George Danezis—Ed. (UCL)
Vasilios Mavroudis (UCL)
Sebastian Meiser (UCL)
Michał Zając (UT)
Rafael Galvez (KUL)
Thomas Zacharias (UEDIN)

Final report

Deliverable D3.3

February 28, 2018
PANORAMIX Project, # 653497, Horizon 2020
http://www.panoramix-project.eu

Horizon 2020
European Union funding
for Research & Innovation

Revision History

Revision	Date	Author(s)	Description
0.1	2018-02-09	Vasilios Mavroudis	Initial Deliverable compilation.
0.2	2018-02-20	George Danezis	Internal WP3 UCL Review.
0.3	2018-02-23	Aggelos Kiayias	External Review (UEDIN).
0.4	2018-02-26	Mirjam Wester	Final Review by Coordinator.
0.5	2018-03-03	Vasilios Mavroudis	Final WP3 UCL Review.
1.0	2018-03-09	Mirjam Wester	Final version submitted to EC.

Executive summary

Deliverable D3.3 summarizes the research & development contributions of the PANORAMIX project partners involved in Work Package 3 (WP3) over months M20–M30 of the project. The work-package as a whole provides solutions to open research problems relating to mix nets that are blocking or otherwise impeding the adoption of advanced mix network designs, within the context of the project, and within the wider community.

Part I presents the project advances in relation to Non-Interactive Zero Knowledge shuffle proofs, cryptographic mechanisms that ensure the correctness of mixing operations, as necessary for applications to electronic voting; Part II and Part III present the evaluation of anonymous communication systems, modelled on the PANORAMIX designs, using advanced attack techniques based on machine learning, and cryptographic techniques to secure mix network routing; Finally, Part IV presents state of the art anonymity definitions and an important resulting new theorem, that formally captures the fundamental trade-offs between anonymity, and system overheads such as bandwidth and latency.

Contents

Executive summary **5**

1 Introduction **8**

I Non-interactive Zero-Knowledge Proofs **13**

2 Efficient non-interactive zero-knowledge shuffles **15**

3 Secure parameter generation **31**

4 Efficient Designated-Verifier Zero-Knowledge Proofs **44**

II Traffic & Routing Analysis **69**

5 Multiparty Routing: Secure Routing for Mixnets **71**

III Communication Patterns & Fingerprinting Techniques **97**

6 Automated Website Fingerprinting through Deep Learning **99**

7 Website Fingerprinting Defenses at the Application Layer **125**

IV Optimal & Adaptive Mixnet Designs **155**

8 Anonymity trilemma: Anonymity, Bandwidth overhead, Latency **157**

9 A cryptographic framework for the privacy of email ecosystems **194**

1. Introduction

This is deliverable D3.3 from the EU PANORAMIX project, and concludes the activity of Work Package 3 relating to the research and advanced development support, largely from academic partners. The work-package as a whole provides solutions to open research problems relating to mix nets that are blocking or otherwise impeding the adoption of advanced mix network designs, within the context of the project, and within the wider community. The precise commitment underlying D3.3 is summarized in the project proposal as:

> Deliverable D3.3 (Final report) final iteration of the NIZK shuffle proof together with security analysis, and an implementation; validation of mix-net design options and refinement of definitions to suit other WPs. [M30]

1.1 Outline of the deliverable

The structure of the deliverable follows closely the description of work as outlined above. In this section, we relate the research papers and technical reports produced to each of the tasks of WP3, and discuss how they relate to deliverable D3.3.

Part I presents the project advances in relation to NIZK shuffle proofs, cryptographic mechanisms that ensure the correctness of mixing operations, as necessary for applications to electronic voting; Part II and Part III present the evaluation of anonymous communication systems, modeled on the PANORAMIX designs, using advanced attack techniques based on machine learning, and cryptographic techniques to secure mix network routing; Finally, Part IV presents state of the art anonymity definitions and an important resulting new theorem, showing that there is a fundamental trade-off between anonymity, and systems overheads such as bandwidth and latency.

1.2 The WP3 results in PANORAMIX and beyond

The results from WP3 directly support other work packages within the project, and are also high quality research results that will support wider engineering and research in the space of anonymous communications. WP4 implements the techniques presented in Part I, as part of supporting the electronic elections use-case fully developed in WP5. WP4 designs, and in particular the Katzenpost mix network, also support cover traffic regimes to foil traffic analysis attacks presented in Part III, that directly support the messaging use-case elaborated in WP7. Part II presents an alternative design strategy for messaging and other mix networks – which can in the future be integrated into practical systems. Finally, the privacy definitions in Part III provide the foundation for how to evaluate the security of mix networks across WP5, WP6 and WP7, how anonymity degrades, and fundamental trade-offs with other desirable properties of networked systems.

Besides the practical relevance to the PANORAMIX project, we are proud to conclude the work in WP3 with a sense that the outputs produced will have a lasting impact beyond the project timeline. In particular the work presented in this deliverable has appeared in very well established computer security and cryptography conferences and journals, such as AsiaCrypt, Conference on Computer and Communications Security, RSA Conference, the Proceedings of Privacy Enhancing Technologies, ACM Transactions on Internet Technology, Network and Distributed System Security Symposium. Besides the dissemination value of such publications, subjecting key aspects of the PANORAMIX innovation through scientific peer review, provides a high degree of external validation for the results presented. This external validation is complementary to the input from partners, and the peer review this specific deliverable has benefited from.

1.3 WP3 tasks and mapping to Deliverable D3.3

In this section, we relate each chapter of this deliverable to the WP3 tasks, and summarize the key contributions to the PANORAMIX project. For each part of the deliverable we introduce some background information, provide the definitions of the relevant tasks and summarize the key contributions of our work.

1.3.1 Part I – Non-interactive Zero-Knowledge Proofs

After the Snowden revelations, there has been a recent surge of interest in constructing cryptographic primitives and protocols secure against active subversion. In the context of PANORAMIX objectives, it seems crucial to provide a secure parameter generation for the protocols developed during the project. Especially taking into account that the proposed zero-knowledge arguments were shown secure in the common reference string (CRS) model, that relies on the honest generation of the CRS. Through WP3 we will disseminate to more use-case oriented packages like WP 5 (e-voting). Our new shuffle arguments were motivated by two factors: 1. Such arguments are crucial for providing privacy in mix networks, especially in the use case of e-voting, 2. Known solutions are usually not efficient enough. In particular, we needed a new argument that scales for a network where the number of messages traversing the mix goes into hundreds of thousands or even millions. Otherwise the whole system would become inefficient with shuffling appointed as a bottleneck, what would deteriorate the usability of the system dramatically.

Definition of Task 3.2.3. The results of WP3.2 will be validated by (a) peer review: that is, by submitting a research paper with a description of the proposed NIZK shuffle proof to one of the top conferences in cryptography or data security, and (b) efficiency and basic correctness test: this will be validated within by an implementation. This subtask starts in parallel with subtask 3.2.2. The main deliverable of this subtask is a result of this validation, consisting of a research paper (that describes the final version of the NIZK shuffle proof together with security proofs; not necessarily published yet) together with an implementation and a short description of thereof.

In compliance with the requirements of the validation Task 3.2.3, we published the following papers in top conferences and journals:

- P. Fauzi and H. Lipmaa "Efficient Culpably Sound NIZK Shuffle Argument Without Random Oracles." In: Sako K. (eds) Topics in Cryptology - CT-RSA 2016. Lecture Notes in Computer Science, vol 9610. 2016.

- P. Fauzi, H. Lipmaa, and M. Zając "A Shuffle Argument Secure in the Generic Model." In Proceedings, Part II, of the 22nd International Conference on Advances in Cryptology — ASIACRYPT 2016.
- H. Lipmaa "Prover-Efficient Commit-and-Prove Zero-Knowledge SNARKs." AFRICACRYPT 2016. Lecture Notes in Computer Science, vol 9646. 2016.
- H. Lipmaa "Optimally Sound Sigma Protocols Under DCRA." 21st International Conference on Financial Cryptography and Data Security 2017.
- H. Lipmaa and K. Pavlyk "Simpler Rate-Optimal CPIR Protocol" 21st International Conference on Financial Cryptography and Data Security 2017.
- P. Fauzi, H. Lipmaa, J. Siim and M. Zając "An Efficient Pairing-Based Shuffle Argument" ASIACRYPT 2017.
- P. Chaidos and G. Couteau. "Efficient Designated-Verifier Non-Interactive Zero-Knowledge Proofs of Knowledge", under submission 2017.

1.3.2 Parts II & III – Traffic & Routing Analysis and Communication Patterns & Fingerprinting Techniques

For the validation Task 3.1.3, we first proposed a series of techniques for traffic and routing analysis, communication patterns inference and fingerprinting and subsequently used those insights to develop efficient countermeasures.

Definition of Task 3.1.3. This task will look at validating both the security and the performance of different design options for mix-nets, including decryption mix-net, re-encryption mix nets and hybrid designs. The security evaluation will focus on rigorous cryptographic arguments of security, as well as the application of modern metrics for the measurement of traffic analysis resistance. The performance evaluation will look at key figures of merit for networks such as latency, bandwidth overhead, and the effect of different optimizations.

Part of this work is outlined in Part II, we released the largest web-fingerprinting dataset ever gathered to date. Our closed-world dataset consists of 900 websites, with traffic traces generated by 2,500 visits each. Our open-world dataset is based on 400,000 unknown websites and 200 monitored websites. We made the generated dataset publicly available, allowing researchers to replicate our results and systematically evaluate new approaches. We then provide a complete overview of existing works and measure their effectiveness, by reevaluating many of them on our dataset and reproduce their results. Moreover, we perform the first systematic exploration of state-of-the-art deep learning (DL) algorithms applied for fingerprinting, and introduce new techniques that reach a high success rate, comparable to the state-of-the-art techniques. Our published papers and technical reports relevant to this task are:

- R. Jansen, M. Juarez, R. Gálvez, T. Elahi, and C. Diaz, "Inside Job: Applying Traffic Analysis to Measure Tor from Within", in Network and Distributed System Security Symposium, IEEE Internet Society, 2018.
- F. Shirazi, M. Simeonovski, M. Asghar, M. Backes, and C. Diaz, "A Survey on Routing in Anonymous Communication Protocols", to appear at ACM Computing Surveys (CSUR), 2018.

- F. Shirazi, E. Andreeva, M. Kohlweiss, and C. Diaz, "Multiparty Routing: Secure Routing for Mixnets", Technical Report, under submission, 2018.
- R. Gálvez and S. Gurses, "The Odyssey: modeling privacy threats in a brave new world", Technical Report, under submission, 2018.

Based on our study on traffic analysis and fingerprinting, we first introduce the first implementation of a server-side web-fingerprinting defense and a simple yet effective lightweight client-side defense, and then explore the space of application-layer defenses specifically designed to counter de-anonymization attacks in anonymous sites (Part III). As part of this work, we have collected the largest (to the best of our knowledge) dataset of sizes and types of content hosted by Tor onion sites. Finally, based on our traffic analysis, fingerprinting, and countermeasure effectiveness measurements, we also proposed a *multiparty routing*, a novel type of anonymous routing that broadens the design space with a fourth kind of routing and comes with important advantages over the previous routing approaches. The papers corresponding this work are:

- E. Balsa, C. Pérez-Solà, and C. Diaz, "Towards Inferring Communication Patterns in Online Social Networks", ACM Transactions on Internet Technology 17(3), 2017.
- V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated Feature Extraction for Website Fingerprinting through Deep Learning", in Network and Distributed System Security Symposium, IEEE Internet Society, 2018.
- B. Overdorf, M. Juarez, G. Acar, C. Diaz, and R. Greenstadt, "How Unique is Your .onion? An Analysis of the Fingerprintability of Tor Onion Services", in ACM Conference on Computer and Communications Security, ACM, pp. 2021-2036, 2017.
- G. Cherubin, J. Hayes, and M. Juarez, "Website Fingerprinting Defenses at the Application Layer", Proceedings on Privacy Enhancing Technologies 2017(2), pp. 187-204, 2017.
- F. Shirazi, E. Andreeva, M. Kohlweiss, and C. Diaz, "Multiparty Routing: Secure Routing for Mixnets", ArXiv preprint arXiv:1708.03387, 2017.

1.3.3 Part IV – Optimal & Adaptive Mixnet Designs

This part outlines the works that fall under Task 3.3.3. Initially, we confirm the previously conjectured relationship between bandwidth overhead, latency overhead and anonymity. We find that there are fundamental bounds on sender and recipient anonymity properties of a protocol that directly depend on the introduced bandwidth and latency overheads. To inform future mixnet decisions and better quantify efficiency objectives, we derive *upper* bounds on anonymity as functions of bandwidth overhead and latency overhead.

Definition of Task 3.3.3. At the validation stage we will apply the designs and evaluate the level that they meet the stated objectives in the contexts of mix-nets. During validation we will employ peer review, invoke the expertise of our advisory board and design simulations in order to establish that the protocols meet their efficiency objectives.

In order for our model to be applicable in different scenarios, we consider two prominent adversary classes: global passive network-level adversaries and strictly stronger adversaries that additionally (passively) compromise some protocol parties (e.g., relays in case of Tor). Using this setup, we analyze the trade-off between latency overhead and bandwidth overhead required to achieve *strong*

anonymity, i.e., anonymity up to a negligible (in a security parameter η) chance of failure. We also assess the practical impact of our results by analyzing prominent anonymous communication protocols. Furthermore, we enable the security analysis of email ecosystems based on the standards of state-of-the art cryptography. Using these techniques, we can formally analyse the anonymity of the PANORAMIX email ecosystem and compare with the anonymity of other existing solutions. The following works are currently in progress or are going through the peer-review process as part of their submission to conferences:

- D. Das, S. Meiser, E. Mohammadi and A. Kate, "Anonymity Trilemma: Strong Anonymity, Low Bandwidth Overhead, Low Latency — Choose Two", 2018 IEEE Symposium on Security and Privacy (SP), pp. 170-188, 2018.
- T. Zacharias and A. Kiayias, "E-Mail Anonymity Modelling", Technical Report, 2018.

Part I

Non-interactive Zero-Knowledge Proofs

2. Efficient non-interactive zero-knowledge shuffles

2.1 Introduction

Consider the case of using mix-networks [12] in e-voting, where n voters individually encrypt their vote using a blindable public-key cryptosystem and send the encrypted votes to a bulletin board. After the vote casting period ends, the first mix-server gets all encrypted votes from the bulletin board. The mix-servers are ordered sequentially, creating a mix-network, and it is assumed that some of them are honest. The kth mix-server obtains input ciphertexts $(\mathfrak{M}_i)_{i=1}^n$, shuffles them, and sends the resulting ciphertext tuple $(\mathfrak{M}'_i)_{i=1}^n$ to the next mix-server. Shuffling means that the mix-server generates a random permutation $\sigma \leftarrow_r S_n$ and a vector $\vec{t}$ of randomizers, and sets $\mathfrak{M}'_i = \mathfrak{M}_{\sigma(i)} + \mathsf{Enc}_{\mathsf{pk}}(\mathfrak{o}; t_i)$. [1]

If at least one of the mix-servers behaves honestly, the link between a voter and his votes is completely hidden. However, in the malicious model, a corrupt mix-server can do an incorrect shuffle, resulting in a set of decrypted votes that do not reflect the original voters' votes. Hence there needs to be some additional steps to achieve security against corruption.

The cryptographically prudent way to proceed is to get each mix-server to prove in zero-knowledge [25] that her shuffle was done correctly. The resulting proof is known as a *(zero-knowledge) shuffle argument*. Based on earlier work [32, 39], in CT-RSA 2016, Fauzi and Lipmaa (FL, [17]) proposed the then most efficient shuffle argument in the common reference string (CRS, [10]) model in terms of prover's computation.[2] Importantly, the FL shuffle argument is based on the standard Elgamal cryptosystem. The culpable soundness [32, 33] of the FL shuffle argument is proven under a knowledge assumption [14] and three computational assumptions. Intuitively, culpable soundness means that if a cheating adversary produces an invalid (yet acceptable) shuffle together with the secret key, then one can break one of the underlying knowledge or computational assumptions. The later scheme, by Fauzi, Lipmaa and Zając [19] made shuffles more efficient for the verifier with a small loss in the prover efficiency. The third scheme by Fauzi, Lipmaa, Siim, Zając [18] gives the most efficient scheme for prover's and verifier's online computation for the time being.

[1] Throughout this chapter, we use additive notation combined with the bracket notation of [16]. We also denote group elements by using the Fraktur script as in $\mathfrak{M}_i$ or $\mathfrak{o}$. Thus, adding $\mathsf{Enc}_{\mathsf{pk}}(\mathfrak{o}; t_i)$ results in a blinded version of $\mathfrak{M}_{\sigma(i)}$.

[2] Many random-oracle model shuffle arguments are known, such as [20, 2, 29]. We will not provide comparisons with such arguments or discussions about the benefits of the CRS vs the random oracle model. We remark that the CRS can be created by using multi-party computation, see, e.g., [6]

2.1.1 Shuffles as a part of the PANORAMIX project

The design of efficient shuffle arguments is an important part of WP3. Through WP3 we will disseminate to more use-case oriented packages like WP 5 (e-voting). The requirement for new shuffle arguments can be easily motivated twofold. The argument itself is crucial for providing privacy of the mix network, especially in the use case of e-voting. On the other hand, known solutions are usually not efficient enough. For a network where the number of messages to mix goes into hundreds of thousands or even millions, new arguments were a must. Otherwise the whole system would become inefficient with shuffling appointed as a bottleneck, what would compromise usability of the system in a great manner.

As a technique to provide robust and private mixing, shuffle arguments are crucial for Tasks 3.1.1 (notions of unlinkability and anonymity) and 3.1.3 (security evaluation). Development of this primitive was stated as of independent interest in Task 3.2. We justify the model we used as it is stated in Task 3.2.1; propose shuffle arguments as required in Task 3.2.2. Furthermore, we validate our results (Task 3.2.3) – all three proposed arguments were accepted to top-notch cryptographic conferences: the argument by Fauzi and Lipmaa to CT-RSA and the arguments by Fauzi, Lipmaa and Zając and Fauzi, Lipmaa, Siim and Zając to Asiacrypt. Task 3.3, focused on making mix networks more efficient will benefit from the below results as well. We emphasize that proposed protocols are currently the fastest CRS-based shuffle arguments (we motivated using this model above).

Knowledge gained during designing these arguments has been proven useful in WP 5 (Use case: e-voting) Tasks: 5.1 (Requirement analysis and specification), 5.2 (Design), 5.3 (Validation and product implementation). The arguments make it possible to process (i.e. mix, re-encrypt and provide a proof) almost 100,000 ciphertexts in about 2 minutes. Verification time for such an amount of messages is under 3 minutes. Both of these numbers were computed on a standard PC with i5 processor and 8 GB of RAM. We can easily assume that these numbers would be much better if the computations are done on a server, what indeed is the case for e-voting. Furthermore, since the numbers are not large, verification of the whole e-voting process could be processed on a PC.

2.1.2 Previous work

During the PANORAMIX project the team working at the University of Tartu published three shuffle-related articles, each of them in the top cryptographic conferences. Given that shuffles of Fauzi, Lipmaa [17] and Fauzi, Lipmaa, Zając [19] were described in details in Deliverable 3.2, we provide here only short description of these results.

Fauzi-Lipmaa shuffle argument

In *Efficient Culpably Sound NIZK Shuffle Argument without Random Oracles* [17] Fauzi and Lipmaa proposed a new, more efficient, shuffle argument in the CRS model. Its online prover's computational complexity was dominated by only two $(n+1)$-wide multi-exponentiations, where n is the number of ciphertexts. Compared to the previously fastest argument by Lipmaa and Zhang [39], it satisfied a stronger notion of soundness (culpable soundness). While the new shuffle argument was still at least 2 times slower than the most efficient known random oracle based shuffle arguments, it had almost optimal *online* prover's computation.

Fauzi-Lipmaa-Zając shuffle argument

In *A Shuffle Argument Secure in the Generic Model* [19] Fauzi, Lipmaa and Zając proposed a random oracle-less NIZK shuffle argument. It had a simple structure, where the first verification equation ascertained that the prover has committed to a permutation matrix, the second verification equation ascertained that the same permutation was used to permute the ciphertexts, and the third verification equation ascertained that input ciphertexts were "correctly" formed. The argument had 3.5 times more efficient verification than the up-to-now most efficient shuffle argument by Fauzi and Lipmaa [17]. Compared to the Fauzi-Lipmaa shuffle argument, they

1. removed the use of knowledge assumptions and prove that the scheme is sound in the generic bilinear group model, and
2. proved the standard soundness, instead of the culpable soundness.

2.1.3 Fauzi-Lipmaa-Siim-Zając shuffle argument

In *An Efficient Pairing-Based Shuffle Argument* [18] Fauzi, Lipmaa, Siim and Zając tried to determine whether it possible to construct a NIZK shuffle argument that shares the best features of the FL and the FLZ shuffle arguments. That is, it would use standard Elgamal (and in only one group), be (non-whitebox) sound, have linear-length CRS, have prover as efficient or better than in the FL shuffle argument, and have verifier as efficient or better than in the FLZ shuffle argument. The answer for the question was positive. They constructed a new pairing-based NIZK shuffle argument that is more efficient than prior work in essentially all parameters. As in [17], they use the Elgamal cryptosystem (with plaintexts in $\mathbb{G}_2$), which means that unlike [19], compatible voting mechanisms are not restricted by the size of the plaintext space. Furthermore, they construct more efficient subarguments, which sometimes leads to a significant efficiency gain. Since the CRS has very few elements from $\mathbb{G}_2$, the new shuffle has much simpler soundness proofs than in the case of the FLZ shuffle argument. Moreover, as in [32, 17] (but not in [39, 19]), they do not give the generic adversary in the soundness proof access to the discrete logarithms of encrypted messages. Their high-level approach in the shuffle argument is similar to the approach in the FL shuffle argument except that we they use (significantly) more efficient subarguments.

Fauzi et al. first let the prover choose a permutation matrix and commit separately to its every row. The prover then proves that the committed matrix is a permutation matrix, by proving that each row is a unit vector, including the last row which is computed explicitly. Then they construct a new unit vector argument based on the square span programs of Danezis *et al.* [15]; it is similar to but somewhat simpler than the 1-sparsity argument of [19]. Basically, to show that a vector $\vec{a}$ is unit vector, they choose polynomials $(P_i(X))_{i\in[0\,..\,n]}$ that interpolate a certain matrix (and a certain vector) connected to the definition of "unit vectorness", and then commit to $\vec{a}$ by using a version of the extended Pedersen commitment scheme, $\mathfrak{c} = \sum_{i=1}^{n} a_i[P_i(\chi)]_1 + r[\varrho]_1$ for trapdoor values (χ, ϱ) and randomizer r. (This commitment scheme, though for different polynomials $P_i(X)$, was implicitly used first by Groth [31] in EUROCRYPT 2016, and then used in the FLZ shuffle argument; similar commitment schemes have been used before [26, 30, 37].) The new unit vector argument differs from the corresponding (1-sparsity) argument in [19] by a small optimization that makes it possible to decrease the number of trapdoor elements by one. If the unit vector argument for each row is accepting, it follows that the committed matrix is a permutation matrix [17]. The knowledge-soundness proof of the new unit vector argument is almost trivial, in contrast to the very complex machine-assisted knowledge-soundness proof in [19].

The authors then use the same high-level idea as previous NIZK shuffle arguments [32, 39, 17, 19] to obtain a shuffle argument from a permutation matrix argument. Namely, they construct a

verification equation that holds tautologically under a corresponding KerMDH [41] assumption. That is, if the permutation matrix argument is knowledge-sound, the mentioned verification equation holds, and the KerMDH assumption holds, then the prover has used his committed permutation matrix to also shuffle the ciphertexts.

However, as in [32, 39, 17, 19], the resulting KerMDH assumption itself will not be secure if one use here the same commitment scheme as before. Intuitively, this is since the polynomials $P_i(X)$ were carefully chosen to make the permutation matrix argument as efficient as possible. Therefore, Fauzi et al. define an alternative version of the extended Pedersen commitment scheme with the commitment computed as $\hat{\mathfrak{c}} = \sum a_i[\hat{P}_i(\chi)]_1 + r[\hat{\varrho}]_1$ for trapdoor values $(\chi, \hat{\varrho})$ and randomizer r. Here, $\hat{P}_i(X)$ are well-chosen polynomials that satisfy a small number of requirements, including that $\{P_i(X)\hat{P}_j(X)\}_{1\leq i,j\leq n}$ is linearly independent.

Moreover, the authors need an efficient argument (that they call, following [17], a *same-message argument*) to show that $\mathfrak{c}$ and $\hat{\mathfrak{c}}$ commit to the same vector $\vec{a}$ (and use the same randomness r) while using different shrinking commitment schemes. They first write down the objective of this argument as the requirement that $\binom{\vec{a}}{r}$ belongs to a subspace generated by a certain matrix $\vec{M}$. After doing that, the authors use the quasi-adaptive NIZK (QANIZK, [34, 35]) argument of Kiltz and Wee (EUROCRYPT 2015, [36]) for linear subspaces to construct an efficient same-message argument. Since it additionally need to be knowledge-sound, the authors give a proof in GBGM.

The new consistency argument is similar to but again more efficient than the consistency arguments of previous pairing-based shuffles. Here, the authors crucially use the fact that neither the permutation matrix argument nor the same-message argument add "too many" $\mathbb{G}_2$ elements to the CRS. Hence, while the Groth-Lu and FL shuffle arguments require two consistency verification equations, for us it suffices to only have one. (The Lipmaa-Zhang [39] and FLZ shuffle arguments have only one consistency verification equation, but this was compensated by using a non-standard cryptosystem with ciphertexts of length 6.)

In fact, they generalize the consistency argument to prove that given a committed matrix $\vec{E}$ and two tuples of ciphertexts $\vec{\mathfrak{M}}'$ and $\vec{\mathfrak{M}}$, it holds that $\mathsf{Dec}_{\mathsf{sk}}(\vec{\mathfrak{M}}') = \vec{E} \cdot \mathsf{Dec}_{\mathsf{sk}}(\vec{\mathfrak{M}})$. Moreover, they prove that the consistency argument is culpably sound [32, 33] under a suitable KerMDH [41] assumption, and prove that the concrete KerMDH assumption holds in the GBGM.

Finally, the authors give a standard (i.e., non-culpable) soundness proof for the full shuffle argument, assuming that the used commitment scheme is computationally binding, the same-message argument and the permutation matrix argument are knowledge-sound, and the consistency argument is culpably sound. Additionally, as in the FLZ shuffle argument, they use batching techniques [4] to speed up verification time. However, they use batching in a more aggressive manner than in the FLZ shuffle argument.

2.1.4 Efficiency Comparison.

Fauzi, Lipmaa, Siim, Zając in [18] provide an implementation of a pairing-based shuffle argument. The implementation is built on top of the freely available libsnark library, [8]. In fact, the authors implement two versions of th shuffle argument, where in the second version they switch the roles of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. In the first case they get better overall prover's computation, while in the second case they get the most efficient online computation for both prover and verifier, and overall the most efficient verifier.

Table 2.1 shows a comparison between both versions of the shuffle argument and prior state of the art CRS-based shuffle arguments with either the best prover's computational complexity or best verifier's computational complexity.

Hence, we for instance do not include in this comparison table prior work by Groth and Lu [32]

Table 2.1: A comparison of the new NIZK shuffle argument and prior work by Fauzi and Lipmaa (FL, [17]), and Fauzi, Lipmaa and Zając (FLZ, [19]). We include shuffling itself to the efficiency analysis of communication and prover's computation.

	FL	FLZ	Current Work	Current Work ($\mathbb{G}_1/\mathbb{G}_2$ switched)
$\lvert\text{CRS}\rvert$ in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$	$(6n+8, 2n+8, 1)$	$(2n+6, n+7, 1)$	$(4n+7, n+7, 1)$	$(n+7, 4n+7, 1)$
Communication	$(7n+2, 2n, 0)$	$(5n+1, 4n+2, 0)$	$(4n-1, 3n+1, 0)$	$(3n+1, 4n-1, 0)$
Prover's computation				
Exp. in $(\mathbb{G}_1, \mathbb{G}_2)$	$(2n-1, 0)$	$(2n, 0)$	$(n, 0)$	$(0, n)$
Fb-exp. in $(\mathbb{G}_1, \mathbb{G}_2)$	$(8n-2, 2n-2)$	$(4n-1, 4n-1)$	$(3n-1, 3n-1)$	$(3n-1, 3n-1)$
M. exp. in $(\mathbb{G}_1, \mathbb{G}_2)$	$(6n+6, 2n+2)$	$(3n+3, 5n+5)$	$(n+1, 2n+2)$	$(2n+2, n+1)$
Units	4.84	5.34	2.87	4.25
Prover's online computation				
M. exp. in $(\mathbb{G}_1, \mathbb{G}_2)$	$(2n+2, 0)$	$(3n+3, 3n+3)$	$(0, 2n+2)$	$(2n+2, 0)$
Units	0.26	1.2	0.54	0.26
Verifier's computation				
Exp. in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$	$(0, 0, 0)$	$(7n+6, 7, 1)$	$(n, 2n+3, 1)$	$(2n+3, n, 1)$
M. exp. in $(\mathbb{G}_1, \mathbb{G}_2)$	$(0, 0)$	$(4n, 3n)$	$(4n-4, 0)$	$(0, 4n-4)$
Pairing product	$18n+6$	$3n+6$	$3n+6$	$3n+6$
Units	38.52	14.75	12.98	12.02
Verifier's online computation				
Exp. in $(\mathbb{G}_1, \mathbb{G}_2)$	$(0, 0)$	$(6n+3, 3)$	$(0, 2n+1)$	$(2n+1, 0)$
M. exp. in $(\mathbb{G}_1, \mathbb{G}_2)$	$(0, 0)$	$(3n, 3n)$	$(0, 0)$	$(0, 0)$
Pairing product	$8n+4$	$2n+3$	$2n+1$	$2n+1$
Units	17.12	11.48	9.32	6.28
Lifted encryption	No	Yes	No	No
Soundness	Culpable	White-box	Full	Full

or Lipmaa and Zhang [39], since their shuffle arguments are slower than [17] and [19] in both prover's and verifier's computation. We also do not include the shuffle argument of González and Ràfols [27] since it has quadratic CRS length. In each row, the argument with best efficiency or best security property is highlighted.

One should compare the number of units, which is a weighted sum of different exponentiations and pairings, and hence takes into account the fact that (say) computations in $\mathbb{G}_1$ and $\mathbb{G}_2$ take different time. Moreover, this table counts separately the number of general exponentiations, multi-exponentiations, and fixed-base exponentiations, since the last two can be much more efficient than general exponentiations. We take this into account by using different unit values for these three types of exponentiations, see Table 2.2 for the number of units required for each type of operation. Note that we use the implementation of each operation in libsnark to compute the number of units.

Table 2.3 gives the running time of the [18] shuffle argument (without and with switching the groups) on our test machine. As seen from this table, the preliminary implementation enables one to prove a shuffle argument in less than 1 minute and verify it in less than 1.5 minutes for $n = 100\,000$. After switching the groups, the prover's online computation takes less than 15 seconds and online verification takes less than 3 minutes for $n = 300\,000$. This means that the new shuffle argument is actually ready to be used in practice.

Table 2.2: Efficiency comparison of various operations based on libsnark. Units in the last column show efficiency relative to n exponentiations in $\mathbb{G}_1$ for $n = 100,000$.

	10,000	100,000	1000,000	units
Multi-exp. $\mathbb{G}_1$	0.26s	2.54s	24.2s	0.13
Fixed-base exp. $\mathbb{G}_1$	0.28s	2.40s	18.44s	0.12
Multi-exp. $\mathbb{G}_2$	0.65s	5.54s	48.04s	0.27
Fixed-base exp. $\mathbb{G}_2$	0.75s	5.62s	44.34s	0.28
Exp. $\mathbb{G}_1$	2.15s	20.29s	207.11s	1
Exp. $\mathbb{G}_2$	5.54s	51.10s	506.26s	2.52
Pairing product	4.38s	43.37s	471.72s	2.14
Pairings	10.24s	97.07s	915.21s	4.78
Exp. $\mathbb{G}_T$	10.65s	100.20s	1110.53s	4.94

Table 2.3: Efficiency of the shuffle implementation (in minutes and seconds) using the libsnark library: the original argument (left) and the one with switched groups (right), for various values of n.

	10,000	100,000	300,000
CRS generation	1.7s	13.4s	37.0s
Prover	6.4s	56.7s	2m38.3s
Prover (online)	1.1s	10.2s	32.3s
Verifier	8.5s	1m27.8s	5m18.8s
Verifier (online)	5.7s	1m0.5s	3m29s

	10,000	100,000	300,000
CRS generation	2.6s	20.5s	55.0s
Prover	9.1s	1m24.0s	4m2.4s
Prover (online)	0.5s	4.1s	13.5s
Verifier	8.3s	1m22.0s	4m49.2s
Verifier (online)	5.0s	49.9s	2m55.5s

2.2 Preliminaries

Let S_n be the symmetric group on n elements. For a (Laurent) polynomial or a rational function f and its monomial μ, denote by $\mathrm{coeff}_\mu(f)$ the coefficient of μ in f. Write $f(\lambda) \approx_\lambda g(\lambda)$, if $f(\lambda) - g(\lambda)$ is negligible as a function of λ.

2.2.1 Bilinear maps

Let λ be the security parameter. Let q be a prime of length $O(\lambda)$ bits. Assume use of a secure bilinear group generator $\mathsf{genbp}(1^\lambda)$ that returns $\mathsf{gk} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are three multiplicative groups of order q, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Here, we denote the elements of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ as in $[1]_1$, $[1]_2, [1]_T$ and by using the Fraktur typeface. We use additive notation for groups operations. It is required that $\hat{e}$ is bilinear (i.e., $\hat{e}([a]_1, [b]_2) = ab \cdot \hat{e}([1]_1, [2]_2)$), efficiently computable, and non-degenerate. Equivalently, we write also $\hat{e}([a]_1, [b]_2) = [a]_1 \bullet [b]_2$ Assume that $[1]_z$ is a generator of $\mathbb{G}_z$ for $i \in \{1, 2\}$, and set $[1]_T \leftarrow ([1]_1 \bullet [2]_2)$.

For $\lambda = 128$, the current recommendation is to use an optimal (asymmetric) Ate pairing over a subclass of Barreto-Naehrig curves. In that case, at security level of $\lambda = 128$, an element of $\mathbb{G}_1/\mathbb{G}_2/\mathbb{G}_T$ can be represented in respectively 256/512/3072 bits.

2.2.2 Zero knowledge

Let $\mathcal{L}_{\mathcal{R}} = \{\mathsf{x} : \exists \mathsf{w}, (\mathsf{x}, \mathsf{w}) \in \mathcal{R}\}$ be an **NP**-language. A *non-interactive zero-knowledge argument system* Ψ for $\mathcal{R}$ consists of six PPT algorithms:

CRS trapdoor generator: K_{tc} is a probabilistic algorithm that, given $(\mathcal{R}, \mathsf{z}_{\mathcal{R}}) \in \text{range}(\mathcal{R}(1^\lambda))$, outputs a *CRS trapdoor* tc. Otherwise, it outputs a special symbol $\perp$.

Simulation trapdoor generator: K_{ts} is a *deterministic* algorithm that, given $(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{tc})$ where $(\mathcal{R}, \mathsf{z}_{\mathcal{R}}) \in \text{range}(\mathcal{R}(1^\lambda))$ and $\mathsf{tc} \in \text{range}(\mathsf{K}_{\mathsf{tc}}(\mathcal{R}, \mathsf{z}_{\mathcal{R}})) \setminus \{\perp\}$, outputs the *simulation trapdoor* ts. Otherwise, it outputs $\perp$.

CRS generator: $\mathsf{K}_{\mathsf{crs}}$ is a *deterministic* algorithm that, given $(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{tc})$, where $(\mathcal{R}, \mathsf{z}_{\mathcal{R}}) \in \text{range}(\mathcal{R}(1^\lambda))$ and $\mathsf{tc} \in \text{range}(\mathsf{K}_{\mathsf{tc}}(\mathcal{R}, \mathsf{z}_{\mathcal{R}})) \setminus \{\perp\}$, outputs crs. Otherwise, it outputs $\perp$. For the sake of efficiency and readability, we divide crs into $\mathsf{crs}_{\mathsf{P}}$ (the part needed by the prover), $\mathsf{crs}_{\mathsf{V}}$ (the part needed by the verifier), and $\mathsf{crs}_{\mathsf{CV}}$ (the part needed only by CV and not by P or V).

Prover: P is a probabilistic algorithm that, given $(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{crs}_{\mathsf{P}}, \mathsf{x}, \mathsf{w})$ for $\mathsf{CV}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{crs}) = 1$ and $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$, outputs an argument π. Otherwise, it outputs $\perp$.

Verifier: V is a probabilistic algorithm that, given $(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{crs}_{\mathsf{V}}, \mathsf{x}, \pi)$, returns either 0 (reject) or 1 (accept).

Simulator: Sim is a probabilistic algorithm that, given $(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{crs}, \mathsf{ts}, \mathsf{x})$ where $\mathsf{CV}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{crs}) = 1$, outputs an argument π.

We also define the CRS generation algorithm $\mathsf{K}(\mathcal{R}, \mathsf{z}_{\mathcal{R}})$ that first sets $\mathsf{tc} \leftarrow \mathsf{K}_{\mathsf{tc}}(\mathcal{R}, \mathsf{z}_{\mathcal{R}})$ and then outputs $(\mathsf{crs} \,\|\, \mathsf{ts}) \leftarrow (\mathsf{K}_{\mathsf{crs}} \,\|\, \mathsf{K}_{\mathsf{ts}})(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{tc})$.

One can remove Sim from the definition of Ψ, and instead require that for each PPT verifier V^* there exists a corresponding PPT simulator Sim.

2.2.3 Security definitions

An NIZK argument has to satisfy various security definitions. The most important ones are *completeness* (an honest prover convinces an honest verifier, and an honestly generated CRS passes the CRS verification test), *computational knowledge-soundness* (if a prover convinces an honest verifier, then he knows the corresponding witness), and *zero knowledge*.

Definition 1 (Perfect Completeness [31])**.** *A non-interactive argument Ψ is* perfectly complete *for $\mathcal{R}$, if for all λ, all $(\mathcal{R}, \mathsf{z}_{\mathcal{R}}) \in \text{range}(\mathcal{R}(1^\lambda))$, $\mathsf{tc} \in \text{range}(\mathsf{K}_{\mathsf{tc}}(\mathcal{R}, \mathsf{z}_{\mathcal{R}})) \setminus \{\perp\}$, and $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$,*

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{K}_{\mathsf{crs}}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{tc}) : \mathsf{V}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{crs}_{\mathsf{V}}, \mathsf{x}, \mathsf{P}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{crs}_{\mathsf{P}}, \mathsf{x}, \mathsf{w})) = 1\right] = 1 \ .$$

Definition 2 (Computational Knowledge-Soundness [31])**.** *Ψ is computationally (adaptively)* knowledge-sound *for $\mathcal{R}$, if for every NUPPT $\mathcal{A}$, there exists a NUPPT extractor $\mathsf{Ext}_{\mathcal{A}}$, s.t. for all λ,*

$$\Pr\left[\begin{array}{l}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}) \leftarrow \mathcal{R}(1^\lambda), (\mathsf{crs} \,\|\, \mathsf{ts}) \leftarrow \mathsf{K}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}), \\ r \leftarrow_r \mathsf{RND}(\mathcal{A}), ((\mathsf{x}, \pi) \,\|\, \mathsf{w}) \leftarrow (\mathcal{A} \,\|\, \mathsf{X}_{\mathcal{A}})(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{crs}; r) : \\ (\mathsf{x}, \mathsf{w}) \notin \mathcal{R} \wedge \mathsf{V}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{crs}_{\mathsf{V}}, \mathsf{x}, \pi) = 1\end{array}\right] \approx_\lambda 0 \ .$$

Here, $\mathsf{z}_{\mathcal{R}}$ can be seen as a common auxiliary input to $\mathcal{A}$ and $\mathsf{X}_{\mathcal{A}}$ that is generated by using a benign [9] relation generator; we recall that we just think of $\mathsf{z}_{\mathcal{R}}$ as being the description of a secure bilinear group. A knowledge-sound argument system is called an *argument of knowledge*.

Next, we define statistically unbounded ZK.

Definition 3 (Statistically Unbounded ZK [28])**.** *Ψ is* statistically unbounded Sub-ZK *for $\mathcal{R}$, if for all λ, all $(\mathcal{R}, \mathsf{z}_{\mathcal{R}}) \in \text{range}(\mathcal{R}(1^\lambda))$, and all computationally unbounded $\mathcal{A}$, $\varepsilon_0^{unb} \approx_\lambda \varepsilon_1^{unb}$, where*

$$\varepsilon_b^{unb} = \Pr[(\mathsf{crs} \,\|\, \mathsf{ts}) \leftarrow \mathsf{K}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}) : \mathcal{A}^{\mathsf{O}_b(\cdot,\cdot)}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{crs}) = 1] \ .$$

Here, the oracle $\mathsf{O}_0(\mathsf{x},\mathsf{w})$ *returns* $\perp$ *(reject) if* $(\mathsf{x},\mathsf{w}) \notin \mathcal{R}$*, and otherwise it returns* $\mathsf{P}(\mathcal{R},\mathsf{z}_{\mathcal{R}},\mathsf{crs}_{\mathsf{P}},\mathsf{x},\mathsf{w})$*. Similarly,* $\mathsf{O}_1(\mathsf{x},\mathsf{w})$ *returns* $\perp$ *(reject) if* $(\mathsf{x},\mathsf{w}) \notin \mathcal{R}$*, and otherwise it returns* $\mathsf{Sim}(\mathcal{R},\mathsf{z}_{\mathcal{R}},\mathsf{crs},\mathsf{ts},\mathsf{x})$*.* Ψ *is* perfectly unbounded Sub-ZK for $\mathcal{R}$ *if one requires that* $\varepsilon_0^{unb} = \varepsilon_1^{unb}$*.*

2.2.4 Generic bilinear group model

The soundness of [19, 18] and new assumption in [17] are proven in the generic bilinear group model.

We start by picking a random asymmetric bilinear group $\mathsf{gk} := (p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,\hat{e}) \leftarrow \mathsf{genbp}(1^\lambda,n)$. Consider a black box $\mathbf{B}$ that can store values from additive groups $\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T$ in internal state variables $\mathsf{cell}_1,\mathsf{cell}_2,\ldots$, where for simplicity we allow the storage space to be infinite (this only increases the power of a generic adversary). The initial state consists of some values $(\mathsf{cell}_1,\mathsf{cell}_2,\ldots,\mathsf{cell}_{|inp|})$, which are set according to some probability distribution. Each state variable cell_i has an accompanying type $\mathsf{type}_i \in \{1,2,T,\perp\}$. We assume initially $\mathsf{type}_i = \perp$ for $i > |inp|$. The black box allows computation operations on internal state variables and queries about the internal state. No other interaction with $\mathbf{B}$ is possible.

Let Π be an allowed set of computation operations. A computation operation consists of selecting a (say, t-ary) operation $f \in \Pi$ together with $t+1$ indices $i_1,i_2,\ldots,i_{t+1}$. Assuming inputs have the correct type, $\mathbf{B}$ computes $f(\mathsf{cell}_{i_1},\ldots,\mathsf{cell}_{i_t})$ and stores the result in $\mathsf{cell}_{i_{t+1}}$. For a set Σ of relations, a query consists of selecting a (say, t-ary) relation $\varrho \in \Sigma$ together with t indices $i_1,i_2,\ldots,i_t$. Assuming inputs have the correct type, $\mathbf{B}$ replies to the query with $\varrho(\mathsf{cell}_{i_1},\ldots,\mathsf{cell}_{i_t})$. In the GBGM, we define $\Pi = \{+,\hat{e}\}$ and $\Sigma = \{=\}$, where

1. On input $(+,i_1,i_2,i_3)$: if $\mathsf{type}_{i_1} = \mathsf{type}_{i_2} \neq \perp$ then set $\mathsf{cell}_{i_3} \leftarrow \mathsf{cell}_{i_1} + \mathsf{cell}_{i_2}$ and $\mathsf{type}_{i_3} \leftarrow \mathsf{type}_{i_1}$.
2. On input $(\hat{e},i_1,i_2,i_3)$: if $\mathsf{type}_{i_1} = 1$ and $\mathsf{type}_{i_2} = 2$ then set $\mathsf{cell}_{i_3} \leftarrow \hat{e}(\mathsf{cell}_{i_1},\mathsf{cell}_{i_2})$ and $\mathsf{type}_{i_3} \leftarrow T$.
3. On input $(=,i_1,i_2)$: if $\mathsf{type}_{i_1} = \mathsf{type}_{i_2} \neq \perp$ and $\mathsf{cell}_{i_1} = \mathsf{cell}_{i_2}$ then return 1. Otherwise return 0.

Since we are proving lower bounds, we will give a generic adversary $\mathcal{A}$ additional power. We assume that all relation queries are for free. We also assume that $\mathcal{A}$ is successful if after τ operation queries, he makes an equality query $(=,i_1,i_2)$, $i_1 \neq i_2$, that returns 1; at this point $\mathcal{A}$ quits. Thus, if $\mathsf{type}_i \neq \perp$, then $\mathsf{cell}_i = F_i(\mathsf{cell}_1,\ldots,\mathsf{cell}_{|inp|})$ for a polynomial F_i known to $\mathcal{A}$.

2.3 FLSZ shuffle argument

In this section we show intuitions behind the new shuffle argument proposed by Fauzi, Lipmaa, Siim and Zając in [18]. We also present and explain subarguments used therein.

Intuitively, in the new shuffle argument the prover first commits to the permutation σ (or more precisely, to the corresponding permutation matrix), then executes three subarguments (the same-message, the permutation matrix, and the consistency arguments). Each of the subarguments corresponds to one check performed by the verifier (see Prot. 2). However, since all subarguments use the same CRS, they are not independent. For example, the permutation matrix argument uses the $((P_i(X))_{i=1}^n, X_\varrho)$-commitment scheme and the consistency argument uses the $((\hat{P}_i(X))_{i=1}^n, X_{\hat{\varrho}})$-commitment scheme for different polynomials $(\hat{P}_i(X))_{i=1}^n$. Both commitment schemes share a part of their trapdoor (χ), while the second part of the trapdoor is different (either ϱ or $\hat{\varrho}$). Moreover, the knowledge-soundness of the same-message argument is a prerequisite for the knowledge-soundness of the permutation matrix argument. The verifier recovers explicitly the commitment to the last row of the permutation matrix (this guarantees that the committed matrix is left stochastic), then verifies the three subarguments.

$\mathsf{K}_{\mathsf{crs}}(\mathsf{gk}, n)$: Generate random $(\chi, \beta, \hat{\beta}, \varrho, \hat{\varrho}, \mathsf{sk}) \leftarrow_r \mathbb{Z}_q^3 \times (\mathbb{Z}_q^*)^2 \times \mathbb{Z}_q$. Denote $\vec{P} = (P_i(\chi))_{i=1}^n$, $P_0 = P_0(\chi)$, and $\vec{\hat{P}} = (\hat{P}_i(\chi))_{i=1}^n$. Let $\mathsf{crs}_{sm} \leftarrow \Big([(\beta P_i + \hat{\beta}\hat{P}_i)_{i=1}^n, \beta\varrho + \hat{\beta}\hat{\varrho}]_1, [\beta, \hat{\beta}]_2^\top\Big)$,

$$\mathsf{crs}_{pm} \leftarrow \begin{pmatrix} [1, P_0, (((P_i + P_0)^2 - 1)/\varrho)_{i=1}^n, \sum_{i=1}^n P_i, \sum_{i=1}^n \hat{P}_i]_1, \\ [P_0, \sum_{i=1}^n P_i]_2, [1]_T \end{pmatrix},$$

$\mathsf{crs}_{con} \leftarrow [\begin{smallmatrix}\vec{P} \\ \hat{\varrho}\end{smallmatrix}]_1$. Set $\mathsf{crs} \leftarrow \Big(\mathsf{pk} = [1, \mathsf{sk}]_2, [\begin{smallmatrix}\vec{P} \\ \varrho\end{smallmatrix}]_1, [\begin{smallmatrix}\vec{P} \\ \varrho\end{smallmatrix}]_2, \mathsf{crs}_{sm}, \mathsf{crs}_{pm}, \mathsf{crs}_{con}\Big)$. Set $\mathsf{td} \leftarrow (\chi, \hat{\varrho})$. Return $(\mathsf{crs}, \mathsf{td})$.

$\mathsf{P}(\mathsf{gk}, \mathsf{crs}, \vec{\mathfrak{M}} \in \mathbb{G}_2^{n\times 2}; \sigma \in S_n, \vec{t} \in \mathbb{Z}_q^n)$:

1. For $i = 1$ to $n-1$: // commits to the permutation σ
   (a) $r_i \leftarrow_r \mathbb{Z}_q$; $\mathfrak{r}_i \leftarrow r_i[\varrho]_1$;
   (b) $\mathfrak{A}_i \leftarrow [P_{\sigma^{-1}(i)}]_1 + \mathfrak{r}_i$; $\mathfrak{b}_i \leftarrow [P_{\sigma^{-1}(i)}]_2 + r_i[\varrho]_2$; $\hat{\mathfrak{A}}_i \leftarrow [\hat{P}_{\sigma^{-1}(i)}]_1 + r_i[\hat{\varrho}]_1$;
2. $\mathfrak{A}_n \leftarrow [\sum_{i=1}^n P_i]_1 - \sum_{j=1}^{n-1} \mathfrak{A}_j$; $\mathfrak{b}_n \leftarrow [\sum_{i=1}^n P_i]_2 - \sum_{j=1}^{n-1} \mathfrak{b}_j$;
3. $\hat{\mathfrak{A}}_n \leftarrow [\sum_{i=1}^n \hat{P}_i]_1 - \sum_{j=1}^{n-1} \hat{\mathfrak{A}}_j$;
4. $r_n \leftarrow -\sum_{i=1}^{n-1} r_i$; $\mathfrak{r}_n \leftarrow r_n[\varrho]_1$;
5. For $i = 1$ to n:
   (a) $\mathfrak{d}_i \leftarrow [\beta P_{\sigma^{-1}(i)} + \hat{\beta}\hat{P}_{\sigma^{-1}(i)}]_1 + r_i[\beta\varrho + \hat{\beta}\hat{\varrho}]_1$;
   (b) $\mathfrak{c}_i \leftarrow r_i \cdot (2(\mathfrak{A}_i + [P_0]_1) - \mathfrak{r}_i) + [((P_{\sigma^{-1}(i)} + P_0)^2 - 1)/\varrho]_1$;
6. $r_t \leftarrow_r \mathbb{Z}_q$; $\mathfrak{t} \leftarrow \vec{t}^\top [\vec{\hat{P}}]_1 + r_t[\hat{\varrho}]_1$;
7. For $i = 1$ to n: $\mathfrak{t}'_i \leftarrow t_i \cdot \mathsf{pk}$;
8. $\vec{\mathfrak{M}}' \leftarrow (\mathfrak{M}_{\sigma(i)} + \mathfrak{t}'_i)_{i=1}^n$; // Shuffling, online
9. $\vec{\mathfrak{N}} \leftarrow \vec{r}^\top \vec{\mathfrak{M}} + r_t \cdot \mathsf{pk}$; // Online
10. $\pi_{sm} \leftarrow \vec{\mathfrak{d}}$; // Same-message argument
11. $\pi_{pm} \leftarrow ((\mathfrak{b}_i)_{i=1}^{n-1}, \vec{\mathfrak{c}})$; // Permutation matrix argument
12. $\pi_{con} \leftarrow ((\hat{\mathfrak{A}}_i)_{i=1}^{n-1}, \mathfrak{t}, \vec{\mathfrak{N}})$; // Consistency argument
13. Return $\pi_{sh} \leftarrow (\vec{\mathfrak{M}}', (\mathfrak{A}_j)_{j=1}^{n-1}, \pi_{sm}, \pi_{pm}, \pi_{con})$.

Protocol 1: The CRS generation and the prover of the new shuffle argument.

The full description of the new shuffle argument is given in Prot. 1 (the CRS generation and the prover) and in Prot. 2 (the verifier). The CRS has entries that allow to efficiently evaluate all subarguments, and hence also both commitment schemes. The CRS in Prot. 1 includes three substrings, crs_{sm}, crs_{pm}, and crs_{con}, that are used in the three subarguments. To prove and verify (say) the first subargument (the same-message argument), one needs access to crs_{sm}. However, the adversary of the same-message argument will get access to the full CRS. For the sake of exposition, the verifier's description in Prot. 2 does not include batching. An explanation of batching technique is given in the full version of the paper.

We will next briefly describe the subarguments. Later, we will give more detailed descriptions of each of them.

For the sake of completeness, we also present a batched version of Prot. 2 as Prot. 3.

$\mathsf{V}(\mathsf{gk}, \mathsf{crs}, \vec{\mathfrak{M}}; \vec{\mathfrak{M}}', (\mathfrak{A}_j)_{j=1}^{n-1}, \pi_{sm}, \pi_{pm}, \pi_{con})$:

1. Parse $(\pi_{sm}, \pi_{pm}, \pi_{con})$ as in the prover's Steps 11–12, abort if unsuccessful;
2. Compute $\mathfrak{A}_n$, $\mathfrak{b}_n$, and $\hat{\mathfrak{A}}_n$ as in the prover's Steps 2 and 3;
3. $\alpha \leftarrow_r \mathbb{Z}_q$;
4. For $i = 1$ to n: check that $\mathfrak{d}_i \bullet [1]_2 \stackrel{?}{=} (\mathfrak{A}_i, \hat{\mathfrak{A}}_i) \bullet \begin{bmatrix} \beta \\ \hat{\beta} \end{bmatrix}_2$; // Same-message argument
5. For $i = 1$ to n: check that // Permutation matrix argument
   $(\mathfrak{A}_i + \alpha[1]_1 + [P_0]_1) \bullet (\mathfrak{b}_i - \alpha[1]_2 + [P_0]_2) \stackrel{?}{=} \mathfrak{c}_i \bullet [\varrho]_2 + (1 - \alpha^2)[1]_T$;
6. Check that // Consistency argument
   $[\vec{P}]_1^\top \circ \vec{\mathfrak{M}}' - \vec{\hat{\mathfrak{A}}}^\top \circ \vec{\mathfrak{M}} \stackrel{?}{=} \mathfrak{t} \circ \mathsf{pk} - [\hat{\varrho}]_1 \circ \vec{\mathfrak{N}}$;

Protocol 2: The non-batched verifier of the new shuffle argument.

$\mathsf{V}(\mathsf{gk}, \mathsf{crs}, \vec{\mathfrak{M}}; (\mathfrak{M}'_i)_{i=1}^n, (\mathfrak{A}_j)_{j=1}^{n-1}, \pi_{uv}, \pi_{sm}, \pi_{con})$:

1. Parse $(\pi_{uv}, \pi_{sm}, \pi_{con})$ as in the prover's Steps 11–12,
2. Compute $\mathfrak{A}_n$, $\mathfrak{b}_n$, and $\hat{\mathfrak{A}}_n$ as in the prover's Step 2,
3. Set $(y_{1j})_{j \in [1\,..\,n-1]} \leftarrow_r [1\,..\,t]^{n-1}$, Set $y_{1n} \leftarrow 1$,
4. Set $y_{21} \leftarrow_r [1\,..\,t]$, $y_{22} \leftarrow 1$,
5. $\alpha \leftarrow_r \mathbb{Z}_q$;
6. Check that // Permutation matrix argument
   $\sum_{j=1}^n ((y_{1j}(\mathfrak{A}_j + \alpha[1]_1 + [P_0(\chi)]_1)) \bullet (\mathfrak{b}_j - \alpha[1]_2 + [P_0(\chi)]_2)) = (\vec{y}_1^\top \vec{\mathfrak{c}}) \bullet [\varrho]_2 + (\sum_{j=1}^n y_{1j})(1 - \alpha^2)[1]_T$;
7. Check that $(\vec{y}_1^\top \vec{\mathfrak{d}}) \bullet [1]_2 = (\vec{y}_1^\top (\vec{\mathfrak{A}}, \vec{\hat{\mathfrak{A}}})) \bullet \begin{bmatrix} \beta \\ \hat{\beta} \end{bmatrix}_2$ // Same-message argument
8. Set $\mathfrak{q} \leftarrow \mathfrak{t} \bullet (\mathsf{pk} \cdot \vec{y}_2)$.
9. Check that // Consistency argument, online
   $[\vec{P}]_1^\top \bullet (\vec{\mathfrak{M}}' \vec{y}_2) - \vec{\hat{\mathfrak{A}}}^\top \bullet (\vec{\mathfrak{M}} \vec{y}_2) = \mathfrak{q} - [\hat{\varrho}]_1 \bullet (\vec{\mathfrak{N}} \vec{y}_2)$.

Protocol 3: The batched verifier of the new shuffle argument.

Same-Message Argument.

Consider the subargument of the new shuffle argument where the verifier only computes $\mathfrak{A}_n$ and then performs the check on Step 4 of Prot. 2 for one concrete i. The authors call it the *same-message argument* [17] and motivate this name, by showing that if the same-message argument accepts, then the prover knows a message $\vec{a}$ and a randomizer r, such that $\mathfrak{A}_i = [\sum a_i P_i(\chi) + r\varrho]_1$ and $\hat{\mathfrak{A}}_i = [\sum a_i \hat{P}_i(\chi) + r\hat{\varrho}]_1$ both commit to $\vec{a}$ with randomizer r, by using respectively the $((P_i(X))_{i=1}^n, X_\varrho)$-commitment scheme and the $((\hat{P}_i(X))_{i=1}^n, X_{\hat{\varrho}})$-commitment scheme.

For the same-message argument to be knowledge-sound, they require that $\{P_i(X)\}_{i=1}^n$ and $\{\hat{P}_i(X)\}_{i=1}^n$ are both linearly independent sets.

The argument consists of four PPT algorithms $\mathsf{K}_{\mathsf{crs}}, \mathsf{P}, \mathsf{Sim}, \mathsf{V}$ that can be formalized as follows:

$\mathsf{K}_{\mathsf{crs}_{sm}}(\mathsf{gk}, [\vec{M}]_1 \in \mathbb{G}_1^{2\times(n+1)})$: $\vec{A} \leftarrow_r \mathcal{D}_k$; $\vec{K} \leftarrow_r \mathbb{Z}_q^{2\times k}$; $[\vec{Q}]_1 \leftarrow [\vec{M}]_1^\top \vec{K} \in \mathbb{Z}_q^{(n+1)\times k}$; $\vec{C} \leftarrow \vec{K}\vec{A} \in \mathbb{Z}_q^{2\times k}$; $\mathsf{crs}_{sm} \leftarrow ([\vec{Q}]_1, [\vec{C}]_2, [\vec{A}]_2)$; $\mathsf{td}_{sm} \leftarrow \vec{K}$; Return $(\mathsf{crs}_{sm}, \mathsf{td}_{sm})$;

$\mathsf{P}_{sm}(\mathsf{gk}, \mathsf{crs}_{sm}, \begin{pmatrix}\mathfrak{A}\\ \hat{\mathfrak{A}}\end{pmatrix}, \begin{pmatrix}\vec{a}\\ r\end{pmatrix})$: Return $\pi_{sm} \leftarrow \begin{pmatrix}\vec{a}\\ r\end{pmatrix}^\top [\vec{Q}]_1 \in \mathbb{G}_1^{1\times k}$;

$\mathsf{Sim}_{sm}(\mathsf{gk}, \mathsf{crs}_{sm}, \mathsf{td}_{sm}, \begin{pmatrix}\mathfrak{A}\\ \hat{\mathfrak{A}}\end{pmatrix})$: Return $\pi_{sm} \leftarrow \begin{pmatrix}\mathfrak{A}\\ \hat{\mathfrak{A}}\end{pmatrix}^\top \vec{K} \in \mathbb{G}_1^{1\times k}$;

$\mathsf{V}_{sm}(\mathsf{gk}, \mathsf{crs}_{sm}, \begin{pmatrix}\mathfrak{A}\\ \hat{\mathfrak{A}}\end{pmatrix}, \pi_{sm})$: Check that $\pi_{sm} \bullet [\vec{A}]_2 \stackrel{?}{=} \begin{pmatrix}\mathfrak{A}\\ \hat{\mathfrak{A}}\end{pmatrix}^\top \bullet [\vec{C}]_2$;

Security of the same message argument has been formalized in the following theorem:

Theorem 4. *Assume* $\mathsf{crs} = (\mathsf{crs}_{sm}, \mathsf{aux})$ *where* aux *does not depend on* β *or* $\hat{\beta}$*. The same-message argument has perfect zero knowledge for*

$$\mathcal{L}_{\vec{M}} = \{\begin{pmatrix}\mathfrak{A}\\ \hat{\mathfrak{A}}\end{pmatrix} : \exists \begin{pmatrix}\vec{a}\\ r\end{pmatrix} \in \mathbb{Z}_q^{n+1} : \begin{pmatrix}\mathfrak{A}\\ \hat{\mathfrak{A}}\end{pmatrix} = [\vec{M}]_1 \begin{pmatrix}\vec{a}\\ r\end{pmatrix}\} \ .$$

. It has adaptive knowledge-soundness in the GBGM.

Permutation Matrix Argument.

Consider the subargument of Prot. 1 and Prot. 2, where (i) the prover computes $\vec{\mathfrak{A}}$ and π_{pm}, and (ii) the verifier computes $\mathfrak{A}_n$ and then checks the verification equation on Step 5 of Prot. 2. This argument is called the *permutation matrix argument*. The authors motivate this name, by proving in the GBGM that if the verifier accepts the permutation matrix argument, then either the prover knows how to open $(\mathfrak{A}_1, \ldots, \mathfrak{A}_n)$ as a $((P_i(X))_{i=1}^n, X_\varrho)$-commitment to a permutation matrix or one can break the same-message argument. For this, the authors first prove the security of a subargument of the permutation matrix argument — the unit vector argument [17] — where the verifier performs the verification Step 5 for exactly one i.

For the unit vector argument to be efficient, one need to make a specific choice of the polynomials $P_i(X)$. For the knowledge-soundness of the unit vector argument, they additionally need that $\{P_i(X)\}_{i=0}^n$ and $\{P_i(X)\}_{i=1}^n \cup \{1\}$ are linearly independent. More precisely, in [19], the prover adds $[\alpha + P_0]_1$ to $\mathfrak{a}_i$, while in this case, it is the verifier that adds $[\alpha]_1 + [P_0]_1$ to $\mathfrak{a}_i$ (and similarly, with $\mathfrak{b}_i$). Due to this small change, the authors could make the CRS independent of α, and let the verifier sample a new α at the time of verification. (In fact, it suffices if the verifier chooses α once and then uses it at each verification.) This makes the CRS shorter, and also simplifies the latter soundness proof. For this optimization to be possible, one has to rely on the same-message argument.

Let $\mathcal{U}_n$ be the set of all unit vectors of length n. The unit vector argument is the following subargument of the new shuffle argument:

$\mathsf{K}_{\mathsf{crs}_{uv}}(\mathsf{gk}, n)$: the same as in Prot. 1.

$\mathsf{P}_{uv}(\mathsf{gk}, \mathsf{crs}, \mathfrak{A}_j, (\vec{a} \in \mathcal{U}_n, r))$: Compute $(\mathfrak{b}_j, \mathfrak{c}_j)$ as in Prot. 1.

$\mathsf{V}_{uv}(\mathsf{gk}, \mathsf{crs}, \mathfrak{A}_j, (\mathfrak{b}_j, \mathfrak{c}_j))$: $\alpha \leftarrow_r \mathbb{Z}_q$; Check that $(\mathfrak{A}_j + [\alpha]_1 + [P_0]_1) \bullet (\mathfrak{b}_j + [-\alpha]_2 + [P_0]_2) \stackrel{?}{=} \mathfrak{c}_i \bullet [\varrho]_2 + [1 - \alpha^2]_T$;

Security of the permutation matrix argument has been formalized in the following theorems:

Theorem 5. *The described unit vector argument is perfectly complete and perfectly witness-indistinguishable. Assume that* $\{P_i(X)\}_{i=1}^n \cup \{1\}$*, and* $\{P_i(X)\}_{i=1}^n \cup \{P_0(X)\}$ *are two linearly independent*

sets. Assume that the same-message argument accepts. The unit vector argument is knowledge-sound in the GBGM in the following sense: there exists an extractor X *such that if the verifier accepts Eq. 6 for* $j = i$*, then* X *returns* $(r_j, I_j \in [1\,..\,n])$*, such that*

$$\mathfrak{A}_j = [P_{I_j}(\chi) + r_j X_\varrho]_1 \ . \tag{2.1}$$

Theorem 6. *The permutation matrix argument of this section is perfectly complete and perfectly witness-indistinguishable. Assume that* $\{P_i(X)\}_{i=1}^n \cup \{1\}$ *is a linearly independent set. The permutation matrix argument is knowledge-sound in the GBGM in the following sense: there exists an extractor* X *such that if the verifier accepts the verification equation on Step 5 of Prot. 2 for all* $j \in [1\,..\,n]$*, and* $\mathfrak{A}_n$ *is explicitly computed as in Prot. 2, then* X *outputs* $(\sigma \in S_n, \vec{r})$*, such that for all* $j \in [1\,..\,n]$*,* $\mathfrak{A}_j = [P_{\sigma^{-1}(j)}(\chi) + r_j \varrho]_1$*.*

Consistency Argument.

Consider the subargument of the new shuffle argument where the prover only computes π_{con} and the verifier performs the check on Step 6 of Prot. 2. We will call it the *consistency argument*. The authors motivate this name by showing that if $\vec{\hat{\mathfrak{A}}}$ $(\{\hat{P}_i(X)\}, X_{\hat{\varrho}})$-commits to a permutation, then $\mathsf{Dec}(\mathfrak{M}'_i) = \mathsf{Dec}(\mathfrak{M}_{\sigma(i)})$ for the same permutation σ that the prover committed to earlier. The authors show that the new consistency argument is culpably sound under a (novel) variant of the KerMDH computational assumption [41]. In particular, the KerMDH assumption has to hold even when the adversary is given access to the full CRS of the shuffle argument.

For the consistency argument to be sound (and in particular, for the KerMDH variant to be secure in the GBGM), it is required that $\{\hat{P}_i(X)\}_{i=1}^n$ and $\{P_i(X)\hat{P}_j(X)\}_{1\le i,j\le n}$ are both linearly independent sets.

$\mathsf{K}_{\mathsf{crs}con}(\mathsf{gk}, n)$**:** Return $(\mathsf{crs}_{con}; \mathsf{td}) = ([\vec{M}]_1 = [\begin{smallmatrix}\vec{P}\\ \hat{\varrho}\end{smallmatrix}]_1, \mathsf{aux}; (\chi, \varrho)) \leftarrow \mathcal{D}_{n+1,1}$.

$\mathsf{P}_{con}(\mathsf{gk}, \mathsf{crs}, (\vec{\mathfrak{M}}, \vec{\mathfrak{M}}'), (\vec{E}, \vec{r}))$**:** // $\vec{\mathfrak{M}}' = \vec{E}\vec{\mathfrak{M}} + (t_i \cdot \mathsf{pk})_{i=1}^n$

$\vec{\hat{\mathfrak{A}}} \leftarrow \begin{pmatrix}\vec{E}\\ \vec{r}^\top\end{pmatrix}^\top [\begin{smallmatrix}\vec{P}\\ \hat{\varrho}\end{smallmatrix}]_1$; $r_t \leftarrow_r \mathbb{Z}_q$; $\mathfrak{t} \leftarrow \vec{t}^\top[\vec{P}]_1 + r_t[\varrho]_1$; $\vec{\mathfrak{N}} \leftarrow \vec{r}^\top\vec{\mathfrak{M}} + r_t \cdot \mathsf{pk}$; Return $(\pi_{con} \leftarrow (\vec{\hat{\mathfrak{A}}}, \mathfrak{t}, \vec{\mathfrak{N}}))$.

$\mathsf{V}_{con}(\mathsf{gk}, \mathsf{crs}, (\vec{\mathfrak{M}}, \vec{\mathfrak{M}}'), \pi_{con})$**:** Check that $[\vec{P}]_1^\top \circ \vec{\mathfrak{M}}' - \vec{\hat{\mathfrak{A}}}^\top \circ \vec{\mathfrak{M}} \stackrel{?}{=} \mathfrak{t} \circ \mathsf{pk} - [\hat{\varrho}]_1 \circ \vec{\mathfrak{N}}$.

Security of the consistency argument has been formalized in the theorem below:

Theorem 7. *Assume that* $\mathcal{D}_n^{con}$ *is a valid CRS distribution, where the matrix distribution outputs* $[\vec{M}]_1 = [\begin{smallmatrix}\vec{\hat{P}}(X)\\ \hat{\varrho}\end{smallmatrix}]_1 \in \mathbb{Z}_q^{n+1}$ *for* $(\chi, \hat{\varrho}) \leftarrow_r \mathbb{Z}_q \times \mathbb{Z}_q^*$*. The consistency argument is perfectly complete and perfectly zero knowledge. Assume that the* $\mathcal{D}_n^{con}$*-KerMDH assumption with an auxiliary input holds in* $\mathbb{G}_1$*. Then the consistency argument is culpably sound using* $\mathcal{R}_{con}^{\mathsf{guilt}}$ *with the CRS* $\mathsf{crs} = ([\vec{M}]_1, \mathsf{aux})$*, where*

$$\mathcal{R}_{con,n}^{\mathsf{guilt}} = \left\{ \begin{array}{l} (\mathsf{gk}, (\vec{\mathfrak{M}}, \vec{\mathfrak{M}}', \vec{\hat{\mathfrak{A}}}), (\vec{E}, \vec{r})) : \\ \vec{\hat{\mathfrak{A}}} = \begin{pmatrix}\vec{E}\\ \vec{r}^\top\end{pmatrix}^\top [\begin{smallmatrix}\vec{P}\\ \hat{\varrho}\end{smallmatrix}]_1 \wedge \mathsf{Dec}_{\mathsf{sk}}(\vec{\mathfrak{M}}') \neq \vec{E} \cdot \mathsf{Dec}_{\mathsf{sk}}(\vec{\mathfrak{M}}) \end{array} \right\} \ .$$

Bibliography

[1] B. Abdolmaleki, K. Baghery, H. Lipmaa, and M. Zając. A Subversion-Resistant SNARK. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017 (3)*, volume 10626 of *LNCS*, pages 3–33, Hong Kong, China, Dec. 3–7, 2017. Springer, Cham.

[2] S. Bayer and J. Groth. Efficient Zero-Knowledge Argument for Correctness of a Shuffle. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280, Cambridge, UK, Apr. 15–19, 2012. Springer, Heidelberg.

[3] M. Bellare, G. Fuchsbauer, and A. Scafuro. NIZKs with an Untrusted CRS: Security in the Face of Parameter Subversion. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016 (2)*, volume 10032 of *LNCS*, pages 777–804, Hanoi, Vietnam, Dec. 4–8, 2016. Springer, Heidelberg.

[4] M. Bellare, J. A. Garay, and T. Rabin. Batch Verification with Applications to Cryptography and Checking. In C. L. Lucchesi and A. V. Moura, editors, *LATIN 1998*, volume 1380 of *LNCS*, pages 170–191, Campinas, Brazil, Apr. 20–24, 1998. Springer, Heidelberg.

[5] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *IEEE SP 2014*, pages 459–474, Berkeley, CA, USA, May 18-21, 2014. IEEE Computer Society.

[6] E. Ben-Sasson, A. Chiesa, M. Green, E. Tromer, and M. Virza. Secure Sampling of Public Parameters for Succinct Zero Knowledge Proofs. In *IEEE SP 2015*, pages 287–304, San Jose, CA, USA, May 17-21, 2015. IEEE Computer Society.

[7] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Scalable Zero Knowledge via Cycles of Elliptic Curves. In J. A. Garay and R. Gennaro, editors, *CRYPTO (2) 2014*, volume 8617 of *LNCS*, pages 276–294, Santa Barbara, California, USA, Aug. 17–21, 2014. Springer, Heidelberg.

[8] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. In K. Fu and J. Jung, editors, *USENIX 2014*, pages 781–796, San Jose, CA, USA, Aug. 20–22, 2014. USENIX Association.

[9] N. Bitansky, R. Canetti, O. Paneth, and A. Rosen. On the Existence of Extractable One-Way Functions. In D. Shmoys, editor, *STOC 2014*, pages 505–514, New York, NY, USA, May 31 – Jun 3, 2014. ACM Press.

[10] M. Blum, P. Feldman, and S. Micali. Non-Interactive Zero-Knowledge and Its Applications. In *STOC 1988*, pages 103–112, Chicago, Illinois, USA, May 2–4, 1988. ACM Press.

[11] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg.

[12] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[13] C. Costello, C. Fournet, J. Howell, M. Kohlweiss, B. Kreuter, M. Naehrig, B. Parno, and S. Zahur. Geppetto: Versatile Verifiable Computation. In *IEEE SP 2015*, pages 253–270, San Jose, CA, USA, May 17-21, 2015. IEEE Computer Society.

[14] I. Damgård. Towards Practical Public Key Systems Secure against Chosen Ciphertext Attacks. In J. Feigenbaum, editor, *CRYPTO 1991*, volume 576 of *LNCS*, pages 445–456, Santa Barbara, California, USA, Aug. 11–15, 1991. Springer, Heidelberg, 1992.

[15] G. Danezis, C. Fournet, J. Groth, and M. Kohlweiss. Square Span Programs with Applications to Succinct NIZK Arguments. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014 (1)*, volume 8873 of *LNCS*, pages 532–550, Kaohsiung, Taiwan, R.O.C., Dec. 7–11, 2014. Springer, Heidelberg.

[16] A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. L. Villar. An Algebraic Framework for Diffie-Hellman Assumptions. In R. Canetti and J. Garay, editors, *CRYPTO (2) 2013*, volume 8043 of *LNCS*, pages 129–147, Santa Barbara, California, USA, Aug. 18–22, 2013. Springer, Heidelberg.

[17] P. Fauzi and H. Lipmaa. Efficient Culpably Sound NIZK Shuffle Argument without Random Oracles. In K. Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 200–216, San Franscisco, CA, USA, February 29–March 4, 2016. Springer, Heildeberg.

[18] P. Fauzi, H. Lipmaa, J. Siim, and M. Zając. An Efficient Pairing-Based Shuffle Argument. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017 (2)*, volume 10625 of *LNCS*, pages 98–127, Hong Kong, China, Dec. 3–7, 2017. Springer, Heidelberg.

[19] P. Fauzi, H. Lipmaa, and M. Zając. A Shuffle Argument Secure in the Generic Model. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016 (2)*, volume 10032 of *LNCS*, pages 841–872, Hanoi, Vietnam, Dec. 4–8, 2016. Springer, Heidelberg.

[20] J. Furukawa and K. Sako. An Efficient Scheme for Proving a Shuffle. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 368–387, Santa Barbara, USA, Aug. 19–23, 2001. Springer, Heidelberg.

[21] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for Cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[22] R. Gennaro, C. Gentry, and B. Parno. Non-Interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482, Santa Barbara, California, USA, Aug. 15–19, 2010. Springer, Heidelberg.

[23] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic Span Programs and NIZKs without PCPs. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645, Athens, Greece, Apr. 26–30, 2013. Springer, Heidelberg.

[24] C. Gentry and D. Wichs. Separating Succinct Non-Interactive Arguments from All Falsifiable Assumptions. In S. Vadhan, editor, *STOC 2011*, pages 99–108, San Jose, California, USA, June 6–8, 2011. ACM Press.

[25] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof-Systems. In R. Sedgewick, editor, *STOC 1985*, pages 291–304, Providence, Rhode Island, USA, May 6–8, 1985. ACM Press.

[26] P. Golle, S. Jarecki, and I. Mironov. Cryptographic Primitives Enforcing Communication and Storage Complexity. In M. Blaze, editor, *FC 2002*, volume 2357 of *LNCS*, pages 120–135, Southhampton Beach, Bermuda, Mar. 11–14, 2002. Springer, Heidelberg.

[27] A. González and C. Ràfols. New Techniques for Non-interactive Shuffle and Range Arguments. In M. Manulis, A.-R. Sadeghi, and S. Schneider, editors, *ACNS 2016*, volume 9696 of *LNCS*, pages 427–444, Guildford, UK, June 19–22, 2016. Springer, Heidelberg.

[28] J. Groth. Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In X. Lai and K. Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459, Shanghai, China, Dec. 3–7, 2006. Springer, Heidelberg.

[29] J. Groth. A Verifiable Secret Shuffle of Homomorphic Encryptions. *J. Cryptology*, 23(4):546–579, 2010.

[30] J. Groth. Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340, Singapore, Dec. 5–9, 2010. Springer, Heidelberg.

[31] J. Groth. On the Size of Pairing-based Non-interactive Arguments. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 305–326, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg.

[32] J. Groth and S. Lu. A Non-interactive Shuffle with Pairing Based Verifiability. In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 51–67, Kuching, Malaysia, Dec. 2–6, 2007. Springer, Heidelberg.

[33] J. Groth, R. Ostrovsky, and A. Sahai. New Techniques for Noninteractive Zero-Knowledge. *Journal of the ACM*, 59(3), 2012.

[34] C. S. Jutla and A. Roy. Shorter Quasi-Adaptive NIZK Proofs for Linear Subspaces. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013 (1)*, volume 8269 of *LNCS*, pages 1–20, Bangalore, India, Dec. 1–5, 2013. Springer, Heidelberg.

[35] C. S. Jutla and A. Roy. Switching Lemma for Bilinear Tests and Constant-Size NIZK Proofs for Linear Subspaces. In J. A. Garay and R. Gennaro, editors, *CRYPTO (2) 2014*, volume 8617 of *LNCS*, pages 295–312, Santa Barbara, California, USA, Aug. 17–21, 2014. Springer, Heidelberg.

[36] E. Kiltz and H. Wee. Quasi-Adaptive NIZK for Linear Subspaces Revisited. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015*, volume 9057 of *LNCS*, pages 101–128, Sofia, Bulgaria, May 26–30, 2015. Springer, Heidelberg.

[37] H. Lipmaa. Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments. In R. Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189, Taormina, Italy, Mar. 18–21, 2012. Springer, Heidelberg.

[38] H. Lipmaa. Succinct Non-Interactive Zero Knowledge Arguments from Span Programs and Linear Error-Correcting Codes. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013 (1)*, volume 8269 of *LNCS*, pages 41–60, Bangalore, India, Dec. 1–5, 2013. Springer, Heidelberg.

[39] H. Lipmaa and B. Zhang. A More Efficient Computationally Sound Non-Interactive Zero-Knowledge Shuffle Argument. In I. Visconti and R. D. Prisco, editors, *SCN 2012*, volume 7485 of *LNCS*, pages 477–502, Amalfi, Italy, September 5–7, 2012. Springer, Heidelberg.

[40] U. M. Maurer. Abstract Models of Computation in Cryptography. In N. P. Smart, editor, *Cryptography and Coding 2005*, pages 1–12, Cirencester, UK, Dec. 19–21, 2005.

[41] P. Morillo, C. Ràfols, and J. L. Villar. The Kernel Matrix Diffie-Hellman Assumption. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016 (1)*, volume 10032 of *LNCS*, pages 729–758, Hanoi, Vietnam, Dec. 4–8, 2016. Springer, Heidelberg.

[42] V. I. Nechaev. Complexity of a Determinate Algorithm for the Discrete Logarithm. *Mathematical Notes*, 55(2):165–172, 1994. Translated from *Matematicheskie Zapiski*, 55(2):91–101, 1994.

[43] B. Parno, C. Gentry, J. Howell, and M. Raykova. Pinocchio: Nearly Practical Verifiable Computation. In *IEEE SP 2013*, pages 238–252, Berkeley, CA, USA, May 19-22, 2013. IEEE Computer Society.

[44] V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In W. Fumy, editor, *EUROCRYPT 1997*, volume 1233 of *LNCS*, pages 256–266, Konstanz, Germany, 11–15 May 1997. Springer, Heidelberg.

3. Secure parameter generation

3.1 Relevance to the PANORAMIX project

After the Snowden revelations, there has been a recent surge of interest in constructing cryptographic primitives and protocols secure against active subversion. In the context of PANORAMIX objectives, it seems crucial to provide a secure parameter generation for the protocols developed during the project. Especially taking into account that the proposed zero-knowledge arguments were shown secure in the CRS [10] model, that relies on the honest generation of the CRS.

In [3], Bellare, Fuchsbauer and Scafuro tackled the problem by studying how much security one can still achieve when the CRS generator cannot be trusted. They proved several negative and positive results. In particular, they showed that it is impossible to achieve subversion soundness and (even non-subversion) zero knowledge simultaneously, the essential reason being that the zero knowledge simulator can be used to break subversion soundness.

In one of their positive solutions, Bellare *et al.* show that it is possible to get (non-subversion) soundness and computational subversion zero knowledge (Sub-ZK, ZK even if the the CRS is not trusted). Their main new idea is to use a knowledge assumption in the Sub-ZK proof, so that the simulator can extract a "trapdoor" from the untrusted CRS and then use this trapdoor to simulate the argument. While neat, the resulting argument system is quite complicated.

Abdolmaleki et al. [1] (see the attached article) shown an efficient way to transform the most efficient, for the time being, zero-knowledge succinct non-interactive argument for QAP [31]. To that end they proposed a number of algorithms that assures that the CRS is correctly generated and that there is an extractable trapdoor that can be provided to a simulator. Although their result does not show implicite how to generate a subversion-resistant shuffle argument, their work shows general design recommendations that should make designing of the new, resistant, shuffle argument much simpler.

3.2 Preliminaries

3.2.1 Quadratic Arithmetic Programs.

Quadratic Arithmetic Program (QAP) was introduced by Gennaro *et al.* [23] as a language where for an input x and witness w, $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$ can be verified by using a parallel quadratic check, and that has an efficient reduction from the well-known language (either Boolean or Arithmetic) Circuit-SAT. Hence, an efficient zk-SNARK for QAP results in an efficient zk-SNARK for Circuit-SAT.

For an m-dimensional vector $\vec{A}$, let $\mathsf{aug}(\vec{A}) = \left(\begin{smallmatrix} 1 \\ \vec{A} \end{smallmatrix}\right)$. For an n-dimensional vector $\vec{M}^{(0)}$ and an $n \times m$ matrix M over finite field $\mathbb{F}$, let $\mathsf{aug}(M) := (\vec{M}^{(0)}, M)$. Let $m_0 < m$ be a non-negative integer. An instance $\mathcal{Q}$ of the QAP language is specified by $(\mathbb{F}, m_0, \mathsf{aug}(U), \mathsf{aug}(V), \mathsf{aug}(W))$ where $U, V, W \in \mathbb{F}^{n \times m}$.

In the case of Arithmetic Circuit-SAT, n is the number of multiplication gates and m to the number of wires in the circuit. Here, we consider arithmetic circuits that consist only of fan-in-2 multiplication gates, but either input of each multiplication gate can be a weighted sum of some wire values, [23].

For a fixed instance $\mathcal{Q}$ of QAP, define the relation $\mathcal{R}$ as follows:

$$\mathcal{R}_{\mathcal{Q}} = \left\{ \begin{array}{l} (\mathsf{x}, \mathsf{w}) \colon \mathsf{x} = (A_1, \ldots, A_{m_0})^\top \wedge \mathsf{w} = (A_{m_0+1}, \ldots, A_m)^\top \wedge \\ (\mathsf{aug}(U) \cdot \mathsf{aug}(\vec{A})) \circ (\mathsf{aug}(V) \cdot \mathsf{aug}(\vec{A})) = \mathsf{aug}(W) \cdot \mathsf{aug}(\vec{A}) \end{array} \right\}$$

where $\vec{a} \circ \vec{b} = (a_i b_i)_{i=1}^n$ denotes the entrywise product of vectors $\vec{a}$ and $\vec{b}$.

In a cryptographic setting, it is more convenient to work with the following alternative definition of QAP and of the relation $\mathcal{R}_{\mathcal{Q}}$. (This corresponds to the original definition of QAP in [23].) Let $\mathbb{F} = \mathbb{Z}_p$, such that ω is the n-th primitive root of unity modulo p. Let $\mathcal{Q} = (\mathbb{Z}_p, m_0, \mathsf{aug}(U), \mathsf{aug}(V), \mathsf{aug}(W))$ be a QAP instance. For $j \in [0\,..\,m]$, define $u_j(X) := L_{\vec{U}^{(j)}}(X)$, $v_j(X) := L_{\vec{V}^{(j)}}(X)$, and $w_j(X) := L_{\vec{W}^{(j)}}(X)$. Thus, $u_j, v_j, w_j \in \mathbb{Z}_p^{(\leq n-1)}[X]$.

An QAP instance $\mathcal{Q}_p$ is specified by the so defined $(\mathbb{Z}_p, m_0, \{u_j, v_j, w_j\}_{j=0}^m)$. This instance defines the following relation, where we assume that $A_0 = 1$:

$$\mathcal{R}_{\mathcal{Q}_p} = \left\{ \begin{array}{l} (\mathsf{x}, \mathsf{w}) \colon \mathsf{x} = (A_1, \ldots, A_{m_0})^\top \wedge \mathsf{w} = (A_{m_0+1}, \ldots, A_m)^\top \wedge \\ \left(\sum_{j=0}^m A_j u_j(X)\right) \left(\sum_{j=0}^m A_j v_j(X)\right) \equiv \sum_{j=0}^m A_j w_j(X) \pmod{\ell(X)} \end{array} \right\}$$

Alternatively, $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$ if there exists a (degree $\leq n-2$) polynomial $h(X)$, s.t.

$$\left(\sum_{j=0}^m A_j u_j(X)\right) \left(\sum_{j=0}^m A_j v_j(X)\right) - \sum_{j=0}^m A_j w_j(X) = h(X)\ell(X) \ .$$

Clearly, $\mathcal{R}_{\mathcal{Q}} = \mathcal{R}_{\mathcal{Q}_p}$, given that $\mathcal{Q}_p$ is constructed from $\mathcal{Q}$ as above.

3.2.2 Definitions: SNARKs and Subversion Zero Knowledge

Let $\mathcal{L}_{\mathcal{R}} = \{\mathsf{x} : \exists \mathsf{w}, (\mathsf{x}, \mathsf{w}) \in \mathcal{R}\}$ be an **NP**-language. A (subversion-resistant) *non-interactive zero-knowledge argument system* Ψ for $\mathcal{R}$ consists of seven PPT algorithms: CRS trapdoor generator, simulation trapdoor generator, CRS generator, prover, verifier and simulator along with an additional algorithm that verifies the correctness of the CRS:

CRS verifier: CV is a probabilistic algorithm that, given $(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{crs})$, returns either 0 (the CRS is incorrectly formed) or 1 (the CRS is correctly formed),

We also define the (non-subverted) CRS generation algorithm $\mathsf{K}(\mathcal{R}, \mathsf{z}_{\mathcal{R}})$ that first sets $\mathsf{tc} \leftarrow \mathsf{K}_{\mathsf{tc}}(\mathcal{R}, \mathsf{z}_{\mathcal{R}})$ and then outputs $(\mathsf{crs} \,\|\, \mathsf{ts}) \leftarrow (\mathsf{K}_{\mathsf{crs}} \,\|\, \mathsf{K}_{\mathsf{ts}})(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{tc})$.

SNARKs.

A non-interactive argument system is *succinct* if the proof size is polynomial in λ and the verifier runs in time polynomial in $\lambda + |\mathsf{x}|$. A succinct non-interactive argument of knowledge is usually called *SNARK*. A zero knowledge SNARK is abbreviated to *zk-SNARK*.

Security definitions

A Sub-ZK SNARK has to satisfy various security definitions. The most important ones are *subversion-completeness* (an honest prover convinces an honest verifier, and an honestly generated CRS passes the CRS verification test), *computational knowledge-soundness* (if a prover convinces an

honest verifier, then he knows the corresponding witness), and *statistical Sub-ZK* (given a possibly subverted CRS, an argument created by the honest prover reveals no side information). Next, we will give definitions of those properties that guarantee both composability and subversion resistance.

Definition 1 (Perfect Subversion-Completeness)**.** *A non-interactive argument* Ψ *is* perfectly subversion-complete for $\mathcal{R}$*, if for all* λ*, all* $(\mathcal{R}, \mathsf{z}_\mathcal{R}) \in \text{range}(\mathcal{R}(1^\lambda))$*,* $\mathsf{tc} \in \text{range}(\mathsf{K}_\mathsf{tc}(\mathcal{R}, \mathsf{z}_\mathcal{R})) \setminus \{\bot\}$*, and* $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$*,*

$$\Pr\left[\begin{array}{l}\mathsf{crs} \leftarrow \mathsf{K}_\mathsf{crs}(\mathcal{R}, \mathsf{z}_\mathcal{R}, \mathsf{tc}) : \mathsf{CV}(\mathcal{R}, \mathsf{z}_\mathcal{R}, \mathsf{crs}) = 1 \wedge \\ \mathsf{V}(\mathcal{R}, \mathsf{z}_\mathcal{R}, \mathsf{crs}_\mathsf{V}, \mathsf{x}, \mathsf{P}(\mathcal{R}, \mathsf{z}_\mathcal{R}, \mathsf{crs}_\mathsf{P}, \mathsf{x}, \mathsf{w})) = 1\end{array}\right] = 1 \ .$$

The following definition of unbounded Sub-ZK differs from the standard definitions as follows. Since we allow the CRS to be subverted, the CRS is generated by a subverter who also returns z_Σ. The adversary's access to z_Σ models the possibility that the subverter and the adversary collaborate. The extractor X_Σ extracts tc from Σ, and then tc is used to generate the simulation trapdoor ts that is then given as an auxiliary input to the adversary and to the oracle O_1.

Definition 2 (Statistically Unbounded Sub-ZK)**.** Ψ *is* statistically unbounded Sub-ZK for $\mathcal{R}$*, if for any NUPPT subverter* Σ *there exists a NUPPT* X_Σ*, such that for all* λ*, all* $(\mathcal{R}, \mathsf{z}_\mathcal{R}) \in \text{range}(\mathcal{R}(1^\lambda))$*, and all computationally unbounded* $\mathcal{A}$*,* $\varepsilon_0^{unb} \approx_\lambda \varepsilon_1^{unb}$*, where* ε_b^{unb} *is defined as*

$$\Pr\left[\begin{array}{l}r \leftarrow_r \mathsf{RND}(\Sigma), (\mathsf{crs}, \mathsf{z}_\Sigma \,\|\, \mathsf{tc}) \leftarrow (\Sigma \,\|\, \mathsf{X}_\Sigma)(\mathcal{R}, \mathsf{z}_\mathcal{R}; r), \\ \mathsf{ts} \leftarrow \mathsf{K}_\mathsf{ts}(\mathcal{R}, \mathsf{z}_\mathcal{R}, \mathsf{tc}) \ : \ \mathsf{CV}(\mathcal{R}, \mathsf{z}_\mathcal{R}, \mathsf{crs}) = 1 \wedge \mathcal{A}^{\mathsf{O}_b(\cdot,\cdot)}(\mathcal{R}, \mathsf{z}_\mathcal{R}, \mathsf{crs}, \mathsf{ts}, \mathsf{z}_\Sigma) = 1\end{array}\right] \ .$$

Here, the oracle $\mathsf{O}_0(\mathsf{x}, \mathsf{w})$ *returns* $\bot$ *(reject) if* $(\mathsf{x}, \mathsf{w}) \notin \mathcal{R}$*, and otherwise it returns* $\mathsf{P}(\mathcal{R}, \mathsf{z}_\mathcal{R}, \mathsf{crs}_\mathsf{P}, \mathsf{x}, \mathsf{w})$*. Similarly,* $\mathsf{O}_1(\mathsf{x}, \mathsf{w})$ *returns* $\bot$ *(reject) if* $(\mathsf{x}, \mathsf{w}) \notin \mathcal{R}$*, and otherwise it returns* $\mathsf{Sim}(\mathcal{R}, \mathsf{z}_\mathcal{R}, \mathsf{crs}, \mathsf{ts}, \mathsf{x})$*.* Ψ *is* perfectly unbounded Sub-ZK for $\mathcal{R}$ *if one requires that* $\varepsilon_0^{unb} = \varepsilon_1^{unb}$*.*

3.2.3 Sub-GBGM and BDH-KE assumption

In previous sections, we introduced the GBG model, here we make it weaker, by allowing adversary to learn group elements without knowing their discrete logarithms. However, we assume that it is not known how to create four elements $[1]_z$, $[a]_z$, $[b]_z$, and $[ab]_z$ without knowing either a or b. The corresponding assumption — that may also be true in the case of symmetric pairings — was named *DH-KE(A)* in [3].

Asymmetric pairings are much more efficient than symmetric pairings. If we work in the type III pairing setting where there is no efficient isomorphism either from $\mathbb{G}_1$ to $\mathbb{G}_2$ or from $\mathbb{G}_2$ to $\mathbb{G}_1$, then clearly an adversary cannot, given $[a]_z$ for $z \in \{1, 2\}$ and an unknown a, compute $[a]_{3-z}$. In the same vein, it seems reasonable to make a stronger assumption (that we call *BDH-KE*, a simplification of the asymmetric PKE assumption of [15]) that if an adversary creates $[a]_1$ and $[a]_2$ then she knows a. Really, since there is no polynomial-time isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ (or back), it seems to be natural to assume that one does not have to worry about an adversary knowing some trapdoor that would break the BDH-KE assumption. Since BDH-KE is not a falsifiable assumption, this does not obviously mean that it must hold for each type III pairing. Instead, the BDH-KE assumption can be interpreted as a *stronger* definition of the type III pairing setting. We formalize the added adversarial power as follows.

We give the generic model adversary an additional power to effectively create new indeterminates Y_i in groups $\mathbb{G}_1$ and $\mathbb{G}_2$ (e.g., by hashing into elliptic curves), without knowing their values. We note since $[Y]_1 \bullet [1]_2 = [Y]_T$ and $[1]_1 \bullet [Y]_2 = [Y]_T$, the adversary that has generated an indeterminate Y in $\mathbb{G}_z$ can also operate with Y in $\mathbb{G}_T$. Formally, this means that Π will contain one more operation create, with the following semantics:

4. On input (create, i, t): if $\mathsf{type}_i = \bot$ and $t \in \{1, 2, T\}$ then set $\mathsf{cell}_i \leftarrow \mathbb{Z}_p$ and $\mathsf{type}_i \leftarrow t$.

The semantics of create dictates that the actual value of the indeterminate Y_i is uniformly random in $\mathbb{Z}_p$, that is, the adversary cannot create indeterminates for which she does not know the discrete logarithm and that yet are not random. This assumption is needed for the lower bound on the generic adversary's time to be provable in Thm. 5.

In the type III setting, this semantics does *not* allow the adversary to create the same indeterminate Y_i in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$; she can only create a representation of a known to her integer in both groups. We formalize this by making the following Bilinear Diffie-Hellman Knowledge of Exponents (*BDH-KE*) assumption: if the adversary, given random generators $\mathfrak{g}_1 = [1]_1 \in \mathbb{G}_1$ and $\mathfrak{g}_2 = [1]_2 \in \mathbb{G}_2$, can generate elements $[\alpha_1]_1 \in \mathbb{G}_1$ and $[\alpha_2]_2 \in \mathbb{G}_2$, such that $[1]_1 \bullet [\alpha_2]_2 = [\alpha_1]_1 \bullet [1]_2$, then the adversary knows the value $\alpha_1 = \alpha_2$. To simplify the further use of the BDH-KE assumption in security reductions, we give the adversary access to $(\mathcal{R}, \mathsf{z}_\mathcal{R}) \in \mathrm{range}(\mathcal{R}(1^\lambda))$. As before, $\mathsf{z}_\mathcal{R}$ just contains gk, that is, the description of the bilinear group together with $[1]_1$ and $[1]_2$.

Definition 3 (BDH-KE)**.** *We say that* genbp *is* BDH-KE secure *for $\mathcal{R}$ if for any λ, $(\mathcal{R}, \mathsf{z}_\mathcal{R}) \in \mathrm{range}(\mathcal{R}(1^\lambda))$, and NUPPT adversary $\mathcal{A}$ there exists a NUPPT extractor $\mathsf{X}_\mathcal{A}$, such that*

$$\Pr\left[\begin{array}{l} r \leftarrow_r \mathsf{RND}(\mathcal{A}), ([\alpha_1]_1, [\alpha_2]_2 \parallel a) \leftarrow (\mathcal{A} \parallel \mathsf{X}_\mathcal{A})(\mathcal{R}, \mathsf{z}_\mathcal{R}; r) : \\ [\alpha_1]_1 \bullet [1]_2 = [1]_1 \bullet [\alpha_2]_2 \wedge a \neq \alpha_1 \end{array}\right] \approx_\lambda 0 \ .$$

The BDH-KE assumption is a simple special case of the PKE assumption as used in the case of asymmetric pairings say in [15]. In the PKE assumption of [15], adversary is given as an input the tuple $\{([\chi^i]_1, [\chi^i]_2)\}_{i=0}^n$ for some $n \geq 0$, and it is assumed that if an adversary outputs $([\alpha]_1, [\alpha]_2)$ then she knows $(a_0, a_1, \ldots, a_n)$, such that $\alpha = \sum_{i=0}^n a_i \chi^i$. In our case, $n = 0$. BDH-KE can also be seen as an asymmetric-pairing version of the original KE assumption [14].

We think that for the following reasons, the BDH-KE assumption is more natural than the DH-KE assumption by Bellare *et al.* [3] which states that if the adversary can create elements $[\alpha_1]_z$, $[\alpha_2]_z$ and $[\alpha_1\alpha_2]_z$ of the group $\mathbb{G}_z$ then she knows either α_1 or α_2.

First, the BDH-KE assumption is well suited to type-III pairings that are by far the most efficient pairings. The DH-KE assumption is tailored to type-I pairings. In the case of type-III pairings, DH-KE assumption can still be used, but it results in inefficient protocols. For example in [3], in security proofs the authors employs an adversary that extracts either α_1 or α_2. Since it is not known a priori which value will be extracted, several elements in the argument system have to be doubled, for the case α_1 is extracted and for the case α_2 is extracted.

Second, most of the efficient SNARKs are constructed to be sound and zero-knowledge in the (most efficient) type-III setting. While the SNARK of Groth [31] is known to be sound in the case of both symmetric and asymmetric pairings, in the case of symmetric pairings it will be much less efficient. To take the advantage of already known efficient SNARKs, it is only natural to keep the type-III setting. In the current paper, we are able to have the best of both worlds. As in the case of [31], we construct a SNARK that uses type-III pairings. On the one hand, we prove it to be Sub-ZK solely under the BDH-KE assumption. On the other hand, we prove that it is (adaptively) knowledge-sound in the Sub-GBGM, independently of whether one uses type-I, type-II, or type-III pairings. This provides a partial hedge against cryptanalysis: even if one were to later find an efficient isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$, this would only break the Sub-ZK of the new SNARK but leave the soundness property intact.

3.3 A Subversion-Resistant SNARK

Combined effort of a large number of recent research papers (to only mention a few, [30, 37, 23, 43, 38, 15, 31]) has made it possible to construct very efficient succinct non-interactive zero-knowledge arguments of knowledge (zk-SNARKs) for both the Boolean and the Arithmetic Circuit-SAT and thus for NP. The most efficient known approach for constructing zk-SNARKs for the Arithmetic Circuit-SAT is based on Quadratic Arithmetic Programs (QAP, [23]).

In a QAP, the prover builds a set of polynomial equations that are then checked by the verifier by using a small number of pairings. QAP-based zk-SNARKs have additional nice properties that make them applicable in verifiable computation [22, 23, 43] where the client outsources some computation to the server, who returns the computation result together with a succinct efficiently-verifiable correctness argument. Especially due to this application, zk-SNARKs have several heavily optimized implementations [43, 7, 8, 13]. Other applications of zk-SNARK include cryptocurrencies [5]. See, e.g., [31] for more references.

One drawback of zk-SNARKs is that they are all based on non-falsifiable assumptions (like the knowledge assumptions [14], or the generic bilinear group model, GBGM [42, 44, 40, 11]). In fact, Gentry and Wichs [24] showed that non-falsifiable assumptions are needed to construct zk-SNARKs for non-trivial languages. The currently most efficient zk-SNARK for Arithmetic Circuit-SAT was proposed by Groth (EUROCRYPT 2016, [31]) who proved it to be knowledge-sound in the GBGM. In Groth's zk-SNARK, the argument consists of only 3 bilinear group elements and the verifier has to check a single pairing equation, dominated by the computation of only 3 bilinear (type III [21]) pairings and m_0 exponentiations, where m_0 is the statement size.

3.3.1 Contributions of [1]

Abdolmaleki et al. [1] takes Groth's zk-SNARK from EUROCRYPT 2016 [31] as a starting point since, since it is currently the most efficient and thus the most attractive zk-SNARK. [1] They propose a minimal modification to Groth's zk-SNARK that makes it computationally knowledge-sound in what the authors call the "subversion generic bilinear group model" (Sub-GBGM) and perfect composable Sub-ZK.

They change Groth's zk-SNARK by adding extra elements to the CRS so that the CRS will become publicly verifiable; this minimal step (clearly, some public verifiability of the CRS is needed in the case the CRS generator cannot be trusted) will be sufficient to obtain Sub-ZK. However, choosing which elements to add to the CRS is not straightforward since the zk-SNARK must remain knowledge-sound even given enlarged CRS; adding too many or just "wrong" elements to the CRS can break the knowledge-soundness.

On the other hand, importantly, the prover and the verifier of the new zk-SNARK are unchanged compared to Groth's SNARK [31]. In the rest of this presentation, we will outline the novel properties of the new SNARK as compared to [31].

Abdolmaleki et al. start by defining perfect subversion-complete (this includes the requirement that an honestly generated CRS is accepted by the CRS verification), computationally adaptively knowledge-sound, and statistically unbounded (or composable) Sub-ZK SNARKs. These definitions are similar to but subtly different from the non-subversion security definitions as given in, say, [28]. First, since one cannot check whether the subverter uses perfectly uniform coins (or, the *CRS trapdoor*) to generate the CRS, they divide the CRS generation into three different algorithms:

[1] While less efficient zk-SNARKs like Pinocchio [43] are used more widely, this situation might change. Moreover, Groth's zk-SNARK is also more efficient from the theoretical perspective: it has both a more complex CRS and more sophisticated soundness proof; hence, we expect that it is easier to achieve Sub-ZK for Pinocchio.

- generation of the CRS trapdoor tc (a probabilistic algorithm K_{tc}),
- creation of the CRS from tc (a deterministic algorithm $\mathsf{K}_{\mathsf{crs}}$), and
- creation of the simulation trapdoor from tc (a deterministic algorithm K_{ts}).

Since one cannot check that K_{tc} works correctly, they guarantee that given a fixed tc, $\mathsf{K}_{\mathsf{crs}}$ has been executed on this tc. More precisely, they require that a Sub-ZK SNARK satisfies a CRS trapdoor extractability property that allows one to extract tc used by the subverter, s.t. if the subverted CRS crs is accepted by the CRS verification algorithm (see below) then K_{tc} maps tc to crs.

The extractability requirement forces a ZK proof to use either a computationally unbounded extractor or a knowledge assumption. Sub-ZK is defined with respect to an efficient subverter and extractor.

In the proof of knowledge-soundness, the authors use (a version of) the GBGM. Using GBGM seems to be well motivates since Groth's non-Sub zk-SNARK is proven knowledge-sound in GBGM and as mentioned above, the use of a knowledge assumption or the generic model in the knowledge-soundness proof is necessary due to the impossibility result of Gentry and Wichs [24]. However, following Bellare *et al.* [3], the authors weaken the usual definition of GBGM by allowing the generic adversary to create (under realistic restrictions) random elements in the source groups without knowing their discrete logarithms. They call the resulting somewhat weaker model the *subversion generic bilinear group model* (Sub-GBGM). Following Groth [31], the authors prove that the new SNARK is (adaptively) knowledge-sound in the Sub-GBGM even in the case of type-I pairings. This provides a hedge against possible future cryptanalysis that finds an efficient isomorphism between the two source groups.

In the proof of perfect composable Sub-ZK, the authors use a well-known knowledge assumption (see, e.g., [15]) that they call BDH-KE. The Sub-ZK proof of the only previously known non-interactive Sub-ZK argument system by Bellare *et al.* [3] also relies on knowledge assumptions. The authors follow the main idea of [3] by first using BDH-KE to extract the CRS trapdoor tc from the CRS and then construct a non-subversion simulator (that gets a part of the tc as an input) to simulate the argument. However, since they construct a zk-SNARK, the concrete approach is different from [3].

Also here, they rely on the existence of the efficient CRS verification algorithm CV. The authors show that if CV accepts a crs, then crs has been computed correctly by $\mathsf{K}_{\mathsf{crs}}$ from a tc bijectively fixed by crs. From this, it follows under the BDH-KE assumption that for any subverter that produces a crs accepted by CV, there exists an extractor that produces tc such that $\mathsf{K}_{\mathsf{crs}}$ given tc outputs crs.

What should be emphasized is fact that security proofs of knowledge-soundness and of Sub-ZK are in incomparable models. The knowledge-soundness proof uses the full power of Sub-GBGM in the case of any pairings (including type-I). The Sub-ZK proof, on the other hand, uses a concrete standard-looking knowledge assumption BDH-KE that holds in the the GBGM but does not hold in the Sub-GBGM in the case of type-I pairings. This allows to construct an efficient Sub-ZK SNARK that uses type-III pairings, while guaranteeing its knowledge-soundness even in the case of type-I pairings.

3.3.2 General Design Recommendations.

It would be unexpected if constructing Sub-ZK SNARKs could be done automatically. In particular since the framework of [1] points to the necessity of making CRS publicly verifiable which potentially means adding new elements to the CRS. Since knowledge-soundness proofs of many SNARKs are very subtle, it seems to be difficult to give a general "theorem" about which SNARKs can be modified to be Sub-ZK or even whether their CRS can be made verifiable without a major reconstruction. Whether a SNARK remains sound after that must be proven separately in each case.

However, the authors give a few recommendations for designing a Sub-ZK SNARK from a non subversion-secure SNARK (or from scratch) when using the same approach as in [1]:

1. **Division of duties:** make sure that K can be divided into randomized K_{tc}, deterministic K_{ts}, and deterministic $\mathsf{K}_{\mathsf{crs}}$.
2. **CRS trapdoor extractability:** for each element of tc, make sure that it can be extracted from the CRS. For this, one can use a generic proof of knowledge, a specific knowledge-assumption, or a computationally unbounded extractor.
3. **CRS verifiability:** the CRS must be publicly verifiable.
4. **Sound approach:** make sure that the previous steps do not hurt the knowledge-soundness. To achieve it, one should aim at designing a SNARK with a very simple CRS or where CRS verifiability comes naturally. Depending on the SNARK in question, this step may be the most difficult one.

3.3.3 On Efficiency.

Since the new zk-SNARK is closely based on the most efficient known non-subversion zk-SNARK of Groth [31], it has comparable efficiency. Importantly, the new CRS verification algorithm CV has to be executed only by the prover (this is since the authors achieve Sub-ZK and non-subversion knowledge-soundness). This means that it suffices for CV to have the same computational complexity as the prover's algorithm. The initial CV we describe in Fig. 3.1 is quite inefficient. However, as it was shown (see the full version [1] for more information), by using batching techniques the CV algorithm can be sped up to be faster than the prover's algorithm (at the information-theoretical security level 2^{-80}) and even faster at the information-theoretical security level 2^{-40}.

3.4 Construction

See Fig. 3.1 and Fig. 3.2 for detained description of the new SNARK. This SNARK uses crucially several random variables, $\chi, \alpha, \beta, \gamma, \delta$. As in [31], α and β (and the inclusion of $\alpha\beta$ in the verification equation) will guarantee that $\mathfrak{A}$, $\mathfrak{b}$, and $\mathfrak{c}$ are computed by using the same coefficients A_i. The role of γ and δ is to make the three products in the verification equation "independent" of each other.

As emphasized before, the new Sub-ZK SNARK is closely based on Groth's zk-SNARK. In fact, the differences between the construction of the two SNARKs can be summarized very briefly:

(i) [1] adds to the CRS $2n + 3$ new elements (see the variable $\mathsf{crs}_{\mathsf{CV}}$ in Fig. 3.1) that are needed for CV to work efficiently.

(ii) They divide the CRS generation algorithm into three algorithms, K_{tc}, K_{ts}, and $\mathsf{K}_{\mathsf{crs}}$. Groth's CRS generation algorithm returns $\mathsf{K}_{\mathsf{crs}}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{K}_{\mathsf{tc}}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}))$ (minus the mentioned $\mathsf{crs}_{\mathsf{CV}}$ part) as the CRS and $\mathsf{K}_{\mathsf{ts}}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{K}_{\mathsf{tc}}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}))$ as the simulation trapdoor.

(iii) They describe an efficient CRS verification algorithm CV (see Fig. 3.1).

It is straightforward to see that Groth's original zk-SNARK does not achieve Sub-ZK. Really, since neither $[\ell_i(\chi)]_1$ nor $\left[\chi^i\right]_1$ are given to the prover, he cannot check the correctness of $[u_j(\chi)]_1$ and $[v_j(\chi)]_1$. This means that a subverter can change those values to some bogus values and due to that, the proof computed by an honest prover and a simulated proof (that relies on the knowledge of trapdoor elements α and β and does not use the CRS elements $[u_j(\chi)]_1$ and $[v_j(\chi)]_1$) will have different distributions.

3.4.1 Subversion Completeness, Soundness and Subversion Zero Knowledge

Security of the new SNARK has been formalized in the following theorems:

$\mathsf{K}_{\mathsf{tc}}(\mathcal{R}, \mathsf{z}_{\mathcal{R}})$: Generate $\mathsf{tc} = (\chi, \alpha, \beta, \gamma, \delta) \leftarrow_r \mathbb{Z}_p^3 \times (\mathbb{Z}_p^*)^2$.
$\mathsf{K}_{\mathsf{ts}}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{tc})$: Set $\mathsf{ts} \leftarrow (\chi, \alpha, \beta, \delta)$.
$\mathsf{K}_{\mathsf{crs}}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{tc})$: Compute $(\ell_i(\chi))_{i=1}^n$ Set $u_j(\chi) \leftarrow \sum_{i=1}^n U_{ij}\ell_i(\chi)$, $v_j(\chi) \leftarrow \sum_{i=1}^n V_{ij}\ell_i(\chi)$, $w_j(\chi) \leftarrow \sum_{i=1}^n W_{ij}\ell_i(\chi)$ for all $j \in \{0, \ldots, m\}$. Let

$$\mathsf{crs}_{\mathsf{P}} \leftarrow \left(\begin{array}{l} \left[\alpha, \beta, \delta, \left(\frac{u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi)}{\delta}\right)_{j=m_0+1}^{m}\right]_1, \\ \left[(\chi^i \ell(\chi)/\delta)_{i=0}^{n-2}, (u_j(\chi), v_j(\chi))_{j=0}^{m}\right]_1, \left[\beta, \delta, (v_j(\chi))_{j=0}^{m}\right]_2 \end{array} \right) ,$$

$$\mathsf{crs}_{\mathsf{V}} \leftarrow \left(\left[\left(\frac{u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi)}{\gamma}\right)_{j=0}^{m_0} \right]_1, [\gamma, \delta]_2, [\alpha\beta]_T \right) ,$$

$$\mathsf{crs}_{\mathsf{CV}} \leftarrow \left([\gamma, (\chi^i)_{i=1}^{n-1}, (\ell_i(\chi))_{i=1}^n]_1, [\alpha, \chi, \chi^{n-1}]_2\right) .$$

Return $\mathsf{crs} \leftarrow (\mathsf{crs}_{\mathsf{CV}}, \mathsf{crs}_{\mathsf{P}}, \mathsf{crs}_{\mathsf{V}})$.
$\mathsf{K}(\mathcal{R}, \mathsf{z}_{\mathcal{R}})$: Let $\mathsf{tc} \leftarrow \mathsf{K}_{\mathsf{tc}}(\mathcal{R}, \mathsf{z}_{\mathcal{R}})$. Return $(\mathsf{crs} \,\|\, \mathsf{ts}) \leftarrow (\mathsf{K}_{\mathsf{crs}} \,\|\, \mathsf{K}_{\mathsf{ts}})(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{tc})$.
$\mathsf{CV}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{crs})$:

1. For $\iota \in \{\gamma, \delta\}$: check that $[\iota]_1 \neq [0]_1$
2. For $\iota \in \{\alpha, \beta, \gamma, \delta\}$: check that $[\iota]_1 \bullet [1]_2 = [1]_1 \bullet [\iota]_2$,
3. For $i = 1$ to $n-1$: check that $\left[\chi^i\right]_1 \bullet [1]_2 = \left[\chi^{i-1}\right]_1 \bullet [\chi]_2$,
4. Check that $([\ell_i(\chi)]_1)_{i=1}^n$ is correctly computed,
5. For $j = 0$ to m:
   (a) Check that $[u_j(\chi)]_1 = \sum_{i=1}^n U_{ij} [\ell_i(\chi)]_1$,
   (b) Check that $[v_j(\chi)]_1 = \sum_{i=1}^n V_{ij} [\ell_i(\chi)]_1$,
   (c) Set $[w_j(\chi)]_1 \leftarrow \sum_{i=1}^n W_{ij} [\ell_i(\chi)]_1$,
   (d) Check that $[v_j(\chi)]_1 \bullet [1]_2 = [1]_1 \bullet [v_j(\chi)]_2$,
6. For $j = m_0 + 1$ to m: check that $[(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\delta]_1 \bullet [\delta]_2 = [u_j(\chi)]_1 \bullet [\beta]_2 + [v_j(\chi)]_1 \bullet [\alpha]_2 + [w_j(\chi)]_1 \bullet [1]_2$,
7. Check that $\left[\chi^{n-1}\right]_1 \bullet [1]_2 = [1]_1 \bullet \left[\chi^{n-1}\right]_2$,
8. For $i = 0$ to $n-2$: check that $\left[\chi^i \ell(\chi)/\delta\right]_1 \bullet [\delta]_2 = \left[\chi^{i+1}\right]_1 \bullet \left[\chi^{n-1}\right]_2 - \left[\chi^i\right]_1 \bullet [1]_2$,
9. Check that $[\alpha]_1 \bullet [\beta]_2 = [\alpha\beta]_T$.

Figure 3.1: The CRS generation and verification of the Sub-ZK SNARK for $\mathcal{R}$

Theorem 4. *The new SNARK of Fig.3.2 is perfectly subversion-complete.*

Theorem 5 (Knowledge-soundness)**.** *Consider the new argument system of Fig.3.2. It is adaptively knowledge-sound in the Sub-GBGM even in the case of symmetric pairings. More precisely, any generic adversary attacking the knowledge-soundness of the new argument system in the symmetric setting requires $\Omega(\sqrt{p/n})$ computation.*

Subversion Zero-Knowledge proof has been divided into a number of lemmas. First the authors shows that if CV algorithm accepts then there is a bijection between trapdoor elements and the output CRS.

Lemma 6. *Let $(\mathcal{R}, \mathsf{z}_{\mathcal{R}}) \in \mathrm{range}(\mathcal{R}(1^\lambda))$, and let crs be any CRS such that $\mathsf{CV}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{crs}) = 1$. Then, with probability 1, $\mathsf{crs} = \mathsf{K}_{\mathsf{crs}}(\mathcal{R}, \mathsf{z}_{\mathcal{R}}, \mathsf{tc})$ for a tc bijectively corresponding to $[\chi, \alpha, \beta, \gamma, \delta]_1$.*

For the sake of simulatability, one has to show that from a given CRS it is possible to extract the trapdoor.

$\mathsf{P}(\mathcal{R}, \mathsf{z}_\mathcal{R}, \mathsf{crs}_\mathsf{P}, \mathsf{x} = (A_1, \ldots, A_{m_0}), \mathsf{w} = (A_{m_0+1}, \ldots, A_m))$:

`/* After executing CV & assuming` $A_0 = 1$`, the prover does: */`

1. Let $a^\dagger(X) \leftarrow \sum_{j=0}^{m} A_j u_j(X)$, $b^\dagger(X) \leftarrow \sum_{j=0}^{m} A_j v_j(X)$,
2. Let $c^\dagger(X) \leftarrow \sum_{j=0}^{m} A_j w_j(X)$,
3. Set $h(X) = \sum_{i=0}^{n-2} h_i X^i \leftarrow (a^\dagger(X) b^\dagger(X) - c^\dagger(X))/\ell(X)$,
4. Set $[h(\chi)\ell(\chi)/\delta]_1 \leftarrow \sum_{i=0}^{n-2} h_i \left[\chi^i \ell(\chi)/\delta\right]_1$,
5. Set $r_a \leftarrow_r \mathbb{Z}_p$; Set $\mathfrak{A} \leftarrow \sum_{j=0}^{m} A_j \left[u_j(\chi)\right]_1 + [\alpha]_1 + r_a [\delta]_1$,
6. Set $r_b \leftarrow_r \mathbb{Z}_p$; Set $\mathfrak{b} \leftarrow \sum_{j=0}^{m} A_j \left[v_j(\chi)\right]_2 + [\beta]_2 + r_b [\delta]_2$,
7. Set $\mathfrak{c} \leftarrow r_b \mathfrak{A} + r_a \left(\sum_{j=0}^{m} A_j \left[v_j(\chi)\right]_1 + [\beta]_1\right) + \sum_{j=m_0+1}^{m} A_j \left[(u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi))/\delta\right]_1 + [h(\chi)\ell(\chi)/\delta]_1$,
8. Return $\pi \leftarrow (\mathfrak{A}, \mathfrak{b}, \mathfrak{c})$.

$\mathsf{V}(\mathcal{R}, \mathsf{z}_\mathcal{R}, \mathsf{crs}_\mathsf{V}, \mathsf{x} = (A_1, \ldots, A_{m_0}), \pi = (\mathfrak{A}, \mathfrak{b}, \mathfrak{c}))$: assuming $A_0 = 1$, check that

$$\mathfrak{A} \bullet \mathfrak{b} = \mathfrak{c} \bullet [\delta]_2 + \left(\sum_{j=0}^{m_0} A_j \left[\tfrac{u_j(\chi)\beta + v_j(\chi)\alpha + w_j(\chi)}{\gamma}\right]_1\right) \bullet [\gamma]_2 + [\alpha\beta]_T \quad .$$

Figure 3.2: The prover and the verifier of the Sub-ZK SNARK for $\mathcal{R}$ (unchanged from Groth's zk-SNARK)

Theorem 7. *The SNARK from Sect. 3.4 has perfect CRS trapdoor extractability under the BDH-KE assumption.*

Finally, the authors show that given $\mathfrak{A}$ and $\mathfrak{b}$ there exists only one $\mathfrak{c}$ that verifier would accept.

Lemma 8. *Let* $(\mathcal{R}, \mathsf{z}_\mathcal{R}) \in \mathrm{range}(\mathcal{R}(1^\lambda))$*, and let* crs *be any CRS such that* $\mathsf{CV}(\mathcal{R}, \mathsf{z}_\mathcal{R}, \mathsf{crs}) = 1$*. Consider any values of* $\mathfrak{A}$*,* $\mathfrak{b}$*, and* $(A_j)_{j=0}^{m_0}$*. Then there exists at most one value* $\mathfrak{c}$*, such that* $\mathsf{V}(\mathcal{R}, \mathsf{z}_\mathcal{R}, \mathsf{crs}_\mathsf{P}, \mathsf{x}, (\mathfrak{A}, \mathfrak{b}, \mathfrak{c})) = 1$.

These lemmas conclude in the following theorem:

Theorem 9. *The SNARK from Sect. 3.4 is perfectly composable Sub-ZK under the BDH-KE assumption.*

Bibliography

[1] B. Abdolmaleki, K. Baghery, H. Lipmaa, and M. Zając. A Subversion-Resistant SNARK. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017 (3)*, volume 10626 of *LNCS*, pages 3–33, Hong Kong, China, Dec. 3–7, 2017. Springer, Cham.

[2] S. Bayer and J. Groth. Efficient Zero-Knowledge Argument for Correctness of a Shuffle. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280, Cambridge, UK, Apr. 15–19, 2012. Springer, Heidelberg.

[3] M. Bellare, G. Fuchsbauer, and A. Scafuro. NIZKs with an Untrusted CRS: Security in the Face of Parameter Subversion. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016 (2)*, volume 10032 of *LNCS*, pages 777–804, Hanoi, Vietnam, Dec. 4–8, 2016. Springer, Heidelberg.

[4] M. Bellare, J. A. Garay, and T. Rabin. Batch Verification with Applications to Cryptography and Checking. In C. L. Lucchesi and A. V. Moura, editors, *LATIN 1998*, volume 1380 of *LNCS*, pages 170–191, Campinas, Brazil, Apr. 20–24, 1998. Springer, Heidelberg.

[5] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *IEEE SP 2014*, pages 459–474, Berkeley, CA, USA, May 18-21, 2014. IEEE Computer Society.

[6] E. Ben-Sasson, A. Chiesa, M. Green, E. Tromer, and M. Virza. Secure Sampling of Public Parameters for Succinct Zero Knowledge Proofs. In *IEEE SP 2015*, pages 287–304, San Jose, CA, USA, May 17-21, 2015. IEEE Computer Society.

[7] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Scalable Zero Knowledge via Cycles of Elliptic Curves. In J. A. Garay and R. Gennaro, editors, *CRYPTO (2) 2014*, volume 8617 of *LNCS*, pages 276–294, Santa Barbara, California, USA, Aug. 17–21, 2014. Springer, Heidelberg.

[8] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. In K. Fu and J. Jung, editors, *USENIX 2014*, pages 781–796, San Jose, CA, USA, Aug. 20–22, 2014. USENIX Association.

[9] N. Bitansky, R. Canetti, O. Paneth, and A. Rosen. On the Existence of Extractable One-Way Functions. In D. Shmoys, editor, *STOC 2014*, pages 505–514, New York, NY, USA, May 31 – Jun 3, 2014. ACM Press.

[10] M. Blum, P. Feldman, and S. Micali. Non-Interactive Zero-Knowledge and Its Applications. In *STOC 1988*, pages 103–112, Chicago, Illinois, USA, May 2–4, 1988. ACM Press.

[11] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg.

[12] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[13] C. Costello, C. Fournet, J. Howell, M. Kohlweiss, B. Kreuter, M. Naehrig, B. Parno, and S. Zahur. Geppetto: Versatile Verifiable Computation. In *IEEE SP 2015*, pages 253–270, San Jose, CA, USA, May 17-21, 2015. IEEE Computer Society.

[14] I. Damgård. Towards Practical Public Key Systems Secure against Chosen Ciphertext Attacks. In J. Feigenbaum, editor, *CRYPTO 1991*, volume 576 of *LNCS*, pages 445–456, Santa Barbara, California, USA, Aug. 11–15, 1991. Springer, Heidelberg, 1992.

[15] G. Danezis, C. Fournet, J. Groth, and M. Kohlweiss. Square Span Programs with Applications to Succinct NIZK Arguments. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014 (1)*, volume 8873 of *LNCS*, pages 532–550, Kaohsiung, Taiwan, R.O.C., Dec. 7–11, 2014. Springer, Heidelberg.

[16] A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. L. Villar. An Algebraic Framework for Diffie-Hellman Assumptions. In R. Canetti and J. Garay, editors, *CRYPTO (2) 2013*, volume 8043 of *LNCS*, pages 129–147, Santa Barbara, California, USA, Aug. 18–22, 2013. Springer, Heidelberg.

[17] P. Fauzi and H. Lipmaa. Efficient Culpably Sound NIZK Shuffle Argument without Random Oracles. In K. Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 200–216, San Franscisco, CA, USA, February 29–March 4, 2016. Springer, Heildeberg.

[18] P. Fauzi, H. Lipmaa, J. Siim, and M. Zając. An Efficient Pairing-Based Shuffle Argument. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017 (2)*, volume 10625 of *LNCS*, pages 98–127, Hong Kong, China, Dec. 3–7, 2017. Springer, Heidelberg.

[19] P. Fauzi, H. Lipmaa, and M. Zając. A Shuffle Argument Secure in the Generic Model. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016 (2)*, volume 10032 of *LNCS*, pages 841–872, Hanoi, Vietnam, Dec. 4–8, 2016. Springer, Heidelberg.

[20] J. Furukawa and K. Sako. An Efficient Scheme for Proving a Shuffle. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 368–387, Santa Barbara, USA, Aug. 19–23, 2001. Springer, Heidelberg.

[21] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for Cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[22] R. Gennaro, C. Gentry, and B. Parno. Non-Interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482, Santa Barbara, California, USA, Aug. 15–19, 2010. Springer, Heidelberg.

[23] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic Span Programs and NIZKs without PCPs. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645, Athens, Greece, Apr. 26–30, 2013. Springer, Heidelberg.

[24] C. Gentry and D. Wichs. Separating Succinct Non-Interactive Arguments from All Falsifiable Assumptions. In S. Vadhan, editor, *STOC 2011*, pages 99–108, San Jose, California, USA, June 6–8, 2011. ACM Press.

[25] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof-Systems. In R. Sedgewick, editor, *STOC 1985*, pages 291–304, Providence, Rhode Island, USA, May 6–8, 1985. ACM Press.

[26] P. Golle, S. Jarecki, and I. Mironov. Cryptographic Primitives Enforcing Communication and Storage Complexity. In M. Blaze, editor, *FC 2002*, volume 2357 of *LNCS*, pages 120–135, Southhampton Beach, Bermuda, Mar. 11–14, 2002. Springer, Heidelberg.

[27] A. González and C. Ràfols. New Techniques for Non-interactive Shuffle and Range Arguments. In M. Manulis, A.-R. Sadeghi, and S. Schneider, editors, *ACNS 2016*, volume 9696 of *LNCS*, pages 427–444, Guildford, UK, June 19–22, 2016. Springer, Heidelberg.

[28] J. Groth. Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In X. Lai and K. Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459, Shanghai, China, Dec. 3–7, 2006. Springer, Heidelberg.

[29] J. Groth. A Verifiable Secret Shuffle of Homomorphic Encryptions. *J. Cryptology*, 23(4):546–579, 2010.

[30] J. Groth. Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340, Singapore, Dec. 5–9, 2010. Springer, Heidelberg.

[31] J. Groth. On the Size of Pairing-based Non-interactive Arguments. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 305–326, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg.

[32] J. Groth and S. Lu. A Non-interactive Shuffle with Pairing Based Verifiability. In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 51–67, Kuching, Malaysia, Dec. 2–6, 2007. Springer, Heidelberg.

[33] J. Groth, R. Ostrovsky, and A. Sahai. New Techniques for Noninteractive Zero-Knowledge. *Journal of the ACM*, 59(3), 2012.

[34] C. S. Jutla and A. Roy. Shorter Quasi-Adaptive NIZK Proofs for Linear Subspaces. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013 (1)*, volume 8269 of *LNCS*, pages 1–20, Bangalore, India, Dec. 1–5, 2013. Springer, Heidelberg.

[35] C. S. Jutla and A. Roy. Switching Lemma for Bilinear Tests and Constant-Size NIZK Proofs for Linear Subspaces. In J. A. Garay and R. Gennaro, editors, *CRYPTO (2) 2014*, volume 8617 of *LNCS*, pages 295–312, Santa Barbara, California, USA, Aug. 17–21, 2014. Springer, Heidelberg.

[36] E. Kiltz and H. Wee. Quasi-Adaptive NIZK for Linear Subspaces Revisited. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015*, volume 9057 of *LNCS*, pages 101–128, Sofia, Bulgaria, May 26–30, 2015. Springer, Heidelberg.

[37] H. Lipmaa. Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments. In R. Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189, Taormina, Italy, Mar. 18–21, 2012. Springer, Heidelberg.

[38] H. Lipmaa. Succinct Non-Interactive Zero Knowledge Arguments from Span Programs and Linear Error-Correcting Codes. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013 (1)*, volume 8269 of *LNCS*, pages 41–60, Bangalore, India, Dec. 1–5, 2013. Springer, Heidelberg.

[39] H. Lipmaa and B. Zhang. A More Efficient Computationally Sound Non-Interactive Zero-Knowledge Shuffle Argument. In I. Visconti and R. D. Prisco, editors, *SCN 2012*, volume 7485 of *LNCS*, pages 477–502, Amalfi, Italy, September 5–7, 2012. Springer, Heidelberg.

[40] U. M. Maurer. Abstract Models of Computation in Cryptography. In N. P. Smart, editor, *Cryptography and Coding 2005*, pages 1–12, Cirencester, UK, Dec. 19–21, 2005.

[41] P. Morillo, C. Ràfols, and J. L. Villar. The Kernel Matrix Diffie-Hellman Assumption. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016 (1)*, volume 10032 of *LNCS*, pages 729–758, Hanoi, Vietnam, Dec. 4–8, 2016. Springer, Heidelberg.

[42] V. I. Nechaev. Complexity of a Determinate Algorithm for the Discrete Logarithm. *Mathematical Notes*, 55(2):165–172, 1994. Translated from *Matematicheskie Zapiski*, 55(2):91–101, 1994.

[43] B. Parno, C. Gentry, J. Howell, and M. Raykova. Pinocchio: Nearly Practical Verifiable Computation. In *IEEE SP 2013*, pages 238–252, Berkeley, CA, USA, May 19-22, 2013. IEEE Computer Society.

[44] V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In W. Fumy, editor, *EUROCRYPT 1997*, volume 1233 of *LNCS*, pages 256–266, Konstanz, Germany, 11–15 May 1997. Springer, Heidelberg.

4. Efficient Designated-Verifier NIZK Proofs

4.1 Introduction

Zero-knowledge proof systems allow a prover to convince someone of the truth of a statement, without revealing anything beyond the fact that the statement is true. After their introduction in the seminal work of Goldwasser, Micali, and Rackoff [34], they have proven to be a fundamental primitive in cryptography. Among them, *non-interactive zero-knowledge proofs* (NIZK proofs), where the proof consists of a single flow from the prover to the verifier, are of particular interest, in part due to their tremendous number of applications in cryptographic primitives and protocols, and in part due to the theoretical and technical challenges that they represent.

For almost two decades after their introduction in [10], NIZKs coexisted in two types: inefficient NIZKs secure under standard assumptions (such as doubly enhanced trapdoor permutations [30]) in the common reference string model, and practically efficient NIZKs built from the Fiat-Shamir heuristic [31, 47], which are secure in the random oracle model [6] (hence only heuristically secure in the standard model). This state of affairs changed with the arrival of pairing-based cryptography, from which a fruitful line of work (starting with the work of Groth, Ostrovsky, and Sahai [37, 38]) introduced increasingly more efficient NIZK proof systems in the standard model. That line of work culminated with the framework of Groth-Sahai proofs [39], which provided an efficient framework of pairing-based NIZKs for a large class of useful languages. Yet, one decade later, pairing-based NIZKs from the Groth-Sahai framework remain the only known efficient NIZK proof system in the standard model. Building efficient NIZKs in the standard model, without pairing-based assumptions, is a major open problem, and research in this direction has proven elusive.

4.1.1 Contributions to the PANORAMIX project

In this chapter, we first introduce a framework for designated-verifier NIZKs on group-dependent languages, in the spirit of the Groth-Sahai framework for NIZKs on languages related to pairing-friendly elliptic curves. Our framework only requires that the underlying abelian group on which it is instantiated has order M, where $\mathbb{Z}_M$ is the plaintext-space of an homomorphic cryptosystem with specific properties, and allows to prove a wide variety statements formulated in terms of the operation associated to this abelian group. In particular, we do not need to rely on pairings. The DVNIZKs obtained with our framework are efficient, as they only require a few group elements and ciphertexts.

The zero-knowledge property of our schemes reduces to the IND-CPA security of the underlying encryption scheme. Additionally, our DVNIZKs enjoy the following properties: they are (adaptively) *knowledge-extractable*; their knowledge-extractability holds *statistically*; their knowledge-

extractability is *unbounded*. We stress that previously, no efficient construction of DVNIZK in the standard model satisfying *any* of the above properties was known. The third property, unbounded soundness, was only claimed to hold for the construction of [23], and this claim was formalized with a proof in an idealized model, but as previously mentioned, we found this claim to be flawed. We also point out that in the Groth-Sahai framework, witness extraction is limited either to statements about group elements, or to statements about exponents committed in a bit-by-bit fashion (making the proof highly inefficient). In contrast, our proof system allows to efficiently extract large exponents, without harming the efficiency of the proof. In addition to the above properties, our DVNIZKs satisfy some other useful properties: they are multi-theorem [30], randomizable [3], and same-string zero-knowledge [27] (*i.e.*, the common reference string used by the prover and the simulator are the same).

Second, our framework comes with a dual variant, where the role of the encryption scheme and the abelian group are reversed, to prove statements, not about elements of the abelian group, but about the underlying homomorphic encryption scheme. This dual variant leads to DVNIZKs satisfying adaptive statistical unbounded soundness, but not knowledge-extractability (i.e. the dual variant does not give proofs of knowledge).

Third, we show that if one is willing to give up unbounded soundness for efficiency, our techniques can be used to construct extremely efficient DVNIZKs with bounded-soundness. The DVNIZKs that we obtain this way are more efficient than *any* previously known construction of non-interactive zero-knowledge proofs, even when considering NIZKs in the random oracle model using the Fiat-Shamir transform: the proofs we obtain are shorter than the proofs obtained via the Fiat-Shamir transform by almost a factor two. To our knowledge, this is the first example of a NIZK construction in the standard model which (conditionally) improves on the Fiat-Shamir paradigm.

4.1.2 Designated-Verifier Non-Interactive Zero-Knowledge

Parallel to the research on NIZKs, an alternative promising line of research has focused on *designated-verifier* non-interactive zero-knowledge proof systems (DVNIZKs). A DVNIZK retains most of the security properties of a NIZK, but is not publicly verifiable: only the owner of some secret information (the designated verifier) can check the proof. Nevertheless, DVNIZKs can replace publicly verifiable NIZKs in a variety of applications. In addition, unlike their publicly-verifiable counterpart, it is known that efficient DVNIZKs secure in the standard model for rich classes of languages can be constructed without pairing-based assumptions [17,23,43,49]. However, to date, research in DVNIZKs has attracted less attention than NIZKs, the previously listed papers being (to our knowledge) the only existing works on this topic, and several important questions have been left open. We list the main open questions below.

Proofs Versus Arguments.

A non-interactive zero-knowledge argument system is a NIZK in which the soundness property is only required to hold against computationally bounded adversaries. In a NIZK *proof* system, however, soundness is required to hold even against unbounded adversaries.

Currently, while several DVNIZK argument systems have been designed in the standard model without pairing-based assumptions, efficient DVNIZK proof systems without pairings remain an open question. In fact, to our knowledge, the only known constructions of (possibly inefficient) DVNIZK proofs rely on publicly-verifiable NIZK proofs.

Soundness Versus Knowledge Extraction.

A non-interactive zero-knowledge proof (or argument) system is a NIZK *of knowledge* if it guarantees that, when the prover succeeds in convincing the verifier, he must *know* a witness for the truth of the statement. This is in constrast with the standard soundness notion, which only guarantees that the statement is true. Formally, this is ensured by requiring the existence of an efficient simulator that can extract a witness from the proof.

Non-interactive zero-knowledge proofs of knowledge are more powerful than standard NIZKs, and the knowledge-extractability property is crucial in many applications. In particular, they are necessary for the very common task of proving relations between values committed with a perfectly hiding commitment scheme, and they are a core component in privacy-preserving authentication mechanisms [4]. Currently, all known DVNIZK argument systems are not arguments of knowledge. Designing efficient DVNIZKs of knowledge without pairing-based assumptions remains an open question.

Bounded Soundness Versus Unbounded Soundness.

The classical soundness security notion for non-interactive zero-knowledge proof systems states that if the statement is not true, no malicious prover can possibly convince the verifier of the truth of the statement with non-negligible probability. While this security notion is sufficient for publicly-verifiable NIZKs, it turns out to be insufficient when considering designated-verifier NIZKs, and corresponds only to a *passive* type of security notion. Indeed, the verification of a DVNIZK involves a secret value, known to the verifier. The fact that a DVNIZK satisfies the standard soundness notion does not preclude the possibility for a malicious prover to learn this secret value, e.g. by submitting a large number of proofs and receiving feedback on whether the proof was accepted or not. Intuitively, this is the same type of issue as for encryption schemes indistinguishable against chosen-plaintext attacks, which can be broken if the adversary is given access to a decryption oracle, or for signature schemes secure against key-only or known-message attacks, which can be broken if the adversary is given access to a signing oracle. Here, an adversary could possibly break the soundness of a DVNIZK if it is given access to a verification oracle.

In practice, this means that as soon as a proof system with bounded soundness is used for more than a logarithmic number of proofs, the soundness property is no longer guaranteed to hold. This calls for a stronger notion of soundness, *unbounded soundness*, which guarantees security even against adversaries that are given arbitrary access to a verification oracle.

Designing a DVNIZK with unbounded soundness has proven to be highly non-trivial. In fact, apart from publicly-verifiable NIZKs (which can be seen as particular types of DVNIZKs where the secret key of the verifier is the empty string), the only known construction of DVNIZK claiming to satisfy unbounded soundness is the construction of [23], where the claim is supported by a proof of security in an idealized model. However, we found this claim to be flawed: there is an explicit attack against the unbounded soundness of any protocol obtained using the compiler of [23], which operates by using slightly malformed proofs to extract the verification key. In the full version of this work [16], we describe our attack, and identify the flaw in the proof of Theorem 5 in [23, Appendix A]. We have notified the authors of our finding and will update future versions of this work with their reply. To our knowledge, in all current constructions, the common reference string and the public key must be refreshed after a logarithmic number of proofs.

Instantiating the Encryption Scheme.

Informally, the security properties we require from the underlying scheme are the following: it must be additively homomorphic, with plaintext space $\mathbb{Z}_M$, random source $\mathbb{Z}_R$, and $\gcd(M, R) = 1$, and it must be *decodable*, which means that a plaintext m can be efficiently recovered from an encryption of m with random coin 0. A natural candidate for the above scheme is the Paillier encryption scheme [45] (and its variants, such as Damgård-Jurik [24]). This gives rise to efficient DVNIZK proofs of knowledge over abelian groups of composite order (e.g. subgroups of $\mathbb{F}_p^*$, with order a prime $p = k \cdot n + 1$ for a small k and an RSA modulus n, or composite-order elliptic curves), as well as efficient DVNIZKs for proving relations between Paillier ciphertexts (using the dual variant of our framework). Alternatively, the scheme can also be instantiated with the more recent Castagnos-Laguillaumie encryption scheme [15] to get DVNIZKs over prime-order abelian groups.

Our framework captures many useful zero-knowledge proofs of knowledge that are commonly used in cryptography. This includes DVNIZK proofs of knowledge of a discrete logarithm, of correctness of a Diffie-Hellman tuple, of multiplicative relationships between Pedersen commitments or ElGamal ciphertexts (or variants thereof), among many others. Our results show that, in the settings where a designated-verifier is sufficient, one can build efficient non-interactive zero-knowledge proofs of knowledge for most statements of interest, under well-known assumptions and with strong security properties, without having to rely on pairing-friendly groups.

4.1.3 Our Method

It is known that linear relations (*i.e.*, membership in linear subspaces) can be non-interactively verified, using the homomorphic properties of cryptographic primitives over abelian groups. Indeed, DVNIZK proofs for linear languages can be constructed, e.g., from hash proof systems [33,41]. In [39], pairings provide exactly the additional structure needed to evaluate degree-two relations, which can be easily generalized to arbitrary relations.

An alternative road was taken in [23] and subsequent works, to obtain non-interactive zero-knowledge proofs for a wide variety of relations, in the designated-verifier setting. To illustrate, let us consider a prover interacting with a verifier, with a common input $(g_1, g_2, h_1, h_2) \in \mathbb{G}^4$ in some group $\mathbb{G}$ of order p, where p is a λ-bit prime. The prover wants to show that (h_1, h_2) have the same discrete logarithm in the basis (g_1, g_2), *i.e.*, there exists x such that $(h_1, h_2) = (g_1^x, g_2^x)$. The standard interactive zero-knowledge proof for this statement proceeds as follows:[1]

1. The prover picks $r \xleftarrow{\$} \{0,1\}^{3\lambda}$, and sends $(a_1, a_2) \leftarrow (g_1^r, g_2^r)$.
2. The verifier picks and sends a uniformly random challenge $e \xleftarrow{\$} \mathbb{Z}_p$.
3. The prover computes and sends $d \leftarrow e \cdot x + r$. The verifier accepts the proof if and only if $(g_1^d, g_2^d) = (h_1^e a_1, h_2^e a_2)$.

The idea of [23] is to squash this interactive protocol into a (designated-verifier) non-interactive proof, by giving the challenge to the prover in advance. As knowing the challenge before sending the first flow gives the prover the ability to cheat, the challenge is encrypted with an additively homomorphic encryption scheme. That way, the prover cannot see the challenge; yet, he can still compute an encryption of the value d homomorphically, using the encryption of e. The verifier, who is given the secret verification key, can decrypt the last flow and perform the above check. Thus, the

[1] More formally, this proof only satisfies zero-knowledge against honest verifiers, but this property is sufficient for the construction of [23].

proof is a tuple (a_1, a_2, c_d), where c_d is an encryption of d computed from (x, r) and an encryption c_e of the challenge e.

Although natural, this intuitive approach has proven quite tough to analyze. In [23], the authors had to rely on a new complexity-leveraging-type assumption tailored to their scheme, which (informally) states that the simulator cannot break the security of the encryption scheme, even if he is powerful enough to break the problem underlying the protocol (in the above example, the discrete logarithm problem over $\mathbb{G}$). Even in the bounded setting, analyzing the soundness guarantees of the protocols obtained by this compilation technique (and its variants) is non-trivial, and it has been the subject of several subsequent works [17, 43, 49]. Additionally, in the unbounded setting, where we must give an efficient simulator that can successfully answer to the proofs submitted by any malicious prover, this compilation technique breaks down. Furthermore, for DVNIZKs constructed with this method, soundness holds only computationally, and security does not guarantee that the simulator can extract a witness for the statement.

Our core idea to overcome all of the above issues is to implement the same strategy in a slightly different way: rather than encrypting the challenge e as the *plaintext* of an homomorphic encryption scheme, we encrypt it as the *random coin* of an encryption scheme which is also homomorphic over the coins. To understand how this allows us to improve over all previous constructions, suppose that we have an encryption scheme Enc which is homomorphic over both the plaintext and the random coins, with plaintext space $\mathbb{Z}_M$ and random source $\mathbb{Z}_R$, and that M is coprime to R. Consider the previously described protocol for proving equality of two discrete logarithms. Given an encryption $\mathsf{Enc}(0; e)$ of 0, where the challenge is the random coin, a prover holding (x, r) can compute and send $\mathsf{Enc}(x; \rho)$ and $\mathsf{Enc}(r; -e\rho)$, for some random ρ. This allows the verifier, who knows e, to compute $\mathsf{Enc}(x \cdot e + r; 0)$, from which she can extract $d = x \cdot e + r \bmod M$ (note that the verifier only needs to know e; unlike in previous work, she does not need to know the decryption key of Enc). Observe that the extracted value depends only on e *modulo* M. At the same time, however, the ciphertext $E(0; e)$ only leaks e *modulo* R, even to an unbounded adversary. By picking e to be sufficiently large ($e > MR$), as M is coprime to R, the verifier can ensure that this leaks *no information* (statistically) about $e \bmod M$. Therefore, we can use a statistical argument to show that the prover cannot cheat when the verification using d succeeds. To allow for efficient simulation of the verifier, we simply give to the simulator the secret key of the scheme, which will allow him to extract all encrypted values, and to check the validity of the equations, without knowing $e \bmod M$. As the simulator is able to extract the values encrypted with Enc, the scheme can be proven to be (statistically) knowledge-extractable. Contrary to previous constructions, the verification key is a random coin rather than the secret key of an encryption scheme. The secret key is only used to extract information in the simulated game.

Example: DVNIZK Proof of Knowledge of a Discrete Logarithm.

We illustrate our method with the classical example of proving knowledge of a discrete logarithm. For concreteness, we describe an explicit protocol using the Paillier encryption scheme; therefore, this section assumes some basic knowledge of the Paillier encryption scheme. All necessary preliminaries can be found in Section 4.2. Let $\mathbb{G}$ be a group of order n, where $n = p \cdot q$ is an RSA modulus (*i.e.*, a product of two strong primes). Let g be a generator of $\mathbb{G}$, and let T be a group element. A prover P wishes to prove to a verifier V that he knows a value $t \in \mathbb{Z}_n$ such that $g^t = T$.

Let $h \leftarrow u^n \bmod n^2$, where u denotes an arbitrary generator of $\mathbb{J}_n$, the subgroup of elements of $\mathbb{Z}_n^*$ with Jacobi symbol 1. The Paillier encryption of a message $m \in \mathbb{Z}_n$ with randomness $r \in \mathbb{Z}_{\varphi(n)/2}$ is $\mathsf{Enc}(m; r) = (1 + n)^m h^r \bmod n^2$. The public key of the DVNIZK is $E = h^e \in \mathbb{Z}_{n^2}^*$, for a random $e \gg n \cdot \varphi(n)/2$; observe that this is exactly $\mathsf{Enc}(0; e)$. The secret key is e. The DVNIZK proceeds as

follows:

The prover P picks $x \xleftarrow{\$} \mathbb{Z}_n$ and a Paillier random coin r, and computes $X \leftarrow g^x$, $T' \leftarrow (1+n)^t h^r \bmod n^2$, and $X' \leftarrow (1+n)^x E^{-r} \bmod n^2$. The verifier V computes $D \leftarrow T^e X \bmod n^2$ and $D' \leftarrow (T')^e X' \bmod n^2$. Then, she checks that D' is of the form $(1+n)^d \bmod n^2$. If so, V computes $d \bmod n$ from D', and checks that $D = g^d$. V accepts iff both checks succeeded.

Let us provide an intuition of the security of this scheme. Correctness follows easily by inspection. Zero-knowledge comes from the fact that T' hides t, under the IND-CPA security of Paillier. For statistical knowledge extractability, note E only reveals $e \bmod \varphi(n)$ to an unbounded adversary, which leaks (statistically) no information on $e \bmod n$ as $\varphi(n)$ is coprime to n. This ensures the value t' encrypted in T' must be equal to t, otherwise the verification equations would uniquely define $e \bmod n$, which is statistically unknown to the prover. The simulator knows $\varphi(n)$ (but not $e \bmod n$) and gets t by decrypting T'.

4.1.4 Applications

A natural application of non-interactive zero-knowledge proofs of knowledge is the design of privacy-preserving non-interactive authentication schemes. This includes classical authentication protocols, but also P-signatures [4] and their many applications, such as anonymous credentials [4], group signatures [20], electronic cash [19], or anonymous authentication [48]. Our framework can lead to a variety of efficient new constructions of designated-verifier variants for the above applications without pairings, whereas all previous constructions either had to rely on the random oracle model, or use pairing-based cryptography.[2] In many scenarios of non-interactive authentication, the designated-verifier property is not an issue.

In addition, the aforementioned applications build upon the Groth-Sahai framework for NIZKs. However, Groth-Sahai NIZKs only satisfy a restricted notion of extractability, called f-extractability in [4]. As a result, constructions of privacy-preserving authentication mechanisms from Groth-Sahai NIZKs require a careful security analysis. Our framework leads to fully extractable zero-knowledge proofs, which could potentially simplify this. We note that our DVNIZKs are additionally randomizable, which has applications for delegatable anonymous credential schemes [3].

Other potential applications of our framework include round-efficient two-party computation protocols secure against malicious adversaries, electronic voting (see e.g. [17]), as well as designated-verifier variants of standard cryptographic primitives, such as verifiable encryption [13], or verifiable pseudorandom-functions [5]. Potential applications to the construction of adaptive oblivious transfers can also be envisioned: in [35], the authors mention that an adaptive oblivious transfer protocol can be designed by replacing the interactive zero-knowledge proofs of the protocol of [14] by non-interactive one. They raise two issues to this approach, namely, that Groth-Sahai proofs are only witness-indistinguishable for the required class of statements, and that they only satisfy a weak form of extractability. None of these restrictions apply to our DVNIZK constructions.

4.1.5 Related Work

Non-interactive zero-knowledge proofs were first introduced in [10]. Efficient publicly-verifiable non-interactive zero-knowledge proofs can be constructed in the random oracle model [31, 32, 47], or in the non-programmable random oracle model [42] (using a common reference string in addition).

[2] These applications typically require a proof-friendly signature scheme, but designated-verifier variants of such scheme can easily be constructed (without pairings) from algebraic MACs [18, 40], by committing to the secret key of the MAC and proving knowledge of the committed value with a DVNIZK; such statements are naturally handled by our framework.

The latter construction was improved in [21]. In the standard model, the main construction of efficient publicly-verifiable NIZKs is the Groth-Sahai framework [39].

Designated-verifier non-interactive zero-knowledge arguments where first introduced in [46], where it was shown that the existence of semantically secure encryption implies the existence of DVNIZK arguments with bounded soundness; however, the construction is highly inefficient and therefore only of theoretical interest. Furthermore, even putting aside efficiency consideration, the construction is inherently limited to arguments (as opposed to proofs) with bounded soundness (as opposed to unbounded soundness).

Designated-verifier NIZKs for linear languages can be constructed from hash proof systems [22, 33, 41]. Such NIZKs are perfectly zero-knowledge and statistically adaptively sound, but are not proofs of knowledge and are restricted to very specific statements, captured by linear equations.

Efficient designated-verifier NIZKs for more general statements were first described in [23]. The authors describe a general compiler that converts any three-round (honest-verifier) zero-knowledge protocol satisfying some (mild) requirements into a DVNIZK. However, the construction has several drawbacks: the soundness only holds under a very specific complexity-leveraging assumption, and only against adversaries making at most $O(\log \lambda)$ proofs (as already mentioned, the paper claims that the construction enjoy unbounded soundness as well, but this claim is flawed, see the full version [16]). In addition, the proofs obtained with this compiler are not proofs of knowledge.

In subsequent works [17, 49], variations of the compilation technique of [23] are described, where the complexity-leveraging assumption was replaced by more standard assumptions (although achieving a more restricted type of soundness) by relying on encryption schemes with additional properties. Eventually, [43] removes some of the constraints of the constructions of [17], and provides new protocols that can be compiled using the transformation. However, all the constructions obtained in these papers are only computationally sound, do not enjoy unbounded soundness, and are not proofs of knowledge; this strongly limits their scope, and in particular, prevents them from being used in the previously discussed applications.

4.1.6 Organization

In Section 4.2, we introduce our notation, and necessary primitives. We refer the reader to the full version of this work [16] for classical preliminaries on commitments and cryptosystems. Section 4.2 also describes the notion of a DVNIZK-friendly encryption scheme, which is central to our framework. In Section 4.3, we introduce our framework for building DVNIZKs of knowledge over an abelian group, illustrate it with practical examples, and prove its security. In Section 4.4, we describe the dual variant of our framework for proving statements over plaintexts of a DVNIZK-friendly encryption scheme. In the full version of this work [16], we additionally describe optimizations on the efficiency of DVNIZKs for relations between plaintexts of a DVNIZK-friendly scheme, by eschewing unbounded soundness, as well as our attack on the unbounded soundness of [23].

4.2 Preliminaries

Throughout this chapter, λ denotes the security parameter. A probabilistic polynomial time algorithm (PPT, also denoted *efficient* algorithm) runs in time polynomial in the (implicit) security parameter λ. A positive function f is *negligible* if for any polynomial p there exists a bound $B > 0$ such that, for any integer $k \geq B$, $f(k) \leq 1/|p(k)|$. An event depending on λ occurs with *overwhelming probability* when its probability is at least $1 - \mathrm{negl}(\lambda)$ for a negligible function negl. Given a finite set S, the notation $x \xleftarrow{\$} S$ means a uniformly random assignment of an element of S to the variable x. We represent adversaries as interactive probabilistic Turing machines; the

notation $\mathscr{A}^{\mathcal{O}}$ indicates that the machine $\mathscr{A}$ is given oracle access to $\mathcal{O}$. Adversaries will sometime output an arbitrary state st to capture stateful interactions.

Abelian Groups and Modules.

We use additive notation for groups for convenience, and write $(\mathbb{G}, \boldsymbol{+})$ for an abelian group of order k. When it is clear from the context, we denote 0 its neutral element (otherwise, we denote it $0_{\mathbb{G}}$). We denote by $\bullet$ the scalar-multiplication algorithm (*i.e.* for any $(x, G) \in \mathbb{Z}_k \times \mathbb{G}$, $x \bullet G = G \boldsymbol{+} G \boldsymbol{+} \ldots \boldsymbol{+} G$, where the sum contains x terms). Observe that we can naturally view $\mathbb{G}$ as a $\mathbb{Z}_k$-module $(\mathbb{G}, \boldsymbol{+}, \bullet)$, for the ring $(\mathbb{Z}_k, +, \cdot)$. For simplicity, we write $\boldsymbol{-}G$ for $(-1) \bullet G$. We use lower case to denote elements of $\mathbb{Z}_k$, upper case to denote elements of $\mathbb{G}$, and bold notations to denote vectors. We extend the notations $(\boldsymbol{+}, \boldsymbol{-})$ to vectors and matrices in the natural way, and write $\mathbf{x} \bullet \mathbf{G}$ to denote the scalar product $x_1 \bullet G_1 \boldsymbol{+} \ldots \boldsymbol{+} x_t \bullet G_t$ (where $\mathbf{x}, \mathbf{G}$ are vectors of the same length t). For a vector $\mathbf{v}$, we denote by $\mathbf{v}^\intercal$ its transpose. By $\mathsf{GGen}(1^\lambda)$, we denote a probabilistic efficient algorithm that, given the security parameter λ, generates an abelian group $\mathbb{G}$ such that the best known algorithm for solving discrete logs in G takes time 2^λ. In the following, we write $(\mathbb{G}, k) \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$. Additionally, we denote by $\mathsf{GGen}(1^\lambda, k)$ a group generation algorithm that allows us to select the order k beforehand.

RSA Groups.

A *strong prime* is a prime $p = 2p' + 1$ such that p' is also a prime. We call *RSA modulus* a product $n = pq$ of two strong primes. We denote by φ Euler's totient function; it holds that $\varphi(n) = (p-1)(q-1)$. We denote by $\mathbb{J}_n$ the cyclic subgroup of $\mathbb{Z}_n^*$ of elements with Jacobi symbol 1 (the order of this group is $\varphi(n)/2$), and by QR_n the cyclic subroup of squares of $\mathbb{Z}_n^*$ (which is also a subgroup of $\mathbb{J}_n$ and has order $\varphi(n)/4$). By $\mathsf{Gen}(1^\lambda)$, we denote a probabilistic efficient algorithm that, given the security parameter λ, generates a strong RSA modulus n and secret parameters (p, q) where $n = pq$, such that the best known algorithm for factoring n takes time 2^λ. In the following, we write $(n, (p, q)) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$.

4.2.1 Encryption Schemes

The formal definition of an IND-CPA-secure public-key encryption scheme is recalled in the full version [16], but in short, a public-key encryption scheme S is a triple of PPT algorithms $(S.\mathsf{KeyGen}, S.\mathsf{Enc}, S.\mathsf{Dec})$, where $S.\mathsf{KeyGen}$ generates a pair $(\mathsf{ek}, \mathsf{dk})$ with an encryption key and a decryption key, decryption (with dk, deterministically) is the reverse operation of encryption (with ek, randomized), and no adversary can distinguish encryptions of one of two messages of its choice (IND-CPA security).

In this work, we will focus on additively homomorphic encryption schemes, which are homomorphic for both the message and the random coin. More formally, we require that the message space $\mathcal{M}$ and the random source $\mathcal{R}$ are integer sets $(\mathbb{Z}_M, \mathbb{Z}_R)$ for some integers (M, R), and that there exists an efficient operation $\oplus$ such that for any $(\mathsf{ek}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$, any $(m_1, m_2) \in \mathbb{Z}_M^2$ and $(r_1, r_2) \in \mathbb{Z}_R^2$, denoting $(C_i)_{i \leq 2} \leftarrow (S.\mathsf{Enc}_{\mathsf{ek}}(m_i; r_i))_{i \leq 2}$, it holds that $C_1 \oplus C_2 = S.\mathsf{Enc}_{\mathsf{ek}}(m_1 + m_2 \bmod M; r_1 + r_2 \bmod R)$. We say an encryption scheme is *strongly additive* if it satisfies these requirements. Note that the existence of $\oplus$ implies (via a standard square-and-multiply method) the existence of an algorithm that, on input a ciphertext $C = S.\mathsf{Enc}_{\mathsf{ek}}(m; r)$ and an integer $\rho \in \mathbb{Z}$, outputs a ciphertext $C' = S.\mathsf{Enc}_{\mathsf{ek}}(\rho m \bmod M; \rho r \bmod R)$. We denote by $\rho \odot C$ the external multiplication of a ciphertext C by an integer ρ, and by $\ominus$ the operation $C \oplus (-1) \odot C'$ for two ciphertexts (C, C'). We will sometimes slightly abuse these notations, and write $C \oplus m$ (resp. $C \ominus m$) for a plaintext m to denote $C \oplus S.\mathsf{Enc}_{\mathsf{ek}}(m; 0)$ (resp. $C \ominus S.\mathsf{Enc}_{\mathsf{ek}}(m; 0)$).

A simple observation on strongly additively homomorphic encryption schemes is that IND-CPA security implies that R must either be equal to $0 \mod M$, or unknown given ek. Otherwise, an IND-CPA adversary would set $(m_0, m_1) = (0, 1)$ and check if $R \odot C$ equals $S.\mathsf{Enc}_{\mathsf{ek}}(0; 0)$ or $S.\mathsf{Enc}_{\mathsf{ek}}(R; 0)$.

The Paillier Encryption Scheme.

The Paillier encryption scheme [45] is a well-known additively homomorphic encryption scheme over $\mathbb{Z}_n$ for an RSA modulus n. We describe here a standard variant [26, 43], where the random coin is an exponent over $\mathbb{J}_n$ rather than a group element. Note that the exponent space of $\mathbb{J}_n$ is $\mathbb{Z}_{\varphi(n)/2}$, which is a group of unknown order; however, it suffices to draw exponents at random from $\mathbb{Z}_{n/2}$ to get a distribution statistically close from uniform over $\mathbb{Z}_{\varphi(n)/2}$.

- $\mathsf{KeyGen}(1^\lambda)$: run $(n, (p, q)) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, pick $g \xleftarrow{\$} \mathbb{J}_n$, set $h \leftarrow g^n \mod n^2$, and compute $\delta \leftarrow n^{-1} \mod \varphi(n)$ (n and $\varphi(n)$ are relatively prime). Return $\mathsf{ek} = (n, h)$ and $\mathsf{dk} = \delta$;
- $\mathsf{Enc}(\mathsf{ek}, m; r)$: given $m \in \mathbb{Z}_n$, for a random $r \xleftarrow{\$} \mathbb{Z}_{n/2}$, compute and output $c \leftarrow (1 + n)^m \cdot h^r \mod n^2$;
- $\mathsf{Dec}(\mathsf{dk}, c)$: compute $x \leftarrow c^{\mathsf{dk}} \mod n$ and $c_0 \leftarrow [c \cdot x^{-n} \mod n^2]$. Return $m \leftarrow (c_0 - 1)/n$.

Note that knowing dk is equivalent to knowing the factorization of n. The IND-CPA security of the Paillier encryption scheme reduces to the decisional composite residuosity (DCR) assumption, which states that it is computationally infeasible to distinguish random n'th powers over $\mathbb{Z}^*_{n^2}$ from random elements of $\mathbb{Z}^*_{n^2}$.[3] It is also strongly additive, where the homomorphic addition of ciphertexts is the multiplication over $\mathbb{Z}^*_{n^2}$.

The ElGamal Encryption Scheme.

We recall the additive variant of the famous ElGamal cryptosystem [28], over an abelian group $(\mathbb{G}, \mathbf{+})$ of order k.

- $\mathsf{KeyGen}(1^\lambda)$: pick $G \xleftarrow{\$} \mathbb{G}$, pick $s \xleftarrow{\$} \mathbb{Z}_k$, set $G \leftarrow s \bullet G$, and return $\mathsf{ek} = (G, H)$ and $\mathsf{dk} = s$;
- $\mathsf{Enc}(\mathsf{ek}, m; r)$: given $m \in \mathbb{Z}_k$, for a random $r \xleftarrow{\$} \mathbb{Z}_k$, output $\mathbf{C} \leftarrow (r \bullet G, (m \bullet G)\mathbf{+}(r \bullet H))$;
- $\mathsf{Dec}(\mathsf{dk}, \mathbf{C})$: parse $\mathbf{C}$ as (C_0, C_1), and compute $M \leftarrow C_1\mathbf{-}(\mathsf{dk} \bullet C_0)$. Compute the discrete logarithm m of M in base G, and return m.

The IND-CPA security of the ElGamal encryption scheme reduces to the decisional Diffie-Hellman (DDH) assumption over $\mathbb{G}$, which states that it is computationally infeasible to distinguish tuples of the form $(G, H, x \bullet G, x \bullet H)$ for random x from uniformly random 4-tuples over $\mathbb{G}$. It is also strongly additive (and the homomorphic operation is the vector addition over $\mathbb{G}$). However, the decryption procedure is not efficient in general, as it requires to compute a discrete logarithm. For the decryption process to be efficient, the message m must be restricted to come from a subset of $\mathbb{Z}_k$ of polynomial size.

[3] In the variant we consider here, we must restrict our attention to elements of $\mathbb{Z}^*_{n^2}$ which have Jacobi symbol 1 when reduced modulo n as $g \in \mathbb{J}_n$, but this can be checked in polynomial time anyway.

DVNIZK-Friendly Encryption Scheme.

We say that a strongly additive encryption scheme is DVNIZK-*friendly*, when it satisfies the following additional properties:

- Coprimality Property: we require that the size M of the plaintext space and the size R of the random source are coprime[4], *i.e.*, $\gcd(M, R) = 1$;
- Decodable: for any $(\mathsf{ek}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$, the function $f_{\mathsf{ek}} : m \mapsto \mathsf{Enc}_{\mathsf{ek}}(m; 0)$ must be efficiently invertible (*i.e.*, there is a PPT algorithm, which is given ek, computing f_{ek}^{-1} on any value from the image of f_{ek}).

One can observe that the Paillier cryptosystem is DVNIZK-friendly ($\gcd(n, \varphi(n)) = 1$, and any message m can be efficiently recovered from $\mathsf{Enc}_{\mathsf{ek}}(m; 0) = (1 + n)^m \bmod n^2$), while the ElGamal cryptosystem is not (it satisfies none of the above properties). Other DVNIZK-friendly cryptosystems include variants of the Paillier cryptosystem [12, 22, 24–26], and the more recent Castagnos-Laguillaumie cryptosystem [15], with prime-order plaintext space. For simplicity, we will also assume that all prime factors of the size M of the plaintext space of a DVNIZK-friendly cryptosystem are of superpolynomial size; our results can be extended to cryptosystems with a small plaintext space (or a plaintext space with small prime factors), but at a cost in efficiency. Note that by the homomorphic property, the decodability property implies that a plaintext can always be recovered from a ciphertext if the random coin is known.

4.2.2 Non-Interactive Zero-Knowledge Proof Systems

In the definitions below, we focus on proof systems for NP-languages that admit an efficient (polynomial-time) prover. For an NP-language $\mathscr{L}$, we denote $R_{\mathscr{L}}$ its associated relation, *i.e.*, a polynomial-time algorithm which satisfies $\mathscr{L} = \{x \mid \exists w, |w| = \mathsf{poly}(|x|) \wedge R_{\mathscr{L}}(x, w) = 1\}$. It is well known that non-interactive proof systems cannot exist for non-trivial languages in the plain model [44]; our constructions will be described in the common reference string model. For conciseness, the common reference string is always implictly given as input to all algorithms. We note that all of our constructions can be readily adapted to work in the registered public-key model as well, a relaxation of the common reference string model introduced by Barak et al in [2].

While languages are naturally associated to *statements of membership*, the constructions of this chapter will mainly consider *statements of knowledge*. We write $\mathsf{St}(x) = \mathsf{K}\{w : R(x, w) = 1\}$ to denote the statement "I know a witness w such that $R(x, w) = 1$" for a word x and a polytime relation R. Similarly, we write $\mathsf{St}(x) = \exists\{w : R(x, w) = 1\}$ to denote the existential statement "there exists a witness w such that $R(x, w) = 1$".

Definition 1. (Non-Interactive Zero-Knowledge Proof System) A non-interactive zero-knowledge (NIZK) proof system Π between for a family of languages $\mathscr{L} = \{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ is a quadruple of probabilistic polynomial-time algorithms $(\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ such that

- $\Pi.\mathsf{Setup}(1^\lambda)$, outputs a common reference string crs (which specifies the language $\mathscr{L}_{\mathsf{crs}}$),
- $\Pi.\mathsf{KeyGen}(1^\lambda)$, outputs a public key pk and a verification key vk,
- $\Pi.\mathsf{Prove}(\mathsf{pk}, x, w)$, on input the public key pk, a word $x \in \mathscr{L}_{\mathsf{crs}}$, and a witness w, outputs a proof π,

[4] In view of our previous observation on IND-CPA security for strongly additive cryptosystems, this implies that R is secret.

- $\Pi.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, x, \pi)$, on input the public key pk, the verification key vk, a word x, and a proof π, outputs $b \in \{0, 1\}$,

which satisfies the completeness, zero-knowledge, and soundness properties defined below.

We assume for simplicity that once it is generated, the common reference string crs is implicitly passed as an argument to the algorithms $(\Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$. In the above definition of NIZK proof systems, we let the key generation algorithm generate a verification key vk which is used by the verifier to check the proofs. We call *publicly verifiable non-interactive zero-knowledge proof system* a NIZK proof system in which vk is set to the empty string (or, equivalently, in which vk is made part of the public key). Otherwise, we call it a *designated-verifier non-interactive zero-knowledge proof system.*

Definition 2. (Completeness) A NIZK proof system $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for a family of languages $\mathscr{L} = \{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ with relations R_{crs} satisfies the (perfect,statistical) completeness property if for $\mathsf{crs} \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda)$, for every $x \in \mathscr{L}_{\mathsf{crs}}$ and every witness w such that $R_{\mathsf{crs}}(x, w) = 1$,

$$\Pr\left[\begin{matrix}(\mathsf{pk}, \mathsf{vk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(1^\lambda), \\ \pi \leftarrow \Pi.\mathsf{Prove}(\mathsf{pk}, x, w)\end{matrix} : \Pi.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, x, \pi) = 1\right] = 1 - \mu(\lambda)$$

where $\mu(\lambda) = 0$ for perfect completeness, and $\mu(\lambda) = \mathrm{negl}(\lambda)$ for statistical completeness.

We now define the zero-knowledge property.

Definition 3. (Composable Zero-Knowledge) A NIZK proof system $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for a family of languages $\mathscr{L} = \{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ with relations R_{crs} satisfies the (perfect, statistical) composable zero-knowledge property if for any $\mathsf{crs} \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda)$, there exists a probabilistic polynomial-time simulator Sim such that for any stateful adversary $\mathscr{A}$,

$$\left|\Pr\left[\begin{matrix}(\mathsf{pk}, \mathsf{vk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(1^\lambda), \\ (x, w) \leftarrow \mathscr{A}(\mathsf{pk}, \mathsf{vk}), \\ \pi \leftarrow \Pi.\mathsf{Prove}(\mathsf{pk}, x, w)\end{matrix} : (R_{\mathsf{crs}}(x, w) = 1) \wedge (\mathscr{A}(\pi) = 1)\right] - \right.$$
$$\left.\Pr\left[\begin{matrix}(\mathsf{pk}, \mathsf{vk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(1^\lambda), \\ (x, w) \leftarrow \mathscr{A}(\mathsf{pk}, \mathsf{vk}), \\ \pi \leftarrow \mathsf{Sim}(\mathsf{pk}, \mathsf{vk}, x)\end{matrix} : (R_{\mathsf{crs}}(x, w) = 1) \wedge (\mathscr{A}(\pi) = 1)\right]\right| \leq \mu(\lambda)$$

where $\mu(\lambda) = 0$ for perfect composable zero-knowledge, and $\mu(\lambda) = \mathrm{negl}(\lambda)$ for statistical composable zero-knowledge. If the composable zero-knowledge property holds against efficient (PPT) verifiers, the proof system satisfies computational composable zero-knowledge.

The composable zero-knowledge property was first introduced in [36]. It strenghtens the standard zero-knowledge definition, in that it explicitly states that the trapdoor of the simulator is exactly the verification key vk of the verifier. This strong security property guarantees that the same common reference string can be used for many different proofs, as the same trapdoor is used for simulating all proofs, which enhances the proof system with composability properties. We note that [36] additionally required indistinguishability between real and simulated common reference string; in our constructions, this will be trivially satisfied, as the simulated crs will be exactly the real one. We define below the notion of (bounded) adaptive soundness, which allows the input to be adversarially picked after the public key is fixed.

Definition 4. (Bounded Adaptive Soundness) A NIZK proof system $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for a family of languages $\mathscr{L} = \{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ with relations R_{crs} satisfies the bounded adaptive soundness property if for $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$, for every adversary $\mathscr{A}$,

$$\Pr\left[\begin{matrix}(\mathsf{pk},\mathsf{vk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(1^\lambda), \\ (\pi, x) \leftarrow \mathscr{A}(\mathsf{pk})\end{matrix} : x \notin \mathscr{L}_{\mathsf{crs}} \wedge \Pi.\mathsf{Verify}(\mathsf{pk},\mathsf{vk},x,\pi)\right] = \mathrm{negl}(\lambda).$$

Definition 4 is formulated with respect to arbitrary adversaries $\mathscr{A}$, which leads to a statistical notion of soundness. A natural relaxation of this requirement is to consider only *efficient* (PPT) adversarial provers. We denote by *computational soundness* this relaxed notion of soundness. Computationally sound proof systems are called *argument systems*.

Unbounded Soundness.

Definition 4 corresponds to a *bounded* notion of soundness, in the sense that soundness is only guaranteed to hold when the prover tries to forge a single proof of a wrong statement, right after the setup phase. However, if the prover is allowed to interact polynomially many times with the verifier before trying to forge a proof, sending proofs and receiving feedback on whether the proof was accepted, the previous definition provides no security guarantees.

Intuitively, in this situation, the distinction between bounded and unbounded soundness is comparable to the distinction between security against chosen plaintext attacks and security against chosen ciphertext attacks for cryptosystems. We define unbounded soundness in a similar fashion, by giving the prover access to a verification oracle $\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]$ (with crs implicitly given as parameter) which, on input (x,π), returns $b \leftarrow \mathsf{Verify}(\mathsf{pk},\mathsf{vk},x,\pi)$.

Definition 5. (Q-bounded Adaptive Soundness) A NIZK proof system $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for a family of languages $\mathscr{L} = \{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ with relations R_{crs} satisfies the Q-bounded adaptive soundness property if for $\mathsf{crs} \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda)$, and every adversary $\mathscr{A}$ making at most Q queries to $\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]$, it holds that

$$\Pr\left[\begin{matrix}(\mathsf{pk},\mathsf{vk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(1^\lambda), \\ (\pi, x) \leftarrow \mathscr{A}^{\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]}(\mathsf{pk})\end{matrix} : x \notin \mathscr{L}_{\mathsf{crs}} \wedge \Pi.\mathsf{Verify}(\mathsf{pk},\mathsf{vk},x,\pi)\right] = \mathrm{negl}(\lambda).$$

Alternatively, the above definition can be formulated with respect to polynomial-time adversarial provers, leading to computational Q-bounded adaptive soundness. Note that the answers of the oracle are bits; therefore, if a NIZK proof system satisfies the bounded adaptive soundness property of Definition 4, it also satisfies the above Q-bounded adaptive soundness property for any $Q = O(\log \lambda)$. Indeed, if Q is logarithmic, one can always guess in advance the answers of the verification oracle with non-negligible (inverse polynomial) probability. We say that a NIZK proof system which is Q-bounded adaptively sound for any $Q = \mathsf{poly}(\lambda)$ satisfies *unbounded adaptive soundness*.

Eventually, we define (unbounded) *knowledge-extractability*, a strenghtening of the soundness property which guarantees that if the prover produces an accepting proof, then the simulator can actually *extract* a witness for the statement. To this aim, we extend the syntax of the Setup algorithm to also output a trapdoor τ, used by the extractor. The knowledge-extractibility guarantee is stronger than soundness, in that the proof guarantees not only that there exists a witness, but also that the prover must know that witness. A NIZK satisfying knowledge-extractability is called a NIZK proof of knowledge.

Definition 6. (Q-bounded Knowledge-Extractability) A NIZK proof system $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for a family of languages $\mathscr{L} = \{\mathscr{L}_{\mathsf{crs}}\}_{\mathsf{crs}}$ with relations R_{crs} satisfies the Q-bounded

knowledge-extractability property if for $(\mathsf{crs}, \tau) \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda)$, and every adversary $\mathscr{A}$ making at most Q queries to $\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]$, there is an efficient extractor Ext such that

$$\Pr\left[\begin{array}{ll}(\mathsf{pk}, \mathsf{vk}) \xleftarrow{\$} \Pi.\mathsf{KeyGen}(1^\lambda), & \\ (\pi, x) \leftarrow \mathscr{A}^{\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]}(\mathsf{pk}), & : R_{\mathsf{crs}}(x, w) \text{ iff } \Pi.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, x, \pi) \\ w \leftarrow \mathsf{Ext}(\pi, x, \tau), & \end{array}\right] \approx 1.$$

4.3 A Framework for Designated-Verifier Non-Interactive Zero-Knowledge Proofs of Knowledge

In this section, we let k be an integer, $(\mathbb{G}, \boldsymbol{+})$ be an abelian group of order k, and (α, β, γ) be three integers. We will describe a framework for proving statements of knowledge over a wide variety of algebraic relations over $\mathbb{G}$, in the spirit of the Groth-Sahai framework for NIZK proofs over bilinear groups. To describe the relations handled by our framework, we describe languages of algebraic relations via linear maps. While this system was previously used to describe membership statements [7–9], we adapt it to statements of knowledge. As previously observed in [7], this system encompasses a wider class of languages than the Groth-Sahai framework.

4.3.1 Statements Defined by a Linear Map over $\mathbb{G}$

Let $\mathbf{G} \in \mathbb{G}^\alpha$ denote a vector of *public parameters*, and let $\mathbf{C} \in \mathbb{G}^\beta$ denote a public *word*. We will consider statements $\mathsf{St}_\Gamma(\mathbf{G}, \mathbf{C})$ defined by a linear map $\Gamma : (\mathbb{G}^\alpha, \mathbb{G}^\beta) \mapsto \mathbb{G}^{\gamma \times \beta}$ as follows:

$$\mathsf{St}_\Gamma(\mathbf{G}, \mathbf{C}) = \text{ג}\{\mathbf{x} \in \mathbb{Z}_k^\gamma \mid \mathbf{x} \bullet \Gamma(\mathbf{G}, \mathbf{C}) = \mathbf{C}\}$$

That is, the prover knows a witness-vector $\mathbf{x} \in \mathbb{Z}_k^\gamma$ such that the equation $\mathbf{x} \bullet \Gamma(\mathbf{G}, \mathbf{C}) = \mathbf{C}$ holds. This abstraction captures a wide class of statements. Below, we describe two examples of statements that can be handled by our framework. They aim at clarifying the way the framework can be used, illustrating its power, as well as providing useful concrete instantiations. The examples focus on the most standard primitives (Pedersen commitments, ElGamal ciphertexts), but the reader will easily recognize they can be naturaly generalized to all standard variants of these primitives (e.g., variants of ElGamal secure under t-linear assumptions [11], or under assumptions from the matrix Diffie-Hellman family of assumptions [29]).

Example 1: Knowledge of Opening to a Pedersen Commitment.

We consider statements of knowledge of an opening (m, r) to a Pedersen commitment C.

- Public Parameters: $(G, H) \in \mathbb{G}^2$;
- Word: $C \in \mathbb{G}$;
- Witness: a pair $(m, r) \in \mathbb{Z}_k^2$ such that $C = m \bullet G \boldsymbol{+} r \bullet H$;
- Linear Map: $\Gamma_{\mathsf{Ped}} : (G, H, C) \mapsto (G, H)^\intercal$;
- Statement: $\mathsf{St}_{\Gamma_{\mathsf{Ped}}}(G, H, C) = \text{ג}\{(m, r) \in \mathbb{Z}_k^2 \mid (m, r) \bullet (G, H)^\intercal = C\}$.

Example 2: Multiplicative Relationship Between ElGamal Ciphertexts.

This type of statement is of particular interest, as it can be generalized to arbitrary (polynomial) relationships between plaintexts.

- Public Parameters: $(G, H) \in \mathbb{G}^2$;
- Word: $\mathbf{C} = ((U_i, V_i)_{0 \le i \le 2}) \in \mathbb{G}^6$;
- Witness: a 5-tuple $\mathbf{x} = (m_0, r_0, m_1, r_1, r_2) \in \mathbb{Z}_k^5$ such that $U_i = r_i \bullet G$ and $V_i = m_i \bullet G + r \bullet H$ for $i = 0, 1$, and $U_2 = m_1 \bullet U_0 + r_2 \bullet G$, $V_2 = m_1 \bullet V_0 + r_2 \bullet H$;
- Linear Map:
$$\Gamma_{\mathsf{EM}} : (G, H, \mathbf{C}) \mapsto \begin{pmatrix} 0 & G & 0 & 0 & 0 & 0 \\ G & H & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & G & U_0 & V_0 \\ 0 & 0 & G & H & 0 & 0 \\ 0 & 0 & 0 & 0 & G & H \end{pmatrix};$$
- Statement: $\mathsf{St}_{\Gamma_{\mathsf{EM}}}(G, H, \mathbf{C}) = \aleph\{\mathbf{x} \in \mathbb{Z}_k^5 \mid \mathbf{x} \bullet \Gamma_{\mathsf{EM}}(G, H, \mathbf{C}) = \mathbf{C}\}$.

Conjunction of Statements.

The above framework naturally handles conjuctions. Consider two statements $(\mathsf{St}_{\Gamma_0}(\mathbf{G_0}, \mathbf{C_0}), \mathsf{St}_{\Gamma_1}(\mathbf{G_1}, \mathbf{C_1}))$, defined by linear maps (Γ_0, Γ_1), with public parameters $(\mathbf{G_1}, \mathbf{G_1})$, words $(\mathbf{C_0}, \mathbf{C_1})$, and witnesses $(\mathbf{x_0}, \mathbf{x_1})$. Let $\mathbf{G} \leftarrow (\mathbf{G_1}, \mathbf{G_1})$, $\mathbf{C} \leftarrow (\mathbf{C_0}, \mathbf{C_1})$, and $\mathbf{x} \leftarrow (\mathbf{x_0}, \mathbf{x_1})$. We construct the linear map Γ associated to $\mathsf{St}_\Gamma(\mathbf{G}, \mathbf{C})$ as $\Gamma \leftarrow ((\Gamma_0, 0)^\intercal, (0, \Gamma_1)^\intercal)$. One can immediatly observe that $\mathsf{St}_\Gamma(\mathbf{G}, \mathbf{C}) = \mathsf{St}_{\Gamma_0}(\mathbf{G_0}, \mathbf{C_0}) \wedge \mathsf{St}_{\Gamma_1}(\mathbf{G_1}, \mathbf{C_1})$. The framework handles disjunction of statements as well, as observed in [1]; we omit the details.

4.3.2 A Framework for DVNIZK Proofs of Knowledge

We now introduce our framework for constructing designated-verifier non-interactive zero-knowledge proofs of knowledge for statements defined by a linear map over $\mathbb{G}$. Let $S = (S.\mathsf{KeyGen}, S.\mathsf{Enc}, S.\mathsf{Dec})$ denote a DVNIZK-friendly encryption scheme with plaintext space $\mathbb{Z}_k$. We construct a DVNIZK of knowledge $\Pi_\mathsf{K} = (\Pi_\mathsf{K}.\mathsf{Setup}, \Pi_\mathsf{K}.\mathsf{KeyGen}, \Pi_\mathsf{K}.\mathsf{Prove}, \Pi_\mathsf{K}.\mathsf{Verify})$ for a statement $\mathsf{St}_\Gamma(\mathbf{G}, \mathbf{C})$ over a word $\mathbf{C} \in \mathbb{G}^\beta$, with public parameters $\mathbf{G} \in \mathbb{G}^\alpha$, defined by a linear map $\Gamma : (\mathbb{G}^\alpha, \mathbb{G}^\beta) \mapsto \mathbb{G}^{\gamma \times \beta}$. Our construction proceeds as follows:

- $\Pi_\mathsf{K}.\mathsf{Setup}(1^\lambda)$: compute $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^\lambda)$. Output $\mathsf{crs} \leftarrow \mathsf{ek}$. Note that ek defines a plaintext space $\mathbb{Z}_k$ and a random source $\mathbb{Z}_R$. As the IND-CPA and strong additive properties of S require R to be unknown, we assume that a bound B on R is publicly available. We denote $\ell \leftarrow 2^\lambda k B$.
- $\Pi_\mathsf{K}.\mathsf{KeyGen}(1^\lambda)$: pick $e \leftarrow \mathbb{Z}_\ell$, set $\mathsf{pk} \leftarrow S.\mathsf{Enc}_\mathsf{ek}(0; e)$ and $\mathsf{vk} \leftarrow e$.
- $\Pi_\mathsf{K}.\mathsf{Prove}(\mathsf{pk}, \mathbf{C}, \mathbf{x})$: on a word $\mathbf{C} \in \mathbb{Z}_k^\beta$, with witness $\mathbf{x}$ for the statement $\mathsf{St}_\Gamma(\mathbf{G}, \mathbf{C})$, pick $\mathbf{x}' \xleftarrow{\$} \mathbb{Z}_k^\gamma$, $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_{2^\lambda B}^\gamma$, compute
$$\mathbf{X} \leftarrow S.\mathsf{Enc}_\mathsf{ek}(\mathbf{x}, \mathbf{r}), \qquad \mathbf{X}' \leftarrow S.\mathsf{Enc}_\mathsf{ek}(\mathbf{x}', 0) \ominus (\mathbf{r} \odot \mathsf{pk}), \qquad \mathbf{C}' \leftarrow \mathbf{x}' \bullet \Gamma(\mathbf{G}, \mathbf{C}),$$
and output $\text{ß} \leftarrow (\mathbf{X}, \mathbf{X}', \mathbf{C}')$.

- $\Pi_{\mathsf{K}}.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, \mathbf{C}, \text{ß})$: parse ß as $(\mathbf{X}, \mathbf{X}', \mathbf{C}')$. Check that $e \odot \mathbf{X} \oplus \mathbf{X}'$ is decodable, and decode it to a vector $\mathbf{d} \in \mathbb{Z}_k^\gamma$. Check that

$$\mathbf{d} \bullet \Gamma(\mathbf{G}, \mathbf{C}) = e \bullet \mathbf{C} \mathbf{+} \mathbf{C}'.$$

  If all checks succeeded, accept. Otherwise, reject.

The proof ß consists of 2γ ciphertexts of S, and β elements of $\mathbb{G}$. Below, we illustrate our construction of DVNIZK on the examples of statements given in the previous section. For the sake of concreteness, we instantiate the DVNIZK-friendly encryption scheme S with Paillier (hence the operation $\mathbf{+}$ is instantiated as the multiplication modulo n^2), so that the message space is $\mathbb{Z}_n$ and the randomizer space is $\mathbb{Z}_{\varphi(n)/2}$ for an RSA modulus n. In the examples, we use a bound $B = n$ and draw Paillier random coins from $\mathbb{Z}_{2^\lambda B}$, following our generic framework. However, observe that in the case of Paillier, we can also draw the coins from $\mathbb{Z}_{n/2}$ to get a distribution statistically close to uniform over $\mathbb{Z}_{\varphi(n)/2}$, which is more efficient.

Example 1: Knowledge of Opening to a Pedersen Commitment.

- $\Pi_{\mathsf{Ped}}.\mathsf{Setup}(1^\lambda)$: Compute $((n, h), \delta) = (\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^\lambda)$. Output $\mathsf{crs} \leftarrow \mathsf{ek}$. Let $\ell \leftarrow 2^\lambda n^2$. Let $\mathbb{G} \xleftarrow{\$} \mathsf{GGen}(1^\lambda, n)$, $(G, H) \xleftarrow{\$} \mathbb{G}^2$.
- $\Pi_{\mathsf{Ped}}.\mathsf{KeyGen}(1^\lambda)$: pick $e \xleftarrow{\$} \mathbb{Z}_\ell$, set $\mathsf{pk} \leftarrow h^e \bmod n^2$ and $\mathsf{vk} \leftarrow e$.
- $\Pi_{\mathsf{Ped}}.\mathsf{Prove}(\mathsf{pk}, C, \mathbf{x})$: on a word $C \in \mathbb{G}$, with witness $\mathbf{x} = (m, r) \in \mathbb{Z}_n^2$ for the statement $\mathsf{St}_{\Gamma_{\mathsf{Ped}}}(\mathbf{G}, \mathbf{C})$, pick $\mathbf{x}' \xleftarrow{\$} \mathbb{Z}_n^2$, $\text{æ} \xleftarrow{\$} \mathbb{Z}_{2^\lambda B}^2$, compute $\mathbf{X} \leftarrow (1+n)^{\mathbf{x}} h^{\text{æ}} \bmod n^2, \mathbf{X}' \leftarrow (1+n)^{\mathbf{x}'} \mathsf{pk}^{-\text{æ}} \bmod n^2, \mathbf{C}' \leftarrow \mathbf{x}' \bullet (G, H)^\intercal$, and output ß $\leftarrow (\mathbf{X}, \mathbf{X}', \mathbf{C}')$.
- $\Pi_{\mathsf{Ped}}.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, \mathbf{C}, \text{ß})$: parse ß as $(\mathbf{X}, \mathbf{X}', \mathbf{C}')$. Check that $\mathbf{X}^e \mathbf{X}'$ is of the form $(1+n)^{\mathbf{d}}$, and recover the vector $\mathbf{d} \in \mathbb{Z}_n^2$. Check that $\mathbf{d} \bullet (G, H)^\intercal = e \bullet \mathbf{C} \mathbf{+} \mathbf{C}'$.

Example 2: Multiplicative Relationship Between ElGamal Ciphertexts.

- $\Pi_{\mathsf{EM}}.\mathsf{Setup}(1^\lambda)$ as $\Pi_{\mathsf{Ped}}.\mathsf{Setup}(1^\lambda)$.
- $\Pi_{\mathsf{EM}}.\mathsf{KeyGen}(1^\lambda)$ as $\Pi_{\mathsf{Ped}}.\mathsf{KeyGen}(1^\lambda)$.
- $\Pi_{\mathsf{EM}}.\mathsf{Prove}(\mathsf{pk}, \mathbf{C}, \mathbf{x})$: on a word $\mathbf{C} \in \mathbb{G}^6$, with witness $\mathbf{x} = (m_0, r_0, m_1, r_1, r_2) \in \mathbb{Z}_n^5$ for the statement $\mathsf{St}_{\Gamma_{\mathsf{EM}}}(\mathbf{G}, \mathbf{C})$, pick $\mathbf{x}' \xleftarrow{\$} \mathbb{Z}_n^5$, $\text{æ} \xleftarrow{\$} \mathbb{Z}_{2^\lambda B}^5$, compute $\mathbf{X} \leftarrow (1+n)^{\mathbf{x}} h^{\text{æ}} \bmod n^2, \mathbf{X}' \leftarrow (1+n)^{\mathbf{x}} \mathsf{pk}^{-\text{æ}} \bmod n^2, \mathbf{C}' \leftarrow \mathbf{x}' \bullet \Gamma_{\mathsf{EM}}(\mathbf{G}, \mathbf{C})$, and output ß $\leftarrow (\mathbf{X}, \mathbf{X}', \mathbf{C}')$.
- $\Pi_{\mathsf{EM}}.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, \mathbf{C}, \text{ß})$: parse ß as $(\mathbf{X}, \mathbf{X}', \mathbf{C}')$. Check that $\mathbf{X}^e \mathbf{X}'$ is of the form $(1+n)^{\mathbf{d}}$, and recover the vector $\mathbf{d} \in \mathbb{Z}_n^5$. Check that $\mathbf{d} \bullet \Gamma_{\mathsf{EM}}(\mathbf{G}, \mathbf{C}) = e \bullet \mathbf{C} \mathbf{+} \mathbf{C}'$.

4.3.3 Security Proof

We now prove the generic DVNIZK construction from Section 4.3.2 is secure.

Perfect Completeness.

It follows from straighforward calculations: $e \odot \mathbf{X} \oplus \mathbf{X}' = S.\mathsf{Enc}_{\mathsf{ek}}(e \cdot \mathbf{x} + \mathbf{x}'; e \cdot \mathbf{r} - e \cdot \mathbf{r}) = S.\mathsf{Enc}_{\mathsf{ek}}(e \cdot \mathbf{x} + \mathbf{x}'; 0)$ is decodable and decodes to $\mathbf{d} = e \cdot \mathbf{x} + \mathbf{x}' \bmod k$. Then, $\mathbf{d} \bullet \Gamma(\mathbf{G}, \mathbf{C}) = e \bullet (\mathbf{x} \bullet \Gamma(\mathbf{G}, \mathbf{C})) \mathbf{+} \mathbf{x}' \bullet \Gamma(\mathbf{G}, \mathbf{C}) = e \bullet \mathbf{C} \mathbf{+} \mathbf{C}'$ by the correctness of the statement $(\mathbf{x} \bullet \Gamma(\mathbf{G}, \mathbf{C}) = \mathbf{C})$ and by construction of $\mathbf{C}'$.

Composable Zero-Knowledge.

We prove the following theorem:

Theorem 1 (Zero-Knowledge of Π_{K})**.** *If the encryption scheme S is* IND-CPA *secure, the* DVNIZK *scheme Π_{K} is composable zero-knowledge.*

We describe a simulator $\mathsf{Sim}(\mathbf{C}, \mathsf{pk}, \mathsf{vk})$ producing proofs computationally indistinguishable from those produced by an honest prover on true statements. The simulator operates as follows: let $\mathbf{d} \xleftarrow{\$} \mathbb{Z}_k^\gamma$, and $\mathbf{C}' \leftarrow \mathbf{d} \bullet \Gamma(\mathbf{G}, \mathbf{C}) - e \bullet \mathbf{C}$. Sample $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_k^\gamma$, $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_{2^\lambda B}^\gamma$, and compute $\mathbf{X} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(\mathbf{x}, \mathbf{r}), \mathbf{X}' \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(\mathbf{d} - \mathbf{e} \cdot \mathbf{x}, -\mathbf{e} \cdot \mathbf{r})$. Output $\pi_s = (\mathbf{X}, \mathbf{X}', \mathbf{C}')$.

Let $\mathscr{A}$ be an adversary that can distinguish Sim from Prove. We will build a reduction against the IND-CPA security of S. The reduction obtains $\mathbf{C}, \mathbf{x}$ from $\mathscr{A}$, samples $\tilde{\mathbf{x}} \leftarrow \mathbb{Z}_k^\gamma$, sends $(\mathbf{x}, \tilde{\mathbf{x}})$ to the IND-CPA game and sets $\mathbf{X}$ to be the challenge from the IND-CPA game. Now, the reduction samples $\mathbf{d} \leftarrow \mathbb{Z}_k^\gamma$ and sets $\mathbf{X}' := S.\mathsf{Enc}_{\mathsf{ek}}(\mathbf{d}; 0) \ominus \mathbf{X} \odot e$. Finally, the reduction sets $\mathbf{C}' := \mathbf{d} \bullet \Gamma(\mathbf{G}, \mathbf{C}) - e \bullet \mathbf{C}$. Send $\pi^* = (\mathbf{X}, \mathbf{X}', \mathbf{C})$ to $\mathscr{A}$.

Direct calculation shows that if the IND-CPA game outputs an encryption of $\tilde{\mathbf{X}}$, then $\mathbf{X}, \mathbf{X}', \mathbf{C}$ are distributed as those produced by Sim, whereas when it outputs an encryption of $\mathbf{X}$ then π^* is distributed identical to a real proof. Thus, whatever advantage $\mathscr{A}$ has in distinguishing Sim from Prove is also achieved by the reduction against IND-CPA. Note that for simplicity, our proof assume that the IND-CPA game is directly played over vectors, but standard methods allow to reduce this to the classical IND-CPA game with a single challenge ciphertext.

Adaptive Unbounded Knowledge-Extractability.

We start by showing that Π_{K} satisfies statistical adaptive unbounded knowledge-extractability. More precisely, we prove the following theorem:

Theorem 2 (Soundness of Π_{K})**.** *There is an efficient simulator* Sim *such that for any (possibly unbounded) adversary $\mathscr{A}$ that outputs an accepting proof $\mathfrak{ß}$ with probability ε on an arbitrary word $\mathbf{C}$ after making at most Q queries to the oracle $\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]$,* Sim *extracts a valid witness for the statement $\mathsf{St}_\Gamma(\mathbf{G}, \mathbf{C})$ with probability at least $\varepsilon - (Q+1)\beta/p_k$, where p_k is the smallest prime factor of k.*

The proof describes an efficient simulator Sim that correctly emulates the verifier, without knowing vk mod k. The simulation is done as follows:

- $\mathsf{Sim.Setup}(1^\lambda)$: compute $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^\lambda)$. Output $\mathsf{crs} \leftarrow \mathsf{ek}$. The encryption key ek defines a plaintext space $\mathbb{Z}_k$ and a random source $\mathbb{Z}_R$ with bound B. Let $\ell \leftarrow 2^\lambda k B$.
- $\mathsf{Sim.KeyGen}(1^\lambda)$: compute $(\mathsf{pk}, \mathsf{vk}) \xleftarrow{\$} \Pi_{\mathsf{K}}.\mathsf{KeyGen}(1^\lambda)$, output pk, store $e_R \leftarrow \mathsf{vk} \bmod R$, and erase vk.
- $\mathsf{Sim.Verify}(\mathsf{pk}, \mathsf{dk}, e_R, \mathbf{C}, \mathfrak{ß})$: parse $\mathfrak{ß}$ as $(\mathbf{X}, \mathbf{X}', \mathbf{C}')$. Using the secret key dk of S, decrypt $\mathbf{X}$ to a vector $\mathbf{x}$, and $\mathbf{X}'$ to a vector $\mathbf{x}'$. Check that $(-e_R) \odot (\mathbf{X} \ominus \mathbf{x}) = \mathbf{X}' \ominus \mathbf{x}'$. Check that $\mathbf{x} \bullet \Gamma(\mathbf{G}, \mathbf{C}) = \mathbf{C}$, and that $\mathbf{x}' \bullet \Gamma(\mathbf{G}, \mathbf{C}) = \mathbf{C}'$. If all checks succeeded, accept. Otherwise, reject.

The simulator Sim first calls $\mathsf{Sim.Setup}(1^\lambda)$ to generate the common reference string (note that our simulator generates the common reference string honestly, hence the simulation of Setup cannot be distinguished from an honest run of Setup), and stores dk. Each time the adversary $\mathscr{A}$ sends a query $(\mathbf{C}, \mathfrak{ß})$ to the oracle $\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]$, Sim simulates $\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]$ (without knowing vk mod k) by running

$\mathsf{Sim.Verify}(\mathsf{pk}, \mathsf{dk}, e_R, \mathbf{C}, \text{ß})$, and accepts or rejects accordingly. When $\mathscr{A}$ outputs a final answer $(\mathbf{C}, \text{ß})$, Sim computes a witness $\mathbf{x}$ for $\mathsf{St}_\Gamma(\mathbf{G}, \mathbf{C})$ by decrypting $\mathbf{C}$ with dk.

Observe that the distribution $\{(\mathsf{pk}, \mathsf{vk}) \xleftarrow{\$} \Pi_\mathsf{K}.\mathsf{KeyGen}(1^\lambda), e_k \leftarrow \mathsf{vk} \bmod k : (\mathsf{pk}, e_k)\}$ is statistically indistinguishable from the distribution $\{(\mathsf{pk}, \mathsf{vk}) \xleftarrow{\$} \Pi_\mathsf{K}.\mathsf{KeyGen}(1^\lambda), e_k \xleftarrow{\$} \mathbb{Z}_k : (\mathsf{pk}, e_k)\}$. Put otherwise, the distribution of $\mathsf{vk} \bmod k$ is statistically indistinguishable from random, even given pk. Indeed, as S is a DVNIZK-friendly encryption scheme, it holds by definition that $\gcd(k, R) = 1$. As $\ell = 2^\lambda Bk \geq 2^\lambda Rk$, the distribution $\{e \xleftarrow{\$} \mathbb{Z}_\ell, e_k \leftarrow e \bmod k, e_R \leftarrow e \bmod R : (e_k, e_R)\}$ is statistically indistinguishable from the uniform distribution over $\mathbb{Z}_k \times \mathbb{Z}_R$, and the value pk only leaks e_R, even to an unbounded adversary (as $S.\mathsf{Enc}_\mathsf{ek}(0; e) = S.\mathsf{Enc}_\mathsf{ek}(0; e \bmod R)$). We now prove the following claim:

Claim 1. *For any public parameters* $\mathbf{G}$ *and word* $\mathbf{C}$*, it holds that*

$$\Pr\left[\begin{array}{ll} (\mathsf{pk}, \mathsf{vk}) \xleftarrow{\$} \Pi_\mathsf{K}.\mathsf{KeyGen}(1^\lambda), & \\ b \leftarrow \mathsf{Sim.Verify}(\mathsf{pk}, \mathsf{dk}, \mathbf{C}, \text{ß}), & : b' = b \\ b' \leftarrow \Pi_\mathsf{K}.\mathsf{Verify}(\mathsf{pk}, \mathsf{vk}, \mathbf{C}, \text{ß}) & \end{array}\right] \geq 1 - \beta/p_k,$$

where p_k *is one of the prime factors of* k*.*

Proof. First, we show that if $b = 1$, then $b' = 1$. Indeed, let us denote $(\mathbf{x}, \mathbf{x}')$ the plaintexts associated to $(\mathbf{X}, \mathbf{X}')$. Let $(\mathbf{r}, \mathbf{r}')$ be the random coins of the ciphertexts $(\mathbf{X}, \mathbf{X}')$. Observe that, by the homomorphic properties of S, the equation $(-e_R) \odot (\mathbf{X} \ominus \mathbf{x}) = \mathbf{X}' \ominus \mathbf{x}'$ is equivalent to $S.\mathsf{Enc}_\mathsf{ek}(0; -e_R \cdot \mathbf{r}) = S.\mathsf{Enc}_\mathsf{ek}(0; \mathbf{r}')$, which is equivalent to $e \odot \mathbf{X} \oplus \mathbf{X}' = S.\mathsf{Enc}(e \cdot \mathbf{x} + \mathbf{x}' \bmod k; e \cdot \mathbf{r} + \mathbf{r}' \bmod R) = S.\mathsf{Enc}(e \cdot \mathbf{x} + \mathbf{x}' \bmod k; 0)$ as $e = e_R \bmod R$. Therefore, the verifier's check that $e \odot \mathbf{X} \oplus \mathbf{X}'$ is decodable succeeds if and only if Sim's first check succeeds, and the decoded value $\mathbf{d} \in \mathbb{Z}_k^\gamma$ satisfies $\mathbf{d} = e \cdot \mathbf{x} + \mathbf{x}' \bmod k$. Moreover, if the equations $\mathbf{x} \bullet \Gamma(\mathbf{G}, \mathbf{C}) = \mathbf{C}$ and $\mathbf{x}' \bullet \Gamma(\mathbf{G}, \mathbf{C}) = \mathbf{C}'$ are both satisfied (*i.e.* Sim's other checks succeed), then it necessarily holds that $\mathbf{d} \bullet \Gamma(\mathbf{G}, \mathbf{C}) = (e \cdot \mathbf{x} + \mathbf{x}') \bullet \Gamma(\mathbf{G}, \mathbf{C}) = e \bullet (\mathbf{x} \bullet \Gamma(\mathbf{G}, \mathbf{C})) \mathbf{+} \mathbf{x}' \bullet \Gamma(\mathbf{G}, \mathbf{C}) = e \bullet \mathbf{C} \mathbf{+} \mathbf{C}'$. This concludes the proof that, conditioned on Sim's checks succeeding, the verifier's checks necessarily succeed.

Now, assume for the sake of contradiction that the converse is not true: suppose that Sim rejected the proof, while the verifier accepted. We already showed that the equation $(-e_R) \odot (\mathbf{X} \ominus \mathbf{x}) = \mathbf{X}' \ominus \mathbf{x}'$ is equivalent to the equation $e \odot \mathbf{X} \oplus \mathbf{X}' = S.\mathsf{Enc}(e \cdot \mathbf{x} + \mathbf{x}' \bmod k; 0)$; therefore, if $e \odot \mathbf{X} \oplus \mathbf{X}'$ is decodable (it has random coin 0), then Sim's check that $(-e_R) \odot (\mathbf{X} \ominus \mathbf{x}) = \mathbf{X}' \ominus \mathbf{x}'$ succeeds. As we assumed that Sim rejects the proof, this means that at least one of Sim's last checks must fail: either $\mathbf{x} \bullet \Gamma(\mathbf{G}, \mathbf{C}) \neq \mathbf{C}$, or $\mathbf{x}' \bullet \Gamma(\mathbf{G}, \mathbf{C}) \neq \mathbf{C}'$. By the first check of the verifier, it holds that $e \odot \mathbf{X} \oplus \mathbf{X}'$ is decodable; denoting $(\mathbf{x}, \mathbf{x}')$ the plaintexts associated to $(\mathbf{X}, \mathbf{X}')$, it therefore decodes to $\mathbf{d} = e \cdot \mathbf{x} + \mathbf{x}' \bmod k$. By the second check of the verifier, it holds that $\mathbf{d} \bullet \Gamma(\mathbf{G}, \mathbf{C}) = e \bullet \mathbf{C} \mathbf{+} \mathbf{C}'$, which implies $e \bullet (\mathbf{x} \bullet \Gamma(\mathbf{G}, \mathbf{C})) \mathbf{+} \mathbf{x}' \bullet \Gamma(\mathbf{G}, \mathbf{C}) = e \bullet \mathbf{C} \mathbf{+} \mathbf{C}'$. This last equation rewrites to

$$e \bullet (\mathbf{x} \bullet \Gamma(\mathbf{G}, \mathbf{C}) \mathbf{-} \mathbf{C}) = \mathbf{C}' \mathbf{-} \mathbf{x}' \bullet \Gamma(\mathbf{G}, \mathbf{C}) \tag{4.1}$$

Now, recall that by assumption, either $\mathbf{x} \bullet \Gamma(\mathbf{G}, \mathbf{C}) \neq \mathbf{C}$, or $\mathbf{x}' \bullet \Gamma(\mathbf{G}, \mathbf{C}) \neq \mathbf{C}'$. Observe that Equation 4.1 further implies, as $e \neq 0$ (with overwhelming probability), that $\mathbf{x}' \bullet \Gamma(\mathbf{G}, \mathbf{C}) \mathbf{-} \mathbf{C}' \neq 0$ if and only if $\mathbf{x} \bullet \Gamma(\mathbf{G}, \mathbf{C}) \mathbf{-} \mathbf{C} \neq 0$. Therefore, conditioned on Sim rejecting the proof, it necessarily holds that $\mathbf{x} \bullet \Gamma(\mathbf{G}, \mathbf{C}) \mathbf{-} \mathbf{C} \neq 0$ and $\mathbf{x}' \bullet \Gamma(\mathbf{G}, \mathbf{C}) \mathbf{-} \mathbf{C}' \neq 0$. Let (μ_i, ν_i) be two non-zero entries of the vectors $(\mathbf{x} \bullet \Gamma(\mathbf{G}, \mathbf{C}) \mathbf{-} \mathbf{C}, \mathbf{C}' \mathbf{-} \mathbf{x}' \bullet \Gamma(\mathbf{G}, \mathbf{C}))$ at the same position $i \leq \beta$; by Equation 4.1, it holds that $e = \nu_i \cdot \mu_i^{-1} \bmod p$ for at least one of the prime factors p of k. However, recall that the value $e \bmod k$ is *statistically hidden* to the prover (and therefore, so is the value $e \bmod p$), hence the probability of this event happening can be upper-bounded by $\beta/p \leq \beta/p_k$. This concludes the proof of the claim. □

Now, consider an adversary $\mathscr{A}$ that outputs an accepting proof $(\mathbf{C}, \text{ß})$ with probability at least ε after a polynomial number Q of interactions with the oracle $\mathcal{O}_{\mathsf{vk}}[\mathsf{pk}]$. By the above claim and a union bound, it necessarily holds that $\mathscr{A}$ outputs an accepting proof $(\mathbf{C}, \text{ß})$ with probability at least $\varepsilon - Q\beta/p_k$ after interacting Q times with $\mathsf{Sim.Verify}(\mathsf{pk}, \mathsf{dk}, e_R, \cdot, \cdot)$; moreover, with probability at least $1 - \beta p_k$, this proof is also accepted by Sim's verification algorithm. Overall, Sim obtains a proof accepted by his verification algorithm with probability at least $\varepsilon - (Q+1)\beta/p_k$. In particular, this implies that the vector $\mathbf{x}$ extracted by Sim from ß satisfies $\mathbf{x} \bullet \Gamma(\mathbf{G}, \mathbf{C}) = \mathbf{C}$ with probability at least $\varepsilon - (Q+1)\beta/p_k$. Therefore, Sim extracts a valid witness for the knowledge statement $\mathsf{St}_\Gamma(\mathbf{G}, \mathbf{C})$ with probability at least $\varepsilon - (Q+1)\beta/p_k$. As the size k of a DVNIZK-friendly cryptosystem has only superpolynomially large prime-factors, it holds that p_k is superpolynomially large. As $(Q+1)\beta$ is polynomial, we conclude that if $\mathscr{A}$ outputs an accepting proof with non-negligible probability, then Sim extracts a valid witness with non-negligible probability.

4.4 Dual Variant of the Framework

In the previous section, we described a framework for constructing efficient DVNIZKs of knowledge for relations between words defined over an abelian group $(\mathbb{G}, +)$, using a cryptosystem with specific properties as the underlying commitment scheme for the proof system. In this section, we show that the framework can also be used in a dual way, by considering languages of relations between the plaintexts of the underlying encryption scheme – we call this variant 'dual variant' of the framework, as the roles of the underlying encryption scheme (which is used as a commitment scheme for the proof) and of the abelian group (which contains the words on which the proof is made) are partially exchanged. This allows for example to handle languages of relations between Paillier ciphertexts. To instantiate the framework, it suffices to have any perfectly binding commitment scheme defined over $\mathbb{G}$. This dual variant leads to efficient DVNIZK proofs for relations between, e.g., Paillier ciphertexts, whose zero-knowledge property reduces to the binding property of the commitment scheme over $\mathbb{G}$ (e.g. the DDH assumption, or its variants), and with statistical (unbounded, adaptive) soundness.

4.4.1 Perfectly Binding Commitment over $\mathbb{G}$

Suppose that we are given a perfectly binding homomorphic commitment $C = (C.\mathsf{Setup}, C.\mathsf{Com}, C.\mathsf{Verify})$, where $C.\mathsf{Com} : \mathbb{Z}_k \times \mathbb{Z}_k \mapsto \mathbb{G}^*$. Assume further that $C.\mathsf{Setup}$ generates a public vector of parameters $\mathbf{G} \in \mathbb{G}^*$, and that there is a linear map Γ_C associated to this commitment such that for all $(m, r) \in \mathbb{Z}_k^2$, $C.\mathsf{Com}(m, r) = (m, r) \bullet \Gamma_C(\mathbf{G})$. Note this implies the commitment scheme is homomorphic over $\mathbb{G}$. ElGamal (Sect. 4.2.1), can be used as a commitment scheme satisfying these properties, is hiding under the DDH assumption and perfectly binding. We do so by using $\mathsf{KeyGen}(1^\lambda)$ in place of $\mathsf{Setup}(1^\lambda)$ to generate group elements (G, H) (the public key of the encryption scheme), and commit (i.e encrypt) via $\Gamma_C(G, H) = ((0, G)^\intercal, (G, H)^\intercal)$. We generalize this to commitments to length-t vectors as follow: we let $\Gamma_{C,t}$ denote the extended matrix such that $C.\mathsf{Com}(\mathbf{m}, \mathbf{r}) = (\mathbf{m}, \mathbf{r}) \bullet \Gamma_{C,t}(\mathbf{G})$, where $(\mathbf{m}, \mathbf{r})$ are vectors of length t ($\Gamma_{C,t}$ is simply the block-diagonal matrix whose t blocks are all equal to Γ_C). Consider now the following statement, where the word is a vector $\mathbf{C}$ of commitments:

$$\begin{aligned}\mathsf{St}_{\Gamma_{C,t}}(\mathbf{G}, \mathbf{C}) &= \aleph\{(\mathbf{m}, \mathbf{r}) \mid (\mathbf{m}, \mathbf{r}) \bullet \Gamma_{C,t}(\mathbf{G}) = \mathbf{C}\} \\ &= \aleph\{(\mathbf{m}, \mathbf{r}) \mid C.\mathsf{Com}(\mathbf{m}, \mathbf{r}) = \mathbf{C}\}.\end{aligned}$$

One can immediatly observe that this statement (which is a proof of knowledge of openings to a vector of commitments with C) is handled by the framework of Section 4.3.

4.4.2 Equality of Plaintexts between C and S

In this section, we describe a simple method to convert a DVNIZK on the statement $\mathsf{St}_{\Gamma_{C,t}}(\mathbf{G},\mathbf{C}) = \lambda\{(\mathbf{m},\mathbf{r}) \mid C.\mathsf{Com}(\mathbf{m},\mathbf{r}) = \mathbf{C}\}$ into a DVNIZK on the statement $\mathsf{St}'(\mathbf{G},\mathbf{C},\mathbf{X_m}) = \exists\{(\mathbf{m},\mathbf{æ_m},\mathbf{r}) \mid \mathbf{X_m} = S.\mathsf{Enc}_{\mathsf{ek}}(\mathbf{m},\mathbf{æ_m}) \wedge \mathbf{C} = C.\mathsf{Com}(\mathbf{m},\mathbf{r})\}$ for a length-t vector $\mathbf{C}$ of commitments with a commitment scheme over $\mathbb{G}$ satisfying the requirements defined in the previous section, and a length-t vector of DVNIZK-friendly ciphertexts $\mathbf{X_m}$. Instantiating the framework of Section 4.3 for the statement $\mathsf{St}_{\Gamma_{C,t}}(\mathbf{G},\mathbf{C})$, we get the following DVNIZK Π:

- $\Pi.\mathsf{Setup}(1^\lambda)$: compute $(\mathsf{ek},\mathsf{dk}) \xleftarrow{\$} S.\mathsf{KeyGen}(1^\lambda)$. Output $\mathsf{crs} \leftarrow \mathsf{ek}$. Note that ek defines the plaintext space $\mathbb{Z}_k$ and the random source $\mathbb{Z}_R$ with bound B. We denote $\ell \leftarrow 2^\lambda kB$.

- $\Pi.\mathsf{KeyGen}(1^\lambda)$: pick $e \leftarrow \mathbb{Z}_\ell$, set $\mathsf{pk} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(0;e)$ and $\mathsf{vk} \leftarrow e$.

- $\Pi.\mathsf{Prove}(\mathsf{pk},\mathbf{C},(\mathbf{m},\mathbf{r}))$: on a word $\mathbf{C} \in \mathbb{Z}_k^t$, with witness $(\mathbf{m},\mathbf{r})$ for the statement $\mathsf{St}_{\Gamma_{C,t}}(\mathbf{G},\mathbf{C})$ (where $\mathbf{G} \xleftarrow{\$} C.\mathsf{Setup}(1^\lambda)$), pick random $(\mathbf{m}',\mathbf{r}')$, random coins $(\mathbf{æ_m},\mathbf{æ_r})$ for S, and compute

$$\begin{aligned}
&\mathbf{X_m} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(\mathbf{m},\mathbf{æ_m}), && \mathbf{X_r} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(\mathbf{r},\mathbf{æ_r}),\\
&\mathbf{X'_m} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(\mathbf{m}',0) \ominus (\mathbf{æ_m} \odot \mathsf{pk}), && \mathbf{X'_r} \leftarrow S.\mathsf{Enc}_{\mathsf{ek}}(\mathbf{r}',0) \ominus (\mathbf{æ_r} \odot \mathsf{pk}),\\
&\mathbf{C}' \leftarrow (\mathbf{m}',\mathbf{r}') \bullet \Gamma_{C,t}(\mathbf{G},\mathbf{C}),
\end{aligned}$$

  and output $\mathbf{ß} \leftarrow (\mathbf{X_m},\mathbf{X'_m},\mathbf{X_r},\mathbf{X'_r},\mathbf{C}')$.

- $\Pi_K.\mathsf{Verify}(\mathsf{pk},\mathsf{vk},\mathbf{C},\mathbf{ß})$: parse $\mathbf{ß}$ as $(\mathbf{X_m},\mathbf{X'_m},\mathbf{X_r},\mathbf{X'_r},\mathbf{C}')$. Check that $e \odot \mathbf{X_m} \oplus \mathbf{X'_m}$ and $e \odot \mathbf{X_r} \oplus \mathbf{X'_r}$ are decodable, and decode them to vectors $(\mathbf{d_m},\mathbf{d_r}) \in (\mathbb{Z}_k^t)^2$. Check that $(\mathbf{d_m},\mathbf{d_r}) \bullet \Gamma_{C,t}(\mathbf{G},\mathbf{C}) = e \bullet \mathbf{C}+\mathbf{C}'$.

By the result of Section 4.3, this is an unbounded statistical adaptive knowledge-extractable DVNIZK of knowledge of an opening for C. Suppose now that we modify the above scheme as follow: we let $\mathbf{X_m}$ be part of the *word* on which the proof is executed, rather than being computed as part of the *proof* by the algorithm $\Pi.\mathsf{Prove}$. That is, we consider words of the form $(\mathbf{C},\mathbf{X_m})$ with witness $(\mathbf{m},\mathbf{r},\mathbf{æ_m})$ such that $(\mathbf{C},\mathbf{X_m}) = (C.\mathsf{Com}(\mathbf{m};\mathbf{r}), S.\mathsf{Enc}_{\mathsf{ek}}(\mathbf{m},\mathbf{æ_m}))$. Let Π' denote the modified proof, in which $\mathbf{X_m}$ is part of the word and $(\mathbf{X'_m},\mathbf{X_r},\mathbf{X'_r},\mathbf{C}')$ are computed as in Π. Observe that the proof of security of our framework immediatly implies that Π' is a secure DVNIZK for *plaintext equality* between commitments with C and encryptions with S: our statistical argument shows that a (possibly unbounded) adversary has negligible probability of outputting a word $\mathbf{C}$ together with an accepting proof $\mathbf{ß} = (\mathbf{X_m},\mathbf{X'_m},\mathbf{X_r},\mathbf{X'_r},\mathbf{C}')$ where the plaintext extracted by the simulator from $\mathbf{X_m}$ is not also the plaintext of $\mathbf{C}$. Hence, it is trivial that the probability of outputting a word $(\mathbf{C},\mathbf{X_m})$ and an accepting proof $\mathbf{ß}' = (\mathbf{X'_m},\mathbf{X_r},\mathbf{X'_r},\mathbf{C}')$ where the plaintext extracted by the simulator from $\mathbf{X_m}$ is not also the plaintext of $\mathbf{C}$ is also negligible. Thus, we get:

Theorem 3. *The proof system Π' is an adaptive unbounded statistically sound proof for equality between plaintexts of C and plaintexts of S, whose composable zero-knowledge property reduces to the* IND-CPA *security of S.*

Note that the proof Π' is no longer a proof of knowledge: while the simulator can extract $(\mathbf{m},\mathbf{r})$ from the prover, he cannot necessarily extract the random coins $\mathbf{æ_m}$ of $\mathbf{X_m}$, which are now part of the witness. Therefore, for the protocol to make sense, it is important that C is perfectly binding .

4.4.3 A Framework for Relations between Plaintexts of S

The observations of the above section suggest a very natural way for designing DVNIZKs for relations between plaintexts $\mathbf{m} \in \mathbb{Z}_k^*$ of the encryption scheme S, which intuitively operates in two steps: first, we create commitments to the plaintexts $\mathbf{m}$ over $\mathbb{G}$ using C and prove them consistent with the encrypted values using the method described in the previous section. Then, we are able to use the framework of Section 4.3 to demonstrate the desired relation holds between the commited values (this is a statement naturally captured by the framework). More formally, on input a vector of ciphertexts $\mathbf{X_m}$ encrypting plaintexts $\mathbf{m}$ with random coins $\mathbf{æ_m}$,

- Pick $\mathbf{r}$ and compute $\mathbf{C} \leftarrow C.\mathsf{Com}(\mathbf{m}, \mathbf{r})$.
- Construct a DVNIZK for the statement $\mathsf{St}'(\mathbf{G}, \mathbf{C}, \mathbf{X_m})$ with witness $(\mathbf{m}, \mathbf{æ_m}, \mathbf{r})$, using the method described in Section 4.4.2.
- Construct a DVNIZK for the statement $\mathsf{St}_\Gamma(\mathbf{G}, \mathbf{C})$ with witness $(\mathbf{m}, \mathbf{r})$, using the framework of Section 4.3.

The correctness of this approach is immediate: the second DVNIZK guarantees that the appropriate relation is satisfied between the plaintexts of the commitments, while the first one guarantees that the ciphertexts indeed encrypt the committed values. This leads to a DVNIZK proof of relation between plaintexts of S, with unbounded adaptive statistical soundness. Regarding zero-knowledge, as the proof starts by committing to $\mathbf{m}$ with C, we must in addition assume that the commitment scheme is hiding (the security analysis is straightforward).

Theorem 4. *The above system is an adaptive unbounded statistically sound proof for relations of plaintexts of S, whose composable zero-knowledge reduces to the* IND-CPA *security of S and the hiding property of C.*

We note that we can also obtain a variant of Theorem 4, where zero-knowledge only relies on the IND-CPA of S, and hiding of C implies the soundness property, using commitment schemes *a la* Groth-Sahai where the crs can be generated in two indistinguishable ways, one leading to a perfectly hiding scheme, and one leading to a perfectly binding scheme (such commitments are known, e.g., from the DDH assumption).

Example: Multiplicative Relationship Between Paillier Ciphertexts.

We focus now on the useful case of multiplicative relationship between plaintexts of Paillier ciphertexts. We instantiate S with the Paillier encryption scheme over an RSA group $\mathbb{Z}_n$, with a public key (n, h) ($h = g^n \bmod n^2$ for a generator g of $\mathbb{J}_n$), and the commitment scheme C with the ElGamal encryption scheme over a group $\mathbb{G}$ of order n, with public key (G, H). Let $(P_0, P_1, P_2) \in (\mathbb{Z}_{n^2}^*)^3$ be three Paillier ciphertexts, and let $(m_0, m_1, m_2, \rho_0, \rho_1, \rho_2)$ be such that $m_2 = m_0 m_1 \bmod n$, and $P_0 = (1+n)^{m_0} h^{\rho_0} \bmod n^2, P_1 = (1+n)^{m_1} h^{\rho_1} \bmod n^2, P_2 = (1+n)^{m_2} h^{\rho_2} \bmod n^2$. Let $E = h^e \bmod n^2$ denote the public key of the verifier. The designated-verifier NIZK for proving that P_2 encrypts $m_0 m_1$ proceeds as follows:

- **Committing over $\mathbb{G}$:** pick (r_0, r_1, r_2) and send $(U_i, V_i)_{0 \le i \le 2} \leftarrow (r_i \bullet G, r_i \bullet H + m_i \bullet G)_{0 \le i \le 2}$ (which are commitments with ElGamal to (m_0, m_1, m_2) over $\mathbb{G}$).
- **Proof of Plaintext Equality:** pick $(m'_i, r'_i, \rho'_i)_{0 \le i \le 2} \xleftarrow{\$} (\mathbb{Z}_n \times \mathbb{Z}_n \times \mathbb{Z}_{n/2})^3$, and send for $i = 0$ to 2, $X_i \leftarrow (1+n)^{r_i} h^{\rho'_i} \bmod n^2, X'_i \leftarrow (1+n)^{r'_i} E^{-\rho'_i} \bmod n^2, P'_i \leftarrow (1+n)^{m'_i} E^{-\rho_i} \bmod n^2$, and $(U'_i, V'_i) \leftarrow (r'_i \bullet G, r'_i \bullet H + m'_i \bullet G)$.

- **Proof of Multiplicative Relationship Between the Committed Values:** apply the proof system of Example 2 from Section 4.3 to the word $(U_i, V_i)_{0 \leq i \leq 2}$, with public parameters (G, H), and the witness $\mathbf{x} = (m_0, r_0, m_1, r_1, r_2 - r_0 m_1)$ which satisfies $(U_0, V_0) = (r_0 \bullet G, r_0 \bullet H \mathbin{+} m_0 \bullet G)$, $(U_1, V_1) = (r_1 \bullet G, r_1 \bullet H \mathbin{+} m_1 \bullet G)$, and $(U_2, V_2) = ((r_2 - r_0 m_1) \bullet G \mathbin{+} m_1 \bullet U_0, (r_2 - r_0 m_1) \bullet H \mathbin{+} m_1 \bullet V_0)$.
- **Proof Verification:** upon receving $(U_i, V_i, X_i, X'_i, P'_i, U'_i, V'_i)_{0 \leq i \leq 2}$ together with the proof of multiplicative relationship between the values committed with $(U_i, V_i)_i$, the verifier with verification key $\mathsf{vk} = e$ checks that $e \odot P_i \oplus P'_i$ and $e \odot X_i \oplus X'_i$ successfully decode (respectively) to values p_i, x_i, and that $e \bullet U_i \mathbin{+} U'_i = x_i \bullet G$ and $e \bullet V_i \mathbin{+} V'_i = x_i \bullet H \mathbin{+} p_i \bullet G$, for $i = 0$ to 2. The verifier additionally checks the multiplicative proof, as in Example 4 from Section 4.3. She accepts iff all checks succeed.

The proof for the multiplicative statement involves 10 Paillier ciphertexts and 3 ElGamal ciphertexts. Overall, the total proof involves 20 Paillier ciphertexts, and 9 ElGamal ciphertexts. However, this size is obtained by applying the framework naively; in this situation, it introduces a lot of redudancy. For instance, instead of computing Paillier encryptions of (m_0, r_0, m_1, r_1) in the third phase, one can simply reuse the word (P_0, P_1) and the ciphertexts (X_0, X_1), as well as reusing $(P'_i, X'_i)_i$ for the corresponding masks $(m'_i, r'_i)_i$, saving 8 Paillier ciphertexts; similar savings can be obtained for the ElGamal ciphertexts, leading to a proof of total size 12 Paillier ciphertexts + 7 ElGamal ciphertexts.

Furthermore, if we eschew unbounded soundness and accept bounds on m_i we are able to produce a much shorter proof, comprising only two Paillier ciphertexts, outperforming even Fiat-Shamir. We detail this in the full version [16].

Bibliography

[1] M. Abdalla, F. Benhamouda, and D. Pointcheval. Disjunctions for hash proof systems: New constructions and applications. LNCS, pages 69–100. Springer, 2015.

[2] B. Barak, R. Canetti, J. B. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *45th FOCS*, pages 186–195. IEEE Computer Society Press, Oct. 2004.

[3] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Randomizable proofs and delegatable anonymous credentials. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 108–125. Springer, Aug. 2009.

[4] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. P-signatures and noninteractive anonymous credentials. In R. Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 356–374. Springer, Mar. 2008.

[5] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. Compact e-cash and simulatable VRFs revisited. In H. Shacham and B. Waters, editors, *PAIRING 2009*, volume 5671 of *LNCS*, pages 114–131. Springer, Aug. 2009.

[6] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993.

[7] F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 449–475. Springer, Aug. 2013.

[8] F. Benhamouda, G. Couteau, D. Pointcheval, and H. Wee. Implicit zero-knowledge arguments and applications to the malicious setting. In *CRYPTO 2015, Part II*, LNCS, pages 107–129. Springer, Aug. 2015.

[9] F. Benhamouda and D. Pointcheval. Trapdoor smooth projective hash functions. Cryptology ePrint Archive, Report 2013/341, 2013. `http://eprint.iacr.org/2013/341`.

[10] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988.

[11] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Aug. 2004.

[12] E. Bresson, D. Catalano, and D. Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In C.-S. Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 37–54. Springer, Nov. / Dec. 2003.

[13] J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 331–345. Springer, Dec. 2000.

[14] J. Camenisch, G. Neven, and a. shelat. Simulatable adaptive oblivious transfer. In M. Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 573–590. Springer, May 2007.

[15] G. Castagnos and F. Laguillaumie. Linearly homomorphic encryption from DDH. In *CT-RSA 2015*, LNCS, pages 487–505. Springer, 2015.

[16] P. Chaidos and G. Couteau. Efficient designated-verifier non-interactive zero-knowledge proofs of knowledge. Cryptology ePrint Archive, Report 2017/1029, 2017. `http://eprint.iacr.org/2017/1029`.

[17] P. Chaidos and J. Groth. Making sigma-protocols non-interactive without random oracles. In *PKC 2015*, LNCS, pages 650–670. Springer, 2015.

[18] M. Chase, S. Meiklejohn, and G. Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. In *ACM CCS 14*, pages 1205–1216. ACM Press, 2014.

[19] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In S. Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 319–327. Springer, Aug. 1990.

[20] D. Chaum and E. van Heyst. Group signatures. In D. W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Apr. 1991.

[21] M. Ciampi, G. Persiano, L. Siniscalchi, and I. Visconti. A transform for NIZK almost as efficient and general as the Fiat-Shamir transform without programmable random oracles. LNCS, pages 83–111. Springer, 2016.

[22] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Apr. / May 2002.

[23] I. Damgård, N. Fazio, and A. Nicolosi. Non-interactive zero-knowledge from homomorphic encryption. In S. Halevi and T. Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 41–59. Springer, Mar. 2006.

[24] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In K. Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 119–136. Springer, Feb. 2001.

[25] I. Damgård and M. Jurik. A length-flexible threshold cryptosystem with applications. In R. Safavi-Naini and J. Seberry, editors, *ACISP 03*, volume 2727 of *LNCS*, pages 350–364. Springer, July 2003.

[26] I. Damgård, M. Jurik, and J. B. Nielsen. A generalization of paillier's public-key system with applications to electronic voting. *International Journal of Information Security*, 9(6):371–385, 2010.

[27] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Aug. 2001.

[28] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Aug. 1984.

[29] A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. An algebraic framework for Diffie-Hellman assumptions. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Aug. 2013.

[30] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st FOCS*, pages 308–317. IEEE Computer Society Press, Oct. 1990.

[31] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Aug. 1987.

[32] M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Aug. 2005.

[33] R. Gay, D. Hofheinz, E. Kiltz, and H. Wee. Tightly CCA-secure encryption without pairings. LNCS, pages 1–27. Springer, 2016.

[34] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[35] M. Green and S. Hohenberger. Universally composable adaptive oblivious transfer. In J. Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 179–197. Springer, Dec. 2008.

[36] J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In X. Lai and K. Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Dec. 2006.

[37] J. Groth, R. Ostrovsky, and A. Sahai. Non-interactive zaps and new techniques for NIZK. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 97–111. Springer, Aug. 2006.

[38] J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for NP. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, May / June 2006.

[39] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Apr. 2008.

[40] E. Kiltz, J. Pan, and H. Wee. Structure-preserving signatures from standard assumptions, revisited. In *CRYPTO 2015, Part II*, LNCS, pages 275–295. Springer, Aug. 2015.

[41] E. Kiltz and H. Wee. Quasi-adaptive NIZK for linear subspaces revisited. LNCS, pages 101–128. Springer, 2015.

[42] Y. Lindell. An efficient transform from sigma protocols to NIZK with a CRS and non-programmable random oracle. LNCS, pages 93–109. Springer, 2015.

[43] H. Lipmaa. Optimally sound sigma protocols under DCRA. Cryptology ePrint Archive, Report 2017/703, 2017. http://eprint.iacr.org/2017/703.

[44] Y. Oren. On the cunning power of cheating verifiers: Some observations about zero knowledge proofs (extended abstract). In *28th FOCS*, pages 462–471. IEEE Computer Society Press, Oct. 1987.

[45] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, May 1999.

[46] R. Pass, a. shelat, and V. Vaikuntanathan. Construction of a non-malleable encryption scheme from any semantically secure one. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 271–289. Springer, Aug. 2006.

[47] D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, May 1996.

[48] I. Teranishi, J. Furukawa, and K. Sako. k-Times anonymous authentication (extended abstract). In P. J. Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 308–322. Springer, Dec. 2004.

[49] C. Ventre and I. Visconti. Co-sound zero-knowledge with public keys. In B. Preneel, editor, *AFRICACRYPT 09*, volume 5580 of *LNCS*, pages 287–304. Springer, June 2009.

Part II

Traffic & Routing Analysis

5. Multiparty Routing: Secure Routing for Mixnets

5.1 Introduction

Anonymous communication networks (ACNs) provide secure communication channels that hide not only data contents but also the communications' metadata, thus providing protection against traffic analysis. By routing data through multiple proxies, ACNs prevent network eavesdroppers from linking the source and destination of messages. Furthermore, ACNs enable senders to communicate with potentially malicious destinations without revealing their identity or location.

ACNs are typically overlay networks consisting of a set of dedicated routers, also called relays, that are connected via the Internet. The first ACN design in the early 1980s [11] introduced the concept of *mixes* for implementing anonymous email. These mixes are relays that collect a batch of N equal-size messages, cryptographically transform them, and output them in a re-shuffled form, so that the output messages cannot be traced to their corresponding inputs based on message content, size, timing or order. The best known ACN is Tor [32]. The Tor network is an ACN that consists of about seven thousands relays run by volunteers, and provides service to an estimated two million daily users who use it primarily to anonymously browse the web [1]. Other examples of ACN networks include for example Mixminion [24] an anonymous re-mailer, Freenet [14, 13] used for anonymous file-sharing, and DC-Nets [16] that can be deployed for broadcast applications such as group messaging.

The goal of ACNs is to anonymize communications by relaying them over multiple routers. There are three main types of anonymous routing in terms of how routers are chosen to form the path.

First, in *deterministic routing*, the paths are predetermined by the system configuration. Chaum's original ACN proposal [11] considered a sequence of mixes organized in a *cascade*. Systems that adopted the cascade network topology in their designs include JAP [8], and voting systems, such as Helios [2]. An advantage of cascades is that routing headers are unnecessary, since the next hop in the path is predetermined. Cascades however, are of limited practicality due to their poor scalability, limited anonymity, and lack of resilience to router failures [42].

Second, in *source routing*, the full path is chosen by the initiator of the communication. This is the routing used by Tor [32] and by anonymous remailers such as Mixmaster [44] and Mixminion [24]. In comparison to cascades, source routing is more resilient against failures of routers [57]; it is more scalable, meaning that the network can grow to accommodate more users; and consequently, can provide better anonymity to those users.

However, an important requirement in source routing is that the initiator must have a full view of the network, since otherwise partial network knowledge can be exploited by an adversary to fingerprint and identify users based on which relays they choose [26]. As a consequence, all users

need to periodically download the full list of currently available routers, their addresses, public keys, and other routing-relevant information. This is a major bottleneck for the scalability of the network, particularly if the routers have a high churn – as is often the case in peer-to-peer networks.

Third, in *hop-by-hop routing* each router locally decides on the next router of the path. Hop-by-hop routing is advantageous over source routing because it does not require entities to maintain an up-to-date view of the full network, as well as allows for better load balancing. The main disadvantage of hop-by-hop routing is that it is vulnerable to *route capture* attacks, where a malicious relay can gain full control over the route of a message, so that it is relayed through adversarial relays [23]. The earliest ACN employing hop-by-hop routing is Crowds [52], and another example is Morphmix [53], which uses witnesses to mitigate route capture attacks.

5.1.1 PANORAMIX Contributions

In this chapter we propose *multiparty routing*, a novel type of anonymous routing that broadens the design space with a fourth kind of routing. Our solution comes with important advantages over the previous routing approaches. Multiparty routing offers a scalability advantage over source routing: in our proposal the communication initiators need to only know one (or a small number) of routers, compared to the full view required in source routing. At the same time, we prevent the route capture attacks of hop-by-hop routing by decentralizing the routing decision using a multiparty computation technique. In addition to anonymity, our protocol guarantees the integrity of both shuffles and routing, which is novel for ACNs.

Our main contribution is a concrete multiparty routing protocol that employs cryptographic primitives, such as commitment schemes, signatures, and cryptographic hash functions. We combine our routing protocol with a mix network that employs provable shuffles, threshold decryption, and re-randomizable encryption to realize secure and verifiable anonymous communication, and load balancing. Our solution achieves security against global active adversaries and offers improved resilience and scalability against relay failures compared to deterministic and source routing. Compared to mix cascades our solution increases the anonymity set without substantially affecting the cost per mix. We achieve close to optimal load balancing by integrating relevant information about the throughput of relays into the routing strategy. Our protocol is a mid-latency anonymous communication network and thus can be deployed for applications that are more tolerant to latency such as anonymous wiki, micro-blogging, voting, and auctions.

5.2 Goals, Assumptions, and System Architecture

In this section, we describe the security model, the goals of the system, the entities and their roles, and the threat model.

5.2.1 Adversary Model

We assume a global adversary that can monitor all the communication links between entities in the system, including channels from users to the system and vice versa. The adversary is active, meaning that it can corrupt a subset of entities in the system, as well as submit messages herself. We assume that all cryptographic primitives are secure and cannot be broken by the adversary. Our protocol does not enforce restrictions on how many users need to be honest. However, a user is anonymous only among honest users that submitted messages to the network during the same time frame, and whose messages have traversed at least one honest relay.

5.2.2 Goals of the System

The goal of our system are as follows:

- Anonymity: the adversary must not be able to link a message leaving the system to the user who sent it, provided that the message has traversed at least one honest relay. This goal is equivalent to sender anonymity that has been defined by Pfitzmann et al.[48].
- Routing Integrity: the adversary must not be able to influence the route followed by a message, and consequently cannot bias the routing by forcing the data to only traverse adversarial relays.
- Availability: the system must be robust to the adversarial removal of relays, and output all messages even if some relays are taken down once the routing has started.
- Load-balancing: we aim at providing load balancing such that relays route an amount of traffic that is proportional to their throughput.

5.2.3 Entities and Roles

We consider a broadcast channel that we call *bulletin board*. Although the bulletin board is readable by all parties in the system: users and servers, users can not write into it.

Our system comprises a set of users and dedicated servers as depicted in Fig. 5.1. The users in the system want to send messages anonymously, whereas the servers cooperate with each other to achieve secure anonymous routing. We distinguish between three sets of servers: $\mathcal{R}$ - relays, $\mathcal{RE}$ - routing entities, and $\mathcal{AS}$- auditing servers. The relays $\mathcal{R}$ are mixes that relay messages via publishing ciphertext messages on the bulletin board and fetching them. More precisely, the mixes first collect from the bulletin board their set of input ciphertexts, then re-encrypt and shuffle those, a process referred to as *mixing*, and then relay their outputs to the bulletin board, so that each of them can be retrieved by the next mix in the path. The relays are organized in l sequential layers formed by disjoint subsets of $\mathcal{R}$.

The second set of servers are called *routing entities* $\mathcal{RE}$. These are responsible for computing the joint randomness that is used in the routing process.

The third set of servers are the *auditors (auditing servers)* $\mathcal{AS}$. Auditors are responsible for: 1. generating and broadcasting the public encryption key; 2. verifying the correctness of message shuffling and routing and 3. inal message decryption.

For clarity, we keep the roles of auditors and routing entities separate from mixes. However, from a practical point of view all three roles can be carried out by the same set of servers.

Network Handler

We suggest to use an untrusted "network handler" who arranges the mixes into layers based on some predefined rules that strategically arranges mixes such that the probability that all mixes in a route cooperate with each other is as low as possible, e.g., due to conflict of interest or high expense. These rules can be for example such as follows:

- Mixes set up by an organization or institution or from a country cannot be placed in more than one layer,
- One layer has to have mixes from at least three distinct countries (IP subnets), organization, or institution

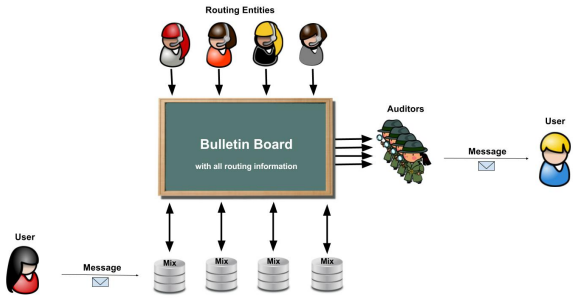


Figure 5.1: This figure depicts the roles of all actors (user, mixes, routing entities, auditing servers) as a message is relayed through the system.

- The network needs to have layers according to the number of incoming messages and the number of mixes in one layer (see Section 5 for analysis for this)

Another responsibility of the network handler is to handle churn. Note, that $\mathcal{RE}$ and $\mathcal{AS}$ are not managed by the network handler and are self-organized. This is important because otherwise the network handler would have power to impact the anonymity of the system. The untrusted network handler can not aid the adversary as long as the network management rules are respected. If the network manager does not follow them his behavior will be visible to all participants in particular $\mathcal{AS}$ who have the power to replace the misbehaving network handler. The network handler can be only one server but ideally would be a set of servers similar to Tor's directory servers [1].

Next, we briefly review the required cryptographic building blocks and proceed by describing our protocol.

5.2.4 Cryptographic Building Blocks

Next, we review relevant cryptographic building blocks.

Public Key Encryption and Signing We assume the reader is familiar with the principles of public key encryption and signature schemes. In our protocol, we employ a variant of public key encryption, referred to as *threshold decryption* [28, 56], which allows for the decryption to be carried out by a quorum of participants (the auditors in our scheme). More precisely, our scheme uses a distributed key generation protocol `KeyGen`, such as the ones of [47, 35]. It generates a public key which is made available to all users of the system, while the corresponding secret key is distributed to all the auditors $\mathcal{AS}$ in a secret shared form. During the decryption protocol `Dec` a threshold of the secret key share holders (auditors) jointly decrypt the input ciphertext. This process can be realized either with a trusted dealer, such as in the work of [60], or without [46], [17].

To conceal how ciphertexts traverse a mix, we use re-randomizable encryption, such as ElGamal [34]. This allows a mix to update the randomness of the traversing ciphertexts. We use standard cryptographic signatures, such as [55], for the routing verification.

Shuffle and Correctness of Shuffling A shuffle algorithm `Shuffle` implements a permutation on its inputs. In our protocol we use shuffles to permute the ciphertexts to conceal their ordering. To guarantee the correct execution of the shuffling algorithm we apply a *proof of correct shuffling* which is realized by a zero-knowledge (ZK) proof systems. In our scheme, given a set of pre-shuffling and post-shuffling ciphertexts, a prover (the mix) needs to prove to a verifier (the auditors or, in case the auditor role is carried out by mixes, other mixes) that the set of post-shuffling ciphertexts is a re-randomization of the set of the pre-shuffling ciphertexts (without revealing the permutation

and the re-randomization randomness used by `Shuffle`). Our protocol utilizes non-interactive variants of ZK proofs, such as the ones proposed in [63][4].

Shuffle-friendly Encryption In addition to invalid shuffling, we use ZK proofs to prevent the double submission of honestly encrypted messages. This is necessary because repetitions in the output reveal the relationship between input and output messages. We follow [6, 43] and use chosen-ciphertext attack (CCA) secure encryption with a homomorphic embeddings. This property is satisfied by ElGamal encryption when combined with a non-interactive ZK proof of plaintext knowledge. After verifying the proof, the non-malleability of CCA secure encryption allows to weed out double submissions. The shuffle uses the re-randomizable ElGamal ciphertext without the proof.

5.3 Multiparty Routing Protocol

5.3.1 System Overview

Messages are relayed by one mix from each layer of the system. Therefore, each message is relayed by a set of mixes. Users send their messages to an arbitrary mix from the first layer of mixes $\mathcal{R}_1$. Only messages that are submitted within the same time frame are mixed with each other. Hence, a user is only anonymous among honest users that submitted their messages at the same time frame.

Each relay mixes all message it receives. The next mix is determined in a hop-by-hop fashion. However, the mixes do not have freedom to choose to which mixes they send any given output message. The routing entities, $\mathcal{RE}$, determine how each message is routed by jointly producing the randomness used for the routing; this process, called *Multiparty Routing*, can be summarized as follows.

1. All routing entities choose privately a random number and use a commitment scheme to commit to their random numbers on the bulletin board.

2. After all routing entities have committed to their random numbers, they all open their commitments.

3. The random numbers are combined to produce a joint random number.

4. This random number is used to compute the routing assignment of the ciphertext messages to the mixes in the next layer in proportion to their capacity/throughput.

This procedure is carried out for the output of each mix. In parallel to this, the auditing servers need to verify the correctness of the shuffling and routing. After a message has been mixed by a mix from each layer and the auditing servers successfully verify the correctness of all routing and shuffling processes the message is delivered to the intended recipient of the message. If at least one of the mixes that has relayed the message is honest, the adversary cannot trace back a message leaving the network to any particular sender.

Figure 5.2 shows the topology for a network that has four layers (disjoint subset) of mixes. After processing input messages, the MPR protocol assigns output messages to mixes in the next layer; meaning that a portion of the output messages of each mix is assigned to mixes in the next layer in proportion to the throughput of those mixes within their own layer. Hence, the routing pattern between the mixes follows a stratified topology as investigated by Diaz et al. [30].

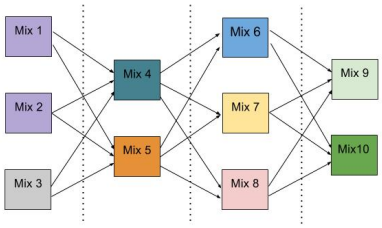


Figure 5.2: In this figure we present an example of how mixes can be arranged for our protocol when the protocol consists of 4 layers. The throughput of all layers needs to be equal, meaning that the total throughput of mixes $1, 2, 3$ needs to be equal to the total throughput of the mixes $4, 5$, and the total throughput of the mixes $6, 7, 8$, ad the total throughput of the mixes $9, 10$. For example, the output ciphertexts from mix 1 are either assigned to mixes 4 or mix 5.

5.3.2 Main Protocol

We denote the set of users by U. The set $\mathcal{R}$ of relays is divided into l layers formed by mutually exclusive subsets $\mathcal{R}_i$ with the union $\bigcup_{i=1}^{l} \mathcal{R}_i = \mathcal{R}$. An individual mix is denoted by r.

The bulletin board contains lists of routing and verification information (input and output ciphertexts, corresponding signatures, commitments and their corresponding openings), and NIZK proofs of shuffles, grouped and indexed by *ctr* going from 1 to l).

Encryption The auditors (auditing servers) $\mathcal{AS}$ are responsible for the generation of the encryption key *pke*. Each auditing server has only a share of the private key $\{skd_{\mathcal{AS}_1}, ..., skd_{\mathcal{AS}_d}\}$ where $|\mathcal{AS}| = d$. Only qualified subsets of the auditing servers can jointly decrypt ciphertexts using their private key shares.

The protocol starts with each user encrypting her message m together with its intended receiver under the public key *pke* (known to all users) and sending the corresponding ciphertext made CCA secure by the proof of knowledge (PoK) to the bulleting board.The set of ciphertexts that enter the system in the same time (by all senders) is denoted by C_0. Our system mixes ciphertexts in multiple layers with layer $\mathcal{R}_i$ producing the ciphertext set C_i. After completing l layers the protocol terminates with a verification step and the decryption of C_l.

Mixing Once a subset of the ciphertexts are assigned to a mix r the *mixing* and *routing* take place. Mixes in the first layer receive ciphertexts from users, while mixes for other layers take ciphertexts from the bulletin board. The mixes in the first layer verify the PoK and after eliminating duplicates remove these proofs. The mixing consists of the *re-encryption* of the ciphertext inputs and their *shuffling* by a random permutation ϕ.

After the output ciphertexts have been submitted to the bulletin board, the next mix to shuffle a ciphertext is determined by the following *routing assignment* procedure.

Routing Assignment We use a coin-tossing protocol such as [9] to obtain joint randomness. In the coin-tossing protocol the routing entities commit to their randomness using a cryptographic commitment schemes. Once the commitments are opened and the random strings are revealed on the bulletin board, a simple function, e.g. bitwise XOR or modular addition, is applied to obtain the joint randomness *Rand*.

Then we apply a function `Assign` on *Rand*. `Assign` then outputs the next layer mix index assignment list $\mathcal{L} = (z_1, \ldots, z_w)$ with k elements in correspondence with the capacity/throughput

of these mixes.

We give an example instantiation of `Assign` via the two procedures `H` and `Map` in Appendix A in Algorithms 2 and 3.

The example in Figure 5.3 to illustrates the routing assignment. Let us have a relay which outputs ciphertexts c_1, c_2, c_3 and the next layer of relays consist of two relays r_1, r_2 with corresponding capacities $b_1 = 1, b_2 = 2$. By applying `Assign` we want to determine which relays in the layer receive which ciphertexts. Namely, for an output vector of indices ($z_1 = 2$, $z_2 = 1$, $z_3 = 2$) output by `Assign` the outgoing ciphertext is going to be fetched in the next step as follows. In our case, r_1 will then fetch the first $\frac{1}{1+2} = \frac{1}{3}$ of the outgoing ciphertexts, namely $c_{z_1} = c_2$, and r_2 fetches the rest $\frac{2}{1+2} = \frac{2}{3}$ ciphertexts, $c_{z_2} = c_1$ and $c_{z_3} = c_3$ respectively.

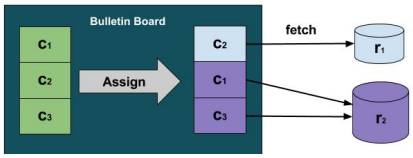


Figure 5.3: This figure depicts how the ciphertexts c_1, c_2, c_3 from our example are assigned to the mixes r_1 and r_2.

Proof of Mixing and Routing The correctness of the mixing process is accommodated by the bulletin board. After mixing the set of input and corresponding set of output ciphertexts the mix needs to sign input and output ciphertexts. For signing, both the routing and mixing, $\mathcal{R}$ use their signing private key sks. The corresponding verifying public key pkv is available to all parties for later verification purposes. These keys are used as long-term keys, unlike $(pke, (skd_1, \ldots, skd_d))$ that are issued fresh for each time-frame.

Then, to prove correctness of protocol execution each mix needs to post on the bulletin board: a) inputs and outputs and their corresponding signatures, and b) a non-interactive zero-knowledge (NIZK) proof of correct shuffling. Also, the bulletin board maintains the commitments for the randomness used for the MPR routing algorithm and their subsequent openings posted by the routing entities.

Verification and Decryption Ciphertext which have passed through all layers are decrypted by the auditing servers upon successful verification of the correctness of: 1) Submission 2) Mixing and 3) Routing. Even though the first layer eliminates duplicates, the auditing servers need to verify that this was done correctly. After all mixes in a layer published their results on the bulletin board, the auditing servers verify their mixing process by verifying all NIZK proofs on the bulletin board. Moreover, the auditing servers carry out routing verification on the input/output ciphertext signed on the bulletin board for each layer as soon as input ciphertexts for the next layer start being published to the bulletin board. The routing verification takes the randomness (commitments openings) and computes the assignment for the next mix layer. Then, correct routing is confirmed by verifying that the computed assignment corresponds to the correct input and output ciphertexts in the bulletin board. With Algorithm 4 we give an example instantiation of the routing verification algorithm in Appendix B.

If routing or mixing verification fails, meaning that a mix has taken wrong ciphertext as input or produced an invalid NIZK proof, the auditing servers record the verification failure of the corresponding mix on the bulletin board. If routing and mixing of a mix does not verify the routing

entities generate a fresh *Rand* to assign the input ciphertexts from the mix that has performed wrong routing to the remaining mixes in his layer. All remaining output ciphertexts in the layer need to wait until this stage is carried out to be routed to next layers.

The integrity and authenticity of bulletin board submissions is verified by the use of the signatures schemes. In our protocol we use randomized algorithms and with $\xleftarrow{\$}$ we denote the randomized output generation, so $\xleftarrow{\$} S_w$ samples randomly from all permutations of size w and s refers to the randomness used by the ciphertext re-randomization algorithms.

Our protocol is summarized in Algorithm 1.

5.3.3 Parameter Choices

Our protocol has similarities to parallel mixing in terms of distributing ciphertexts among a set of mixes, hence, we revisit suggested parameters by the literature and adopt these parameters for our system.

The number of mixes that relay the ciphertexts, or so-called path length, is an important aspect of our system because it determines how many times a batch of ciphertexts C is re-arranged. The path length has a direct effect on whether a ciphertext batch C entering the system is fully mixed with the other batches entering at the same time. Our result confirms the intuition that a larger number of layers leads to a better mixing.

For a passive adversary, Czumaj shows that the path length in any switching networks should be $\log n$, where n is the total number of mixes in the system [20] [19]. Similarly, Klonowski et al. [41] have also shown that in parallel mixing the number of distribution steps should be $\log n$.

For an active adversary, Klonowski [41] shows that the number of mixing layers must depend on the number of messages that are relayed through the system. Similarly, Goodrich and Mitzenmacher [38] have shown that if some mixes are controlled by an active adversary then the number of layers needs to be $\log N$ where N is the number of input messages in a given time frame.

Since, we are considering an active adversary we set the number of layers to $\log N$. For example, if we assume the routing of 1000 messages our system needs at least 3 layers where each layer has the throughput to relay 1000 messages. For relaying 100,000 messages the system needs to have at least 5 layers where each layer has the throughput to relay 100,000.messages. To improve availability each layer should have at least two mixes.

Furthermore, to increase the cost of an adversary we assume that mixes are set up across distinct IP subnets by multiple organizations or institutions. For example, an organization can offer more than one mix for the network, however, all mixes of an organization need to be in the same layer. This is important because otherwise a message can be effectively mixed by a single organization.

Another important parameter is the capacity or throughput of the mixes. From a load balancing point of view the total throughput of mixes in each layer needs to be equal in order to prevent any congestion in our system.

5.4 Related Work

In the last three decades there have been numerous ACNs proposed in the literature using random walks, mixes and onion routing, dummy traffic, the dining cryptographer concept [12], multi-party computations, and broadcasting. Often an ACN combines several of these techniques to achieve anonymity and avoid de-anonymization attacks. Some examples of ACN are as follows. Random walks are used by peer-to-peer systems such as Freenet [14] and Octpus [62] but also in onion routing protocols such as Tor [33] and I2P [54]; dummy traffic [31] is used by system such as Loopix [49];

Setup
Auditing servers $\mathcal{AS}$ generate encryption keys:
$(pke, skd_{\mathcal{AS}_1}, ..., skd_{\mathcal{AS}_d}) \xleftarrow{\$} \texttt{KeyGen}$
publish (pke) to BB

Encrypt
Each sender:
$c \xleftarrow{\$} \texttt{Enc}_{pke}(\text{receiver's address}, m; \rho)$
$\Pi' \leftarrow \texttt{Prove}_{\text{PoK}}(c, pke; m, \rho)$
sends c and Π' to BB
$ctr \leftarrow 1$

while $ctr \leq l$ do

Mixing and Prove Mixing
Mixes $r \in R_1$ verify the PoKs, drop invalid or duplicate ciphertext, and remove proofs
Each $r \in \mathcal{R}_{ctr}$, with inputs $C = \{c_1, \ldots, c_w\}$ and $\phi \xleftarrow{\$} S_w$:
shuffles $C^{\phi} \xleftarrow{\$} \texttt{Shuffle}(C, \phi)$
re-encrypts $(C', s) \xleftarrow{\$} \texttt{ReEnc}(C^{\phi})$

Each $r \in \mathcal{R}_{ctr}$:
computes $\pi \xleftarrow{\$} \texttt{Prove}_{\text{NIZK}}(C, C'; \phi, s)$
publish π to bulletin board
publish $(\texttt{Sign}_{skv_r}(C'), C')$ to BB

Next hop and Prove Routing
For each $r \in \mathcal{R}_{ctr}$, all v routing entities $\mathcal{RE}$:
publish $\texttt{Com}(rand_1), \ldots, \texttt{Com}(rand_v)$ to BB
publish $rand_1, \ldots, rand_v$ to BB
For each $r_{out} \in \mathcal{R}_{ctr}$, each $r \in \mathcal{R}_{ctr+1}$ computes:
$Rand \leftarrow \texttt{SUM}(rand_1, \ldots, rand_v)$
$(z_1, \ldots, z_w) \leftarrow \texttt{Assign}(Rand)$
For $i = 1$ to w $r_{z_i} \in R_{ctr+1}$ fetch c_i

Verify Mixing and Routing
For each $r \in \mathcal{R}_{ctr}$, a quorum of $\mathcal{AS}_z \subset \mathcal{AS}$ runs
$\texttt{Verify}_{NIZK}(C_r, C'_r, \pi_r)$
$\texttt{Verify}_{pkv_r}(\texttt{Sign}_{sks_r}(C_r), C_r)$
$\texttt{Verify}_{pkv_r}(\texttt{Sign}_{sks_r}(C'_r), C'_r)$
and computes
$\texttt{VerifyRouting}(Rand_r, C'_r, \mathcal{R}_{ctr+1})$

$ctr \leftarrow ctr + 1$
end while

Decrypt
Each $r \in \mathcal{R}_l$:
sends all c' to $\mathcal{AS}$
For a threshold z of auditing servers $\mathcal{AS}$:
if ***Verify Mixing and Routing*** is successful for all layers
(receiver's address, m) $\leftarrow \texttt{Dec}_{skd_1, \ldots skd_z}(c')$
otherwise returns an error
send m to *receiver* from U

Algorithm 1: Multiparty Routing

dining cryptographers is used by Dissent [64] [16], multi-party computation is used by MCMix [3], and broadcasting is used by Dissent and CAR [59] (see [58] for a survey on ACNs). We focus on the closest related work.

Like Parallel Mixes [37] and Atom [42], a recent improvement on Parallel Mixes, our approach addresses the scalability and robustness issues of mix cascades [11].

Golle and Juels introduced Parallel Mixes [37], an ACN that uses both deterministic and hop-by-hop routing in two different phases that they call *rotation* and *distribution* phases, respectively. Typical configurations start with a rotation phase, then go through a distribution phase, and finish with a rotation phase. During a rotation phase, mixes exchange their entire output ciphertexts with another mix in a rotating fashion (that is, the first mix passes its ciphertexts to the second, and so on, and the last mix passes them to the first). In the distribution phase, each mix splits output ciphertexts among other mixes. The main purpose of the distribution phase is to mix the inputs with each other to achieve a bigger anonymity set. Our routing protocol has similarities with the distribution phase of Parallel Mixes, but removes one primary requirement of Parallel Mixes: The mixes in the distribution phase must not be aware of the sender of any of the ciphertexts. Consequently, users must submit input ciphertexts uniformly to the system and ciphertexts must be mixed by at least one honest mix during the rotation phase. This is necessary to prevent route-capture attacks, because mixes in the distribution phase are free to choose how to distribute their output ciphertexts to other mixes (contrary to our protocol, where this decision is made jointly by multiple entities). These countermeasures are both costly and insufficient when the adversary controls a fraction of messages as demonstrated by Borisov's attack [10]. Kwon et al. [42] proposed Atom a recent parallel mixing proposal that considers active adversaries and achieves random permutation of messages using iterated butterfly networks.

Movahedi et al. [45] proposed a multiparty shuffling protocol that allows multiple parties to jointly compute a private random permutation of input messages. The parties that perform the multiparty shuffling protocol are split up into quorums that use a sorting scheme on inputs that have been shared among the quorum to achieve random permutation. While the approach is resilient to parties aborting the protocol or dropping out because of technical difficulties, there is no mechanism for preventing traffic tampering before the shuffling starts. Therefore, parties who hold user inputs at the beginning of the shuffling can replace them with arbitrary adversarial inputs. The approach is therefore only suitable for passive adversary. Furthermore, this scheme requires that the links between honest parties must be private and cannot be observed by the adversary; and the adversary needs to be static, meaning it cannot corrupt alternative parties once the protocol is running.

5.5 Security Analysis

In this section we start by revisiting the security properties of interest and go on to analyze the security of our protocol based on those security goals against a global, active adversary. We show that an active adversary in our solution has no advantage over a passive one.

Security Properties

- *Routing Integrity*: guarantees that an active adversary is not able to influence the routing decision to deanonymize messages by forcing them to route through adversarial mixes.

- *Routing Correctness*: guarantees the detection of malicious routing and subsequently only correctly routed messages leaving the system.

- *Anonymity (routing confidentiality)*: guarantees that the adversary cannot find the relationship between messages leaving the system and a user sending a message to the system, provided that the message has traversed at least some honest mixes.

- *Availability*: guarantees that the protocol is robust against removal of a subset of entities by an adversary and that the messages entering the system will be output by the system in the face of such attacks.

5.5.1 Routing Integrity

To address the integrity property we answer the following question: *Can a malicious mix (or a subset of mixes) influence the routing of a ciphertext c to a mix of his choice* $r_{\mathcal{A}}$*?* We investigate two ways of manipulating the routing decision. The first way deals with adversaries who try to influence the shuffling procedure. This attack path is however not possible in our protocol because all mixes publish their shuffled ciphertexts before the routing decision (*Rand* outcome) is revealed. The second way to manipulate the routing decision is through adversarial bias in the computation of the joint randomness *Rand*.

More concretely, we investigate the advantage of an active adversary $\mathcal{A}$ who gains control over a subset of routing entities denoted as $\mathcal{RE}_A$. In order to influence the routing decision $\mathcal{A}$ has to be able to bias the computation of *Rand* where $Rand = Rand_{honest} + rand_{\mathcal{A}}$ is the mod sum addition of the random numbers from all honest entities $Rand_{honest}$ and the adversarial controlled randomness $rand_{\mathcal{A}}$. The goal of $\mathcal{A}$ is to produce a new $rand'_{\mathcal{A}}$, such that the updated sum $Rand' = Rand_{honest} + rand'_{\mathcal{A}}$ fits its routing purposes: the target ciphertext is routed to $r_{\mathcal{A}}$ where $\mathcal{A}$'s capacity/throughput is b/B. In our protocol *all* routing entities $\mathcal{RE}$ first need to *commit* to their random numbers and only then *reveal* the openings of their commitments. $\mathcal{A}$ has control only over its own commitment $T = \mathtt{Com}(rand'_{\mathcal{A}})$ and the corresponding opening $rand'_{\mathcal{A}}$. The best strategy for $\mathcal{A}$ is to wait for the openings of all honest routing entities and then reveal his own $rand'_{\mathcal{A}}$.

Since $\mathcal{A}$ does not know in advance the openings of the honest routing entities, $\mathcal{A}$ needs to 'predict' in advance a valid $rand'_{\mathcal{A}}$, such that $T = \mathtt{Com}(rand'_{\mathcal{A}}))$. Our assumption here is that Com is a secure scheme against collision, preimage and second preimage attacks. Under this assumption $\mathcal{A}$'s best strategy is to perform precomputation to find the largest set of preimages $\mathcal{N}$ with $|\mathcal{N}| = n$ mapping to a commitment value T. Then, once $\mathcal{A}$ commits to T, he 'hopes' that the needed randomness for his routing purposes $rand'_{\mathcal{A}}$ belongs to the precomputed set $\mathcal{N}$. Then, what is left to find is the size of the set of valid randomness values $\mathcal{V}$ for the routing purposes of $\mathcal{A}$. Since the adversary controls b/B of the throughput w ciphertexts, then the set of valid values $\mathcal{V}$ amounts to $\frac{2^{|Rand|}}{w} \times \frac{b}{B} \times w$ where $\frac{2^{|Rand|}}{w}$ is the fraction of random values resulting in a given router assignment as defined by our routing algorithm. Finally, the probability that $\mathcal{A}$'s set of precomputed values $\mathcal{N}$ ends up intersecting with one of the valid random values $rand'_{\mathcal{A}}$ is $\frac{n}{2^{|Rand|} \times b/B}$. In other words, $\frac{n}{2^{|Rand|} \times b/B}$ amounts to the probability that $\mathtt{Com}^{-1}(T) = rand'_{\mathcal{A}}$ where $rand'_{\mathcal{A}} = (Rand' - Rand_{honest}) \in \mathcal{V} \cap \mathcal{N}$.

Note that, as we assumed, if Com is collision secure, finding $n = 2$ values takes already precomputation time of birthday bound complexity $2^{|Rand|/2}$. This means that unless $\mathcal{A}$ has a precomputational power of order $2^{|Rand|/2}$ its success probability in biasing the joint randomness *Rand* is $\frac{1}{2^{|Rand|} \times b/B}$.

Furthermore, if the adversary removes the mix that c is assigned to to force a new routing assignment his chances to be assigned to c are increasing only if he removes a large proportion of throughput of the layer of $r_{\mathcal{A}}$.

5.5.2 Routing Correctness

In this scenario we investigate if a mix can deviate from the routing assignment without being noticed. The adversary goal (acting as one of the mixes $\mathcal{R}$) is to get a ciphertext of his choice regardless of the ciphertext's next hop assignment. To achieve her goal the adversary here needs to manipulate the *routing verification*. To do so the adversary needs to control a subset of the auditing entities that read the routing information from the bulletin board and carry out the verification. But this is impossible in our protocol because this adversarial behavior is going to be discovered in the routing verification phase if at least one of the auditing servers is honest. Remember that one of the tasks of the auditing servers is the verification of the correct input/output ciphertext routing and their signatures.

5.5.3 Anonymity

We review our system for vulnerabilities that lead to weakened anonymity. As usual, our system only provides anonymity if at least one honest mix has routed the message. Our system provides anonymity by mixing a message with other messages, if a single message goes through our system we provide no anonymity, therefore, another measure we use for evaluating anonymity is the anonymity set size as discussed in [29]. There are, however, other means to deanonymize messages and we review those attacks too.

Traffic Analysis

Traffic analysis [51] is one of the main attack methods against anonymous communication. The adversary uses one or a combination of criteria to match incoming and outgoing traffic of an AC system in order to correlate them. The common criteria used for traffic analysis are as follows:

- Timing: in MPR timing correlation are not possible, because all communication that is sent through the system is accumulated periodically and sent in a synchronous manner that eliminates timing signatures.

- Size: if MPR is used for applications other than voting or micro-blogging (e.g., twitter has a limited message size 140 formerly and recently 280 characters) where only fixed sizes of messages can be sent, the adversary cannot use the message sizes for deanonymization. If MPR is used for applications that need more variation is message size, then a solution to avoid traffic analysis would be to define a number of message sizes (e.g., 2 sizes, one for messages and one for media) and design two MPR networks that each mix message of the same size. Hence, message in different sizes are not mixed together.

- Statistical disclosure: If MPR is used for anonymous messaging the communication pattern, the pattern of using the system or even just being online by two users can reveal they are communicating together by statistical disclosure attacks [21]. The same holds for the number of message that are sent by the sender and received by the receiver that can be used to deanonymize users. However, if the application that uses MPR receives the communication to only a certain platform then this attacks becomes harder. But if the outgoing communication is sent to multiple destinations such as in email applications, our system is vulnerable to long-term statistical disclosure attacks [8], and in such cases additional measures such as being always online and sending dummy traffic needs to be used in combination with our system.

- Appearance: since we are using re-encryption the incoming and outgoing traffic have a different appearance and can not be correlated as long as the cryptographic primitives remain secure.

Replay and Tagging Attacks

Danezis discovered replay and tagging attacks against re-encryption mix networks[22]. In these attacks, the attacker exploits the malleability of re-randomizable (and homomorphic) encryption to either resend or tag a message observed while entering the system. While mix networks with a proof of correct shuffle prevent the tampering of message that already entered the system, they are defenseless against tampering (duplicating, replacing, or tagging) messages before entering the mix network. In electronic voting, such as Civitas [15], this problem is sometimes solved using voting credentials that are distributed when users register for voting. We opt for a solution (see 5.2.4) introduced to prevent replay in the Helios voting system [6]. It relies on a ZK-proof that makes the ElGamal ciphertext non-malleable during submission, so that replayed messages can be weeded out. This proof is discarded once the message is mixed by the first layer of the system.

Tracing Messages

Here we assume the adversary's goal is to trace the path of a ciphertext c until it leaves the system. To link an input ciphertext to an output ciphertext the adversary needs to control all mixes that have relayed the input ciphertext. Otherwise, if at least one mix on the ciphertext's route is honest, the adversary will not be able to trace this link. (Note that he may still be able to reduce the anonymity set) Hence, our adversary needs to control at least one mix in each layer of the system. To estimate the probability that a ciphertext follows a fixed path, denoted by $c \to p$, then we just need to multiply the probability of that ciphertext being routed by the individual mixes in each layer of the path. Let us assume that a path p consists of an ordered list of mixes $p = \{r_1, r_2, \ldots, r_l\}$, where l is the number of layers of the system.

$$Pr[c \to p] = \prod_{i=1}^{l} \frac{b_i}{B}.$$

Moreover, the probability of choosing each of the mixes in a path is computed as follows. where b_i is the relative throughput of the mix r compared to the total throughput B of all mixes (recall that it is the same for all layers). If all mixes in the subset have the same capacity b, then $Pr[c \to p] = (\frac{b}{B})^l$

Our decision to follow a stratified topology disallows the adversary to gain any advantage from placing his efforts (i.e. throughput) unevenly as compared to balancing out its efforts for each layer. Since the number of total messages that will be routed by adversarial mixes is the multiplicative product (see above) of the throughput that the adversary is controlling at each layer, hence, he gains ore success by distributing his resources evenly among layers. For example, if an adversary controls 20% of the total throughput of the system, we assume that he is in fact controlling a 20% of the throughput of the mixes in each layer.

We are interested in the probability that a ciphertext is routed only by adversarial mixes because this leads to deanonymize the ciphertext leaving the system by the adversary. We compare the probability that the ciphertext relayed by our MPR protocol is only routed through adversarial mixes to to the probability that the ciphertext relayed by regular parallel mixing system is only relayed by adversarial mixes. Figure 5.4 shows the probability that a ciphertext routed through only adversarial mixes, for our MPR protocol and for parallel mixes system, when the adversary is controlling $0.1\%, 0.15\%, 0.20\%, 0.25\%, 0.30\%, 0.35\%$, and 0.40% of the throughput in each layer

of the system. For this figure we assume the system has 4 layers. When regular parallel mixing is used the probability that a ciphertext is only routed along adversarial mixes is in proportion to the resources of the adversary. While if our MPR protocol is used, this probability is increasing much slower due to the fact that adversarial mixes cannot steer ciphertexts toward further adversarial mixes.

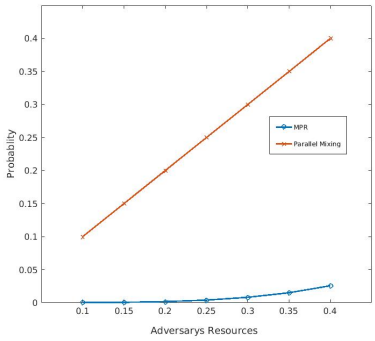


Figure 5.4: This figure depicts the probability of a message being routed through mixes that are all controlled by the adversary and shows that this probability increases for stronger adversaries in MPR and in parallel mixing. For both MPR and regular parallel mixing the system has 4 layers. The adversary is controlling mixes in each layer by a proportion of 0.1, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40.

Figure 5.5 shows the probability that a ciphertext is routed through only adversarial mixes, for our MPR protocol and for regular parallel mixes system, when the system has $3, 4, 5, 6$, or 7 layers. We assume for this figure that the adversary is controlling 0.25% of the throughput of each layer of the system. If regular parallel mixing is used, the probability that a ciphertext is only routed along adversarial mixes is constant and equal to the proportion of adversarial mixes irrespective of the number of layers. Wile when our MPR protocol is used, this probability is significantly lower compared to parallel mixes even when the system consists of only 3 layers. This probability decreases further as the number of layers in the system increases.

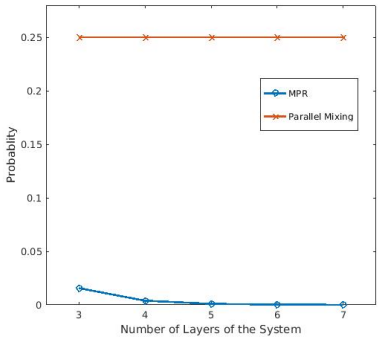


Figure 5.5: This figure depicts the probability of a message being routed through mixes that are all controlled by the adversary and shows that this probability increases when the system has more layers of mixing. In both MPR and regular parallel mixing, the adversary is controlling 0.25 of mixes of each layer.

5.5.4 Availability and Resilience

The goal of the adversary is to take out stakeholders of the protocol in order to obstruct the protocol from running.

There is a restriction on how many mixes, routing entities, and auditors the adversary can take out. Taking out auditors, $\mathcal{AS}$ is least beneficial, because the attack is not resulting in any advantage for the adversary. While taking out routing entities, $\mathcal{RE}$ increases the adversary's success in biasing routing to a very small extent. Taking out mixes will not abrupt the protocol execution but leads to traffic congestion. Below we investigate the effect of removing each type of entity from the system.

Routing entities $\mathcal{RE}$: If the routing entity fails before committing to a random number the protocol continues to operate as long as it is assumed that at least one honest routing entity has remained. If the routing entity fails after committing to a random number it is considered an abortion, although it might have been a system failure. If this happens more than once, the remaining routing entities vote the corresponding routing out.

Auditors $\mathcal{AS}$: If a limited number of auditors fail to carry out the distributed decryption, then the protocol is guaranteed to continue execution because we are using threshold cryptography. To carry out the routing and mixing verification the only restriction is that at least one remaining auditor is not controlled by the adversary.

Mixes $\mathcal{R}$: Our protocol is resilient to the failure of a number of mixes and preserves load balanced among the remaining mixes. We describe how the protocol recovers after the failure of a mix or a number of mixes in different positions as follows.

1. If a mix fails at the first layer after the users have sent their ciphertexts the users machine needs to notice in the bulletin board that her ciphertext is not in the input of any of the entry mixes and send their ciphertext to another entry mix.

2. If the adversary takes out all mixes in the first layer the user's machine will notice that the ciphertext sent by her is not in the bulletin board and the auditors notice the absence of any output from the first layer and introduce new entry points for the users.

3. If a mix fails at one of the layers except the first layer before generating the output ciphertext and the corresponding routing information, the auditors notice this and inform the routing entities to generate a new *Rand* that assigns the input ciphertext of the failed mix to the remaining mixes in its layer. If the proof that the mix has produced cannot be verified, the same happens. Note, that the ciphertexts shuffled by other mixes in the layer do not need to be re-shuffled again.

4. If the adversary takes out all mixes in one layer except the first layer, the auditors inform the next layer of mixes to take over with fresh random numbers.

Note that if the adversary takes out l honest mixes from each layer of the system he can increase his success in tracing messages as follows. Assume the adversary initially controlling b^A/B of the throughput of all layers of the system where B is the throughput of the layers. Before the attack the adversary is able to trace:
$\prod_{i=1}^{l} \frac{b_i^A}{B} \times \mid M \mid$
messages, where b_i^A is the throughput that the adversary is controlling in layer i and M is the total number of distinct messages that were submitted to the system. After taking out a mix from each layer he can trace:

$\prod_{i=1}^{l} \frac{b_i^A}{B-b_{DoS}} \times \mid M \mid$
messages, where b_{DoS} is the throughput he has taken out of each layer.

5.5.5 Scalability

Our system provides scalability by allowing the users to only need to know the key of the corresponding time-frame and connection details of one entry mix to the system opposed to source-routing protocols where the users needs to be aware of the whole system such as Tor [32] and mixes for mix networks such as Mixminion [24]. Moreover, the user is oblivious to the network changes that are managed by the network handler, only other mixes need to be keeping track of this. In addition, MPR is also more scalable compared to mix cascades, because increasing the anonymity set of the system can be increased by adding throughput to each layer that can be added by adding even small mixes to each layer while in mix cascades each mix of the cascade needs to increase its throughput individually to result in an increase of the anonymity set. Adding an additional mix cascade while is useful for reducing congestion and expanding the system but is not increasing the anonymity set.

5.5.6 Performance Evaluation

For a system with l layers and each layer having k mixes and $R = l * k$ total number of mixes, where M distinct messages were submitted for a given time-frame and the proportion M/k messages are sent to each mix. In a given time-frame each protocol participant has to perform the following operations. For cryptographic operations we adopt implementation techniques from electronic voting [27].

Lets assume our system is using the following cryptographic primitives. We suggest to implement commitments using HMAC. This is a well known efficient technique justified under the assumption that the hash function is collision resistant and, when keyed by the opening information, pseudo-random [5]. For the NIZK we assume [4] is used. and for the PoK we assume [7] is used. The signatures can be realized using [55] and the distributed decryption ([17] and the distributed key generation [36]. Our system has roughly O(1) operations for the user. The mixes carry out $O(|R| * |M|/|k| * log(\sqrt{(\frac{|M|}{|k|})})$ operations, the $\mathcal{RE}$ need to carry out $O(|R|)$ operations and the $\mathcal{AS}$ need to carry out $O(|AS| * |M| * |R|)$ operations.

5.5.7 Comparing Topologies

We review the differences between stratified topology and mix cascades. We omit a comparison to free-route topologies, because using them for verifiable mix networks is complicated by all mixes needing to produce proofs of shuffling. As for many aspect of anonymous communication networks topologies also present a trade-off between latency, anonymity, scalability and resilience.

On the one hand, if the adversarial mixes constitute a large proportion of all mixes, mix cascade topology is the best choice because a mix cascade provides anonymity as long at least one mix is honest. While a stratified topology (or a free-route topology) falls short on this scenario. On the other hand, if the adversary is controlling only a proportion of the mixes it is worthwhile considering stratified topology. We explain their advantages with the help of a simple example.

Let us assume the adversary is controlling 3 out of 9 mixes available for the system. In a mix cascade setting we can either have a mix cascade with 9 mixes, which is going to be very slow or two mix cascades one with 4 mixes and one with 5 mixes. In this latter case, we achieve better latency, however, the anonymity set is split in to, meaning that messages of one cascade are not mixed with the messages of the other. This is important because it becomes easier for the adversary

for example to carry out a $n-1$ attack. In terms if scalability, adding 3 mixes to the system does not improve latency or throughput. However, if 4 mixes are added another mix cascade can be built, where again the anonymity set becomes even more divided.

If we use a stratified topology we can use a network with 4-5 where each layer has 2 mixes. In this setting, we do not gain any latency advantage over the mix cascade setting, however, the network has a larger anonymity set. Let us assume we use a network with 3 layers where each layer has 3 mixes. In such a setting, the adversary can deanonymize 1/27 of all messages that enter the system but we are faster than the 3 previous settings because each message is going through fewer mixes. In terms of scalability, if three mixes are added in the first setting (4-5 layers and two mixes in each layer) the system can rearrange (4 layers and three mixes each).

5.5.8 Comparison to Parallel Mixes

Recall that in *Parallel Mixes* it is required to distribute the input messages uniformly before submitting them to the system to decrease the chances of carrying out attacks such as blending attacks. However, permuting incoming messages before entering the system to obtain a uniform distribution would need another random distribution mechanism and complicates the trust model. Furthermore, Parallel Mixes require two rotating phases before and after a single re-distribution phase where the input ciphertexts are rotated $m+1$ times, where m is the number of malicious mixes. Using MPR there is no need to enter only uniformly distributed messages to the system and the rotating phases of Parallel Mixes can be eliminated; this allows for multiple distribution phases instead of only one, which contributes to obtaining a complete random permutation of all ciphertexts.

5.5.9 Comparison to Atom

Atom has many similarity to our system in terms of the cryptographic primitives and the layers/groups of servers/mixes.

Atom uses a square permutation suggested by Hastad []. Atom uses a stratified topology mix network, where each mix is replaced by a sequence of servers (called a group) that has at least one honest server among them. In this sense, Atom has similarity to Parallel Mixes by Juels and Golle [], which used forwarding phases using cascades and a distribution phase to mix the input of the cascades for the next forwarding phase.

Atom offers two variants for checking the correctness of shuffling. One option is similar to our use of NIZKs and the second option is to use trap-based checking that is weaker in anonymity and offers better efficiency. However, using trap-based checking has two main flaws. First, it increases the input of the system significantly. In order to make the probability of hitting (eliminating or replacing) a trap message compared to hitting (eliminating or replacing) a real message 50 % for each message submitted to the system a trap message needs to be sent too. Second, it makes the system vulnerable towards users and users can make system halt, which eliminates any resistance. Atom suggests to identity malicious users, but users can join the system using another IP or alias, hence, making this countermeasure ineffective.

In comparison to Atom, we have the same cryptographic primitives and we also use stratified topology, with the difference that we are not replacing the mixes in the topology with cascades, but instead the distribution phase in our protocol needs a randomness that needs to be produced after a mix has mixed the input ciphertexts and the output ciphertext are distributed according to this randomness.

AC System	Adversary[1]	Topology	Routing Decision	Routing Integrity	Scalability	Availability
MPR	G/A/I	Stratified	Random	Yes	High	High
Parallel Mixes	G/A/E	Stratified	Mix decision	No	High	Low
cMix	G/A/I	Cascade	Fixed	Yes	Low	Low
Loopix	G/A/I	Stratified	User-decision	Yes	High	Low
Atom	G/A/I	Stratified	Mix decision	No	High	Low

Table 5.1: Comparison with other mix-nets.

5.5.10 Comparison to cMix, Loopix, and Tor

Javani et al. designed in 2017 *cMix* which is an mix network consisting of a set of mixes organized in a mix cascade, where expensive computation are precomputed to improve the online performance.

Loopix is a mix-based anonymous communication system that uses Poisson mixes which are a simplification of [40], where the users choose the sequence of mixes based on the given stratified topology and sets a delay for each mix routing their message. In Loopix, users and mixes monitor whether the network is forwarding traffic correctly by loops, similar to RGB mixes [25].

In Table 5.1 we summarize the comparison of our system with the AC systems reviewed above.

Tor [32] is the most commonly used anonymous communication system with over 2 million users per day [50]. In Tor, the communication sender initiates the anonymous route by randomly selecting three Tor servers and encrypting the communication in a layered fashion. When the Tor servers receive the communication it removes a layer of encryption and forward it to the next Tor server. Tor provides low-latency service because the communication is immediately forwarded and there is no delay in the routing phase. However, this makes Tor traffic vulnerable to timing attacks, hence, Tor's threat model is assuming an adversary that has only local reach. This is helpful in many use cases such as anonymous web browsing, but is not strong enough if the user votes or casts an opinion against a very powerful adversary. Hence, all the input doesn't need to be shuffled with each other and all outgoing connections are not roughly equally likely corresponding to a given input. Local adversaries are reasonable if the system has a large user-base and many relays, Tor has 7000 volunteer relays. The anonymity Tor provides relies on the fact that Tor servers are wide-spread geographically and only a very strong adversary would be able to observe all Tor relays (or at least the entry and exit relay) that are re-routing targeted communication. Hence, Tor does not provide verifiable anonymity but rather provides good odds.

5.6 Discussion

Decentralization Advantages: MPR decentralizes the routing decision. Troncoso et al. [61] present numerous advantages of decentralization in protocol design such as increasing the adversaries cost and facilitating public verifiability. Our MPR protocol also increases the cost of the adversary for attacking the routing integrity and facilitates public verifiability.

Bulletin Broad The Bulletin Boards is the main communication channel that 1) guarantees accessibility, 2) is append-only, and 3) keeps the memory of all communication until a given time-frame ends. Bulletin boards are an essential component of many cryptographic designs, in particular, when verifiability is needed such as electronic voting protocols [15]. The information of the bulletin board needs to be there until the end of the time-frame, after that it needs to be erased.

[1] G=Global, A=Active, I=Internal, E=External

Mixes, routing entities, and auditing servers can write on the bulletin board during the protocol run of a time-frame and user can write on the bulletin board before a time-frame starts. Everyone can read the information on the bulletin board. Since, our bulletin board is the main channel of communication among the all the entities and since it contains all the messages that are sent through the system it needs to be large in size. Since, broadcasting is easiest way to implement bulletin boards, they are often referred to as Broadcast Channels with memory. The design of our protocol also relies on the secure and robust implementation of the bulletin board. Literature has suggested several methods for implementing bulletin boards. Examples of implementations of bulleting board are [18] [39].

Latency Our protocol is a medium-latency anonymous communication network and thus can be deployed for applications that are more tolerant to latency such as anonymous wiki, micro-blogging, voting, and auctions. Note that for application such as voting with homomorphic tallying the decryption phase of our protocol may not be necessary for all messages. We estimate for a system with 9 mixes having equal throughput, arranged in 3 layers each layer having the throughput to relay 900 messages, and where 1000 messages were sent to the system, messages takes about 20 seconds to traverse the system (before the decryption process starts). This is, however, only a preliminary result tested on a Intel Xeon E5-2640 CPU, 2.50 GHz machine with 24 cores, where all computation are performed online.

Load Balancing: Since our MPR protocol offers load balancing, the messages leaving the system have anonymity sets with more uniform sizes compared to mixnets using source routing where routes are selected without load balancing. This increases the anonymity a message can achieve. Accommodating load balancing comes with the risk that an internal adversary can attract more messages than other mixes. However, limiting load balancing would result in fixed size mixes and this would lead to wasting otherwise available bandwidth. Considering that a system providing service for a larger number of users also provides better anonymity this trade-off needs to be considered. Another solution is to place mixes that have the same size in the same layer. Since the total throughput of each layer has to be equal, layers with smaller mixes will have more mixes than layers with big mixes.

Limitation of our System MPR addresses the scalability issue of source routing in terms of not requiring a complete system view for users. Growing our system is, however, not as scalable as growing a system with a free topology because the system's throughput can be only expanded by adding mixes to all layers. When there is no restrictions for selecting a relay, adding a single relay increases the system's overall throughput.

Appendices

In this section, we give instantiations of routing assignment and routing verification and a comparison to other AC systems.

A. Routing Assignment

To route a ciphertext, at each layer, a relay is assigned to the ciphertext using a mapping function. The mapping function is composed of a permutation function keyed by the jointly generated random number *Rand* and a load-balancing function that distributes ciphertext based on the throughput of relays in the next layer.

The permutation algorithm is the function `H` out of hash function `h`. `H` is applied iteratively on the inputs: joint random number *Rand* and the ciphertext indexes $1, \ldots, w$ and returns the per-

$\texttt{H}(Rand, 1, ..., w)$
  $Z = \emptyset$
  $i = 1$
  for $i = 1$ to w
    $j = 0$
    $z_i = h(Rand \| i \| j)$
    while $z_i \in Z$
      $j = j + 1$
      $z_i = h(Rand \| i \| j)$
    end while
    $Z \leftarrow Z \cup z_i$
  end for
 $output(z_1, \ldots, z_w)$

Algorithm 2: Permuting with Jointly Obtained *Rand*

$\texttt{Map}(z_1, ..., z_w, r_1, ..., r_k, b_1, ..., b_k)$
  $B = \texttt{SUM}(b_1, ..., b_k)$
  for $j = 1$ to k
    r_j `fetch` $w * bj/B$ ciphertexts from bulletin board
  end for
 Assigns ciphertext to mixes in proportion to the mixes throughput.

Algorithm 3: Mapping ciphertexts

muted indexes $z_1, \ldots, z_w$. To realize the permutation functionality we have to avoid hash function collisions by performing a check of membership on a set Z of already computed index values.

We apply the mapping function `Map`, e.g., Algorithm 3 in appendices, to make the assignment of the permuted ciphertext (indexed by `H`) to the relays in the next layer in proportion to their throughput. We refer to the throughput of relays by b, where throughput refers to the number of messages it can route in proportion to all the number of message that can be routed by a layer and the throughput of a single layer is denoted B. For this purpose we use the function `fetch`, which ciphertexts to mixes in the next layer. These relays fetch the ciphertext that is assigned to them as their input from the bulletin board.

B. Routing Verification

Let C_r refer to the set of input ciphertext that relay r has submitted and signed and C'_r to the set of output ciphertexts that relay r has submitted and signed. Moreover, let us denote the *ctr*-th layer of routers in our system by R_{ctr}.

Then all the auditors must carry out the routing verification as described below. Our algorithm applies a Boolean `VerifyRand` function which simply verifies both the correctness of the opening of the commitments `Com`($rand$) with regard to the actual $rand$ on the bulletin board and checks that their sum equals $Rand$. Moreover, the algorithm verifies whether the input of all relays in the $ctr + 1$ layer is correct according to the output ciphertexts of relays in layer ctr and their corresponding $Rand$s.

```
VerifyRouting(Rand_r,C'_r, R_{ctr+1})
    if VerifyRand returns 1
      (c_1, ..., c_w) =| C'_r |
      (z_1, ..., z_w) = H(Rand_r, 1, ..., w)
      (r_1, ..., r_w) = Map(c_1, ..., c_w, R_{ctr+1}, B_{ctr+1})
      for j = 1 to w
        if relay r_j signed c_j
          continue
        else return ("Routing error", r_j)
        end if
      end for
    else if return ("Commitment error")
    return (1)
```

Algorithm 4: Verify Routing

Bibliography

[1] Tor Project: Anonymity Online. `https://www.torproject.org/`. Last accessed: July 30, 2017.

[2] B. Adida. Helios: Web-based open-audit voting. In *Proceedings of the 17th Conference on Security Symposium*, SS'08, pages 335–348, Berkeley, CA, USA, 2008. USENIX Association.

[3] N. Alexopoulos, A. Kiayias, R. Talviste, and T. Zacharias. Mcmix: Anonymous messaging via secure multiparty computation. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1217–1234, Vancouver, BC, 2017. USENIX Association.

[4] S. Bayer and J. Groth. *Efficient Zero-Knowledge Argument for Correctness of a Shuffle*, pages 263–280. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[5] M. Bellare, R. Canetti, and H. Krawczyk. Message authentication using hash functions- the hmac construction. *CryptoBytes*, 2, 1996.

[6] D. Bernhard, V. Cortier, O. Pereira, B. Smyth, and B. Warinschi. Adapting helios for provable ballot privacy. In *Computer Security - ESORICS 2011 - 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings*, pages 335–354, 2011.

[7] D. Bernhard, V. Cortier, O. Pereira, B. Smyth, and B. Warinschi. *Adapting Helios for Provable Ballot Privacy*, pages 335–354. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[8] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A system for anonymous and unobservable internet access. In *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings*, pages 115–129, 2000.

[9] M. Blum. Coin flipping by telephone a protocol for solving impossible problems. *SIGACT News*, 15(1):23–27, Jan. 1983.

[10] N. Borisov. An analysis of parallel mixing with attacker-controlled inputs. In *Proceedings of the 5th International Conference on Privacy Enhancing Technologies*, PET'05, pages 12–25, Berlin, Heidelberg, 2006. Springer-Verlag.

[11] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[12] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[13] I. Clarke, O. Sandberg, M. Toseland, and V. Verendel. Private communication through a network of trusted connections: The dark freenet. *Network*, 2010.

[14] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, pages 46–66. Springer-Verlag New York, Inc., 2001.

[15] M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a secure voting system. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, SP '08, pages 354–368, Washington, DC, USA, 2008. IEEE Computer Society.

[16] H. Corrigan-Gibbs and B. Ford. Dissent: Accountable anonymous group messaging. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 340–350, 2010.

[17] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Proceedings of the 16th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'97, pages 103–118, Berlin, Heidelberg, 1997. Springer-Verlag.

[18] C. Culnane and S. Schneider. A peered bulletin board for robust use in verifiable voting systems. In *Proceedings of the 2014 IEEE 27th Computer Security Foundations Symposium*, CSF '14, pages 169–183, Washington, DC, USA, 2014. IEEE Computer Society.

[19] A. Czumaj. Random permutations using switching networks. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 703–712, New York, NY, USA, 2015. ACM.

[20] A. Czumaj, P. Kanarek, K. Loryś, and M. Kutylowski. *Switching Networks for Generating Random Permutations*, pages 25–61. Springer US, Boston, MA, 2001.

[21] G. Danezis. *Statistical Disclosure Attacks*, pages 421–426. Springer US, Boston, MA, 2003.

[22] G. Danezis. Breaking four mix-related schemes based on universal re-encryption. *International Journal of Information Security*, 6(6):393–402, 2007.

[23] G. Danezis and R. Clayton. Route fingerprinting in anonymous communications. In *Peer-to-Peer Computing, 2006. P2P 2006. Sixth IEEE International Conference on*, pages 69–72. IEEE, 2006.

[24] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *2003 IEEE Symposium on Security and Privacy (SP 2003), 11-14 May 2003, Berkeley, CA, USA*, pages 2–15, 2003.

[25] G. Danezis and L. Sassaman. Heartbeat traffic to counter (n-1) attacks: Red-green-black mixes. In *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*, WPES '03, pages 89–93, New York, NY, USA, 2003. ACM.

[26] G. Danezis and P. Syverson. Bridging and fingerprinting: Epistemic attacks on route selection. In *Proceedings of the 8th International Symposium on Privacy Enhancing Technologies*, PETS '08, pages 151–166, Berlin, Heidelberg, 2008. Springer-Verlag.

[27] A. M. Davis, D. Chmelev, and M. R. Clarkson. Civitas: Implementation of a threshold cryptosystem. Technical report, Department of Computer Science Cornell Universiy, 2008.

[28] Y. G. Desmedt and Y. Frankel. Threshold cryptosystems. In *Proceedings on Advances in Cryptology*, CRYPTO '89, pages 307–315, New York, NY, USA, 1989. Springer-Verlag New York, Inc.

[29] C. Diaz. Anonymity metrics revisited. In S. Dolev, R. Ostrovsky, and A. Pfitzmann, editors, *Anonymous Communication and its Applications*, number 05411 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2006. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[30] C. Diaz, S. J. Murdoch, and C. Troncoso. Impact of network topology on anonymity and overhead in low-latency anonymity networks. In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, PETS'10, pages 184–201, Berlin, Heidelberg, 2010. Springer-Verlag.

[31] C. Díaz and B. Preneel. Reasoning about the anonymity provided by pool mixes that generate dummy traffic. In J. Fridrich, editor, *Information Hiding*, volume 3200 of *Lecture Notes in Computer Science*, pages 309–325. Springer Berlin Heidelberg, 2005.

[32] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM '04, pages 303–320. USENIX Association, 2004.

[33] R. Dingledine, V. Shmatikov, and P. F. Syverson. Synchronous batching: From cascades to free routes. In *Privacy Enhancing Technologies, 4th International Workshop, PET 2004, Toronto, Canada, May 26-28, 2004, Revised Selected Papers*, pages 186–206, 2004.

[34] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. Blakley and D. Chaum, editors, *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer Berlin Heidelberg, 1985.

[35] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. *Secure Distributed Key Generation for Discrete-Log Based Cryptosystems*, pages 295–310. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[36] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20(1):51–83, 2007.

[37] P. Golle and A. Juels. Parallel mixing. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, CCS '04, pages 220–226, New York, NY, USA, 2004. ACM.

[38] M. T. Goodrich and M. Mitzenmacher. *Anonymous Card Shuffling and Its Applications to Parallel Mixnets*, pages 549–560. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[39] S. Hauser and R. Haenni. A generic interface for the public bulletin board used in univote. In *2016 Conference for E-Democracy and Open Government (CeDEM)*, pages 49–56, May 2016.

[40] D. Kesdogan, J. Egner, and R. Büschkes. Stop-and-Go-MIXes providing probabilistic anonymity in an open system. In *Information Hiding, Second International Workshop, Portland, Oregon, USA, April 14-17, 1998, Proceedings*, pages 83–98, 1998.

[41] M. Klonowski and M. Kutyłowski. *Provable Anonymity for Networks of Mixes*, pages 26–38. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[42] A. Kwon, H. Corrigan-Gibbs, S. Devadas, and B. Ford. Atom: Scalable anonymity resistant to traffic analysis. *arXiv preprint arXiv:1612.07841*, 2016.

[43] J. Loftus, A. May, N. P. Smart, and F. Vercauteren. On cca-secure somewhat homomorphic encryption. In *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, pages 55–72, 2011.

[44] U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster protocol - version 2, 2003.

[45] M. Movahedi, J. Saia, and M. Zamani. *Secure Multi-party Shuffling*, pages 459–473. Springer International Publishing, Cham, 2015.

[46] T. P. Pedersen. A threshold cryptosystem without a trusted party. In *Proceedings of the 10th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'91, pages 522–526, Berlin, Heidelberg, 1991. Springer-Verlag.

[47] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '91, pages 129–140, London, UK, UK, 1992. Springer-Verlag.

[48] A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity - A proposal for terminology. In *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings*, pages 1–9, 2000.

[49] A. M. Piotrowska, J. Hayes, T. Elahi, S. Meiser, and G. Danezis. The loopix anonymity system. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1199–1216, Vancouver, BC, 2017. USENIX Association.

[50] T. T. Project. Tor metrics. `https://metrics.torproject.org/`. Last accessed: August 05, 2015.

[51] J.-F. Raymond. *Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems*, pages 10–29. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[52] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, November 1998.

[53] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-peer based anonymous internet usage with collusion detection. In *Proceedings of the 2002 ACM Workshop on Privacy in the Electronic Society*, WPES '02, pages 91–102, New York, NY, USA, 2002. ACM.

[54] L. Schimmer. Peer profiling and selection in the I2P anonymous network. In *Proceedings of PET-CON 2009.1*, pages 59–70, March 2009.

[55] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '89, pages 239–252, London, UK, UK, 1990. Springer-Verlag.

[56] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.

[57] F. Shirazi, C. Diaz, and J. Wright. Towards measuring resilience in anonymous communication networks. In *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society*, WPES '15, pages 95–99, New York, NY, USA, 2015. ACM.

[58] F. Shirazi, M. Simeonovski, M. R. Asghar, C. Diaz, and M. Backes. Survey on Routing in Anonymous Communication Networks. Technical report, 2016.

[59] R. Shokri, N. Yazdani, and A. Khonsari. Chain-based anonymous routing for wireless ad hoc networks. In *2007 4th IEEE Consumer Communications and Networking Conference*, pages 297–302, Jan 2007.

[60] V. Shoup and R. Gennaro. *Securing threshold cryptosystems against chosen ciphertext attack*, pages 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[61] C. Troncoso, G. Danezis, M. Isaakidis, and H. Halpin. Systematizing decentralization and privacy: Lessons from 15 years of research and deployments. *arXiv preprint arXiv:1704.08065*, 2017.

[62] Q. Wang and N. Borisov. Octopus: A secure and anonymous DHT lookup. *CoRR*, abs/1203.2668, 2012.

[63] D. Wikström. A commitment-consistent proof of a shuffle. In *Information Security and Privacy, 14th Australasian Conference, ACISP 2009, Brisbane, Australia, July 1-3, 2009, Proceedings*, pages 407–421, 2009.

[64] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson. Dissent in numbers: Making strong anonymity scale. In *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, OSDI '12, pages 179–192. USENIX Association, 2012.

Part III

Communication Patterns & Fingerprinting Techniques

6. Automated Website Fingerprinting through Deep Learning

6.1 Introduction

The Onion Router (Tor) is a communication tool that provides anonymity to Internet users. It is an actively developed and well-secured system that ensures the privacy of its users' browsing activities. For this purpose, Tor encrypts the contents and routing information of communications, and relays the encrypted traffic through a randomly assigned route of nodes such that only a single node knows its immediate peers, but never the origin and destination of a communication at the same time. Tor's architecture thus prevents ISPs and local network observers from identifying the websites users visit.

As a result of previous research on Tor privacy, a serious side-channel of Tor network traffic was revealed that allowed a local adversary to infer which websites were visited by a particular user [14]. The identifying information leaks from the communication's meta-data, more precisely, from the directions and sizes of encrypted network packets. As this side-channel information is often unique for a specific website, it can be leveraged to form a unique fingerprint, thus allowing network eavesdroppers to reveal which website was visited based on the traffic that it generated.

The feasibility of Website Fingerprinting (WF) attacks on Tor was assessed in a series of studies [25, 31, 19, 24, 32]. In the related works, the attack is treated as a classification problem. This problem is solved by, first, manually engineering features of traffic traces and then classifying these features with state-of-practice machine learning algorithms. Proposed approaches have been shown to achieve a classification accuracy of 91-96% correctly recognized websites [30, 24, 13] in a set of 100 websites with 100 traces per website. Their works show that finding distinctive features is essential for accurate recognition of websites. Moreover, this tasks can be costly for the adversary as he has to keep up with changes introduced in the network protocol [4, 20, 9]. The WF research community thus far has not investigated the success of an attacker who automates the feature extraction step for classification. This is the key problem that we address in this work.

An essential step of traditional machine learning is feature engineering. Feature engineering is a manual process, based on intuition and expert knowledge, to find a representation of raw data that conveys characteristics that are most relevant to the learning problem. Feature engineering proved to be even more important than the choice of the specific machine learning algorithm in many applications, including WF [12, 19].

When developing a new WF attack, prior work on WF typically focuses on feature engineering to compose and select the most salient features for website identification. Moreover, these attacks are actually defined by a fixed set of features derived from this process. Thus, these attacks are sensitive to changes in the traffic that would distort those features. In particular, deploying countermeasures in the Tor network that conceal the features is sufficient to defend against such

attacks. This enables an arms-race between attacks and defenses: new attacks defeat defenses because they exploit features that had not been considered before and, conversely, new defenses are designed to conceal the features that those attacks exploited.

In this chapter, we propose a novel WF attack based on deep learning. Our attack incorporates automatic feature learning and, thus, it is not defined by a particular feature set. This may be a game-changer in the arms-race between WF attacks and defenses, because the deep learning based attack is designed to be adaptive to any perturbations in the features introduced by defenses. The attack we present in this work is the first automated WF attack and it is at least as effective as the state-of-the-art, manual approaches.

6.1.1 Key Contributions for PANORAMIX

- This study provides the first systematic exploration of state-of-the-art deep learning (DL) algorithms applied to WF, namely feedforward, convolutional and recurrent deep neural networks. We design, tune and evaluate three models – Stacked Denoising Autoencoder (SDAE), Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM). Our DL models are capable of *automatically* learning traffic features for website recognition at the expense of using more data. Moreover, we automate the model selection to find the best network hyperparameters. We demonstrate that our DL-based WF attack reaches a high success rate, comparable to the state-of-the-art techniques.

- We reevaluate prior work on our dataset and reproduce their results. We find that state-of-the-art WF approaches benefit from using more training data, similar to DL. As a result of a systematic comparison of our novel DL-based methods to previous WF approaches for the closed and open world settings, we demonstrate comparable recognition results with slight improvements of up to 2%. Furthermore, we show that our DL attack reveals more general and stable website features than the state-of-the-art methods, which makes them more robust to concept drift caused by highly dynamic web content.

- The dataset collected for the evaluation is the largest WF dataset ever gathered to date. Our closed-world dataset consists of 900 websites, with traffic traces generated by 2,500 visits each. Our open-world dataset is based on 400,000 unknown websites and 200 monitored websites. We made the generated dataset publicly available, allowing researchers to replicate our results and systematically evaluate new (DL) approaches to WF[1].

6.2 Background

This section reviews recent related work on Tor WF attacks relying on traditional machine learning algorithms, and the application of deep learning.

Anonymous communications systems such as Tor [11] provide confidentiality of communications and conceal the destination server's address from network eavesdroppers. However, in the last decade, several studies have shown that, under certain conditions, an attacker can identify the destination website only from encrypted and anonymized traffic.

In WF, the adversary collects traffic from his own visits to a set of websites that he is interested in monitoring, visiting each site multiple times. Next, the adversary builds a website *template* or *fingerprint* from the traffic traces collected for that site. The fingerprints are built using a supervised

[1]The dataset and implementation can be found on the following URL: `https://distrinet.cs.kuleuven.be/software/tor-wf-dl/`.

learning method that takes the traffic traces labeled as their corresponding site, extracts a number of features that identify the site and outputs a statistical model that can be used for classification of new, unseen traffic traces. Finally, the attacker applies the classifier on unlabeled traffic traces collected from communications initiated by the victim and makes a guess based on the output of the classifier. To be able to deploy the attack, the adversary must be able to observe the traffic generated by the victim and be able to identify the user (see Section 6.3 for more details on the threat model).

The first WF studies evaluated the effectiveness of the attack against HTTPS [8], encrypted web proxies [27, 16], OpenSSH [22] and VPNs [14] and it was not until 2009 that the first evaluation of a WF attack was performed in Tor [14]. This first attack in Tor was based on a Naive Bayes classifier and the features were the frequency distributions of packet lengths [14]. Even though their evaluation showed the attack achieved an average accuracy of only 3%, the attack was improved by Panchenko et al. using a Support Vector Machine (SVM) [25]. In addition, Panchenko et al. added new features that were exploiting the distinctive *burstiness* of traffic and increased the accuracy of the attack to more than 50%.

These works were succeeded by a series of studies that claimed to boost the attacks and presented attacks with more than 90% success rates. First, Cai et al. [5] used an SVM with a custom kernel based on an edit-distance and achieved more than 86% accuracy for 100 sites. The edit distance allowed for delete and transpose operations, that are supposed to capture drop and retransmission of packets respectively. Following a similar approach, Wang and Goldberg [31] experimented with several custom edit distances and improved Cai et al.'s attack to 91% accuracy for the same dataset.

However, these evaluations have been criticized for making unrealistic assumptions on the experimental settings that give an unfair advantage to the adversary compared to real attack settings [19]. For instance, they evaluated the attacks on small datasets and considered adversaries who can perfectly parse the traffic generated by a web-page visit from all the traffic that blends into the Tor network. Furthermore, they assume users browse pages sequentially on one single browser tab and never interrupt an ongoing page-load. Recent research has developed new techniques to overcome some of these assumptions, suggesting that the attacks may be more practical than previously expected [32].

The three most recent attacks in the literature outperform all the attacks described above and, for this reason, we have selected them to compare with our DL-based attack. Each attack uses a different classification model and feature sets and work as follows:

Wang-kNN [30]: this attack is based on a k-Nearest Neighbors (k-NN) classifier with more than 3,000 traffic features. This large amount of features is obtained by varying the parameters of set of fewer feature *families*. For instance, the number of outgoing packets in spans of X packets and the lengths of the Y packets in the same direction. In order to mitigate the curse of dimensionality, they proposed to weigh the features of a custom distance metric, minimizing the distance among traffic samples that belong to the same site. Their results show that this attack achieves 90% to 95% accuracy on 100 websites [30].

CUMUL [24]: CUMUL is based on an SVM with a Radial Basis Function (RBF) kernel. CUMUL uses the cumulative sum of packet lengths to derive the features for the SVM. The cumulative sum is computed by adding the lengths of outgoing packets and subtracting the lengths of incoming packets. However, since the RBF kernel, in contrast to the aforementioned edit-distance based SVM kernel, expects feature vectors to have the same dimension, they interpolated 100 points from the cumulative sums. Furthermore, they prepend the total incoming and outgoing number of packets and bytes. As a result, they ended with 104 features to represent a traffic instance. Their evaluations demonstrate an attack success that ranges between 90% and 93% for 100 websites. It

is worth mentioning that their dataset is the most realistic up to the date, including *inner* pages of sites that have spikes of popularity such as Google searches or Twitter links. Despite the high success rate of their attack, the authors conclude that the WF attack does not scale when applied in a real-world setting, as an adversary would need to train the classifier on a large fraction of all websites.

k-Fingerprinting (k-FP) [13]: Hayes and Danezis's k-FP attack is based on Random Forests (RF). Random Forests are ensembles of decision trees that are randomized and averaged so that they can generalize better than simple decision trees. Their feature sets include 175 features developed from features available in prior work, as well as timing features that had not been considered before, such as the number of packets per second. The random forest is not used to classify but as a way to transform these features into a different feature space: they use the leafs of the random forest to encode a new representation of the sites they intent to detect that is relative to all the other sites in their training set. Next, the new representation of the data is fed to a k-NN classifier for the actual classification. Their results show that this attack is as effective as CUMUL and achieves similar accuracy scores for the same number of sites.

All these attacks have selected their features mostly based on expertise and their technical knowledge on how Tor and the HTTP protocol work and interact with each other. As a result of manual feature engineering and standard feature selection, each proposed attack can be represented by a set of fingerprinting features. It is still unknown whether WF can be successfully deployed through automatic feature engineering based on implicit uninterpretable traffic features.

To the best of our knowledge, the only research that successfully applies deep learning to a similar problem is the network protocol recognition on encrypted traffic with a Stacked Denoising Autoencoder (SDAE) done by Wang [34]. His approach achieves a 90% recognition rate, which is a promising indicator for deep learning application to anonymized traffic.

The first effort to apply a DL-based approach to WF was made by Abe and Goto [1], where they evaluated a SDAE on the Wang-kNN's dataset. Their classifiers do not outperform the state-of-the-art, but nevertheless achieve a convincing 88% on a closed world of 100 classes. It is fair to assume that the lower performance is due to the lack of a sufficient amount of training data for a deep neural network, which, as we confirm later in our paper, is essential for the deep learning performance. Moreover, the work does not assess applicability of other deep learning algorithms to the problem. In this chapter we explore three deep learning methods when applied to a significantly larger closed world of varying sizes, trained on sufficient amounts of data and evaluated in context of dynamic changes of web content over time. We provide a more extensive tuning of the DL-based attacks and finally achieve a similar accuracy to the state-of-the-art WF attacks.

6.3 Threat model

In this chapter we consider an adversary similar to the one considered in prior work in WF, namely a *passive* and *local* network-level adversary. Figure 6.1 shows an overview of this WF scenario. A passive adversary only records network packets transmitted during the communication and may not modify them or cause them to drop, and may not insert new packets into the stream of packets. A local adversary has a limited view of the network. In particular, in Tor, such an adversary typically owns the entry node to the Tor network (also known as *entry guard*), or has access to the link between the client and the entry. Examples of entities that have this level of visibility range from Internet Service Providers (ISP), Autonomous Systems (AS) or even local network administrators. Note that an adversary that owns the entry guard can decrypt the first layer of encryption and access Tor protocol messages. In this work, we assume an ISP-level adversary that collects traffic at

the TCP layer and infers the cells from TCP packets [31]. Obviously, all work on WF assumes the adversary cannot decrypt the encryption provided by Tor, as message contents would immediately reveal the identity of the website.

In the WF literature, it is common for the evaluation of the attack to assume a closed world of websites. This means that the user can only visit pages that the adversary has been able to train on. This assumption, commonly known as the *closed-world* assumption, has been deemed unrealistic [25] as the size of the Web is so large that an adversary can only train on a tiny fraction of the Web. For this reason, many studies have also evaluated the more realistic *open world*, where the user is allowed to visit pages that the adversary has not trained on. The closed world is still useful to compare existing attacks and defenses. In this study, we evaluated both the closed world and the open world.

6.4 Data collection

One of the prerequisites for deep learning is an abundance of training data required to learn the underlying patterns. Processing sufficient amounts of representative data enables the deep neural network to not only precisely reveal the identifying features but also generalize better to unseen test instances. In prior work on WF in the context of Tor, the datasets that were collected are relatively limited in size, both in terms of classes (i.e. the number of unique websites) as well as instances (i.e. the number of traffic traces per website). To properly evaluate our proposed deep learning approach and explore how existing models can benefit from extra training data, we used a distributed setup to collect various new datasets that accommodate these requirements.

6.4.1 Data collection methodology

For the data collection process, we used 15 virtual machines on our OpenStack-based private cloud environment. Each VM was provisioned with 4 CPUs and 4GB of RAM. To each VM, 16 worker threads were assigned, which each had their separate `tor` process (version 0.2.8.11). Page-visit tasks, consisting of starting the Tor browser (version 6.5) and loading the target web page, were then distributed among the 240 concurrent worker threads. Web pages were given 285 seconds to load, before the browser was killed and the visit marked as invalid. Upon loading the page, it was left open for an additional 10 seconds, after which the browser was closed and any profile information was removed.

By leveraging network namespaces and `tcpdump`, we isolated and captured the traffic of each `tor` process. Due to storage constraints, and since the packet payloads are encrypted and thus do not have value for the adversary, we extract meta-data from the traffic trace and discard the encrypted payload. More precisely, we capture (1) the timing information, (2) the direction and

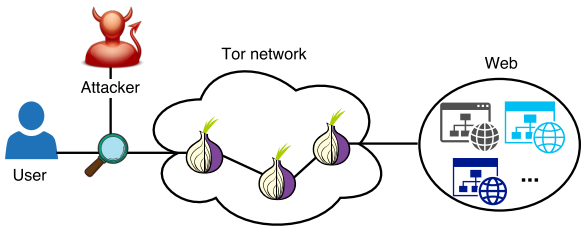


Figure 6.1: The client visits a website over the Tor network. The adversary can observe the (encrypted) traffic between the client and the *entry* to the Tor network.

(3) the size of the TCP packet. We follow the approach proposed by Wang and Goldberg [31] to extract Tor cells from the captured TCP packets. Our final representation of the traffic trace is a sequence of cells, where each cell is encoded as 1 when transmitted from the client to the website and as -1 when captured in the opposite direction. For the purpose of sanity checks and validation, information on the Tor circuit that was used for the page visit is also recorded.

It should be noted that, in contrast to prior work [31], the Tor entry guard node was not pinned over the course of our experiments. The reason for this is twofold. First, compared to prior data collection, we use significantly more concurrent processes. If the same entry guard would be used by the 240 browser instances, this could overload the entry guard, possibly affecting the network traces. Second, by using a variety of entry guards, the trained models are agnostic to the intrinsics of a specific entry guard. This means that the model of the adversary is not only applicable in a targeted attack on a single victim, but can be launched against any Tor user.

6.4.2 Datasets

Since the WF adversary's goals might vary widely and as there are no statistics about which pages Tor users browse to, there can be no definitive set of sensitive websites for WF research. Moreover, since we aim to compare various approaches with each other, the actual choice of websites is not essential as long as it is consistent. The list of websites we chose for our evaluation comes from the Alexa Top Sites service, the source widely used in prior research on Tor.

In total, we evaluate our deep learning approach in comparison with traditional methods on three different datasets. This section details how these datasets were chosen and obtained.

Closed world

For the dataset under the *closed world* assumption, we collected up to 3,000 network traces for visits to the homepage of the 1,200 most popular websites according to Alexa. The list of popular websites was first filtered to remove duplicate entries that only differ in the TLD, e.g. in the case of `google.com` and `google.de`, only the former was included in the list. Data for these 1,200 websites was collected in four iterations, consisting of 300 websites each. An iteration was again split up into 30 batches, with each batch performing 100 network traces per websites. After each batch, the 240 `tor` processes were restarted and data directories were removed, forcing new circuits to be built with (new) randomly selected entry guards. Network traces for each of the four iterations were collected over approximately 14 days per group, starting from January 2017.

After collecting data on the 3.6 million page visits, we filtered out invalid entries, which were due to a timeout, or a crash of the browser or Selenium driver. Websites with a high amount of invalid page visits were removed from our dataset. Additionally, using the similarity hash of the web page's HTML content [7] and the perceptual hash of the screenshot [3], we detected and excluded websites with exactly the same content. Moreover, we filtered out websites that had no content, denied all requests coming from Tor, or showed a CAPTCHA for every visit. Finally, we balanced the dataset to ensure the uniform distribution of instances across different sites by fixing the same number of traces for every site. After this filtering process, our biggest closed world dataset consists of 900 websites, with 2,500 valid network traces each. In the remainder of the text, we refer to this dataset as CW_{900}. Similarly, for datasets that are composed of a subset of this one we use a corresponding representation: the datasets for the top 100, 200 and 500 websites are referred to as CW_{100}, CW_{200} and CW_{500} accordingly.

Revisit over time

For the top 200 websites, we obtained additional periodic measurements. More precisely, for these websites we collected 100 test network traces per website 3 days, 10 days, 4 weeks, 6 weeks and 8 weeks after the end of the initial data collection for these 200 websites. Each test set is collected within one day. As a result, our *revisit-over-time* dataset provides 500 network traces for each of the top 200 websites collected over a 2-month period (CW_{200} was collected over 2 weeks).

Open world

Since the *open world* data is only used for testing purposes (which differs from some of the open world evaluations), we collected only a single instance for each page in the open world. In total, we collected network traces for the top 400,000 of Alexa websites.

We collected additional 2,000 test traces for each website of the monitored closed world CW_{200} (400,000 instances in total). As a result, we conduct the open world evaluation on 800,000 test traffic traces, half from the closed world and half from the open world (a 4-fold increase compared to the largest dataset considered in prior work [13, 24]). We provide the motivation for this experimental setting in Section 6.5.2.

6.4.3 Ethical considerations & data access

For our data collection experiments, we performed around 4 million page visits over Tor. It is highly unlikely that this had any impact on the top websites, which each receive multiple millions of requests every day. We consider the impact on the Tor network to be limited as well: The Tor Project estimates that during the time we performed our experiments, approximately 2 million clients were concurrently connected to the Tor network. As such, the 240 clients we used are only a minor fraction of the total number of active clients. Furthermore, we made the data publicly available, allowing other researchers to evaluate other approaches without having to collect new data samples.

6.5 Evaluation

In this section, we conduct a reevaluation of the state-of-the-art WF methods discussed in the related work of Section 6.2 to confirm their reproducibility on our dataset. We then evaluate the proposed attacks based on the three chosen deep learning (DL) algorithms and compare them to the previously known techniques.

6.5.1 Reevaluation of state-of-the-art

We aim to enable a systematic comparison between our work and that of Wang et al. [30], Panchenko et al. [24] and Hayes et al. [13], not only to guarantee a fair assessment by evaluating on new data, but also to analyze (1) the practical feasibility of the attack on a significantly larger set of websites, (2) the impact of collecting more instances or traces per website on the classification accuracy, and (3) the resilience of trained models to concept drift with a growing time gap between training and testing.

The goal of the first closed world experiment is to confirm whether we can reproduce the three WF attacks of prior work [30, 24, 13] and to assert whether we obtain similar classification results as those reported by the respective authors, but on a different training and testing dataset similar in size. We reuse the original implementation of the authors to carry out the feature extraction

and subsequently execute the training and testing steps. All results reported in this section are computed via 10-fold cross-validation.

The following results were obtained on a Dell PowerEdge R620 server with 2x Intel Xeon E5-2650 CPUs, 64GB of memory and 8 cores on each CPU with hyperthreading, resulting in 32 cores in total each running at 2GHz. Wang's k-NN based attack ran on a single core as the stochastic gradient descent method to find the best weights for k-NN classification could not be parallelized without sacrificing some classification accuracy. Panchenko's CUMUL attack trains an SVM model which requires a grid search to find the best C and γ parameter combination for the RBF kernel. As the native libSVM library is not multi-core enabled, the parameter combination tests ran as parallel processes each on a single core, with the time reported being the one of the slowest C and γ parameter combination test.

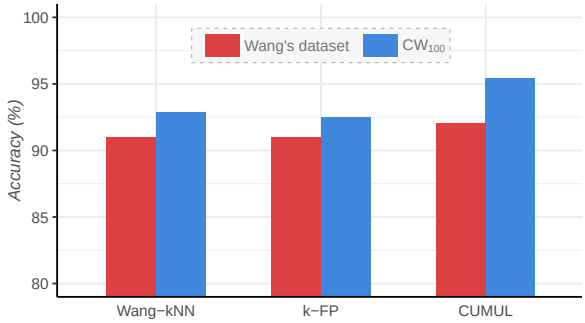


Figure 6.2: Re-evaluation of traditional WF attacks on new data

Figure 6.2 shows the closed world classification accuracy obtained through cross-fold validation for the three traditional WF attacks on a CW_{100} dataset with 100 traces per website. For the same set of website instances, the k-NN algorithm of Wang et al. reports a classification accuracy of 92.87% on our new data set, whereas the CUMUL algorithm of Panchenko et al. and the k-FP attack by Hayes et al. respectively report accuracy results of 95.43% and 92.47%. The obtained results are in line with those originally reported by the authors themselves albeit on other data sets. For this particular setup, the CUMUL WF attack turned out to be the most accurate.

In the second experiment, we evaluate the same traditional methods on 100 websites, but with a growing number of traces per website, to investigate whether the classification accuracy improves significantly when provided with more training data and whether one WF attack method is consistently better than another.

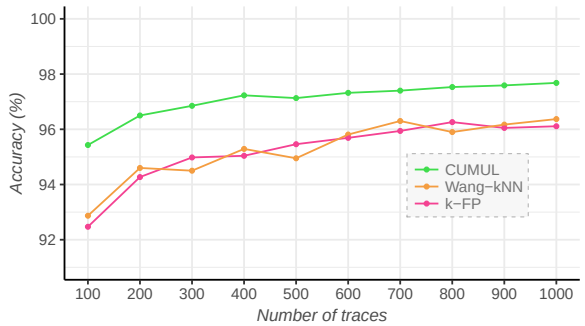


Figure 6.3: Impact on the classification accuracy for a growing number of website traces

In Figure 6.3, we depict the classification accuracy in a closed world experiment where the

number of website instances grows from 100 to 1,000 traces. Our results show that the CUMUL attack consistently outperforms the two other methods. For all methods, the improvement becomes less evident after about 300 website traces. Another interesting observation is that each WF attack − when given sufficient training data − converges to a classification accuracy of approximately 96-97%. However, we experienced scalability issues with the k-NN based attack by Wang et al., given that the classification running times were at least an order of magnitude higher than those of the CUMUL and k-FP attacks.

In a third experiment, we assess how the classification accuracy drops when the number of websites increases for a fixed amount of training instances. Given that the CUMUL attack consistently outperformed the other two methods on our dataset, and was superior in resource consumption, we only report the results for CUMUL. We reevaluate the CUMUL classifier on our closed worlds CW_{100}, CW_{200}, CW_{500} and CW_{900} with a fixed number of traffic traces: 300 per website.

Table 6.1 illustrates that the CUMUL attack obtains a reasonable 92.73% 10-fold cross-validation accuracy for 900 websites using 300 instances each, and a parameter combination of $log_2(C) = 21$ and $log_2(\gamma) = 5$. In general, we observe that the performance degrades gradually with a growing size of the closed world. Moreover, doubling the initial amount of instances gives an advantage of up to 2%, while the amounts higher than 300 stop providing any significant improvement. The biggest weakness is that for each experiment one must execute the grid search to ensure the best classification results, and certain parameter combination tests take a long time to converge with no guarantee of a gain in accuracy.

Table 6.1: CUMUL accuracy for a growing closed world (with 100 traces per website, 300 traces, and the best achieved accuracy for a varying number of traces).

Dataset	**CUMUL (100tr)**	**CUMUL (300tr)**	**CUMUL (best)**
CW_{100}	95.43%	96.85%	97.68% (2000tr)
CW_{200}	93.58%	95.93%	97.07% (2000tr)
CW_{500}	92.30%	94.22%	95.73% (1000tr)
CW_{900}	89.82%	92.73%	92.73% (300tr)

Table 6.2 gives an overview of the running times (in minutes) to find the best C and γ parameter values for the RBF kernel. We aborted those experiments where the grid search took more than four days to complete. While there is a trend of increasing values for these parameters with a growing number of websites and instances, we could not find a strong correlation that would enable us to eliminate the grid search altogether.

As a result, we choose CUMUL as the reference point for comparing our proposed method with the state-of-the art. This decision is driven by the fact that CUMUL performed the best on our closed worlds, and proved to be more practically feasible. We acknowledge that the k-FP attack has the potential to work better in our open world evaluation. However, over the course of our scalability experiments, k-FP did not scale to 50,000 training instances. The experiment consumed more than 64GB memory and took longer than the allocated 4 days, and thus was aborted.With our open world datasets consisting of 800,000 instances (and 400,000 training instances), such high resource consumption demands strongly limit large scale evaluation. CUMUL on the other hand scales up to 400,000 training instances. Therefore, we further evaluate our DL-based approach in

Table 6.2: Time required to find optimal RBF parameter values for C and γ for SVM based classification.

Traces	CW_{100}	CW_{200}	CW_{500}	CW_{900}
100	3 min	8 min	139 min	771 min
200	10 min	48 min	684 min	3027 min
300	19 min	99 min	1230 min	4031 min
400	29 min	134 min	1490 min	$>$ 6000† min
500	34 min	169 min	1541 min	$>$ 6000† min
1000	41 min	844 min	5016 min	$>$ 6000† min
2000	41 min	844 min	5016 min	$>$ 6000† min

†Aborted experiments.

comparison to CUMUL, which outperformed the other traditional WF techniques and which was practically feasible on a larger scale.

6.5.2 Deep Learning for Website Fingerprinting

Here we provide a detailed outline of our DL-based methodology. DL provides a broad set of powerful machine learning techniques with deep architectures. Deep neural networks (DNN), which underlie DL, exploit many layers of non-linear mathematical data transformations for automatic hierarchical feature extraction and selection. DNN demonstrate a superior ability of feature learning for solving a wide variety of tasks. In this study we apply three major types of DNNs to WF: a feedforward SDAE, a convolutional CNN and a recurrent LSTM.

Problem definition

In our proposed method, we follow prior work and formulate WF as a classification problem. Namely, we perform a supervised multinomial classification, where we train a classifier on a set of labeled instances and test the classifier by assigning a label out of a set of multiple possible labels to each unlabeled instance. In WF, a traffic trace t captured from a single visit to a website is an instance of the form ($\mathbf{f_t}$, c_t), where $\mathbf{f_t}$ is the feature vector of the traffic trace and c_t is the class label that corresponds to the website that generated this traffic. Assuming a closed world of N possible websites, label c_t belongs to the set $\{0, 1, \ldots, N-1\}$. As such, we state the WF problem as follows: assign a class label to each anonymous traffic trace in a dataset based on its features.

The classifiers used in related work successfully solved this problem by carefully constructing feature vectors, as described in Section 6.2. Our proposed classifier, based on a DNN, integrates feature learning within the training process, enabling it to classify traffic traces simply based on their initial representation. Thus, for a DL classifier, the form of the input instance changes to ($\mathbf{r_t}$, c_t), where $\mathbf{r_t}$ is a raw representation of a traffic trace that can be interpreted by a neural network.

In essence, we represent a traffic trace as a sequence of successive Tor cells that form the communication between the target user and the visited website. As a result, an input instance of our DNN-based classifier is a series of 1 and -1 of variable length, based on which model performs feature learning and website recognition. Our choice of this format is also supported by the fact

that neural networks generally work with real numbers from the compact interval $[-1, 1]$ due to the nature of the mathematical operations they perform. Moreover, by providing the input data in such a format, we avoid having to rescale and/or normalize the values and thus mitigate a possible information loss coupled with the preprocessing step.

Out of all existing types of DNNs and corresponding DL algorithms, we evaluate three major types of neural networks: feedforward, convolutional and recurrent. We choose to apply the models that provide the capabilities and architectural characteristics to perform the task of automated feature extraction and to benefit from the nature of our input data. We refer to the Appendix for a more elaborate and in-depth discussion on the DL algorithms, which we consider to be conceptually the most well-suited for the WF task at hand.

The first DNN we apply is a classifier called Stacked Denoising Autoencoder (SDAE) – a deep feedforward neural network composed of Denoising Autoencoders (DAE). An Autoencoder (AE) is a feedforward network specifically designed for feature learning through dimensionality reduction. Stacking multiple AEs as building blocks to form a deep model allows for hierarchical extraction of the most salient features of the input data and performing classification based on the derived features, which makes SDAE a promising model for our WF problem.

The next proposed DNN is a Convolutional Neural Network (CNN) – a classifier built on a series of convolutional layers. Convolutional layers are also used for feature extraction, starting with low-level features at the first layer and building up to more abstract concepts going deeper in the network. CNN's methodology for achieving that differs from that of SDAE. Convolutional layers learn numerous filters that reveal regions in the input data containing specific characteristics. These input instances are then downsampled with the special regions preserved. In such a way the CNN searches for the most important features to base the classification on. Furthermore, while SDAE has to be pretrained block by block, CNN requires minimum preprocessing.

The final chosen DNN is yet another type of a neural network, very different in its fundamental properties from the first two. A classifier called Long-Short Term Memory network (LSTM) is a special type of a recurrent neural network that has enhanced memorization capabilities. Its design allows for learning long-term dependencies in data, enabling the classifier to interpret time series. Our input traffic traces are essentially time series of Tor cells, and temporal dynamics in these series are expected to be highly revealing of the contained website fingerprint, thus the choice of the model.

We used `Keras`[10] with `Theano`[28] backend for the implementation of the DNN classifiers. The source code is publicly available on the following webpage: `https://distrinet.cs.kuleuven.be/software/tor-wf-dl/`.

Hyperparameter tuning and model selection

The adversary has to empirically select a DNN model to apply for WF. For that, the adversary should tune the hyperparameters of the DNN to achieve the best classification performance and, at the same time, enhance its capabilities to generalize well to unseen traffic traces.

Performing an automatic search of the best hyperparameters – be that an exhaustive grid search, a random search or another search algorithm – is highly effective but computationally expensive at the same time. In our work, we evaluate the DL algorithms applied to WF by performing semi-automatic hyperparameter tuning, where we exploit the knowledge of each hyperparameter's impact. Namely, the main strategy is as follows:

- The adversary chooses a *representative subsample* of the given dataset and splits it *randomly* into training set, validation set and test set in the following proportion: 90% - 5% - 5%

- Next, the adversary defines the limits of the model capacity based on the amount of available training data. On the one hand, the model has to be expressed with a sufficient amount of parameters in order to be able to learn the problem. On the other hand, there has to be *much fewer* trainable parameters than available training instances in order to avoid overfitting. The model's capacity is defined through its structure and hyperparameters, different for each DNN. The adversary has to define the search spaces for each hyperparameter.
- In our evaluation a special form of Bayesian optimization is applied for hyperparameter tuning, specifically a Tree of Parzen Estimators (TPE)[2] implemented in `hyperopt` library. Through this algorithm the adversary automates the tuning process within previously defined search spaces.
- The optimization algorithm returns the best combination of values and the network structure based on the test results. If the adversary finds the model's test performance satisfactory, he selects this model. Otherwise, he adjusts the search spaces and repeats the tuning procedure.
- Finally, the adversary builds and initializes the selected learning model and applies it to the *whole* dataset to deploy the actual WF attack.

Traditional machine learning methods used for WF in the related work (such as SVM, k-NN and RF, as presented in Section 6.2) also require hyperparameter tuning, but on a smaller scale than DL. Nevertheless, tuning the parameters of the DL model becomes even more feasible in comparison to traditional models due to the parallelism of DL algorithms. As learning algorithms of neural networks are inherently parallel, graphical processing units (GPUs) can take advantage of this characteristic. Performing hyperparameter tuning on GPUs compromises for intense computational requirements allows for rapid feedback of the model. For our DL experiments we use two Nvidia GeForce GTX 1080 GPUs with 8GB memory and 2560 cores each and one TITAN Xp with 12GB memory and 3840 cores to accommodate parallelized training of the DNNs. The training runtime reported in this chapter should therefore be interpreted in association with said platforms.

Table 6.3 includes the list and the values of the hyperparameters we tuned, together with the corresponding intervals within which we vary the values. Each hyperparameter controls a certain aspect of the DL algorithm: architecture (structural complexity of the network), learning (the training process) and regularization (constraint of the learning capabilities applied order to avoid *overfitting*, which occurs when the model memorizes the training data instead of learning from it). Note that in order to reduce the search space, we limited our models to the same learning and regularization parameters for each network layer.

The adversary is supposed to select the DL-based model *once* given a sample crawled for a desired closed world of websites. Similarly, we perform the model selection on the CW_{100} dataset, as defined in Section 6.4, in order to limit the computational requirements. Given a proper tuning procedure and a sufficiently large amount of training instances for each class, the chosen model is expected to learn the problem (learn to extract the fingerprints), and at the same time generalize well to the other closed world datasets. In fact, the adversary capable of crawling large amounts of data can compensate on hyperparameter tuning.

The final selected models of SDAE, CNN and LSTM used for evaluation are described in Table 6.3. The amount of LSTM units has to be adjusted for the bigger closed worlds to increase expressive capacity. Note that due to the LSTM's backpropagation through time constraints, we have to trim the traffic traces to the first 150 Tor cells (we elaborate on the reason for that in Appendix).

Further in this subsection we present the experimental results of the DL-based WF attack on the crawled dataset. Namely, we evaluate the three chosen DNNs on the closed worlds of various sizes

Table 6.3: Tuned hyperparameters of the selected DL models.

	SDAE		CNN		LSTM	
Hyperparameter	**Value**	**Space**	**Value**	**Space**	**Value**	**Space**
optimizer	SGD	SGD, Adam RMSProp	RMSProp	SGD, Adam RMSProp	RMSProp	SGD, Adam RMSProp
learning rate	0.001	0.0001 .. 0.1	0.0011	0.0009 .. 0.0025	0.001	0.0001 .. 0.1
decay	0.0	0.0 .. 0.9	0.0	0.0 .. 0.9	0.0	0.0 .. 0.9
batch size	32	8 .. 256	256	8 .. 256	128	32 .. 256
training epochs	≤30	1 .. 100	3-6	1 .. 20	≤50	1 .. 100
number of layers	5	3 .. 7	8	6 .. 10	4	3 .. 6
input units	5000	200 .. 5000	3000	200 .. 5000	150	70 .. 1000
hidden layers units	1000, 500, 300	200 .. 3000	—	—	64, 64 / 128, 128	64 .. 256
dropout	0.1	0.0 .. 0.5	0.1	0.0 .. 0.5	0.22	0.0 .. 0.5
activation	tanh	tanh, sigmoid, relu	relu	tanh, relu	tanh	tanh, sigmoid, relu
pretraining optimizer	SGD	SGD, Adam	—	—	—	—
pretraining learning rate	0.1	0.01 .. 0.1	—	—	—	—
kernels	—	—	32	4 .. 128	—	—
kernel size	—	—	5	2 .. 50	—	—
pool size	—	—	4	2 .. 16	—	—

and on the open world. We also assess their generalization capabilities by testing their resilience to concept drift on data periodically collected over 2 months. Furthermore, we compare results to CUMUL, being the most accurate traditional WF method.

Closed world evaluation

In this study, we evaluate the SDAE, CNN and LSTM networks on four closed worlds of different sizes, namely CW_{100}, CW_{200}, CW_{500} and CW_{900}. We use the models selected by performing hyperparameter tuning on the CW_{100} dataset, according to the aforementioned methodology. To ensure the reliability of our experiments, we estimate the models' performance by conducting a 10-fold cross-validation on each dataset. We use two performance metrics to evaluate and compare the models with each other: the test *accuracy* (classification success rate, which needs to be maximized) and the test *loss* (a cost function that reflects the significance of classification errors made by the model, namely the categorical cross-entropy, that needs to be minimized, as explained in the Appendix).

The aspect that had the greatest impact on the performance over the course of our experiments was the amount of training data (i.e. the amount of traffic traces for each website), which is in line with our expectations and justifies the extensive data collection. Indeed, for every closed world experiment, we observed significant improvements for a growing amount of traces. One example of this trend is given in Figure 6.4 for the CW_{100} dataset, where we vary the amount of instances from 100 to all available 2,500 per class. The Table 6.4 reports on the actual metrics' values and the corresponding runtimes.

First and foremost, from these results we can confirm the *feasibility of the WF attack based on a DL approach with automatic feature learning*. We observe how classification accuracy and loss function gradually improve for all models, in the end reaching the 95.46, 96.66 and 94.02% success rate for SDAE, CNN and LSTM model accordingly. These results are comparable to the ones achieved by traditional approaches in Section 6.5.1.

If we compare the three DNNs with each other, we observe that the SDAE and CNN networks consistently perform better than the LSTM in terms of classification accuracy, with CNN being the most performant. Nevertheless, knowing that the LSTM classifies traffic traces based solely

Table 6.4: Accuracy, loss and runtime of the DL models (SDAE, CNN, LSTM) for CW_{100} and a growing number of traces.

	SDAE			CNN			LSTM		
Traces	**Accuracy**	**Loss**	**Runtime**	**Accuracy**	**Loss**	**Runtime**	**Accuracy**	**Loss**	**Runtime**
100	85.00%	0.5902	0 min	81.25%	0.8276	0 min	40.60%	2.2132	9 min
200	87.30%	0.5252	1 min	86.63%	0.5793	0.5 min	57.30%	1.5471	17 min
500	91.34%	0.3576	1 min	91.43%	0.3877	1 min	79.54%	0.7848	40 min
1000	92.64%	0.2950	2 min	94.72%	0.2545	1.5 min	91.63%	0.3555	63 min
1500	94.49%	0.2314	4 min	95.95%	0.1855	2 min	91.93%	0.3055	66 min
2000	95.17%	0.1955	6 min	96.14%	0.1699	3 min	93.98%	0.3277	67 min
2500	95.46%	0.1968	7 min	96.26%	0.1784	5 min	94.02%	0.3204	76 min

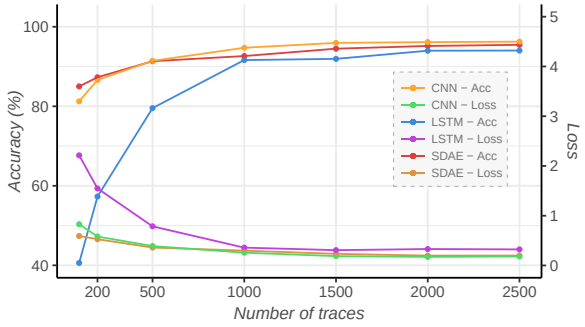


Figure 6.4: Accuracy, loss and evaluation time of the DL models (SDAE, CNN, LSTM) for CW_{100} and a growing number of traces

on their first 150 Tor cells (compared to the SDAE and CNN that use up to 5,000 and 3,000 cells from each trace), the achieved performance still appears promising. Our interpretation is that even a small part of the traffic trace is sufficient for website recognition up to 94% accuracy when deploying a model that is able to exploit temporal dependencies of the input sequence. Notably, LSTM performs much poorer when trained on fewer traffic traces than SDAE and CNN, but later gains comparable recognition rate at 1000 training instances per class.

Next, we assess whether the selected DL models tuned on CW_{100} perform similarly when applied to the larger datasets: CW_{200}, CW_{500} and CW_{900}. The results of the DL-based WF for all closed world datasets are presented in Table 6.5, expressed in classification accuracy, loss function and runtime. The time reported in the table is the average time required to build, train and evaluate a model. We observe that for larger closed worlds the performance of the three DL models gradually decreases following a similar trend. The closed world evaluation results remain comparable to CUMUL's results presented in Table 6.1 in the previous subsection. Figure 6.5 compares the DL-based methods to CUMUL. This comparison illustrates that our DL-based attack can indeed successfully learn the fingerprinting features in an automated manner. Furthermore, the training method itself is highly parallelizable on GPU hardware resulting in a faster and therefore more practical closed world WF attack.

The presented experiments on the closed world reflect the model's ability to classify traffic traces that are collected at the same moment as the training data. Even though we prove that such a

Table 6.5: Accuracy, loss and runtime of the DL models (SDAE, CNN, LSTM) for each closed world and 2,500 traces.

	SDAE			CNN			LSTM		
Dataset	**Accuracy**	**Loss**	**Runtime**	**Accuracy**	**Loss**	**Runtime**	**Accuracy**	**Loss**	**Runtime**
CW_{100}	95.46%	0.1968	7 min	**96.66%**	0.1699	5 min	94.02%	0.3204	76 min.
CW_{200}	95.76%	0.1822	14 min	**96.52%**	0.1774	8 min	93.10%	0.3292	91 min
CW_{500}	**95.04%**	0.2243	34 min	92.31%	0.3732	12 min	90.80%	0.3163	257 min
CW_{900}	**94.25%**	0.2530	52 min	91.79%	0.4278	20 min	88.04%	0.3601	276 min

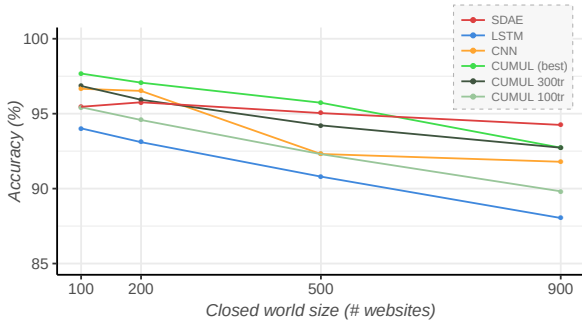


Figure 6.5: DL (SDAE, CNN, LSTM) vs. CUMUL for a growing size of the closed world from 100 to 900 websites.

WF attack is possible, we do not address the question of eliciting the concrete data features that the models take decisions upon. In other words, just based on this experiment, we cannot certainly infer if the DNN reveals the actual website fingerprint for deanonymization, or also learns occasional dynamics in the traffic data instead that just happens to enable recognition. The next experiment is intended to reveal how well our DNNs are able to extract the fingerprint and generalize to new data.

Concept drift evaluation

The challenge of recognizing traffic traces collected over time was first addressed by Juarez et al. [19]. They showed that classification accuracy drops drastically when testing the model on traffic captured 10 days after training. This time effect is explained by constant content changes of the websites, which of course may affect the identifying fingerprints. Another possible reason for the performance drop is that the classifier trained and evaluated at one moment in time might overlook the stable fingerprint and learn the temporary features instead. In general such an occurrence is known as *concept drift* – a change over time in the statistical properties of the class that the model is trying to predict. Therefore, the recognition might become less accurate over time. A model resilient against concept drift is the one that manages to capture the salient traffic features maximally correlated with the website fingerprint and thus remains performant over time. To reveal if our DNNs detect the actual website fingerprints and assess how well they perform in case

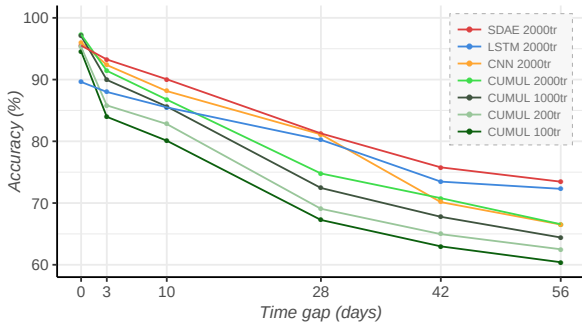


Figure 6.6: DL (SDAE, CNN, LSTM) vs. CUMUL resilience to concept drift: evaluation of CW_{200} over time.

of traffic changes, we train the models on a closed world and test them on data collected from visiting websites of the same closed world periodically over 2 months. In order to fairly compare DL-based methods to CUMUL, we have to evaluate them on the same dataset with the same amount of traces. Due to CUMUL's scalability issue, the biggest dataset possible to use for this evaluation is CW_{200} with 2,000 training instances. Even though this is not the largest dataset we collected, it is still twice bigger than the closed worlds normally used in prior works. Thus we train models on the whole CW_{200} dataset (with 2,000 training traces) and test them on the *revisit-over-time* dataset (as defined in Section 6.4).

The results are depicted in Figure 6.6 for DL and traditional CUMUL. The plot indicates the WF performance of various models trained on CW_{200} and evaluated on traffic re-collected 3 days, 10 days, 4 weeks, 6 weeks and 8 weeks after training.

The figure demonstrates how the classification accuracy decreases and the classification loss increases gradually and drastically over time. These results illustrate the high generalizing abilities of both the evaluated models. Despite a significant 2-month time gap between the moment of training and the last evaluation, the DL algorithms are still capable to correctly deanonymize at least 66% out of 2,000 website visits. We witness a rather small accuracy drop in the first 3 and 10 days for all three DL models, which may be acceptable for an adversary who would prefer to use the built WF classifier for several more days rather than repeat the data collection and training process every day. In total, SDAE loses 22% of accuracy over 2 months, CNN loses 29%, while LSTM only loses 17%. Notably, being the most performant DL model on the day of training, CNN generalized worse than SDAE or LSTM. Despite the fact that the LSTM model (which still makes decision just based on the first 150 cells in the input sequence) is initially outperformed by both SDAE and CNN, after one month its accuracy catches up with that of the SDAE. Moreover, after 1 month the LSTM loss values are lower than those of the SDAE, which means that even though the LSTM outputs less correct predictions, it is overall more certain of these predictions. This obviously speaks in favor of LSTM's high generalization abilities, in line with our best expectations.

Our SDAE and CNN approaches outperform CUMUL with up to 7% over the course of 2 months. In total CUMUL loses 31%. LSTM network starts outperforming CUMUL after approximately 2 weeks. As such, this comparison not only shows that our approach indeed automates the feature engineering, but also that the learned implicit features (hidden in the neural network) are more robust against website changes over time. Notably, CUMUL is found to significantly improve its generalization abilities when trained on larger amounts of traffic traces per website, which proves that DL-based classifiers are not alone in their requirement for a bigger training data for the highest performance.

The main conclusion here is that the DL-based classifiers are capable of extracting stable identifying information from the closed world traffic which allows for its deanonymization with a high success rate, even several days after training.

Open world evaluation

This study compares DL-based WF attacks and CUMUL for the open world evaluation. The goal is to assess the classifier's ability to distinguish a traffic trace generated by a visit to one of the monitored websites from a traffic trace generated by a visit to any other unknown website. Our methodology for the open world evaluation differs from prior work in several aspects. We aim to provide a fair comparison of the classifiers by reducing possible bias. To this purpose we have to depart from the realistic WF setting and adapt the following assumptions:

- We model the monitored websites by training the classifier solely on the traffic traces of the websites an adversary is aiming to detect. By doing so, we assess the abilities of the learning algorithms to distinguish seen and unseen websites. In previous studies on WF, it has been argued that an adversary may improve the attack by additionally collecting and training on traffic of known websites that he is not interested in identifying, which is of course a possibility given sufficient resources. But here we do not provide any helping patterns of the open Web to the classifiers to not distort their actual performance.

- We test the classifiers on balanced datasets: monitored and unknown websites in proportion 50%-50% (meaning that random classification would be accurate on average 50% of a time). Thus, we do not attempt to infer the realistic ratio, especially knowing that modeling an open world of a realistic scale poses large issues: (1) the effect of the hypothesis space complexity, as shown by Panchenko et al. [24], and (2) the *base rate fallacy*, demonstrated by Juarez et al. [19]: even a highly accurate classifier trained on the monitored websites with a very low prior probabilities of visit cannot be fully confident of its predictions. Instead we assume a standard uniform probability distribution of visits to the monitored and unknown sets. With such evaluation the classifier's errors are more prominent and allow for a clearer comparison.

- Following the earlier reasoning, we use Alexa websites for both, monitored and unknown sets. Choosing a particular set of monitored websites characterized by patterns that are not common to the whole Web would introduce classification bias with unpredictable impact on comparison. In order to objectively compare the studied classifiers, we demonstrate their abilities to distinguish seen and unseen fingerprints belonging to the websites of the same category (in our case, most popular websites).

We evaluate the open world WF attack for an adversary who monitors a set of 200 websites, while the target user may visit 400,000 more unknown websites. As a result, our open world dataset consists of 800,000 visits through Tor: one-time visits to 400,000 various websites in the Web and 400,000 visits to the monitored CW_{200}. We train the models solely on 2,000 instances of CW_{200} (thus obtaining the classifiers identical to those used for the closed world evaluation). Recall that earlier in the closed world evaluation section we already assessed their multinomial classification performance; the reported success rates indicate the ability of the classifiers to identify the exact visited monitored website. In this section we perform binary classification by testing the same models on our open world dataset. With this experiment we assess the classifiers' ability to recognize the input instance as a visit to a monitored or an unknown, earlier unseen website. The classifier makes decisions based on the cross-entropy loss function, which reflects its confidence in made predictions (Appendix elaborates on the cross-entropy as a measure of classification confidence). If

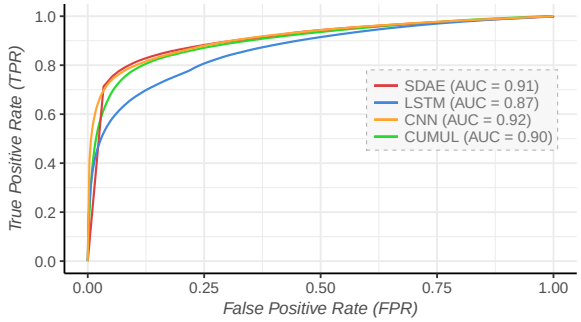


Figure 6.7: DL (SDAE, CNN and LSTM) vs. CUMUL in the open world setting for a monitored set of CW_{200}.

the loss value is low enough, the adversary assumes that the classified website visit belongs to a set of monitored websites. If the entropy is bigger than a certain *confidence threshold*, the adversary decides to not trust the classifier's class prediction and concludes that the tested traffic trace was generated by an unknown website, thus causing the prediction uncertainty. By varying the confidence threshold, the adversary balances the True Positive and False Positive Rate according to their priorities.

In our evaluation, we plot the ROC curve for the three DL classifiers in order to define the optimal confidence threshold which separates the monitored websites traffic from unknown websites traffic. Both CNN and SDAE again outperform CUMUL, if only slightly, as demonstrated by Area Under Curve values in the same figure. The ROC curves for SDAE, CNN and LSTM are depicted in Figure 6.7 and demonstrate the relative performance of the suggested open world WF DL-based attacks within 200 monitored and 400,000 unknown websites. We observe that the CNN model performs better than SDAE, and both perform significantly better than the LSTM model. However, the adversary may improve the models by using the open world traces for validation during hyperparameter tuning . LSTM classifier is outperformed by two other DL models because it only processes the first 150 Tor cells, opposed to 5,000 by SDAE and 3,000 by CNN.

According to the ROC curves, an adversary may optimize the confidence threshold depending on their priority. For 200 classes, the categorical cross-entropy E varies between 0 (absolute confidence of the classifier's prediction) to 5.3 (absolute uncertainty). The optimization examples are given in Table 6.6, where reduced thresholds allow to decrease FPR.

Table 6.6: DL vs. CUMUL in the open world setting.

	Optimized for TPR			**Optimized for FPR**		
Model	E	**TPR**	**FPR**	E	**TPR**	**FPR**
SDAE	0.005	80.25%	9.11%	0.001	71.30%	3.40%
CNN	0.033	80.11%	10.53%	0.013	70.94%	3.82%
LSTM	0.062	76.19%	19.78%	0.010	53.39%	3.67%
CUMUL	0.048	78.00%	9.89%	0.018	62.57%	3.58%

Our open world evaluation considers a large set of unknown sites in which the adversary cannot train, allowing us to test the generalization of our models in a large sample of the Web. Similarly to the state-of-the-art, we observe how our DL-based approach withstands a challenging open world scenario, providing high accuracy on the largest set of unknown sites.

In the previous subsections, we have shown the relative performance of various DL models in comparison with each other and with the traditional CUMUL classifier. In certain experimental settings we improved beyond the state-of-the-art, e.g. in resilience to content changes and in success rate on the largest closed world. The success rates of WF attacks proved to depend on the closed world size, the amount of training data available to the adversary and the computational resources that can be used to train the classifier. For the evaluations performed in this chapter, we used the resources available at our institution, but we acknowledge that a more powerful attacker could most likely further improve the attack by using more resources for data collection, model selection and training.

6.6 Discussion

In this section, we enumerate the limitations of this work and discuss remaining open challenges with regard to both the threat model and the deep learning methods we presented.

As in virtually all prior work on WF, we analyzed the attacks only on visits to homepages and omitted other pages within the considered websites. We acknowledge this is an unrealistic assumption. However, as our main goal was to perform a fair comparison with existing attacks, we used the same experimental settings. As the models developed in prior work were tailored to these particular settings, the evaluation of techniques that consider inner web pages was deemed out of scope for this chapter. Nevertheless, we find automatic feature learning a promising approach to this problem.

We do not try to approximate the probability of visiting a closed world site vs. a site from the open world in our experiments. We assume that all open world sites have the same prior probability and all closed world sites have the same prior probability. We acknowledge this does not reflect reality but one can only hypothesize on the actual popularity distribution of websites over Tor without risking the privacy of Tor users. It is a limitation of our study and previous work.

Deep learning allows us to replace manual feature engineering with automatic feature learning. Therefore, the resulting attack is not defined by an explicit set of features that would be easily interpretable by a human analyst, but is instead based on abstract implicit non-interpretable features, being learnable parameters of the neural network. Moreover, these features have proven to be more robust to web content changes in comparison to those suggested in prior literature. Consequentially, the corresponding countermeasure cannot focus on concealing specific features as it was done earlier, but in order to defend against the DL-based attack we have to challenge the DL algorithm itself. Therefore, future work should focus on defending against the automated WF attacks, such as deep neural networks presented in this study.

One line of research for future work could be to investigate whether it is possible to mislead the deep neural network predictions. For instance, such research could base on the latest work on *adversarial examples* [6]. These are inputs to the learning model specifically crafted to fool the neural network into classifying them into a wrong class. Adversarial examples can be explored as a defense strategy against DL-based WF in order to protect Tor user's privacy.

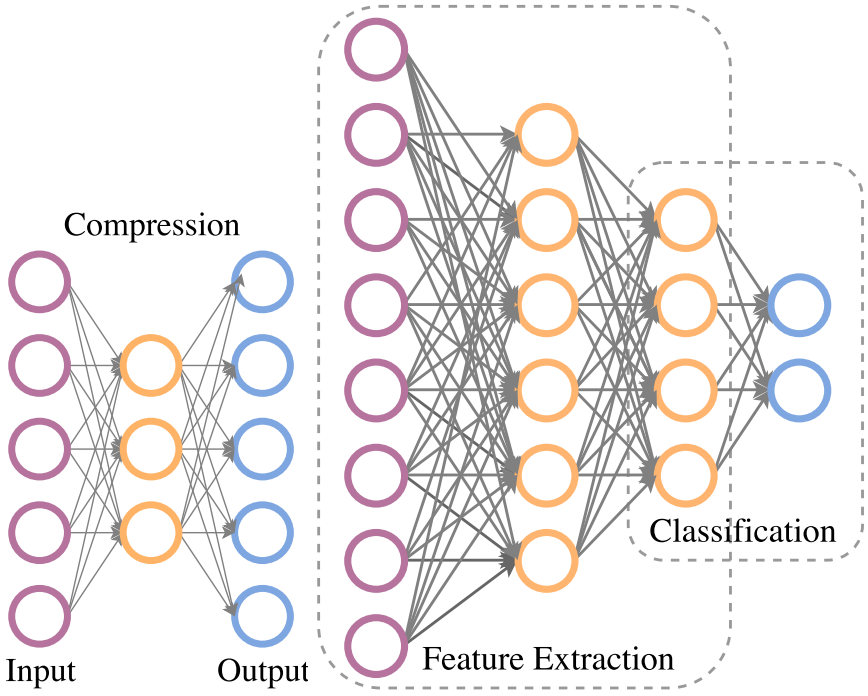
In the very recent work by Wang and Goldberg [33], a defense technique based on half-duplex communication and burst molding is proposed. The authors claim that this defense defeats all WF attack techniques known to date. It would be interesting to validate whether the author's claims still hold in the presence of automatic feature learners such as DL.

A. Appendix

This section elaborates further on the DNN models and learning algorithms we used in our WF attack.

A.1 Stacked Denoising Autoencoder

Autoencoder (AE) is a shallow feedforward neural network designed for learning meaningful data representations [29]. It is composed of an input layer, one hidden layer and an output layer, as shown in FigureA.1a. The input layer acts as an encoder that transforms data and passes it to the hidden layer $\mathbf{h} = f(\mathbf{x})$, and the output layer of the same size acts as a decoder that reconstructs the data back from the hidden layer $\mathbf{r} = g(\mathbf{h})$, intending to produce maximally similar values.



(a) Autoencoder (b) SDAE from two autoencoders

Figure A.1: Stacked Denoising Autoencoder

The size of the hidden layer plays a crucial role in the AE's working algorithm: it defines the representation of the input used for reconstructing the data. The hidden layer $\mathbf{h}$ is constrained to have fewer neurons than the input $\mathbf{x}$. Then such an *undercomplete* AE is forced to compress the

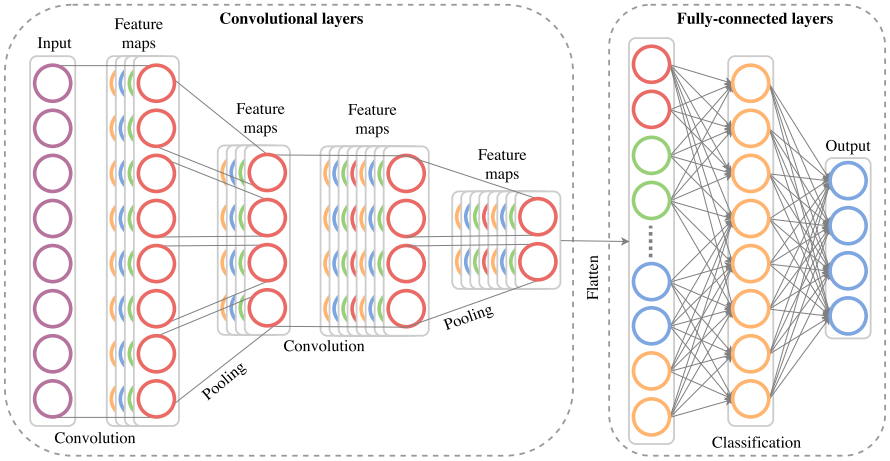


Figure A.2: Convolutional Neural Network.

input and can only output its approximation rather than the identity. In order to reconstruct the data from a compressed representation with a minimal loss, the network has to prioritize between properties of the data during compression.

In case of a traffic trace as an input, AE will learn certain combinations and transformations of the input values that allow to reconstruct the same trace with the highest accuracy. As a result, the hidden layer will contain the most salient features of the traffic trace. The training is performed by backpropagating the reconstruction errors expressed via the loss function that has to be optimized by the network. The loss function $L(\mathbf{x}, g(f(\mathbf{x})))$, such as mean squared error, reflects the difference between the input $\mathbf{x}$ and its reconstruction $\mathbf{g}(\mathbf{f}(\mathbf{x}))$, and reaches its minimum value in case of a total similarity between the two. We use a mean squared error for this purpose, which measures the average of the squares of the deviations: $L(\mathbf{x}, g(f(\mathbf{x}))) = \frac{1}{N}\sum_{i=1}^{N}(g(f(x_i)) - x_i)^2$, where N is the number of neurons of the input (and the output) layer.

Since the undercomplete AE cannot learn a total identity function but only an approximation, its training stops once having minimized the loss function, and thus ensures a good learned representation of data. The AE, as a building block of our future classifier, has to learn representations which reflect statistical properties of the whole data distribution beyond the training examples. This is necessary to achieve a high performance of the model on unseen data, a property of the machine learning models known as a *generalization* capability. The AE that performs during training much better than on traffic unseen before, has *overfitted* to the training data, and thus shows poor generalization capabilities.

To ensure generalization, we apply *regularization* by using *dropout*, when a randomly chosen fraction of input values is set to 0 at each training iteration. AE with dropout is a *Denoising Autoencoder* (DAE) which is more robust to overfitting [26].

Stacked Denoising Autoencoder is a deep feedforward neural network built from multiple DAEs by stacking them together, in a manner depicted in Figure A.1b. SDAE stacks the DAEs representation layers: the hidden layer of the first DAE is used as the input layer of the successive DAE, and so forth. Chaining several DAEs enables the model to hierarchically extract data from the input to learn features of different levels of abstraction. We chain 3 DAEs to form a 5-layered SDAE. Deeper models produce final features of higher abstraction, which are meant to be used for classification on the concluding layer. The classification layer has one neuron for each possible class, or in our

case for each website. Output neurons compute the probability of the input instance to belong to a class. The neuron that produced a maximum probability assigns its label to the training instance.

It was discovered by Hinton et al.[15] that in order to achieve a better performing DNN, it has to first be pretrained in an unsupervised fashion, that is without using the knowledge of labels of the training data. This strategy is known as the *greedy layer-wise unsupervised pretraining* that initializes the SDAE. This stage is followed by a *supervised fine-tuning* of the whole model, that learns to classify the input by backpropagating the classification errors. The loss function that expresses the errors is a categorical entropy $E = -\frac{1}{N}\sum_i^N(p_i log_2 p_i)$, where p_i is a returned probability for the predicted class with N websites in total. A classifier confident of its decisions gives a high probability for each predicted class which results into a minimized entropy.

A.2 Convolutional Neural Network

A deep network called *Convolutional Neural Network* (CNN) is another feedforward network trained with backpropagation similarly to SDAE, but has a different structure, designed for minimal preprocessing [21]. CNN's main building block is a convolutional layer, which performs a linear *convolution* operation instead of a regular matrix multiplication. The learnable parameters of the convolutional layers are *kernels* or *filters* – multidimensional arrays that are convolved with the input data to create *feature maps*, as depicted in Figure A.2. The kernel is applied spatially to small regions of the input, thus enabling sparse connectivity and reducing the actual parameter learning in comparison to fully-connected layers. The kernel aims to learn an individual part of an underlying feature set, e.g. the website fingerprint in a traffic trace. The convolution function is followed by a non-linear activation, typically a *rectifier* [23]. The rectified feature maps are stacked together along the depth dimension to produce the output.

The next operation of the CNN is typically a *pooling* layer that performs a subsampling operation by replacing the output of the convolution layer with a summary statistics of the nearby outputs. We use a max pooling layer that reports the maximum outputs within regions of the feature maps. Pooling helps the representation become invariant to minor changes of the input. For instance, such subsampling allows to find the prominent identifying parts of the website fingerprint within the traffic trace, despite its slight shifts in location and ignoring the surrounding traffic.

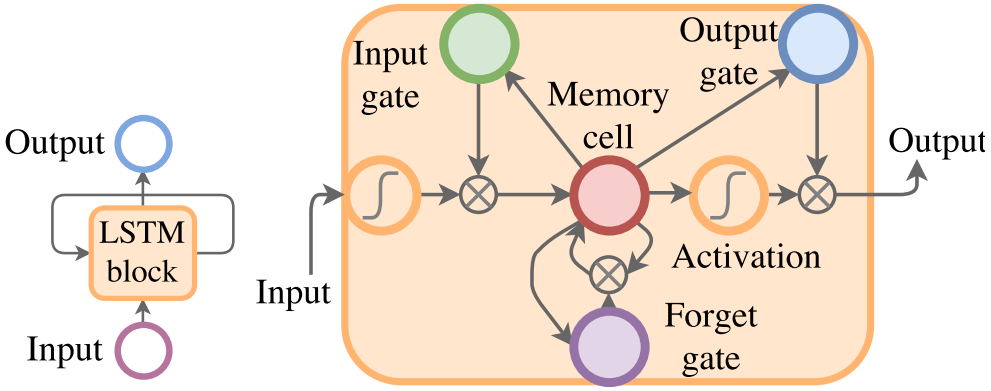
The network can include a whole series of convolution and pooling layers in order to extract more abstract features. We use two sets of such layers. The resulting feature maps need to be flattened and concluded by at least one regular fully-connected layer prior to classification. Because of the risk of overfitting, we apply dropout and limit the amount of learnable parameters of the network by using only two fully-connected hidden layers. The final layer outputs the predictions.

A.3 Long Short Term Memory

Recurrent neural network (RNN) is a network with feedback connections, which enable it to learn temporal dependencies [17]. RNN can interpret the input as a sequence, taking into account its temporal properties.

Long short term memory network (LSTM) [18] shown in Figure A.3a is a special type of a RNN that accommodates so-called LSTM building block to model long-term memory, which allows the network to learn longer input sequences.

The LSTM block processes sequences time step by time step, passing the data through its *memory cells*, and input, output and forget *gates*, as depicted in Figure A.3b.



(a) LSTM (b) LSTM block

Figure A.3: Long Short Term Memory

The memory cell represents the so-called *internal state* of the network. LSTM is able to remove or add information to the cell, regulating these operations by gates. Gates are composed of a sigmoid neural network layer and a pointwise product, and are parameterized by a set of learnable weights. Gates learn to carefully choose whether to let the information through them in order to modify the internal state, to forget information or to produce the output when deemed necessary. The output of an LSTM block is formed by the number of memory units.

LSTM layer's depth depends on the length of processed sequences: due to the feedback connection, they basically have one layer for every processed time step of a sequence. Such structure can be obtained by unrolling the loop in Figure A.3a. Classification errors are backpropagated through many layers "through time", which limits the training process: first it significantly slows down training in compare to the feedforward networks, and secondly, in practice it only allows to backpropagate up to 100-200 layers.

LSTM layers can be stacked to form deeper networks. The intuition is the same that higher LSTM layers can capture more abstract concepts. We chain two hidden LSTM layers and form a 4-layered LSTM network (with each layer "unrolled" to as many layers as there are time steps in the processed sequence), which allowed to obtain the best performance.

Bibliography

[1] K. Abe and S. Goto, "Fingerprinting attack on tor anonymity using deep learning," *Proceedings of the Asia-Pacific Advanced Network*, vol. 42, pp. 15–20, 2016.

[2] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 1. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 115–123. [Online]. Available: http://proceedings.mlr.press/v28/bergstra13.html

[3] J. Buchner, "ImageHash," https://github.com/JohannesBuchner/imagehash, 2017.

[4] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses," in *ACM Conference on Computer and Communications Security (CCS)*. ACM, 2014, pp. 227–238.

[5] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a Distance: Website Fingerprinting Attacks and Defenses," in *ACM Conference on Computer and Communications Security (CCS)*. ACM, 2012, pp. 605–616.

[6] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy (S&P)*, 2017, pp. 39–57.

[7] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. ACM, 2002, pp. 380–388.

[8] H. Cheng and R. Avnur, "Traffic Analysis of SSL Encrypted Web Browsing," *Project paper, University of Berkeley*, 1998, Available at http://www.cs.berkeley.edu/~daw/teaching/cs261-f98/projects/final-reports/ronathan-heyning.ps.

[9] G. Cherubin, J. Hayes, and M. Juarez, ""Website Fingerprinting Defenses at the Application Layer"," in *Privacy Enhancing Technologies Symposium (PETS)*. De Gruyter, 2017, pp. 168–185.

[10] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[11] R. Dingledine, N. Mathewson, and P. F. Syverson, ""Tor: The Second-Generation Onion Router"," in *USENIX Security Symposium*. USENIX Association, 2004, pp. 303–320.

[12] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2012, pp. 332–346.

[13] J. Hayes and G. Danezis, "k-fingerprinting: a Robust Scalable Website Fingerprinting Technique," in *USENIX Security Symposium*. USENIX Association, 2016, pp. 1–17.

[14] D. Herrmann, R. Wendolsky, and H. Federrath, "Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier," in *ACM Workshop on Cloud Computing Security*. ACM, 2009, pp. 31–42.

[15] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[16] A. Hintz, "Fingerprinting Websites Using Traffic Analysis," in *Privacy Enhancing Technologies (PETs)*. Springer, 2003, pp. 171–178.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] ——, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A critical evaluation of website fingerprinting attacks," in *ACM Conference on Computer and Communications Security (CCS)*. ACM, 2014, pp. 263–274.

[20] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an Efficient Website Fingerprinting Defense," in *European Symposium on Research in Computer Security (ESORICS)*. Springer, 2016, pp. 27–46.

[21] Y. LeCun and Y. Bengio, "The handbook of brain theory and neural networks," M. A. Arbib, Ed. Cambridge, MA, USA: MIT Press, 1998, ch. Convolutional Networks for Images, Speech, and Time Series, pp. 255–258. [Online]. Available: http://dl.acm.org/citation.cfm?id=303568.303704

[22] M. Liberatore and B. N. Levine, ""Inferring the source of encrypted HTTP connections"," in *ACM Conference on Computer and Communications Security (CCS)*. ACM, 2006, pp. 255–263.

[23] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, J. Frnkranz and T. Joachims, Eds. Omnipress, 2010, pp. 807–814. [Online]. Available: http://www.icml2010.org/papers/432.pdf

[24] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, "Website fingerprinting at internet scale," in *Network & Distributed System Security Symposium (NDSS)*. IEEE Computer Society, 2016, pp. 1–15.

[25] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *ACM Workshop on Privacy in the Electronic Society (WPES)*. ACM, 2011, pp. 103–114.

[26] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[27] Q. Sun, D. R. Simon, and Y. M. Wang, "Statistical Identification of Encrypted Web Browsing Traffic," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2002, pp. 19–30.

[28] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: http://arxiv.org/abs/1605.02688

[29] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[30] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective Attacks and Provable Defenses for Website Fingerprinting," in *USENIX Security Symposium*. USENIX Association, 2014, pp. 143–157.

[31] T. Wang and I. Goldberg, "Improved Website Fingerprinting on Tor," in *ACM Workshop on Privacy in the Electronic Society (WPES)*. ACM, 2013, pp. 201–212.

[32] ——, "On realistically attacking tor with website fingerprinting," in *Proceedings on Privacy Enhancing Technologies (PoPETs)*. De Gruyter Open, 2016, pp. 21–36.

[33] ——, "Walkie-talkie: An efficient defense against passive website fingerprinting attacks," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 1375–1390. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-tao

[34] Z. Wang, "The applications of deep learning on traffic identification," *BlackHat USA*, 2015.

7. Website Fingerprinting Defenses at the Application Layer

7.1 Introduction

Website Fingerprinting (WF) attacks allow a passive local adversary to infer which webpage a client is viewing by identifying patterns in network traffic that are unique to the webpage. These attacks are possible even if the client is browsing through anonymity networks such as Tor and the communication is encrypted [12]. Tor routes a client's traffic through volunteer relays before connecting to the communication's destination, so that local eavesdroppers cannot link both sender and receiver of the communication [8]. However, the WF attack, if successful, breaks the *unlinkability* property that Tor aims to provide to its users.

Moreover, a 2015 study has shown that `.onion` sites can be distinguished from regular sites with more than 90% accuracy [16]. This substantially narrows down the classification space in Tor and suggests the attack is potentially more effective at identifying `.onion` sites than regular pages. *Onion services* are websites with the `.onion` domain hosted over Tor, allowing a client to visit a website without requiring it to publicly announce its IP address. These sites tend to host sensitive content and may be more interesting for an adversary, turning the WF attack into a major threat for connecting users. In this chapter, we propose the first set of defenses specifically designed and evaluated for Tor `.onion` sites.

WF defenses are often theorized at the network level, and try to disrupt statistical patterns via inserting dummy messages in to the packet stream [9, 2, 4]. Some defenses try to alter the network traffic of a webpage to mimic that of another webpage that is not interesting to the attacker [32]. However, a defense at the network level may require substantial changes of Tor or even the TCP stack, which would make its deployment unrealistic. Furthermore, there is no need to hide patterns at the network layer because few webpage-identifying features, if any, are introduced by low layers of the stack (e.g., TCP, IP). In this work, we consider application-layer defenses, arguing that this approach is more natural for WF defenses and facilitates their development.

Existing WF defenses have been engineered to protect the link between the client and the entry to the Tor network, assuming this is the only part of the network observable by the adversary. We propose both WF defenses at the client- and server-side. A server-side defense is more usable as it does not require any action from the user. More and more, certain types of websites, such as human rights advocacy websites, have the motivation to provide WF defenses as a service to its user base, who may be of particular interest to an adversary. For this reason, we believe that, in contrast to *normal* websites, `.onion` site operators not only have the incentive to provide defenses against WF attacks, but can also achieve a competitive advantage with respect to other `.onion` sites by doing so.

As a real life motivating example, we were contacted by *SecureDrop* [27], an organization that provides onion services for the anonymous communication between journalists and whistleblowers. They are concerned that sources wishing to use their service can be de-anonymized through WF. As a consequence, they are interested in using a server-side WF defense. We have included a SecureDrop website in all the datasets used for the evaluation of defenses.

We introduce two variants of a server-side defense operating at the application layer, which we call *Application Layer Padding Concerns Adversaries* (ALPaCA). We evaluate it via a live implementation on the Tor network. We first crawl over a significant fraction of the total Tor `.onion` site space, retrieving not only the network level traffic information – as is standard in WF research – but also the `index.html` page and HTTP requests and responses. We then analyze the size distribution for each content type, e.g. PNG, HTML, CSS. Using this information, ALPaCA alters the `index.html` of a page to conform to an "average" `.onion` site page. ALPaCA runs periodically, changing the page fingerprint on every user request.

Due to the expected slow adoption of server-side WF defenses, client-side defenses must still be used. We therefore implement a simple client-side WF defense, dubbed *Lightweight application-Layer Masquerading Add-on* (LLaMA), that works at the application layer by adding extra delays to the HTTP requests. These delays alter the order of the requests in a similar way to randomized pipelining (RP) [23], a WF countermeasure implemented in the Tor browser that has been shown to fail in several evaluations [5, 31, 14]. Besides delaying HTTP requests, our defense sends redundant requests to the server. We show most of the protection provided by this defense stems from the extra requests and not from the randomization of legitimate requests.

7.1.1 Contributions to PANORAMIX

Our contributions are, as a result of a real life demand, the first implementation of a server-side WF defense and a simple yet effective lightweight client-side defense. With these two approaches we explore the space of application-layer defenses specifically designed to counter WF in `.onion` sites. In addition, we have collected the largest – to the best of our knowledge – dataset of sizes and types of content hosted by Tor `.onion` sites. We provide an evaluation of the overhead and efficacy of our defenses and compare it to some of the most practicable existing WF defenses.

The source code and datasets of both ALPaCA and LLaMA have been made publicly available on GitHub[1]. The original code is also available on an `.onion` site[2], which is protected using our defense.

7.2 Threat Model

As depicted in Figure 7.1, we consider an adversary who has access to the communication between the client and the entry point to the Tor network, known as *entry guard*. A wide range of actors could have access to such communications, ranging from malicious or corrupted relay operators, who can target all clients connecting to the guards they control; to ASes, ISPs and local network administrators, who can eavesdrop on Tor clients located within their infrastructure.

The adversary eavesdrops the communication to obtain a *trace* or *sample instance* of the encrypted network packets. He can observe and record these packets, but he cannot decrypt them.

[1] `http://github.com/camelids/`
[2] `http://3tmaadslguc72xc2.onion`

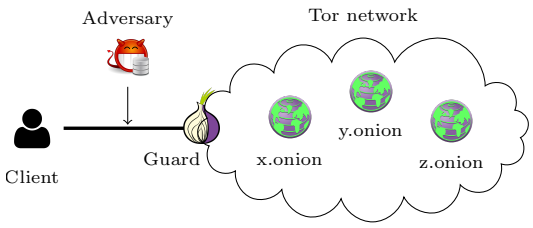


Figure 7.1: A client visits an `.onion` site over Tor. The attacker eavesdrops the encrypted link between the Tor client and the entry *guard* to the Tor network. Between the client and the destination onion service there is a six-hop Tor circuit that we have omitted to simplify the figure.

Furthermore, we will assume a *passive* adversary: he cannot remove or modify the packets, nor drop or add new packets to the stream.

The objective of the adversary is to infer the websites that were visited by the client from the traffic samples. The adversary can build a template for a number of websites with his own visits and then match the traffic generated by the client. It has been shown that, for a small set of websites, such an attacker can achieve high success rates achieving over 90% accuracy [5].

These attacks have however been criticized for making a number of unrealistic assumptions that favor the adversary [14]. For instance, they assume webpages are static, although some pages have frequent content updates; the client only visits pages that the attacker has trained on, also known as the *closed-world* assumption; and the attacker is able to perfectly parse the fraction of the continuous stream of traffic corresponding to a specific page download, assuming there is a gap between one visit and the next one.

In 2015, Kwon et al. showed that an attacker falling within this threat model can effectively distinguish visits to `.onion` sites from regular websites [16]. They also revisited the assumptions for which prior work on WF had been criticized [14] and found that many of these assumptions hold when considering only `.onion` sites. In contrast to the open Web, the world of `.onion` sites is small and comparable to a closed world, they are also more static than regular websites and their streams are isolated by domain [16]. As in virtually all prior work on WF, they still assumed the client visits only home pages, ignoring other pages in the website such as inner pages and logged-in or personalized pages that are not available to the attacker. In our evaluation, we follow them and only collect data for the home page of the `.onion` sites we have crawled.

We assume an adversary is only interested in fingerprinting `.onion` sites, and already has a classifier to tell `.onion` traffic apart from the bulk of client traffic. We focus on defenses that protect against the WF attack in the "onion world" because it is a more threatening setting than the one studied in most prior WF work on Tor; visits to `.onion` sites tend to be more sensitive than to pages whose IP address is visible to clients. Luo et al. argue that a WF defense must be implemented at the client-side because web servers have no incentive to offer such a service [19]. However, we believe `.onion` site operators are aware of the privacy concerns that Tor clients have and would make the necessary (minor) modifications in the server to implement our defense.

For the design of ALPaCA, we will assume there is no dynamic content. This includes content generated at the client-side (e.g., AJAX) as well as the server-side (e.g., a PHP script polling a database). This assumption simplifies the design of the server-side defense: ALPaCA requires the size of the web resources being loaded and it is hard to estimate the size of dynamic content a priori.

To assume that no JavaScript will run in the browser is not as unrealistic as it may seem given the high prevalence of JavaScript in the modern Web. The Tor Browser's security slider allows

users to select different levels of security, disabling partially or totally JavaScript. Furthermore, SecureDrop pages already ask their clients to disable JavaScript to prevent attacks such as cross-site scripting. It is reasonable to think that clients who protect themselves against WF will first disable JavaScript to prevent these other attacks.

7.3 Related Work

WF is typically modeled as a supervised learning problem. The attacker collects traffic traces for a large sample of websites that aims to identify and builds a classifier that outputs a label, with a certain level of confidence. Since the first WF classifiers were proposed in the late nineties [7], the attacks have been developed with improved classification models to defeat a wide variety of privacy enhancing technologies such as encrypting web proxies [28, 13], SSH tunnels[17], VPNs, and even anonymity systems such as Tor and JAP [12].

7.3.1 Attacks

The latest attacks against Tor achieve more than 90% accuracy in a *closed-world* of websites, where the attacker is assumed to have samples for all the websites a target user may visit [5, 31, 30, 11, 20]. This assumption is unrealistically advantageous for the attacker [14] and a recent study has shown that the attack does not scale to large open-worlds [20]. However, the `.onion` space is significantly smaller than the Web and may be feasible for an adversary to train on a substantial fraction of all `.onion` websites. Furthermore, the closed-world evaluation provides a lower bound for the efficacy of the defense. For a complete evaluation of the performance of our defenses, in this chapter we will provide results for both open and closed-world scenarios.

We have selected the most relevant attacks in the literature to evaluate our defenses:

k-NN [30]: Wang et al. proposed a feature set of more than 3,000 traffic features and defined an adaptive distance that gives more weight to those features that provide more information. To classify a new instance, the attack takes the label of the k Nearest Neighbors (k-NN) and only makes a guess if all the neighbors agree, minimizing the number of false positives.

CUMUL [20]: The features of this attack are based on the cumulative sums of packet sizes. The authors interpolated a fixed number of points from this cumulative sum to build the feature vectors that they use to feed a Support Vector Machine (SVM).

k-FP [11]: Hayes and Danezis used Random Forests (RF) to transform, based on the leafs of the trees, an initial feature set to another feature set that encodes the similarity of an instance with respect to its neighbors. Then, they also used a k-NN for final classification.

7.3.2 Defenses

Most WF defenses in the literature are based on *link-padding*. The traffic morphing approach attempts to transform the traffic of a page to resemble that of another page [32, 18, 21], or to generalize groups of traffic traces in to anonymity sets [30, 3]. The main downside of this type of defenses is that they require a large database of traffic traces that would be costly to maintain [14].

Link-padding aims to conceal patterns in web traffic by adding varying amounts of dummy messages and delays in flows. Link-padding has been used for traffic morphing to cause confusion in the classifier by disguising the target page fingerprint as that of another page [32]. However, as Dyer et al. note [9], traffic morphing techniques produce high bandwidth overheads as some packets

must be buffered for a long period. The strategy we follow in ALPaCA is different from traffic morphing in that page contents are not disguised as other pages', but rather the content is modified to become less *fingerprintable*. The intuition behind ALPaCA is to make each resource look like an "average" resource, according to the distribution of resources in the world of pages. This approach reduces the overheads with respect to morphing, as resizing an object to an average size will tend to require less amount of padding than to an object of a specific page. We have experimented with morphing the contents in a page to make it look like another page. This can be seen as the application-level counterpart of *traffic morphing* and the results can be found in chapter 7.7.

In 2012, Dyer et al. presented *BuFLO* [9], a defense based on constant-rate link-padding. Although *BuFLO* is a proof-of-concept defense and has high bandwidth overheads, other defenses have been developed from the original *BuFLO* design. *Tamaraw* [4] and *CS-BuFLO* [2] optimize *BuFLO*'s bandwidth and latency overheads to make its deployment feasible. Both of these defenses address the issue of padding the page's tail. BuFLO padded all pages up to a certain maximum number of bytes producing the high bandwidth overheads. CS-BuFLO and Tamaraw proposed a strategy to pad pages to multiples of a certain parameter, which groups pages in anonymity sets by size and significantly reduces the bandwidth overhead over BuFLO. We follow a similar strategy for one of the modes of ALPaCA.

Recently, a lightweight defense based on Adaptive Padding has also been proposed to counter WF [15]. In order to offer low latency overheads, this defense only pads time gaps in traffic that are statistically unlikely to happen. To empirically determine the likelihood of a gap they sampled a large number of pages over Tor and built a distribution of inter-arrival times used to sample the delays for the dummy messages.

Our main concern with these designs is that padding is applied at the network layer. There is no need to apply the defense at the network layer because layers below HTTP do not carry identifying information about the webpage. One could argue that latency and bandwidth identify the web server. However, these features vary depending on network conditions and are shared by all pages hosted in the same server or behind the same CDN. Moreover, the implementation of such defenses may require modifications in the Tor protocol and even the TCP stack, as they generate Tor cells that are sent over Tor's TLS connections.

Application layer defenses act directly on the objects that are fingerprinted at the network layer. The padding is also added directly to these objects. As opposed to network-layer defenses that must model legitimate traffic to generate padding, application-layer defenses inject the padding inside the encrypted payload and is, consequently, already indistinguishable from legitimate traffic at the network layer. In addition, defenses at the application layer do not require modifications in the source code of the Tor protocol, which make them more suitable for deployment.

In this chapter we present and explore two novel approaches for application layer defenses at both client and server-side. In the rest of this section we describe the state of the art on application-layer defenses.

Server-side

To the best of our knowledge, there is only a prototype of a server-side defense that was drafted by Chen et al. and it was designed for a slightly different although related problem [6]. They studied WF in the context of SSL web applications, where the attacker is not trying to fingerprint websites, but specific pages within one single website. Their main contribution was to show that a local passive adversary can identify fine-grained user interactions within the website. The authors devise a defense that requires modifications at both client and server sides, which allows padding to be added to individual HTTP requests and responses.

Client-side

There are only two application-layer defenses proposed in the WF literature: *HTTPOS* [19] and *Randomized Pipelining* (RP) [23]. Luo et al. proposed *HTTPOS* as a client-side defense arguing that server-side or hybrid defenses would see little adoption in the wild due to lack of incentives [19]. In that study, the authors pinpoint a number of high-level techniques that alter the traffic features exploited by WF attacks. For instance, they modify the HTTP headers and inject fake HTTP requests to modify the length of web object sizes.

RP is the only WF countermeasure that is currently implemented in the Tor browser. It operates by batching together a single clients requests in the HTTP pipeline to randomize their order before being sent to the server. Several studies have applied WF attacks on data collected with a RP-enabled Tor Browser and all of them have shown that the defense was not effective at decreasing the accuracy of the WF attack in the closed world [5, 31, 14]. The reason why RP does not work is not clear and has not been investigated in these evaluations.

7.4 Defenses

WF attacks are possible because different webpages serve different content. High level features such as the number of requests the browser makes to download a page, the order of these requests and the size of each response, induce distinctive low level features observed in the network traffic [9, 21]. For instance, the number of requests sent by the browser corresponds to the number of objects embedded in the page such as images, scripts, stylesheets, and so on.

Most existing defenses propose to add spurious network packets to the stream to hide these low-level features [9, 2, 4], However, effectively concealing these features at network level poses technical challenges, as the operation of underlying protocols, i.e. TLS, TCP, IP, obfuscates the relation between low and high level features. For this reason, we believe adding the padding to the actual contents of the page is a more natural strategy to hide traffic features than sending dummy packets: if the defense successfully conceals high-level features, the low-level features will follow.

In this section, we describe in detail the strategies that we propose at the application layer at both server (ALPaCA) and client side (LLaMA) to mitigate WF attacks.

7.4.1 ALPaCA

ALPaCA is a server-side defense that pads the contents of a webpage and creates new content with the objective of concealing distinctive features at the network level. We demonstrate that this strategy is not only effective, but also practical to deploy. We have implemented and evaluated ALPaCA as a script that periodically runs on a server hosting an `.onion` site.

We first show that it is possible to pad the most popular types of webpage objects (e.g., images, HTML) to a desired size, without altering how they look to a user. We then propose two variants of server-side defenses, referred to as P-ALPaCA and D-ALPaCA. At a high level, the defenses choose, for a page to morph, a suitable list of sizes T, that we call *target*. A target specifies the number and size of the objects of the morphed page; P-ALPaCA and D-ALPaCA differ in how they select such a target. Then, the objects of the original page are padded to match the sizes defined in T. If T contains more elements than the page's objects, then new objects ("padding objects") are created and referenced from the morphed HTML page (Algorithm 1). Figure 7.2 gives a high level overview of this process.

Padding an object to a target size

This section describes how we can pad most types of objects. It is important to note that an adversary looking at encrypted packets cannot: i) distinguish the type of objects that are being downloaded, ii) infer how much padding was added to such objects or whether they were padded at all. By padding an object directly on the server, we can control how large it will look like at the network level. While this control is not complete (because of compression in the HTTP protocol), experiments show that this discrepancy does not largely affect on our defenses.

Table 7.1 shows the types of objects that we can pad up to a desired size, and their frequency within the `.onion` site world. To pad text objects (e.g., HTML and CCS) we can add the desired amount of random data into a comment. To pad binary objects (e.g., images), it is normally sufficient to append random data to the end of the file; in fact, the file structure allows programs to recognize the end of the file even after this operation.

We verified that binary files would not be corrupted after appending random bytes to them as follows. We used ImageMagick's `identify` program[3] for verifying the validity of PNG, ICO, JPEG, GIF, and BMP files after morphing. The program only raised a warning "length and filesize do not match" for the BMP file; the image was, nevertheless, unaffected, as it could be opened without any errors. We used `mp3val`[4] to check MP3 files; the program returned a warning "Garbage at the end of the file", but the file was not corrupted, and it could be played. We used `ffmpeg`[5] to verify AVI files; the program did not return any errors or warnings.

It is thus possible to morph the most common object types. We suspect that many other types of object can be morphed analogously, by appending random bytes or by inserting data in comments or unused sections of the type structure. We remark, however, that in experiments we did not remove content we could not morph from webpages.

Morphing a page to a target T

We introduce Algorithm 1, which morphs the contents of a page to match the sizes defined by a target T. The target is selected differently by the two versions of ALPaCA, as presented later, and it defines the size of the objects that the morphed page should have.

The algorithm keeps two lists: M, containing the morphed objects, and P, which keeps track of the sizes in T that have not been used for moprhing an object; both lists M and P are initially empty. The algorithm sequentially considers the objects of the original page from the smallest to the largest; for object o, it seeks the smallest size $t \in T$ which o can be padded (i.e., for which $size(o) \leq t$). Once it has found such a t, it removes all the elements of T smaller than t, and pads o to size t; the elements removed from T at this stage (except t) are put into P. After all the original objects have been morphed, the sizes remaining in T are appended to P. New "padding objects" (objects containing random bytes) are generated according to the sizes in P. We make sure that padding objects will be downloaded by a browser, but will not be shown, by inserting a reference to them in the HTML page as if they were hidden images[6]. Finally, the HTML page itself is padded to a target size by the defense.

[3] `http://www.imagemagick.org/`

[4] `http://mp3val.sourceforge.net/`

[5] `https://ffmpeg.org/`

[6] To add an invisible object called "rnd.png" to an HTML page we insert `<img src="rnd.png" style="visibility:hidden">'`. The browser will consider this a PNG file and it will download it, but it will not attempt to show it. The file, thus, needs not to respect the PNG format, and it can just contain random bytes.

Content type	Morphing	Frequency
PNG, ICO, JPG, GIF, BMP	Append random bytes to the file.	51%
HTML	Insert random data within a comment “¡!–”, “–¿”.	13%
CSS	Insert random data within a comment “/*” “*/”.	12%
JS	Insert random data within a comment “/*” “*/”.	13%
MP3	Append random bytes to the file.	N.O.
AVI	Append random bytes to the file.	N.O.

Table 7.1: Padding the most frequent objects in `.onion` sites to a desired size. “N.O.” stands for “not observed”. We assume JavaScript is disabled, although it is possible to morph JS files as shown.

P-ALPaCA

P-ALPaCA (Probabilistic-ALPaCA) generates a target by randomly sampling from a distribution that represents real-world `.onion` sites. Specifically, it has access to three probability distributions D_n, D_h and D_s, defined respectively on the number of objects a page has, the size of the HTML page and the size of each of its objects. The defense samples a target T using these distributions, and then morphs the original page as shown in Algorithm 1.

We estimated D_n, D_h and D_s using Kernel Density Estimation (KDE) from $5,295$ unique `.onion` websites we crawled. Details about crawling and analysis of these websites are in Section 7.5. In Table 7.7 we show the resulting distributions D_n, D_h and D_s, and provide details on how we used KDE to estimate them.

The defense first samples the number of objects n for the morphed page according to D_n. Then, it samples the size of the morphed HTML from D_h, and n sizes from D_s which constitute a target T. Finally, it attempts to morph the original page to T (Algorithm 1); if morphing fails, the procedure is repeated. The algorithm is shown in Algorithm 2.

Because sampling from the distributions can (with low probability) produce very large targets T, we introduced a parameter *max_bandwidth* to P-ALPaCA. Before morphing, the defense checks that the total page size is smaller than or equal to this parameter: $\sum_{t \in T} t \leq max\_bandwidth$. If not, the sampling procedure is repeated.

A simple alternative to sampling from a distribution that represents the present state of the `.onion` world, is to sample the number and size of padding objects uniformly at random. We expect that this alternative approach would also set a maximum bandwidth parameter, which would serve as the upper bound of the size of the morphed page. One could imagine that a naive implementation of this alternative approach which sets a high maximum would cause extremely high bandwidth

Algorithm 1 Pad a list of objects to a target

Input: O: list of original objects
T: list of target sizes
Output: M: list of morphed objects

$M \leftarrow [\,]$
$P \leftarrow [\,]$
▷ Morph the original objects.
while $|M| < |O|$ **do**
  $o \leftarrow \arg\min_{o \in O} size(o)$
  ▷ Remove the target sizes smaller than $size(o)$.
  while $min(T) < size(o)$ **do**
    Remove $min(T)$ from T
    Append $min(T)$ to P
  end while
  if T is empty **then**
    ▷ Cannot morph O to T
    fail
  end if
  ▷ Note: the current $min(T)$ is larger than $size(o)$
  $t \leftarrow min(T)$
  $m \leftarrow o$ padded to size t
  Append m to M
end while
▷ Add padding objects.
Merge P and T into P
for p in P **do**
  $m \leftarrow$ New padding object of size p
  Append m to M
end for

overheads. However, reducing this maximum parameter would constrain the morphed page to look like a small subsection of the onion world, removing altogether the possibility that the page is morphed to resemble a large `.onion` site. Our approach allows a large maximum bandwidth parameter to bet set while ensuring bandwidth overheads will be low. With our approach, the probability that a small page, say `A.onion`, is morphed to the size of a large `.onion` site, say `B.onion`, directly corresponds to the ratio of the number of `.onion` sites within the `.onion` world that are of an equal size to `B.onion`. Meaning a small `.onion` site can have the entire `.onion` world as an anonymity set while ensuring a low bandwidth overhead.

D-ALPaCA

We propose a second server-side defense, D-ALPaCA (Deterministic-ALPaCA), which decides deterministically by how much a page's objects should be padded. The defense is inspired by Tamaraw [4], which pads the number of packets in a network trace to a multiple of a padding parameter L. D-ALPaCA has the advantage of introducing less overheads than P-ALPaCA, but experimental results suggest this defense is slightly less effective against a WF adversary.

Algorithm 2 P-ALPaCA

Input: O: list of original objects
D_n: distribution over the number of objects
D_h: distribution over the size of HTML pages
D_s: distribution over the size of objects
$html\_size$: size of the original HTML page
$max\_bandwidth$: maximum page size

▷ We use $x \leftarrow^{\$} D$ to indicate that x is sampled from distribution D
$morphed \leftarrow False$
while not $morphed$ **do**
  $T \leftarrow [\,]$
  $h \leftarrow^{\$} D_h$
  if $h < html\_size$ **then**
    continue
  end if
  $n \leftarrow^{\$} D_n$
  for i in $1..n$ **do**
    $s \leftarrow^{\$} D_s$
    Append s to T
  end for
  if $sum(T) < max\_bandwidth$ **then**
    Try morphing O to target T (Algorithm 1)
    If successful, $morphed \leftarrow True$
  end if
end while
Pad the HTML page to size h

D-ALPaCA (Algorithm 3) accepts as input three parameters: λ, σ and max_s, where max_s should be a multiple of σ. It pads the number of objects of a page to the next multiple of λ, and the size of each object to the next multiple of σ. Then, if the target number of objects is larger than the original number of objects, it creates padding objects of size sampled uniformly at random from $\{\sigma, 2\sigma, ..., max\_s\}$. Experiments in Section 7.6 evaluate how different sets of parameters influence security and overheads.

Practicality of the defenses

Both P-ALPaCA and D-ALPaCA are practical to use in real-world applications. In fact, they only require a script to morph the contents of a page periodically. This can be done by setting up a `cron` job running the defense's code, which we release.

Since it is preferable to morph a page after each client's visit, and it may be difficult for the server operator to decide how frequently they should run the `cron` job, we propose a more sophisticated (and flexible) alternative. The defense should preemptively morph the web page many times, and place the morphed pages within distinct directories on the server. Then, the server should be configured to redirect every new request to a different directory. Once the content of a directory has been loaded, the directory is removed, and a new one can be created.

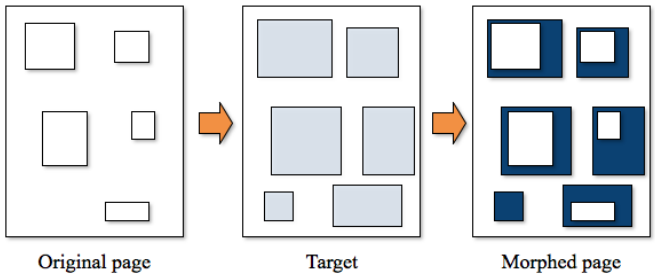


Figure 7.2: Graphical representation of the server side defenses. Server side defenses P-ALPaCA and D-ALPaCA first select a target for the original web page. Then, they pad the contents of the original page as defined by the target (Algorithm 1), and generate new padding objects if needed. The original and morphed page will look identical to a user.

Third-party content

A limitation of ALPaCA is that it can only pad resources hosted in the web server, thus content linked from third parties cannot be protected. In the evaluation of the defense, we have intentionally omitted all third-party content because only two out of the 100 pages in our dataset had resources from third parties.

To understand the impact of this assumption on a larger scale, we have analyzed the prevalence of third-party resources in a crawl of 25K `.onion` sites: only 20% of these sites create requests to third-party domains. Furthermore, for half the pages with third-party content, the third-party requests account for less than 40% of total requests observed within a webpage. However, we found a handful of sites that had more than 90% of their content hosted in third parties. They seem to act as proxies to existing websites. With such a high percentage of unprotected content, the defense is most likely to fail at providing protection against website fingerprinting.

Since the average cost in terms of disk space is 5MB, a possible solution for sites with a large proportion of third-party content would be to cache the third-party resources in the server running the defense. We strongly discourage this approach as if not implemented properly, the `.onion` site, attempting to keep these resources updated, may become vulnerable to timing correlations attacks by the third parties serving the content. In fact, we recommend `.onion` site operators minimize the amount of third-party content they embed to their pages and only cache static content that does not require periodic updates.

7.4.2 LLaMA

LLaMA is inspired by Randomized Pipelining (RP) [23]. RP modifies the implementation of HTTP pipelining in Firefox to randomize the order of the HTTP requests queued in the pipeline. However, RP has been shown to fail at thwarting WF attacks in several evaluations [5, 31, 14].

LLaMA is implemented as an add-on for the Tor browser that follows a similar strategy to RP: it alters the order in which HTTP requests are sent. The main advantage of a WF defense as a browser add-on is ease of deployment: it does not require modifications to the Tor source code. Thus, a user can install the add-on to enable the protection offered by the defense independently or, if the Tor Project decides to, it could be shipped with the Tor Browser Bundle.

RP exposes a debug flag that logs extra information about its use of the HTTP pipeline [22]. A dataset collected with this flag enabled, visiting the same webpages that the aforementioned

Algorithm 3 D-ALPaCA

Input: O: list of original objects
σ: size parameter
λ: number of objects parameter
$html\_size$: size of the original HTML page
max_s: maximum size of a padding object (should be a multiple of σ)

▷ We use $x \leftarrow^{\$} S$ to indicate that x is sampled uniformly at random from a set S
$T \leftarrow [\,]$
$h \leftarrow$ next multiple of σ greater or equal to $html\_size$
for o in O **do**
    $s \leftarrow$ next multiple of σ greater or equal to $size(o)$
    Append s to T
end for
$n \leftarrow$ next multiple of λ greater or equal to $size(O)$
while $size(T) < n$ **do**
    $s \leftarrow^{\$} \{\sigma, 2\sigma, ..., max\_s\}$
    Append s to T
end while
Morph O to target T (Algorithm 1)
Pad the HTML page to size h

evaluations did, provided evidence of a suboptimal usage of the HTTP pipeline by RP [24]. Either the design of those pages or the low adoption of HTTP pipelining on the servers of these pages or CDNs in between may account for the low performance of RP [1]. Since our defense does not depend on HTTP pipelining, it allows us to test whether these hypotheses hold or it is actually the randomization strategy which is flawed.

Delaying requests. In order to randomize the order of the HTTP requests, the add-on intercepts all requests generated during a visit to a website and adds a different random delay to each one (see Figure 7.3). We use the statistics extracted from Section 7.5.2 to set the distribution of delays for the requests. We take the median page load time in our crawl and set a uniform distribution from zero to half the median load time. As a result, on average, each request will be delayed within a window of half the page load time. In the worst case, this approach will introduce 50% latency overhead if the last request is delayed by the maximum time in the distribution.

Extra requests. As shown in Figure 7.3, every time a request is sent or a response is received, the extension can be configured to send an extra request. It tosses a coin to decide whether to make an additional HTTP request or not. These fake HTTP requests are sent to a web server that serves custom-sized resources: a parameter in the URL indicates the size of the resource that will be sent in the response body. This allows us to fake random responses from the client-side. Tor isolates streams in different circuits per domain, since such fake requests are made to a different domain they will be sent through a different circuit. This should not be a problem because the attacker cannot distinguish them from legitimate third-party requests. However, as we discuss in the following section, third-party content in `.onion` sites has low prevalence. In addition, this approach requires a trusted server that can be queried from the add-ons. To avoid these issues, the extension implements an alternative method to generate extra responses: it keeps a hash table with domains as keys and lists of request URLs sent to that domain during a browser session as

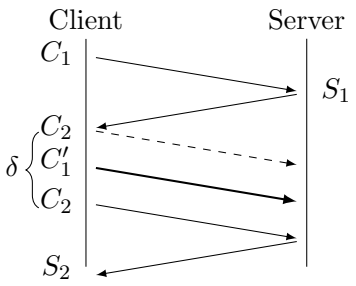


Figure 7.3: Graphical representation of the LLaMA's operation. δ is the delay added to C_2. C'_1, in bold, requests the same resource as C_1.

values. To generate a new request, it uniformly samples a URL from the list corresponding to the current first-party domain and sends a request to that URL.

To change the size of legitimate requests we would require cooperation of the server. We acknowledge that previous defenses have proposed this approach [6], but our focus for this defense is to not require any change at the server-side.

7.5 Methodology

In this section we describe the methodology that we followed to collect the data and evaluate the defenses. This data was also used to create the probability distribution used by P-ALPaCA.

7.5.1 Data collection

For the collection of the dataset we used the `tor-browser-crawler`[7], a web crawler that provides a driver for the Tor Browser, allowing the automation of web page visits in conditions similar to those of regular Tor users. We added support for the Tor Browser Bundle 5.5.5, the latest version at the time of our crawls (March 2016) and extended the crawler to intercept all HTTP requests and responses for future inspection. The crawler logs the size and the URL for each HTTP request and response. The crawler also allows to modify browser preferences. We used this feature to disable JavaScript and RP when needed.

We crawled a list of `.onion` sites obtained from *Ahmia*[8], the most popular search engine for onion services. Ahmia maintains a blacklist of illegal `.onion` sites and thus are excluded from our crawls. The crawl consisted of $25,000$ `.onion` instances, after removing time-outs and failed loads, we captured $18,261$ instances of an `.onion` site load from $5,295$ unique addresses. This dataset serves as both the basis for which we conduct WF attack experiments with our defense in place, as a source of information when inferring the distribution of objects that the server-side defense should conform to, and as a source of load time statistics for which the client-side defense decides when to inject additional requests.

7.5.2 Data analysis

[7] `https://github.com/webfp/tor-browser-crawler`
[8] `https://ahmia.fi`

From the 18,261 instances, a total of 177,376 HTTP responses and 7,095 HTTP requests were captured. The average amount of uploaded data per `.onion` site was 256B, while the median amount of uploaded data per `.onion` site was 158B. The average amount of downloaded data per `.onion` site was 608KB, while the median amount of downloaded data per `.onion` site was 45KB. The average size of one response was 55KB; the average size of a request was 87B. Clearly the amount of downloaded data surpasses the amount of uploaded data as clients are simply issuing a HTTP request for objects within the server.

The average number of requests to an `.onion` site was 3, while the average number of responses was 11.

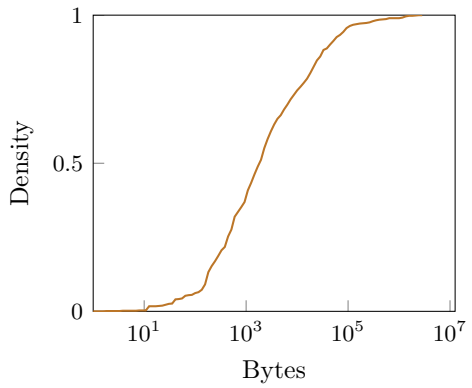


Figure 7.4: CDF of the HTTP response size in the 25K crawl (in log scale).

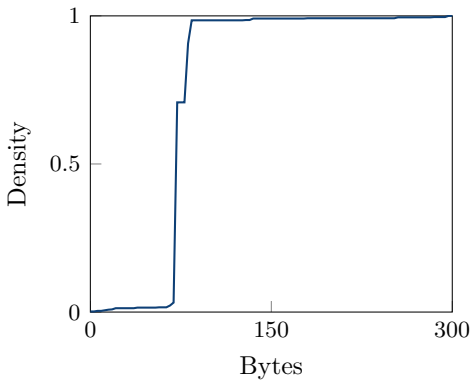


Figure 7.5: CDF of the HTTP request size in the 25K crawl.

The average size of an `.onion` site then is a little over 608KB. In 2015, the average standard website was just over 2MB, and the average number of objects was over 100 [25, 29], much larger than the average size and number of objects of an `.onion` site. Clearly there is a distinct difference between standard websites and `.onion` sites; standard websites are much larger and contain a greater number of objects within the HTML index page, we note however that the space of all standard websites is orders of magnitude greater than the space of all `.onion` sites and so contains much greater variance in both size and number of objects.

From Figure 7.5 we see that nearly all HTTP requests were less than 100 bytes, combining this with the knowledge that there are on average just three HTTP requests to download the `.onion` site, we can infer it is most common to download the entire site with just one or two requests after the initial HTTP GET request. From Figure 7.4, 99% of HTTP responses are less than 1MB in length, and nearly 70% are less than 10KB.

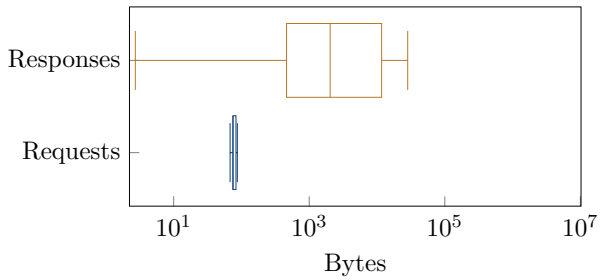


Figure 7.6: Boxplot of the HTTP request and response sizes for 25K `.onion` sites.

From Figure 7.6 we see that the majority of requests are between $70 - 100$B, with relatively few outliers. There is a large skew between the majority of responses of size less than a few KB's and a comparatively (to the number of request outliers) large number of response outliers that are orders of magnitude larger in size than the average response size.

7.6 Evaluation

To assess the effectiveness of our defenses against WF attacks, we have crawled the same set of pages with and without the defenses in place. Comparing the accuracy of state-of-the-art attacks on both datasets provides an estimate of the protection offered by the defenses.

7.6.1 P-ALPaCA & D-ALPaCA Evaluation

We evaluate the server-side defenses when a server does not wish to transform its network traffic to look like another `.onion` site but wishes to morph their traffic so it resembles an "average" `.onion` site. We use results from Section 7.5.2 to extract information such as the average number of objects and the average size of these objects across all `.onion` sites. A participating server can then use such information to modify their `index.html` page, resulting in an `.onion` site resembling, at the network layer, many different `.onion` sites rather than a specific targeted site.

The object distributions statistics may change over time and require periodic updates. However, to determine whether they change and how often is out of the scope of this chapter and leave it for future research. Such an update mechanism could be served by a trusted entity in the Tor network (e.g., a directory authority) that supplies `.onion` sites with this information.

In addition to transforming the network traffic of an `.onion` site to resemble many different "average" `.onion` sites rather than a targeted site, this method allows the server to control the bandwidth overheads at a more fine grained level, since the server can decide the amount and size of extra objects placed in the `index.html` page.

The server also has control over how often their site is morphed. The frequency of morphing depends on the estimation of how quickly an adversary can mount an attack. If an adversary,

	Latency		Volume	
	%	Avg. (s)	%	Avg. (KB)
Undefended	–	3.99	–	175
P-ALPaCA	52.6	6.09	86.2	326
D-ALPaCA (2, 500, 5000)	66.3	6.63	3.66	182
D-ALPaCA (2, 5000, 5000)	56.1	6.22	9.84	193
D-ALPaCA (5, 2500, 5000)	61.7	6.44	15.1	202
D-ALPaCA (10, 5000, 5000)	41.7	5.65	44	254

Table 7.2: P-ALPaCA & D-ALPaCA latency and bandwidth overheads.

can train on network traffic from the server and monitor during a period where the site remains unchanged, the defense will not be of any use. However, the time to train and launch an attack on a number of `.onion` sites will likely be in the order of hours not minutes[9], as long as a server morphs the site in a shorter period than this, the training data the attacker gathers will be of little use.

To confirm this assertion, we collected 40 network traffic loads, which we call an instance, for each site of 100 `.onion` sites. We chose 100 `.onion` sites that resembled the average size of an `.onion` site[10], in terms of total page size and number of objects. We also collected 40 P-ALPaCA morphed instances for each of the `.onion` sites, such that each instance is the result of a new morphing process[11] . We then check whether an adversary, training on different morphed versions of an `.onion` site, can still correctly determine the `.onion` site of origin.

More specifically, for each of the 100 `.onion` sites, we collect 40 instances. Resulting in 4000 overall traces. We then apply our server-side defense and re-visit the newly defended sites, resulting in another 4000 traces. We then apply, separately, WF attacks to both undefended and defended `.onion` sites, training on 60% of traces and testing on the remaining 40%. We consider the defense successful if the WF attack accuracy on the defended `.onion` sites is dramatically lower than attack accuracy on the undefended `.onion` sites.

To explore the parameter space, we also evaluated D-ALPaCA, under four different parameter choices. We collected 20 instances for the same 100 `.onion` sites and compared attack accuracy against both the undefended and P-ALPaCA defended `.onion` sites. The parameter choices were: λ - the defended page will have a multiple of λ objects, σ - each of the defended page's objects will have a size which is multiple of σ, max_s - when generating new padding objects, sample uniformly within the set [σ, 2*σ, 3*σ, ..., max_s]. Specifically, we chose the following parameter values for (λ, σ, max_s): $(2, 500, 5000)$, $(2, 5000, 5000)$, $(5, 2500, 5000)$, $(10, 5000, 5000)$.

User Experience: in Table 7.2, we see that average latencies are approximately 40-60% greater in the protected traces than in the unprotected ones. In seconds, the extra time that the user

[9]For example, we used a total of 100 `.onion` sites in experiments, visiting each `.onion` sites 40 times. We trained on 60% of data. The average page load time was around 4 seconds. Therefore an attacker, using one machine for crawling and gathering training data, would be able to initiate an attack after 9600 seconds. However, we note an attacker can parallelize this process for faster attacks.

[10]Via Section 7.5.

[11]As proposed in Section 7.4.1, the `.onion` site is differently morphed upon every client visit.

will spend loading the pages is between two and three seconds. We also measured the times to load the original resources in the protected traces with respect to loading all content, since serving extra padding resources once all the original content is sent does not impact on user experience. We call the time between the first request to the last legitimate request *UX-time*. However, the average difference between UX-time and the time to load all resources in a protected page is less than 200ms. We notice that the randomization of RP often sends original requests at the end of the transmission which explains the mild difference between UX-time and total page load time.

	k-NN (%)	k-FP (%)	CUMUL (%)
Undefended	45.6	69.6	55.6
P-ALPaCA	0.2	9.5	15.6
D-ALPaCA (2, 500, 5000)	9.5	22.7	27.0
D-ALPaCA (2, 5000, 5000)	12.5	34.4	40.0
D-ALPaCA (5, 2500, 5000)	5.8	22.3	30
D-ALPaCA (10, 5000, 5000)	7.2	22.9	33.0
Decoy [21]	4.9	11.2	X
Tamaraw [4]	6.8	14.0	X
BuFLO [9]	5.3	13.3	X

Table 7.3: Closed world classification for `.onion` sites morphed via P-ALPaCA and D-ALPaCA, with other defenses added for comparison. CUMUL depends on packet lengths and so some defenses that only operate on packet time information cannot be applied.

Closed World classification: we performed a closed world WF attack on P-ALPaCA defended, D-ALPaCA defended and undefended `.onion` sites. If our server-side defenses are successful, defended `.onion` sites should, at the network level, look similar to one another and result in a low classification accuracy. We use CUMUL [20], k-FP [11] k-NN [30] for evaluation[12]. The number of neighbours used for classification is fixed at two.

7.3 shows the closed-world classification results of undefended `.onion` sites against `.onion` sites with each instance uniquely defended using P-ALPaCA or D-ALPaCA. WF attacks are ineffective under both defenses, and in fact P-ALPaCA improves upon *Tamaraw* and *BuFLO*. D-ALPaCA does slightly worse than the P-ALPaCA in terms of defending `.onion` sites, but as can be seen from Table 7.2, has real advantages in terms of limiting bandwidth overheads. For example, D-ALPaCA with parameters (2, 500, 5000), reduced k-FP accuracy from 69.6% to 22.7%, compared to the P-ALPaCA which reduced attack accuracy to 10%. But, D-ALPaCA (2, 500, 5000) required 23.6 times less bandwidth than P-ALPaCA to achieve these results. A server operator wishing to provide a defense to its clients while limiting the increase in bandwidth may then consider this a worthwhile trade-off and choose to use D-ALPaCA over P-ALPaCA.

Open World classification: in addition to closed world experiments, we evaluated the server-side defenses in the open world setting, where we include network traffic instances of `.onion` sites that are not of interest to the attacker. We observe how the classification accuracy is affected

[12]We use Tobias Pulls' implementation of the k-NN website fingerprinting attack [26].

	k-NN (%)		k-FP (%)		CUMUL-k-FP (%)	
	TPR	FPR	TPR	FPR	TPR	FPR
Undefended	37.0	1.0	62.1	0.8	49.7	5.4
P-ALPaCA	0.4	0.2	3.6	0.2	1.1	1.3
D-ALPaCA (2, 500, 5000)	4.5	0.2	12.0	0.4	21.4	1.4
D-ALPaCA (2, 5000, 5000)	7.5	0.4	12.6	0.4	28.8	1.2
D-ALPaCA (5, 2500, 5000)	6.0	0.3	12.7	0.3	18.7	1.3
D-ALPaCA (10, 5000, 5000)	3.4	0.3	13.3	0.3	27.3	1.0

Table 7.4: Open world classification for `.onion` sites morphed P-ALPaCA and D-ALPaCA.

in this setting, which is intended to reflect a more realistic attack. We use 5,259 unique `.onion` sites, from Section 7.5.2, as background traffic instances[13] and set the number of neighbours used for classification at two. Note that CUMUL only does binary classification in the open world, classifying as either a background instance or a foreground instance of interest, whereas k-FP and k-NN attempt to classify an instance to the correct `.onion` site if it is flagged as a non-background instance. In order to compare the results of the attacks in the open-world, we have used the feature vectors of CUMUL while applying the k-FP classification process. To make sure that the classification model does not affect the accuracy of the attack, we evaluated the CUMUL features with k-FP in a closed-world and achieved a similar accuracy to SVM.

As we can see from Table 7.4 there is a dramatic decrease in attack accuracy when both P-ALPaCA and D-ALPaCA are used, showing that if a server morphs their site at a higher rate than the adversary can gather training data, the site will be almost perfectly concealed.

D-ALPaCA parameter choices: 7.3 and 7.4 show there is no notable difference in attack accuracy when changing parameters. However, as expected, smaller parameter choices led to smaller bandwidth overheads.

7.6.2 LLaMA Evaluation

We have crawled the same list of `.onion` sites as in the evaluation of ALPaCA, under four different conditions:

JS enabled: we collected our data with no defense installed and JavaScript enabled, the default setting in the Tor Browser.

JS disabled: we repeated the same crawl as with JS enabled but disabling JavaScript in the Tor Browser. We keep JS disabled for the rest of our crawls.

RP with delays: we collected data with the defense only delaying requests, altering the order of the requests as described in Section 7.4.

Extra requests: we crawled the same data with the defense adding delays and extra requests as described in the previous section.

[13]For k-FP, we train on 1,000 of the 5,259 background traces and for each `.onion` site we train on 50-75% of instances. Whereas k-NN uses Leave-one-out cross-validation on the entire dataset.

We note that we have disabled RP in the Tor Browser for all the crawls above by disabling the browser preference `network.http.pipelining`.

In Table 7.5, we show the results for the three classifiers in the closed world of 100 onion sites. We do not observe much difference in accuracy between JavaScript enabled and disabled. This shows that our assumption of no dynamic content holds for the list of onion sites used in our evaluation.

	k-NN (%)	k-FP (%)	CUMUL (%)
JS enabled	64.0	55.8	52.4
JS disabled	60.8	53.4	52.7
RP with delays	46.8	47.9	49.6
Extra requests	31.5	36.0	34.8

Table 7.5: Closed world classification for `.onion` sites under different countermeasures.

When the defense only adds delays to requests, the accuracy of the classifiers decreases 10% in the k-NN classifier and has limited effect on k-FP and CUMUL. The mild impact on the accuracy of the classifier may imply that the hypothesis that RP does not work because servers do not support HTTP pipelining does not hold, suggesting that the request randomization strategy is flawed, as previous evaluations have argued [5, 31].

We also evaluated the scenario in which the countermeasure, besides adding delays, repeats previous HTTP requests. We observe a significant decrease in accuracy to almost half the accuracy obtained in the unprotected case for the k-NN classifier.

In Table 7.6, we show the overheads of LLaMA for its two different modes. We see that overheads are around 10%. Even though the protection provided by the defense is considerably lower than the server-side defense or other defenses in the literature, its simplicity and the small overhead that it introduces makes it a good candidate for a WF countermeasure.

	Latency		Volume	
	%	Avg. (s)	%	Avg. (KB)
JS disabled	–	5.01	–	126
RP with delays	8.4	5.42	X	X
Extra requests	9.8	5.49	7.14	135

Table 7.6: Latency and bandwidth overheads of the client-side defense in the closed world.

7.7 Discussion and Future Work

Both the ALPaCA and LLaMA have performed at least as well as state-of-the-art defenses, showing that application layer WF defenses do indeed protect against attacks. Next we discuss potential avenues for future research.

Ease of Deployment. We argue that application layer defenses are simpler to implement than previously proposed approaches as they require no modifications to existing protocols or participation from a relay in the circuit. The only expensive part of ALPaCA comes in the form of the gathering of statistics for the probabilistic based morphing approach. However, we suggest this cost can be amortized over all participating servers by allowing a centralized entity to collect this information, such as is done by directory authorities now to collect Tor relay descriptors. Future research could determine how often these statistics must be updated. Implementation of the client-side defense is simple, as we developed it as a browser add-on. This could be made available to Tor clients either by direct integration in to the Tor browser bundle, or through an add-on store.

Rate of Adoption. Initially, we expect relatively few `.onion` sites to implement server-side defenses. Over time if a significant number of `.onion` sites adopt ALPaCA, it is possible that a large fraction of sites will morph their page to resemble one another. In turn, this will create stable anonymity sets of `.onion` sites that have the same network traffic patterns. Finding the rate and size of these anonymity sets is left for future work.

Clearly, smaller `.onion` sites are easier to protect than larger ones, as it is impossible to morph a larger site to resemble network traffic patterns of a smaller site. Thus, we expect larger `.onion` sites to be more difficult to protect over time. However, as Section 7.5.2 show, the majority of `.onion` sites are small and so should be relatively simple to defend against WF attacks.

Latency and Bandwidth Overheads. All WF defenses come at the expense of added latency and bandwidth. Our defenses allow the exact overheads to be tuned by the participating client or server. We saw from Section 7.6.1 that P-ALPaCA adds, on average, 52.6% extra waiting time and 86.2% additional bandwidth. We note, that compared to previous works, these overheads are relatively small, and that due to the nature of `.onion` sites, even the morphed pages are small in size compared to standard web pages. LLaMA improves on striking a balance between overhead limitation and protection against WF attacks. By issuing additional HTTP requests, WF attack accuracy is halved, while only adding 9.8% in waiting time and 7.14% in bandwidth. We also saw comparably small overheads in our D-ALPaCA defense which significantly reduced WF attack accuracy at the expense of an additional 3.66% of bandwidth.

Natural WF Defenses. We note that compared to related works, the attack accuracy on `.onion` sites seems alarmingly low. Wang et al. [30] achieved accuracies of over 90% when fingerprinting the top 100 Alexa websites, whereas our experiments on 100 `.onion` sites resulted in an accuracy of only 45.6% using the same classifier. We have validated the results of Wang et al. on the top 100 Alexa websites, removing the possibility of a bug or some irregularity in our own crawler. We conclude that this reduction in accuracy is an artifact of the size and type of the majority of `.onion` sites. The average size of a `.onion` site is substantially smaller than that of a standard web page; resulting in less information being leaked to a classification process, allowing for the increase in chance of misclassifications. We also found that a large number of `.onion` sites are log-in pages to various forums, that are based on standard designs and so bear a resemblance to one another. The small size and design of `.onion` sites provide a natural defense against WF. By restricting the amount of information available to a classification process, and conforming to standard website designs, despite the small world size of `.onion` sites we conclude that successful website fingerprinting attacks are considerably more difficult than on standard websites.

HTTP/2. HTTP/2 is the upcoming new version of the HTTP protocol and is already supported by some of the domains that receive most traffic volume in the Web [1]. HTTP/1.1 tried to provide parallelism of HTTP messages with HTTP pipelining. However, the deployment of HTTP pipelining has not been ideal, as many intermediaries (e.g., CDNs) do not implement it correctly [1].

HTTP/2 supports parallelism of HTTP conversations natively and overcomes one of the main limitations of HTTP/1.1. From our experiments with request randomization performed with LLaMA, our intuition is that randomization of HTTP/2 will not provide better results than RP. HTTP/2 also allows to add padding in HTTP messages to mitigate cryptographic attacks [10]. We devise the use of HTTP/2 padding as a primitive for application-layer WF defenses.

Bibliography

[1] HTTP/2 specs. "`https://http2.github.io/`", 2015. (accessed: August, 2016).

[2] X. Cai, R. Nithyanand, and R. Johnson. CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense. In *Workshop on Privacy in the Electronic Society (WPES)*, pages 121–130. ACM, 2014.

[3] X. Cai, R. Nithyanand, and R. Johnson. Glove: A Bespoke Website Fingerprinting Defense. In *Workshop on Privacy in the Electronic Society (WPES)*, pages 131–134. ACM, 2014.

[4] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg. A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses. In *ACM Conference on Computer and Communications Security (CCS)*, pages 227–238. ACM, 2014.

[5] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson. Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *ACM Conference on Computer and Communications Security (CCS)*, pages 605–616. ACM, 2012.

[6] S. Chen, R. Wang, X. Wang, and K. Zhang. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *IEEE Symposium on Security and Privacy (S&P)*, pages 191–206. IEEE, 2010.

[7] H. Cheng and R. Avnur. Traffic Analysis of SSL Encrypted Web Browsing. *Project paper, University of Berkeley*, 1998. Available at `http://www.cs.berkeley.edu/~daw/teaching/cs261-f98/projects/final-reports/ronathan-heyning.ps`.

[8] R. Dingledine, N. Mathewson, and P. F. Syverson. "Tor: The Second-Generation Onion Router". In *USENIX Security Symposium*, pages 303–320. USENIX Association, 2004.

[9] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton. Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In *IEEE Symposium on Security and Privacy (S&P)*, pages 332–346. IEEE, 2012.

[10] Y. Gluck, N. Harris, and A. Prado. Breach: reviving the crime attack. *Unpublished manuscript*, 2013.

[11] J. Hayes and G. Danezis. k-fingerprinting: a Robust Scalable Website Fingerprinting Technique. In *USENIX Security Symposium*. USENIX Association, 2016.

[12] D. Herrmann, R. Wendolsky, and H. Federrath. Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier. In *ACM Workshop on Cloud Computing Security*, pages 31–42. ACM, 2009.

[13] A. Hintz. Fingerprinting Websites Using Traffic Analysis. In *Privacy Enhancing Technologies (PETs)*, pages 171–178. Springer, 2003.

[14] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt. A critical evaluation of website fingerprinting attacks. In *ACM Conference on Computer and Communications Security (CCS)*, pages 263–274. ACM, 2014.

[15] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright. Toward an Efficient Website Fingerprinting Defense. In *European Symposium on Research in Computer Security (ESORICS)*, pages 27–46. Springer, 2016.

[16] A. Kwon, M. AlSabah, D. Lazar, M. Dacier, and S. Devadas. Circuit fingerprinting attacks: passive deanonymization of tor hidden services. In *USENIX Security Symposium*, pages 287–302. USENIX Association, 2015.

[17] M. Liberatore and B. N. Levine. "Inferring the source of encrypted HTTP connections". In *ACM Conference on Computer and Communications Security (CCS)*, pages 255–263. ACM, 2006.

[18] L. Lu, E. Chang, and M. Chan. Website Fingerprinting and Identification Using Ordered Feature Sequences. In *European Symposium on Research in Computer Security (ESORICS)*, pages 199–214. Springer, 2010.

[19] X. Luo, P. Zhou, E. W. W. Chan, W. Lee, R. K. C. Chang, and R. Perdisci. HTTPOS: Sealing Information Leaks with Browser-side Obfuscation of Encrypted Flows. In *Network & Distributed System Security Symposium (NDSS)*. IEEE Computer Society, 2011.

[20] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel. Website fingerprinting at internet scale. In *Network & Distributed System Security Symposium (NDSS)*. IEEE Computer Society, 2016.

[21] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website fingerprinting in onion routing based anonymization networks. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 103–114. ACM, 2011.

[22] M. Perry. Committed to the official Tor Browser git repository, `https://gitweb.torproject.org/tor-browser.git/commit/?id=354b3b`.

[23] M. Perry. Experimental Defense for Website Traffic Fingerprinting. Tor project Blog. "`https://blog.torproject.org/blog/experimental-defense-website-traffic-fingerprinting`", 2011. (accessed: October 10, 2013).

[24] M. Perry, G. Acar, and M. Juarez. personal communication.

[25] A. Pinto. Web Page Sizes: A (Not So) Brief History of Page Size through 2015. yottaa.com. "`http://www.yottaa.com/company/blog/application-optimization/a-brief-history-of-web-page-size/`", 2015. (accessed: April 18, 2016).

[26] T. Pulls. A golang implementation of the kNN website fingerprinting attack. "`https://github.com/pylls/go-knn`", 2016. (accessed: May, 2016).

[27] SecureDrop. securedrop.org. "`https://securedrop.org/`", 2016. (accessed: April 20, 2016).

[28] Q. Sun, D. R. Simon, and Y. M. Wang. Statistical Identification of Encrypted Web Browsing Traffic. In *IEEE Symposium on Security and Privacy (S&P)*, pages 19–30. IEEE, 2002.

[29] MobiForge. mobiforge.com. "`https://mobiforge.com/research-analysis/the-web-is-doom`", 2016. (accessed: April 20, 2016).

[30] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg. Effective Attacks and Provable Defenses for Website Fingerprinting. In *USENIX Security Symposium*, pages 143–157. USENIX Association, 2014.

[31] T. Wang and I. Goldberg. Improved Website Fingerprinting on Tor. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 201–212. ACM, 2013.

[32] C. V. Wright, S. E. Coull, and F. Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *Network & Distributed System Security Symposium (NDSS)*. IEEE Computer Society, 2009.

Appendices: Onion service target experiments

In addition to morphing a page via P-ALPaCA and D-ALPaCA, we evaluate the efficacy of our server-side defense on a number of `.onion` sites via morphing to a target `.onion` site. We dispense with applying a new morphing process for each capture of an `.onion` site load. Instead, we morph the `.onion` site once and capture 40 instances of this morphed site. We show that even if a server morphs their network traffic once, if it is morphed towards a targeted `.onion` site, this is enough to thwart WF attacks.

A. SecureDrop

To protect its users, SecureDrop may want to morph the network traffic pattern of its page load to look like that of an `.onion` site which would not raise suspicion on a visit. We collected 40 instances of network traffic when visiting SecureDrop; we then chose 40 target `.onion` sites which our server-side defense would morph SecureDrop's traffic to look like.

We considered a powerful adversary, who knows all sites that the defense would like to morph traffic to look like. For each target site, the adversary could train on all the undefended SecureDrop network traffic and the network traffic of the target `.onion` site, and they must classify an unknown traffic instance as either SecureDrop or the target `.onion` site. In our experiment, all new traffic instances were the morphed SecureDrop page; under a perfect defense all should have been classified as the target site

Using k-FP with 1,000 trees [11], the average binary classification accuracy over the 40 different `.onion` sites was 0.372 ± 0.416. Overall, our server-side defense was successful in obscuring which site a client was visiting, though we saw a large variation: some onion sites perfectly concealed the true label while others failed.

The average communication cost (incoming and outgoing size of packets) of the SecureDrop page was 15 KB, and it loaded on average in 4.62 seconds. The average communication cost of the morphed page was 373 KB and it loaded in 6.70 seconds. The size of the morphed page entirely depends on the target page we chose to morph the SecureDrop page towards, if a smaller target page had been chosen this would result in a smaller bandwidth overhead. However, the average bandwidth overhead is still smaller than that of a standard website.

B. Facebook

To generalize our defense beyond SecureDrop we chose 100 `.onion` sites that may also wish to protect visiting clients from WF attacks, by morphing their traffic to that of the Facebook `.onion` site[14]. We collected 40 traffic instances for each `.onion` site. All WF attacks were applied in the same manner as in Section 7.6.1.

Binary classification: the average binary classification accuracy over the 100 `.onion` sites was 0.098 ± 0.253. Even when the adversary knows undefended and target site, the attack's accuracy is below 10%.

Closed World classification: we also compared a closed world attack on the 100 undefended `.onion` sites and the same attack after morphing those sites to look like Facebook `.onion` site. If our server side defense is successful the 100 morphed `.onion` sites should, at the network level, look like the Facebook `.onion` site, resulting in a low classification accuracy.

[14] `https://facebookcorewwwi.onion`

Table 8, shows as expected, attack accuracy decreases when onion sites are morphed to resemble Facebook's network traffic patterns.

Table 7: Facebook experiment latency and bandwidth overheads.

	Latency		Volume	
	%	Avg. (s)	%	Avg. (KB)
Undefended	–	3.99	–	175
Defended	27.3	5.08	80	315

Table 8: Closed world classification for `.onion` sites morphed to Facebook's `.onion` site.

	k-NN (%)	k-FP (%)	CUMUL (%)
Undefended	45.6	69.6	55.6
Defended	9.4	55.6	53.6

Open World classification: in addition to closed world experiments, we evaluated the server-side defense in the open world setting, where we included instances of `.onion` sites that were not of interest to the attacker. We used 5,259 unique `.onion` sites, from Section 7.5.2, as background traffic instances. Table 9 shows, as expected, attack's accuracy decreases when sites are morphed to resemble Facebook's network traffic patterns.

Table 9: Open world classification for `.onion` sites morphed to Facebook's `.onion` site.

	k-NN (%)		k-FP (%)		CUMUL-k-FP (%)	
	TPR	FPR	TPR	FPR	TPR	FPR
Undefended	30.8	2.6	59.3	5.2	53.2	5.7
Defended	7.8	0.9	44.9	1.8	44.4	2.0

Table 7 shows the average time to load a page only increases by 1.09s when morphing a page to the Facebook `.onion` site. We also see that the bandwidth overhead is, compared to previous works, quite tolerable. The total cost of communication rises by only 140KB.

C. KDE distributions

We used Kernel Density Estimation (KDE) to estimate the distributions of number of objects (Figure 7), size of html pages (Figure 8) and size of objects (Figure 9). KDE is a non-parametric method for estimating a probability distribution given a data sample, which provides smoother estimates than histograms. KDE requires to specify a kernel (Gaussian, in our case) and a bandwidth. The bandwidth impacts on the smoothness of the estimate: a larger bandwidth tends to provide better smoothness, but less fidelity to the original data. To determine the bandwidth for each of

our distributions, we first performed Grid Search Cross Validation using `scikit-learn` library[15], to obtain a rough idea of the bandwidth ranges. Then, we manually trimmed the bandwidth to achieve what visually seemed to reflect well the variance of data, but also provided smooth distributions. For our purposes, it was important to have smooth estimates to guarantee a good quality in sampling (e.g., to avoid spikes). We used a bandwidth of 2 for the distribution over objects, and of 2000 for both the HTML and object sizes distributions.

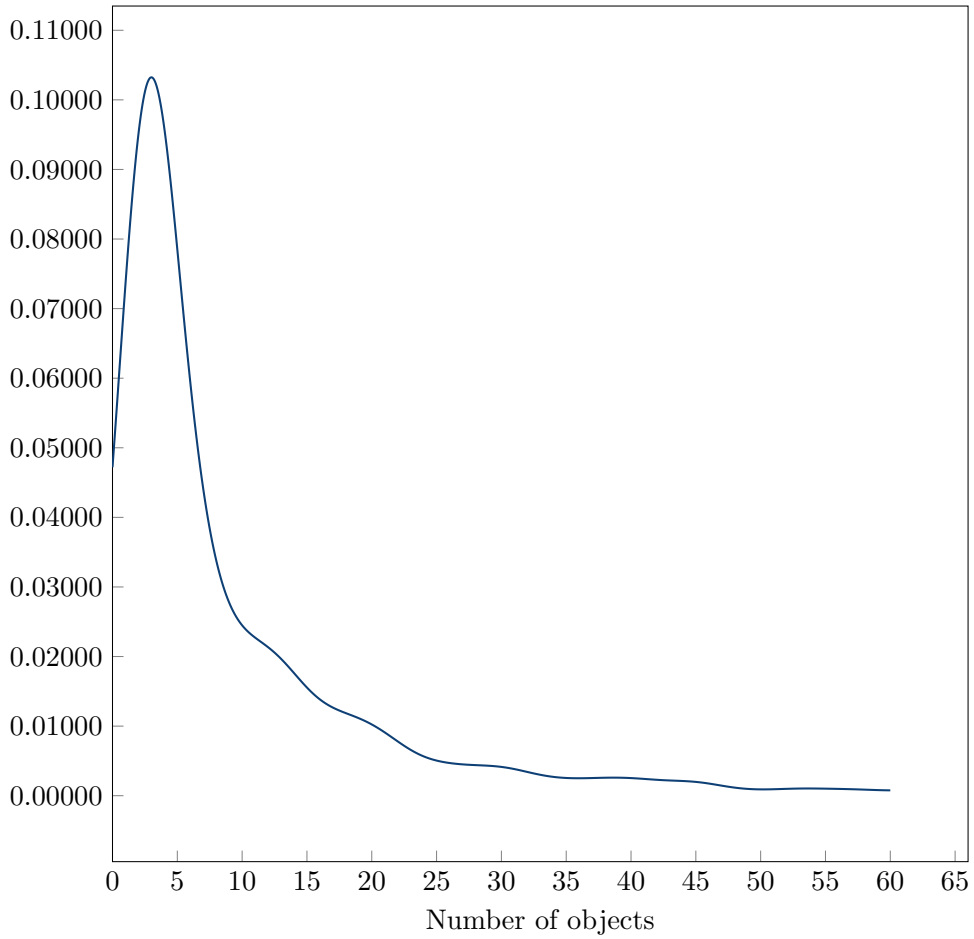


Figure 7: KDE distribution of the number of objects

[15] `http://scikit-learn.org/`

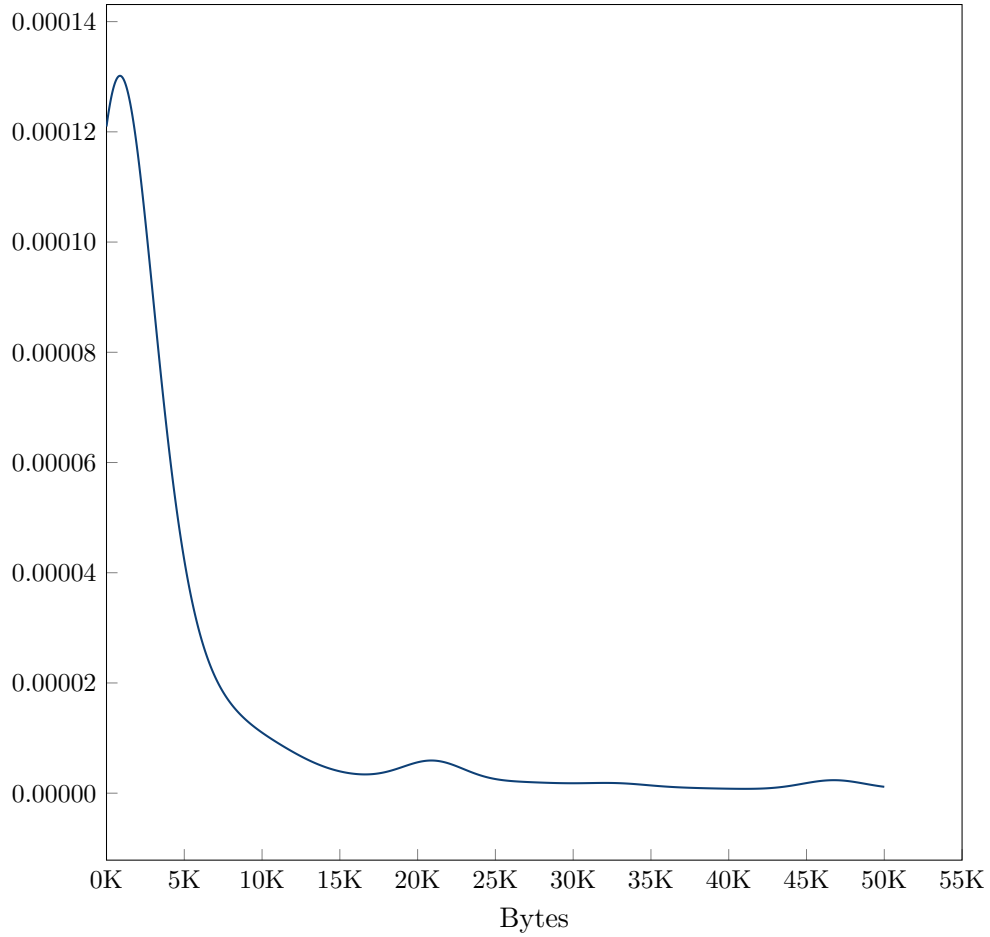


Figure 8: KDE distribution of the HTML sizes

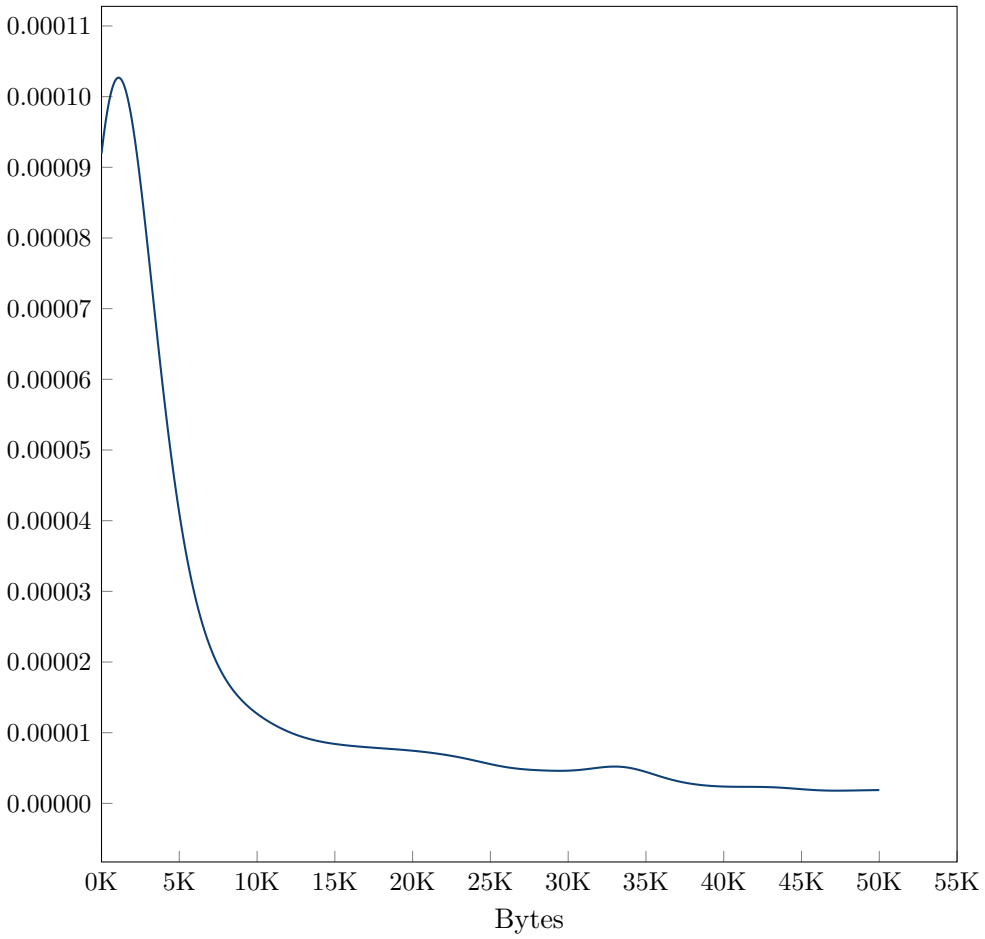


Figure 9: KDE distribution of the object sizes

Part IV

Optimal & Adaptive Mixnet Designs

8. Privacy Trilemma: Anonymity, Overhead, Latency

8.1 Introduction

Millions of users from all over the world employ anonymous communication networks, such as Tor [48], to protect their privacy over the Internet. The design choice made by the Tor network to keep the latency and bandwidth overheads small has made it highly attractive to its geographically diverse user-base. However, over the last decade, the academic literature [4, 23, 26, 28, 37, 38, 46] has demonstrated Tor's vulnerability to a variety of traffic correlation attacks. In fact, Tor also has been successfully attacked in practice [47].

It is widely accepted that low-latency low-bandwidth overhead of anonymous communication (AC) protocols, such as Tor [16], can only provide a weak form of anonymity [6]. In the anonymity literature, several AC protocols were able to overcome this security barrier to provide a stronger anonymity guarantee (cryptographic indistinguishability based anonymity [21, 25]) by either increasing the latency overhead or the bandwidth overhead. In particular, high-latency approaches (such as threshold mix networks [44]) can ensure strong anonymity by introducing significant communication delays for users messages, while high-bandwidth approaches (such as Dining Cryptographers network [7] and its extensions [10, 24, 43]) can provide strong anonymity by adding copious noise (or dummy) messages.

There have been a few efforts to propose hybrid approaches [11, 32, 34, 41, 49, 51] that try to provide anonymity by simultaneously introducing latency and bandwidth overhead. However, it is not clear how to balance such system parameters to ensure strong anonymity while preserving practical performance.

In general, in the last 35 years a significant amount of research efforts have been put towards constructing novel AC protocols, deploying them, and attacking real-world AC networks. However, unlike other security fields such as cryptography, our understanding regarding the fundamental limits and requirements of AC protocols remains limited. This work takes some important steps towards answering fundamental question associated with anonymous communication. "Can we prove that strong anonymity cannot be achieved without introducing large latency or bandwidth overhead? When we wish to introduce the latency and bandwidth overheads simultaneously, do we know the overhead range values that still fall short at providing stronger anonymity?"

8.1.1 Contribution to the PANORAMIX project

We confirm a previously conjectured [33, 41] relationship between bandwidth overhead, latency overhead and anonymity. We find that there are fundamental bounds on sender and recipient anonymity properties [2, 21, 25, 35] of a protocol that directly depend on the introduced bandwidth

and latency overheads.

This work presents a generic model of AC protocols using petri nets [27, 42] such that different instantiations of this model will represent different AC protocols, covering most practical AC systems in the literature. We derive *upper* bounds on anonymity as functions of bandwidth overhead and latency overhead, against two prominent adversary classes: global passive network-level adversaries and strictly stronger adversaries that additionally (passively) compromise some protocol parties (e.g., relays in case of Tor). These bounds constitute necessary constraints for anonymity. Naturally, the constraints are valid against any stronger adversary class as well.

For both adversary classes, we analyze two different user distributions (i.e., distributions that determine at which time or rate users of the AC protocol send messages): (i) synchronized user distributions, where users globally synchronize their messages, and (ii) unsynchronized user distributions, where each user locally decides when to send his messages independent of other users.

We analyze the trade-off between latency overhead and bandwidth overhead required to achieve *strong anonymity*, i.e., anonymity up to a negligible (in a security parameter η) chance of failure. For any AC protocol where only a fraction of $\beta \in [0, 1]$ users send noise messages per communication round, and where messages can only remain in the network for $\ell \geq 0$ communication rounds, we find that against a global network-level adversary no protocol can achieve strong anonymity if $2\beta\ell < 1 - 1/\text{poly}(\eta)$ even when all the protocol parties are honest. In the case where a strictly stronger adversary additionally passively compromises c (out of K) protocol parties, we show that strong anonymity is impossible if $2(\ell - \mathsf{c})\beta < 1 - 1/\text{poly}(\eta)$ (for $\mathsf{c} < \ell$), or $2\beta\ell < 1 - 1/\text{poly}(\eta)$ and $\ell \in \mathcal{O}(1)$ (for $\mathsf{c} \geq \ell$).

We also assess the practical impact of our results by analyzing prominent AC protocols. Our impossibility results naturally only offer necessary constraints for anonymity, but *not* sufficient conditions for the AC protocol. However, these necessary constraints for sender and recipient anonymity are crucial for understanding bi-directional anonymous communication. In fact, we find that several AC protocols in the literature are asymptotically close to the suggested constraints. Moreover, designers of new AC protocols can use our necessary constraints as guidelines for avoiding bad trade-off between latency and bandwidth-overhead.

8.2 Overview

8.2.1 Formalization and Adversary Model

AC Protocols as Petri Nets We define a view of AC protocols as petri nets [27, 31, 42], i.e., as graphs with two types of labeled nodes: *places*, that store colored tokens, and *transitions*, that define how these tokens are sent over the graph. In our case, each colored token represents a message, places are the protocol parties that can receive, hold and send messages, and transitions describe how parties exchange and relay messages. Our model captures all AC protocols under the assumption that messages are transmitted directly, i.e., in order for Bob to receive a message from Alice, Alice has to send the message and the message (albeit relayed, delayed and cryptographically modified) eventually has to reach Bob. While this requirement may sound strict, as elaborated in Section 8.4.2, we effectively only exclude few esoteric protocols.

User Distributions, Communication Rounds, Bandwidth Overhead, and Latency We consider two types of *user distributions*. In the first user distribution (*synchronized*) N users send their messages in exactly N rounds (see Figure 8.1 for notations). Per round, exactly one user sends a message. The protocol decides which users send noise messages in each round. In the second

ℓ	Latency overhead for every message
β	Bandwidth overhead for every user per round
p	Probability to send a message per user per round
K	Number of (internal) protocol parties
c	Number of compromised protocol parties
N	Number of online users (that may send messages)
δ	Adversarial advantage in the anonymity game
Π	A protocol. $\Pi \in M$: Π is within our model
η	The security parameter
ϵ	A (very small, but non-negligible) function

Figure 8.1: Notation

user distribution (*unsynchronized*) each user independently decides whether to send a message in a round using a coin flip, with a success probability p.

The model considers synchronous communication *rounds* as in [10,22,43,45]. We model latency overhead ℓ as the number of rounds a message can be delayed by the protocol before being delivered. We formalize bandwidth overhead β as the number of noise messages per user that the protocol can create in every round, i.e., the dummy message rate.

Our two types of user distributions cover a large array of possible scenarios. Results for our user distributions imply results for similar distributions, if a reduction proof can show that they are less favorable to the protocol.[1]

Adversaries We consider global passive *non-compromising* adversaries, that can observe all communication between protocol parties; and strictly stronger *partially compromising* (passive) adversaries, that can compromise protocol parties to learn the mapping between inputs and outputs for this party.

Anonymity Property We leverage an indistinguishability based anonymity notion for sender anonymity: the adversary has to distinguish two senders of its own choosing [21,25].

For a security parameter η, we say that a protocol achieves *strong anonymity*, if the adversary's advantage remains negligible in η. Strong anonymity is relative to a strength η, which is bound to system parameters or analysis parameters such as the number of users or protocol parties, the latency overhead and the bandwidth overhead. These parameters typically increase as η increases, which improves the protocol's anonymity.[2] Anonymity in relation to η unifies a wide variety of possible analyses on how the anonymity bound changes with changing system parameters, and user numbers and behaviors.

8.2.2 Brief Overview of the Proof Technique

As *non-compromising* adversaries are a subset of *partially compromising* adversaries, our proof technique for the former is a simplified case of the latter. In general, we derive our results in four

[1] Such distributions might contain usage patterns, irregularities between users and synchronization failures that the adversary can exploit.

[2] In some analyses, individual parameters may reduce with increasing η, such as the bandwidth overhead per user, as the other parameters, such as the number of users, increase.

main steps.

First, we define a concrete adversary $\mathcal{A}_{paths}$, that uses a well established strategy: upon recognizing the challenge message (as soon as it reaches a receiver) $\mathcal{A}_{paths}$ constructs the possible paths this message could have taken through the network, and tries to identify the user who has sent the message.

Second, given the concrete adversary $\mathcal{A}_{paths}$, we identify a necessary invariant that any protocol has to fulfill in order to provide anonymity. Intuitively: *both challenge users chosen by the adversary must be active (i.e., send at least one message) before the challenge message reaches the recipient, and it must be possible for these messages to meet in at least one honest party along the way.* We prove that indeed this natural invariant is necessary for anonymity.

Next, we propose an ideal protocol Π_{ideal} that is optimal in terms of satisfying the invariant: The probability that Π_{ideal} fulfills the necessary invariant is at least as high as for any protocol within our model (limited by the same constraints for β and ℓ). Moreover, whenever Π_{ideal} satisfies the invariant, the advantage of $\mathcal{A}_{paths}$ is zero. Thus, Π_{ideal} is at least as good as any protocol within our model at winning against $\mathcal{A}_{paths}$.

Finally, we calculate the advantage of $\mathcal{A}_{paths}$ against Π_{ideal} to obtain a lower bound on the adversarial advantage against all protocols within our model.[3]

8.2.3 Scenarios and Lower Bounds

We devise necessary constraints for four different scenarios. Let Π be a protocol in our model, with N users, restricted by bandwidth overhead $\beta \in [0,1]$ and latency overhead $\ell \geq 0$. For the *compromising* cases, the adversary can compromise c out of K protocol parties. We derive the following lower bounds for δ-sender anonymity in the respective scenarios.

Synchronized Users, Non-compromising Adversaries:

$\delta \geq 1 - f_\beta(\ell)$, where $f_\beta(x) = \min\left(1, \left(\frac{x+\beta \mathsf{N} x}{\mathsf{N}-1}\right)\right)$.

Synchronized Users, Partially Compromising Adversaries:

$$\delta \geq \begin{cases} 1 - [1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}] f_\beta(\ell) & \mathsf{c} \geq \ell \\ 1 - [1 - 1/\binom{\mathsf{K}}{\mathsf{c}}] f_\beta(\mathsf{c}) - f_\beta(\ell - \mathsf{c}) & \mathsf{c} < \ell. \end{cases}$$

Unsynchronized Users, Non-compromising Adversaries:

$\delta \geq 1 - [1/2 + f_p(\ell)]$, where for $p \approx \beta$ we have $f_p(x) = \min(1/2,\ 1 - (1-p)^x)$ for a positive integer x.

Unsynchronized Users, Partially Compromising Adv.:

$$\delta \geq \begin{cases} 1 - [1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}][1/2 + f_p(\ell)] & \mathsf{c} \geq \ell \\ \left(1 - [1 - 1/\binom{\mathsf{K}}{\mathsf{c}}][1/2 + f_p(\mathsf{c})]\right) & \\ \quad \cdot \left(1 - [1/2 + f_p(\ell - \mathsf{c})]\right) & \mathsf{c} < \ell. \end{cases}$$

[3] $\mathcal{A}_{paths}$ is a possible adversary against all protocols within our model. If $\mathcal{A}_{paths}$ has an advantage of δ against our ideal protocol Π_{ideal} (bounded by β and ℓ), then $\mathcal{A}_{paths}$ will also have an advantage of at least δ against any protocol within our model (that is also bounded by β and ℓ). Thus, our bound for δ describes a lower bound on the adversarial advantage against any protocol within the model, while against particular protocols there can be other adversaries (in the same adversary class) with an even higher advantage.

To keep the presentation concise, we focus on how to derive bounds for sender anonymity. As the bounds for recipient anonymity are obtained analogously, we only explain the adjustments in the proofs and the corresponding resulting bounds. The omitted canonical analysis can be found in [15].

8.2.4 Interpretation and Interesting Cases

Our first and third lower bounds, for respectively synchronized and unsynchronized user behaviors against in a non-compromised AC network, suggest an anonymity trilemma. Both lower bounds can be simplified under some natural constraints to the following simplified lemma:

Lemma 1 (Informal Trilemma)**.** *For security parameter η, no protocol can achieve strong anonymity if $2\ell\beta < 1 - \epsilon(\eta)$, where $\epsilon(\eta) = \frac{1}{\eta^d}$ for any positive constant d.*

Ideal asymptotic values for latency overhead is $\ell = O(1)$ (i.e., a constant number of hop separation from the receiver), while ideal asymptotic values for bandwidth overhead is $\beta = O(1/\mathsf{N}) = O(1/poly(\eta))$ (i.e., a constant number of message per round from all $\mathsf{N} = poly(\eta)$ users combined). It is easy to see that for this ideal overhead $\ell\beta = O(1/poly(\eta))$, the trilemma excludes strong anonymity, while, with latency overhead $\ell = \mathsf{N} = O(poly(\eta))$ or with bandwidth overhead $\beta = O(1)$, the trilemma does not exclude strong anonymity.

We find some interesting possible overhead constraints for strong anonymity (e.g. $\ell = O(\eta)$ and $\beta = O(1/\eta)$) demanding some compromise in both latency and bandwidth. These constraints can help understand and improve existing AC protocols as well as inform the design of future AC protocols.

For partially compromised scenarios the requirements are naturally stronger. All constraints discussed for compromised case in the following part are in addition to the requirements from the non-compromised case. While bandwidth overhead might be sufficient against non-compromising adversaries, it is not sufficient if parts of the protocol are compromised. With $\ell = \eta$ and $\frac{K}{c} = constant$ strong anonymity may be possible, whereas with $\ell = \mathcal{O}(1)$, strong anonymity is impossible, even for $\mathsf{K} \in poly(\eta)$ and $\mathsf{c} = \mathcal{O}(1)$.

In case $\mathsf{c} < \ell$, strong anonymity guarantees may be possible only if $2(\ell - \mathsf{c})p > 1 - \epsilon(\eta)$, where $p = p' + \beta$ combines the genuine user messages p' with their bandwidth overhead β. Our result shows a connection between the expected usage behavior p and the latency ℓ. If p is not particularly large, the latency cannot be low; otherwise, the path-length cannot be sufficiently high to ensure mixing at an honest node. In other words, unless p is very large (as should be the case for some file sharing applications), a low latency renders the AC protocol cheap to compromise, i.e., c can be low.

Our necessary constraints enable protocol designers of AC protocols to avoid bad trade-offs between latency and bandwidth overhead. For a given expected user behavior and a given target attacker against which the AC shall provide anonymity, our constraints clearly state which combinations of latency and bandwidth overhead to avoid.

8.2.5 Related Work

In contrast to previous work, our work provides necessary constraints for strong anonymity w.r.t. to bandwidth and latency overhead. While there is a successful line of work on provable anonymity guarantees [2,5,19,21,30,35,50], it is incomparable since it provides lower bounds on anonymity for specific protocols, and does not prove any general statements about sufficient conditions for strong anonymity.

Previous work on attacks against anonymous communication protocols, except for Oya et al. [39], solely provides upper bounds on anonymity for specific protocols [12,14,20,40]. Oya et al. [39] cast their attack in a general model and provide a sophisticated generic attacker. However, they only compute bounds w.r.t. a dummy message rate against timed pool mixes, not against other protocols and not w.r.t. latency and compromisation rate. Even more important, none of these results discuss the relationship of the lower bounds for latency and bandwidth overheads.

8.3 Anonymity Definition and User Distributions

8.3.1 AnoA-Style Anonymity Definition

We define our anonymity notions with a challenge-response game similar to AnoA [2, 35], where the challenger simulates the protocol and the adversary tries to deanonymize users. The challenger $\texttt{Ch}(\Pi, \alpha, b)$ allows the adversary to adaptively control user communication in the network, up to an uncertainty of one bit for challenges, and is parametric in the following parts: (i) the AC protocol Π to be analyzed, (ii) the so called *anonymity function* α, that describes the specific variant of anonymity such as sender anonymity, recipient anonymity and relationship anonymity, (iii) and the challenge bit b which determines the decision the challenger takes in challenge inputs from the adversary.

Given a security parameter η, we quantify the anonymity provided by the protocol Π simulated by $\texttt{Ch}(\Pi, \alpha, b)$ in terms of the advantage the probabilistic polynomial time (PPT) adversary $\mathcal{A}$ has in correctly guessing Ch's challenge bit b. We measure this advantage in terms of indistinguishability of random variables additively, where the random variables in question represent the output of the interactions $\langle \mathcal{A} | \texttt{Ch}(\Pi, \alpha, 0) \rangle$ and $\langle \mathcal{A} | \texttt{Ch}(\Pi, \alpha, 1) \rangle$.

Definition 1 ((α, δ)-IND-ANO)**.** *A protocol* Π *is* (α, δ)-IND-ANO [4] *for the security parameter* η*, an adversary class* C*, an anonymity function* α *and a distinguishing factor* $\delta(\cdot) \geq 0$*, if for all* ppt *machines* $\mathcal{A} \in \mathrm{C}$*,*
$\Pr[0 = \langle \mathcal{A} | \texttt{Ch}(\Pi, \alpha, 0) \rangle] \leq \Pr[0 = \langle \mathcal{A} | \texttt{Ch}(\Pi, \alpha, 1) \rangle] + \delta(\eta)$*.*

For an anonymity function α, we say that a protocol Π provides *strong anonymity* [21, 25] if it is $(\alpha, \delta) - \texttt{IND-ANO}$ with $\delta \leq neg(\eta)$ for some negligible function *neg*. If δ is instead *non-negligible* in η, then we say that Π provides *weak anonymity*. Note that η does not measure the size of the anonymity set, but the computational limitation of the adversary.

Sender Anonymity Sender anonymity characterizes the anonymity of users against a malicious server through the inability of the server (or some intermediary) to decide which of two *self-chosen* users have been communicating with the server. We borrow the sender anonymity α_{SA} definition from the AnoA framework [2], where α_{SA} selects one of two possible challenge users and makes sure that the users cannot be distinguished based on the chosen recipient(s) or message(s).

Definition 2 (Sender anonymity)**.** *A protocol* Π *provides* δ-sender anonymity *if it is* (α_{SA}, δ)-IND-ANO *for* α_{SA} *as defined in Figure 8.2.*

Recipient Anonymity Recipient anonymity characterizes that the recipient of a communication remains anonymous, even to observers that have knowledge about the sender in question. Similar

[4] AnoA also allows a multiplicative factor ε; we use the simplified version with $\varepsilon = 0$, such that δ directly corresponds to the adversarial advantage.

Adaptive AnoA Challenger $\texttt{Ch}(\Pi, \alpha, b)$

Upon message $(\texttt{Input}, u, R, m)$**:** *RunProtocol*$(u, R, m)$

Upon message $(\texttt{Challenge}, u_0, u_1, R_0, R_1, m)$**:**

if this is the first time, such a message is received **then**
Compute $(u^*, R^*) \leftarrow \alpha(u_0, u_1, R_0, R_1, b)$
RunProtocol$(u^*, R^*, m))$
end if

RunProtocol(u, R, m)**:**

Run Π on $r = (u, R, m)$ and forward all messages that are sent by Π to the adversary $\mathcal{A}$ and send all messages by the adversary to Π.

$\alpha_{SA}(u_0, u_1, R_0, R_1, b) = (u_b, R_0)$

$\alpha_{RA}(u_0, u_1, R_0, R_1, b) = (u_0, R_b)$

Figure 8.2: Adaptive AnoA Challenger [2]

to sender anonymity, we borrow the recipient anonymity α_{RA} definition from the AnoA framework, where α_{RA} selects one of two possible recipients for a message and makes sure that the recipients cannot be distinguished based on the chosen sender(s) or message(s).

Definition 3 (Recipient anonymity)**.** *A protocol* Π *provides* δ-recipient anonymity *if it is* (α_{RA}, δ)-$\texttt{IND-ANO}$ *for* α_{RA} *as defined in Figure 8.2.*

We omit the detailed technical notation of the anonymity functions in the following sections, and write $\Pr[0 = \mathcal{A}|b = i]$ instead of $\Pr[0 = \langle \mathcal{A}|\texttt{Ch}(\Pi, \alpha_{SA}, i)\rangle]$.

8.3.2 Game Setup

Let $\mathcal{S}$ be the set of all senders, $\mathcal{R}$ be the set of all recipients, and P be the set of protocol parties that participate in the execution of the protocol (like relays/mix-nodes in Tor/mix-nets, for DC-net or P2P mixing users and protocol parties are the same). We consider a system of total $|\mathcal{S}| = \mathsf{N}$ senders. Given our focus on *sender anonymity*, we need only a single element in $\mathcal{R}$. We allow the adversary to set the same entity (say R) as the recipient of all messages, and expect R to be compromised by the adversary. The adversary uses a challenge (as defined in Figure 8.2) of the form $(u_0, u_1, R, \_, m_0)$, where $u_0, u_1 \in \mathcal{S}$, for our sender anonymity game.

We consider a completely connected topology, which means any party can send a message directly to any other party. We assume a standard (bounded) synchronous communication model as in [10, 22, 43, 45], where a protocol operates in a sequence of communication rounds.[5] In each round, a party performs some local computation, sends messages (if any) to other party through an authenticated link. By the end of the round, every party receives all messages sent by the other

[5]While a time-sensitive model [3] would be more accurate, e.g., for low-latency protocols like Tor [17], such a model would only strengthen the attacker. As we present necessary constraints, our results also hold for the more accurate setting.

parties to her the same round. With our focus on computing lower bounds, our model abstracts from the time the computations at the node take and also the length of the messages. Nevertheless, as we are interested in quantifying the communication/bandwidth overhead, unlike [10,22,43], we do not assume that the parties have access to ready-made broadcast communication channels; Parties are expected to communicate with each other to implement broadcast features [18, 45]. Lastly, the use of the asynchronous communication model offers more capabilities to the attacker, and thus, our impossibility results for the synchronous model naturally apply to the asynchronous model as well.

We define the latency overhead ℓ as the number of rounds a message can be delayed by the protocol before being delivered. We define the bandwidth overhead β as the number of noise messages per user that the protocol can create in every round (i.e., the dummy message rate) and we do not restrict the time these noise messages reside within the protocol.

We consider two types of *global passive* adversaries: Our *non-compromising* adversaries (which model network-level eavesdroppers) can observe all communication between all protocol parties, but do not compromise any party of the AC protocol except the recipient R. We say that the AC protocol is *non-compromised*. Our strictly stronger *partially compromising* adversaries (which model hacking and infiltration capabilities) can additionally compromise some of the AC parties in the setup phase of the game to obtain these parties' mapping between the input messages and output messages during the protocol's runtime. We say that the AC protocol is *partially compromised*.

8.3.3 User Distributions

We consider two kinds of user distributions in our anonymity games and both of them assume an N sized set $\mathcal{S}$ of users that want to send messages. In both cases, the adversary can choose any two senders $u_0, u_1 \in \mathcal{S}$. However, the time and method by which they actually send messages differs:

- In the *synchronized* user distribution the users globally synchronize who should send a message at which point in time. We assume that each user wants to send exactly one message. Consequently, we choose a random permutation of the set of users $\mathcal{S}$ and the users send messages in their respective round. In every single round out of a total of N rounds exactly one user sends a message. Since the users globally synchronize their sending of messages, we allow the protocol to also globally decide on the bandwidth overhead it introduces. Note that here the requirements are identical to those of the Bulk protocol in [10].
- In the *unsynchronized* user distribution each of the N users wants to send messages eventually and we assume that each user locally flips a (biased) coin every round to decide whether or not to send a message. In this case we define the bandwidth overhead as an increased chance of users sending messages. Since the protocol does not globally synchronize the input messages, for noise messages also we allow the users to decide it locally and send noise messages with a certain probability.

8.4 A Protocol Model for AC Protocols

An AC protocol allows any user in the set of users $\mathcal{S}$ to send messages to any user in $\mathcal{R}$, via a set of anonymizing parties P. We define protocols that are under observation of an eavesdropping adversary $\mathcal{A}$ that may have compromised a set of c parties $\mathsf{P_c} \subseteq \mathsf{P}$ and that furthermore observes the communication links between any two parties, including users.

Technically, whenever a party $P_1 \in \mathsf{P} \cup \mathcal{S}$ sends a message to another party $P_2 \in \mathsf{P} \cup \mathcal{R}$, the adversary is able to observe this fact together with the current round number. However, we assume the protocol applies sufficient cryptography, s.t., the adversary can not read the content of any

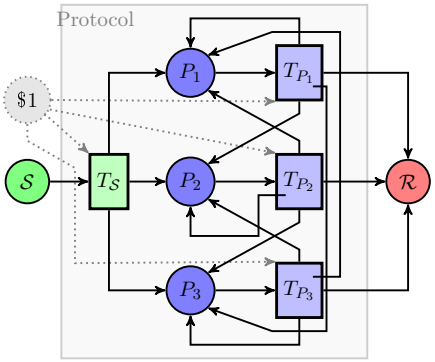


Figure 8.3: Petri net of an AC protocol with $\mathsf{K} = 3$ parties.

message except the messages sent to the malicious recipient, which technically results in simply being able to additionally recognize when the challenge reaches the recipient.

For an actual protocol, the sets $\mathcal{S}$, $\mathcal{R}$, and P might not be mutually exclusive [7, 24, 43]. Since we have only one malicious party in $\mathcal{R}$, and the content of a message can only be read when it reaches its final recipient, we consider $\mathcal{R}$ to be mutually exclusive from $\mathcal{S} \cup \mathsf{P}$ for the purpose of simplicity.

With the above preliminaries in mind, we shall now formally define our generic AC protocol using a petri net model.

8.4.1 Protocol Model

We model any AC protocol with K parties by a timed colored petri net [27, 31, 42] M, consisting of places $\mathcal{S}$ for the users, $P_1, \ldots, P_{\mathsf{K}}$ symbolizing the protocol parties, \$1 for randomness and R for recipients of messages, and colored tokens m symbolizing the messages (real or noise) sent by clients or protocol parties, and transitions $T_{\mathcal{S}}$ for inserting messages into the network and $T_{P_1}, \ldots, T_{P_{\mathsf{K}}}$ as functions for sending the messages from one party to another. The structure of the petri net with its places, tokens and transitions remains the same for every AC protocol. However, the implementation of the guards within the transitions is different for different protocols: protocols can choose to which party messages are to be sent next and whether they should be delayed. But, protocols in M are oblivious to the challenge message or the challenge users. We refer to Figure 8.3 for a graphical depiction of petri net model M.

Definition 4 (Colored token)**.** *A* colored token *is represented by the tuple* $m = \langle \mathsf{msg}, \mathsf{meta}, \mathsf{t_r}, \mathsf{ID_t}, \mathsf{prev}, \mathsf{next}, \mathsf{ts} \rangle$*, where,* msg *is the content of the message,* meta *is the internal protocol meta-data for this message,* $\mathsf{t_r}$ *is the time the message can remain in the network,* $\mathsf{ID_t}$ *is a new unique ID generated by each transition for each token by honest parties; dishonest parties instead keep* $\mathsf{ID_t}$ *untouched to allow the adversary to link incoming and outgoing messages,* prev *is party/user that sent the token and* next *is the user/party that receives the token. Finally,* ts *is the time remaining for the token to be eligible for a firing event (a feature of timed petri-net). Here,* ts *either describes when new messages are introduced into the petri net or is set to the next round, such that messages can be processed in every round as soon as they enter the network.*

The four fields $\mathsf{ID_t}, \mathsf{prev}, \mathsf{next}, \mathsf{ts}$ are public, and are visible to the adversary. The remaining three fields msg, meta and $\mathsf{t_r}$ in a token are private and can not be observed by the adversary, with the exception that msg can be observed when a message reaches its destination, i.e, is received by

$\mathbf{T}_{\mathcal{S}}$ on tokens $q = \langle \mathsf{msg}, \_, \_, \_, u, \_, \mathsf{ts} \rangle$ **from** $\mathcal{S}$ **and** \$ **from** \$1**:**

$(P_i, \mathsf{meta}) = f_\Pi(q, \$)$; $\mathsf{ID_t}$ = a fresh randomly generated ID
r = current round; $t = \langle \mathsf{msg}, \mathsf{meta}, \ell, \mathsf{ID_t}, u, P_i, 1 \rangle$
if $P_i = R$ **then** $\mathsf{Tokens} = \mathsf{Tokens} \cup (\langle \mathsf{msg}, \_, \_, \mathsf{ID_t}, u, P_i, 1 \rangle, r)$
else $\mathsf{Tokens} = \mathsf{Tokens} \cup (\langle \_, \_, \_, \mathsf{ID_t}, u, P_i, 1 \rangle, r)$
Output: token t at P_i

$\mathbf{T}_{\mathbf{P_i}}$ on tokens $q = \langle \mathsf{msg}, \_, \mathsf{t_r}, \mathsf{ID_t}, \_, P_i, \mathsf{ts} \rangle$ **from** P_i**,** \$ **from** \$1**:**

$(P', \mathsf{meta}') = f_\Pi(q, \$)$; r = current round
if $\mathsf{t_r} - 1 = 0$ **then** $P' = R$
if P_i is honest **then** $\mathsf{ID_t}' =$ a fresh randomly generated ID
else if P_i is compromised **then** $\mathsf{ID_t}' = \mathsf{ID_t}$
$t = \langle \mathsf{msg}, \mathsf{meta}', \mathsf{t_r} - 1, \mathsf{ID_t}', P_i, P', 1 \rangle$
if $P_i = R$ **then** $\mathsf{Tokens} = \mathsf{Tokens} \cup (\langle \mathsf{msg}, \_, \_, \mathsf{ID_t}', P_i, P', 1 \rangle, r)$
else $\mathsf{Tokens} = \mathsf{Tokens} \cup (\langle \_, \_, \_, \mathsf{ID_t}', P_i, P', 1 \rangle, r)$
Output: token t at P'

f_Π: A function provided by Π to determine routing and the meta field.

Figure 8.4: Transitions in petri net model M

a recipient. Formally, we introduce a set Tokens, that is initially empty and in which we collect the pair (t, r), where t is a copy of a token and r the round number in which the token was observed.

Places Any AC protocol with K parties $\mathsf{P} = \{P_1, \ldots, P_\mathsf{K}\}$ consists of the following places:

- $\mathcal{S}$: A token in $\mathcal{S}$ denotes a user message (real or noise) which is scheduled to enter the network after ts rounds.
- \$1: This place is responsible for providing randomness. Whenever a transition picks a token from this place, the transition basically picks a random value.
- P_i with $P_i \in \mathsf{P}$: A token in P_i denotes a message which is currently held by the party $P_i \in \mathsf{P}$.
- R: A token in R denotes a message which has already been delivered to a recipient.

Transitions As part of the *initial configuration*, the challenger populates $\mathcal{S}$ on behalf of the protocol. All other places are initially empty. The transitions then consumes tokens from one place and generate tokens to other places, to modify the *configuration* of the petri-net. The event of consumption of a token from one place by a transition and generation of a new token represents the movement of a message from one party to another. We define the following transitions (refer to Figure 8.4 for the pseudocodes of the transitions):

- $T_\mathcal{S}$: takes a token $\langle \mathsf{msg}, \_, \_, \_, u, \_, \mathsf{ts} \rangle$ from $\mathcal{S}$ and a token from \$1 to write $t = \langle \mathsf{msg}, \mathsf{meta}, \ell, \mathsf{ID_t}, u, P_i, \mathsf{ts} = 1 \rangle$ to P_i; the values of i and meta are decided by the AC protocol.
- T_{P_i}: takes a token $\langle \mathsf{msg}, \mathsf{meta}, \mathsf{t_r}, \mathsf{ID_t}, \_, P_i, \mathsf{ts} \rangle$ from P_i and a token from \$1 to write $t = \langle \mathsf{msg}, \mathsf{meta}', \mathsf{t_r} - 1, \mathsf{ID_t}', P_i, P', 1 \rangle$ to P'. If P_i is an honest party $\mathsf{ID_t}'$ is freshly generated, but if P_i is a compromised party $\mathsf{ID_t}' = \mathsf{ID_t}$. The place $P' \in \{P_1, \ldots, P_\mathsf{K}\} \cup \{R\}$ and meta′ are decided by the AC protocol, with the exception that if $\mathsf{t_r} = 0$, P' always is R.

In either case, the transition also adds an element (t', r) to the set Tokens, where r is the current round number and t' is a copy of the respective (new) token t, with the fields meta and $\mathsf{t_r}$ are removed. If the place where t was written to is not $\mathcal{R}$, then additionally the field msg is removed.

Game Setting Recall that we define anonymity as a game between a PPT adversary $\mathcal{A}$ and an honest challenger Ch.

Validity of the Protocol Model The above protocol model M behaves as expected (more details in Lemma 2 in Appendix 8.11). We show in Lemma 2 that the protocols indeed have a bandwidth overhead of β and a latency overhead of ℓ. For every message that is sent from one party in $\mathcal{S} \cup \mathsf{P}$ to another party in $\mathsf{P} \cup \mathcal{R}$, the adversary learns the time, the sender, and the receiver. When a message leaves the network, the attacker learns whether it was the target (i.e., the challenge) message. The attacker also learns the mapping between the input and output messages of compromised parties.

8.4.2 Expressing Protocols

Our protocol model M allows the expression of any AC protocol with very few, esoteric exceptions.

Mix networks can be naturally embedded into our model, in particular any stop-and-go mix [29] that uses discrete distribution and even AC protocols with specialized path selection algorithms [13, 36]. For the sake of our necessary constraints, low-latency protocols (with time-bounded channels) that are not round-based (e.g., Tor [17]) can be expressed in a round-based variant, since it only strengthens the protocols anonymity properties. This section illustrates embedding techniques into our model for some other kinds of protocols, but a much larger variety of protocols can be expressed in our model.

Users as protocol parties In peer-to-peer protocols like dining cryptographers networks (DC net) [24, 43], there are no separate protocol parties, users act as a type of relays. Also, any noise sent by users counts into the bandwidth overhead of the protocol (we will see in Claim 2 that noise sent by nodes that are not users can be treated differently). Whenever a user wants to send a message it should use the transition $T_{\mathcal{S}}$, but when it acts as a relay it should use the transition T_{P_i}. For interested readers, we show in Appendix 8.11 how to model a specific DC net type protocol using our petri net model.

Splitting and Recombining Messages We model protocols that split and later re-combine messages by declaring one of the parts as the main message and the other parts as noise, which may count into the bandwidth overhead. This declaration is mainly required for the analysis, i.e., for evaluating the success of the adversary and for quantifying the amount of noise messages introduced by the protocol. We do not restrict the strategy by which the protocol decides which message is "the main share" (i.e., the message that is sent on) and which is "an additional share" (i.e., a fresh noise message). A more complex scenario involves threshold schemes in which a smaller number of shares suffices for reconstructing the message and in which some shares are dropped randomly. In such cases we consider the protocol to decide beforehand which of the constructed shares will be dropped later and to declare one of the remaining shares the "main share".

Broadcasting Messages If the protocol chooses to copy or broadcast messages to several receivers, we consider the copy sent to the challenge receiver to be the main message and copies sent to other receivers to be noise (which, if the copies are created by nodes that are not users, will not count into the bandwidth overhead).[6]

[6] We note that in some cases, where users act as nodes and broadcast messages to other users, our quantification of the bandwidth overhead might be a bit harsh. If the group of users to which the broadcast will be sent is known

Private Information Retrieval In schemes based on private information retrieval we require that the receiver retrieves the information sufficiently fast (within the latency limit). Otherwise, our method is similar to the broadcasting of messages: the receiver of interest will retrieve the main message, whereas other receivers will retrieve copies that are modeled as noise.

Excluded Protocols For this work we exclude protocols that cannot guarantee the delivery of a message within the given latency bound (except if this occurs with a negligible probability). Moreover, we cannot easily express the exploitation of side channels to transfer information, e.g., sending information about one message in the meta-data of another message, or sending bits of information by not sending a message.

8.4.3 Construction of a Concrete Adversary

Given two challenge users u_0 and u_1 and the set of observed tokens $(t, r) \in \mathsf{Tokens}$, where t is the token and r the round in which the token was observed, an adversary can construct the sets S_j (for $j \in \{0, 1\}$). Assume the challenge message arrives at the receiver R in a round r. We construct possible paths of varying length k, s.t., each element $p \in S_j$ represents a possible path of the challenge message starting from u_j ($j \in \{0, 1\}$) and the challenge message then arrives at R in round $r_k = r$. With challenge bit b, S_b cannot be empty, as the actual path taken by the challenge message to reach R has to be one element in S_b.

$$
\begin{aligned}
S_j = \{p = (t_1.\mathsf{prev}, \ldots, t_k.\mathsf{prev}, t_k.\mathsf{next}) : \\
((t_1, r_1), \ldots, (t_k, r_k)) \in \mathsf{Tokens} \text{ s.t.} \\
t_1.\mathsf{prev} = u_j \wedge t_k.\mathsf{next} = R \\
\wedge\, t_k.\mathsf{msg} = \texttt{Challenge} \wedge k \leq \ell \\
\wedge\, \forall_{i \in \{1,\ldots,k-1\}} (t_i.\mathsf{next} = t_{i+1}.\mathsf{prev} \wedge r_{i+1} = r_i + 1 \\
\wedge\, (\ \exists t'_{i+1} : (t'_{i+1}, r_{i+1}) \in \mathsf{Tokens}\ \wedge t'_{i+1}.\mathsf{prev} = t_i.\mathsf{next} \\
\qquad \wedge\, t'_{i+1}.\mathsf{ID_t} = t_i.\mathsf{ID_t}) \Rightarrow t'_{i+1} = t_{i+1})\}
\end{aligned}
$$

Definition 5 (Adversary $\mathcal{A}_{paths}$)**.** *Given a set of users $\mathcal{S}$, a set of protocol parties P of size K, and a number of possibly compromised nodes c, the adversary $\mathcal{A}_{paths}$ proceeds as follows: 1. $\mathcal{A}_{paths}$ selects and compromises c different parties from P uniformly at random. 2. $\mathcal{A}_{paths}$ chooses two challenge users $u_0, u_1 \in \mathcal{S}$ uniformly at random. 3. $\mathcal{A}_{paths}$ makes observations and, based upon those, constructs the sets S_0 and S_1. For any $i \in \{0, 1\}$, if $S_i = \emptyset$, then $\mathcal{A}_{paths}$ returns $1 - i$. Otherwise, it returns 0 or 1 uniformly at random.*

$\mathcal{A}_{paths}$ thus checks whether both challenge users *could have* sent the challenge message. We explicitly ignore differences in probabilities of the challenge users having sent the challenge message, as those probabilities can be protocol specific. Naturally, when $\mathsf{c} = 0$, $\mathcal{A}_{paths}$ represents a *non-compromising* adversary; but when $\mathsf{c} \neq 0$, $\mathcal{A}_{paths}$ is *partially compromising.*

8.4.4 Protocol Invariants

We now investigate the robustness of protocols against our adversary. We define an invariant that, if not satisfied, allows $\mathcal{A}_{paths}$ to win against any protocol. Moreover, we present a protocol that maximizes the probability of fulfilling the invariant. Moreover, we show that whenever the invariant

in advance (i.e., if messages are broadcast to all users or to pre-existing groups of users), we can allow the protocol to use a single receiver for these messages instead.

is fulfilled by our protocol, the advantage of $\mathcal{A}_{paths}$ reduces to zero (as it is forced to randomly guess b).

Necessary invariant for protocol anonymity It is necessary that at least both challenge users send messages in one of the ℓ rounds before the challenge message reaches the recipient, as otherwise there is no way both of them could have sent the challenge message. Moreover, on the path of the actual challenge message, there needs to be at least one honest (uncompromised) party, as otherwise the adversary can track the challenge message from the sender to the recipient (S_b will have exactly one element and S_{1-b} will be empty). Those two conditions together form our *necessary protocol invariant.*

Invariant 1. *Let u_0 and u_1 be the challenge users; let b be the challenge bit; and let t_0 be the time when u_b sends the challenge message. Assume that the challenge message reaches the recipient at r. Assume furthermore that u_{1-b} sends her messages (including noise messages) at $V = \{t_1, t_2, t_3, \ldots, t_k\}$. Now, let $T = \{t : t \in V \wedge (r - \ell) \leq t < r\}$. Then,*

(i) the set T is not empty, and

(ii) the challenge message passes through at least one honest node at some time t' such that, $t' \in \{\min(T), \ldots, r - 1\}$.

Claim 1 (Invariant 1 is necessary for anonymity)**.** *Let Π be any protocol $\in M$ with latency overhead ℓ and bandwidth overhead β. Let u_0, u_1, b and T be defined as in Invariant 1. If Invariant 1 is not satisfied by Π, then our adversary $\mathcal{A}_{paths}$ as in Definition 5 wins.*

We refer to Appendix 8.12 for the proof. We next claim that it suffices to consider noise messages sent by users that also remain within the system for at most ℓ rounds, i.e., noise messages that follow the same rules as real messages. Note that we consider every new message originating from any user's client as a fresh noise message.

Claim 2 (Internal noise does not influence Invariant 1)**.** *Any message not originating from an end user $u \in \mathcal{S}$ does not influence the probability for Invariant 1 being true. Moreover, noise messages do not contribute to the probability for Invariant 1 being true after they stayed in the network for ℓ rounds.*

We refer to Appendix 8.12 for the proof. We henceforth consider noise messages as a protocol input.

8.4.5 Ideal Protocol

We construct a protocol Π_{ideal} that maximizes the probability of fulfilling Invariant 1. We show that the invariant is sufficient for Π_{ideal} to win against $\mathcal{A}_{paths}$, i.e., to reduce $\mathcal{A}_{paths}$'s advantage to 0. Claim 1 shows that for any protocol in our model $\mathcal{A}_{paths}$ wins whenever Invariant 1 does not hold. Thus, an upper bound on the probability that Π_{ideal} satisfies Invariant 1 yields an upper bound for all these protocols.

Given the set of all protocol parties $\mathsf{P} = \{P_0, \ldots, P_{\mathsf{K}-1}\}$ of size K, the strategy of Π_{ideal} is as follows: in a round r, Π_{ideal} delivers all messages scheduled for delivery to a recipient. All other messages (including the messages that enter Π_{ideal} in round r) are sent to the protocol party P_i with $i = r \mod \mathsf{K}$. For every message that enters the protocol, Π_{ideal} queries an oracle O for the number of rounds the message should remain in the protocol. We define the following events:

- $u.\text{sent}(x, y)$: user u has sent at least one message within rounds from x to y. For a single round we use $u.\text{sent}(x)$.
- $\text{Cmpr}(x)$: $\mathcal{A}_{paths}$ has compromised the next x consecutive parties on the path.
- $\neg H$: NOT of event H.

Given a message sent at t_0 by sender x, and delivered to the recipient at $(t_0 + t)$, we define P_t for sender $v \in \mathcal{S} \setminus \{x\}$:

$$\begin{aligned} P_t = &\sum_{j=r-\ell}^{t_0} \Pr\left[v.\text{sent}(j) \wedge \neg v.\text{sent}(j+1, t_0)\right] \cdot \Pr\left[\neg\text{Cmpr}(t)\right] \\ &+ \sum_{j=t_0+1}^{r} \Pr\left[v.\text{sent}(j) \wedge \neg v.\text{sent}(r-\ell, j-1)\right] \\ &\qquad \cdot \Pr\left[\neg\text{Cmpr}(r-j)\right] \end{aligned}$$

When $v = u_{1-b}$, and the message is the challenge message, P_t is the probability of fulfilling Invariant 1, for the strategy above. For each message, oracle O chooses an *optimal* t that maximizes the expectation of P_t over all users. After the oracle has decided the latencies for all messages, it sets the time t for the messages from u_{1-b} to ℓ. Since the oracle uses the knowledge of u_{1-b}, Π_{ideal} is slightly more powerful than protocols in M. Due to the over-approximation with this (not realizable) oracle, the resulting protocol is optimal w.r.t. Invariant 1 (Refer to Claim 3 and Claim 4).

Claim 3 (Ideal protocol is ideal for the invariant)**.** *Against the given adversary $\mathcal{A}_{paths}$, Π_{ideal} satisfies Invariant 1 with probability at least as high as any other protocol in M.*

Claim 4 (Ideal protocol wins)**.** *If Π_{ideal} satisfies Invariant 1, $\mathcal{A}_{paths}$ has an advantage of zero:*

$$\Pr[b = \mathcal{A}_{paths} \mid \textit{Invariant 1 holds}] = \tfrac{1}{2}$$

We refer to Appendix 8.12 for the proofs of Claim 3 and Claim 4.

8.5 Synchronized Users with Non-compromising Adversaries

Our first scenario is a protocol-friendly user distribution U_B, where inputs from all users are globally synchronized: over the course of N rounds, exactly one user per round sends a message, following a random permutation that assigns one round to each user. Analogously, the protocol globally instructs the users to send up to $\beta \in [0, 1]$ noise messages **per user** per round, or $B = \beta\mathsf{N}$ noise messages per round in total.

In real life, the user distribution is independent of the protocol. However, to make the user distribution protocol-friendly in our modeling we consider a globally controlled user distribution. For this scenario, we consider *non-compromising* passive adversaries that can observe all network traffic.

8.5.1 Lower Bound on Adversarial Advantage

Theorem 1. *For user distribution U_B, no protocol $\Pi \in M$ can provide δ-sender anonymity, for any $\delta < 1 - f_\beta(\ell)$, where $f_\beta(x) = \min(1, ((x + \beta\mathsf{N}x)/(\mathsf{N} - 1)))$.*

Proof. By Claim 3 and Claim 4, we know that Π_{ideal} is an optimal protocol against $\mathcal{A}_{paths}$; and with $\mathsf{c} = 0$, $\mathcal{A}_{paths}$ is our representative *non-compromising* adversary. Thus, it suffices to calculate the advantage of $\mathcal{A}_{paths}$ against Π_{ideal} as a lower bound of the adversary's advantage against any protocol.

Let, u_0 and u_1 be the users chosen by the adversary and let b be the challenge bit. Let t_0 be the round in which u_b sends the challenge message and let r be the round in which the challenge message reaches the recipient.

Recall that Invariant 1 is necessary for the protocol to provide anonymity; u_{1-b} sends her messages (can be a noise message) at $V = \{t_1, t_2, t_3, \ldots, t_k\}$, then $T = \{t : t \in V \wedge (r-\ell) \leq t < r\}$. Since we are considering a non-compromising adversary, $\Pr[\text{Invariant 1 is true}] = \Pr[\text{T is not empty}]$. With the above in mind, let us define the following events:

H_1: In ℓ rounds u_{1-b} sends at least one noise message.

H_2: u_{1-b} sends his own message within the chosen ℓ rounds.

H_3: there is at least one message from u_{1-b} within the chosen ℓ rounds $\equiv T$ is not empty $\equiv$ Invariant 1 is true.

Consider any slice of ℓ rounds around the challenge message, there are exactly $(\ell-1)$ user messages other than the challenge message. Hence, any slice of ℓ rounds yields the same probability of containing a user message from u_{1-b}, except when $r < \ell$ OR $r > \mathsf{N}$ where the probability is smaller. Thus, no matter what value of t is returned by O, $\Pr[H_2] \leq \frac{\ell-1}{\mathsf{N}-1}$.

Given any values $\ell, \beta \geq 0$, $\mathcal{A}_{paths}$ has the least chance of winning, if for a given interval of ℓ rounds, $\beta\mathsf{N}\ell$ unique users are picked to send the noise messages in such a way that they are not scheduled to send their own messages in that interval.

$$\Pr[\neg H_3] = \Pr[\neg H_1, \neg H_2] \geq \max(0, (\mathsf{N} - \ell - \beta\mathsf{N}\ell)/(\mathsf{N}-1)).$$
$$\Pr[H_3] = 1 - \Pr[\neg H_3] \leq \min(1, ((\ell + \beta\mathsf{N}\ell)/(\mathsf{N}-1))).$$

Thus, we can bound the probability for the adversary as $Pr[0 = \mathcal{A}_{paths}|b = 1] = Pr[1 = \mathcal{A}_{paths}|b = 0] = \frac{1}{2}\Pr[H_3]$; and $Pr[0 = \mathcal{A}_{paths}|b = 0] = 1 - \frac{1}{2}\Pr[H_3]$. And therefore, since $\delta \geq Pr[0 = \mathcal{A}_{paths}|b = 0] - Pr[0 = \mathcal{A}_{paths}|b = 1]$, $\delta \geq 1 - \Pr[H_3] \geq 1 - f_\beta(\ell)$. □

8.5.2 Impossibility for Strong Anonymity

We now investigate under which constraints for ℓ and β Theorem 1 rules out strong anonymity.

Theorem 2. *For user distribution U_B with $\ell < \mathsf{N}$ and $\beta\mathsf{N} \geq 1$, no protocol $\Pi \in M$ can achieve strong anonymity if $2\ell\beta < 1 - \epsilon(\eta)$, where $\epsilon(\eta) = \frac{1}{\eta^d}$ for a positive constant d.*

We refer to Appendix 8.12 for the proof.

Interesting Cases For illustration, we now discuss a few examples for different values of ℓ, β, and N.

1. If $\ell = \mathsf{N}$, we can have $\delta = 0$ even for $\beta = 0$. Anonymity can be achieved trivially by accumulating all messages from all N users and delivering them together at round $(\mathsf{N}+1)$. In this case $2\ell\beta = 0 < 1 - \epsilon(\eta)$, but also $\beta\mathsf{N} = 0 < 1$.

2. $\beta = \frac{1}{\eta}$, $\ell = \eta$: We have $\delta \geq \frac{\mathsf{N}-\eta-\mathsf{N}}{\mathsf{N}} \geq \frac{-\eta}{\mathsf{N}}$. In ℓ rounds the protocol can send $\ell\beta\mathsf{N} = \mathsf{N}$ noise messages and achieve strong anonymity (all N users send a noise message each).

3. $\beta = \frac{1}{2\tau}$, $\ell = \tau$, where τ is a positive integer: Here we have, $\delta \geq \frac{\mathsf{N}-\tau-\frac{\mathsf{N}}{2}}{\mathsf{N}} = \frac{1}{2} - \frac{\tau}{\mathsf{N}}$. Here, strong anonymity is possible if $\frac{\tau}{\mathsf{N}} \geq \frac{1}{2} - neg(\eta)$. Even though $2\ell\beta = 1 > 1 - neg(\eta)$, anonymity depends on the relation between τ and N.

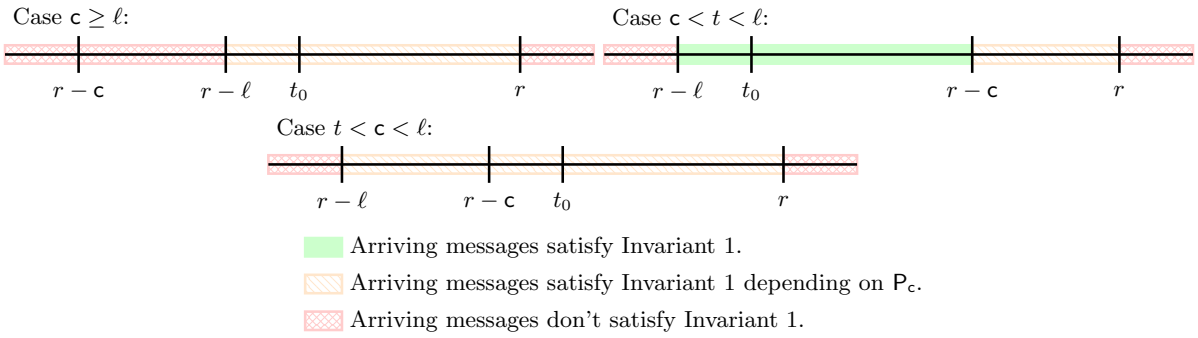


Figure 8.5: Satisfying Invariant 1 depending on the arrival time of messages from u_{1-b} in the cases of the proof for Theorem 3.

4. $\beta = \frac{1}{9}$, $\ell = 3$: For $\eta > 3$ and $\mathsf{N} > 4$, which is a very natural assumption, we have $2\ell\beta = \frac{2}{3} < 1 - neg(\eta)$. Then, $\delta \geq \frac{\mathsf{N}-3-\frac{N}{3}}{\mathsf{N}} > neg(\eta)$. In ℓ rounds Π_{ideal} receives only $(\frac{\mathsf{N}}{3} + 3)$ messages, and thus, with high probability u_{1-b} does not send a message. Hence, Π_{ideal} cannot achieve strong anonymity.

8.6 Synchronized Users with Partially Compromising Adversaries

We now extend our analysis of the previous section by having compromised protocol parties. Given the set of protocol parties P, now our adversary $\mathcal{A}_{paths}$ can compromise a set of c parties $\mathsf{P_c} \subset \mathsf{P}$. If $\mathcal{A}_{paths}$ can compromise all the parties in P, anonymity is broken trivially - that's why we do not analyze that case separately. Recall from Section 8.4.3 that $\mathcal{A}_{paths}$ picks the c parties from P uniformly at random. We consider the same user distribution U_B as in Section 8.5.

8.6.1 Lower Bound on Adversarial Advantage

Theorem 3. *For user distribution U_B, no protocol $\Pi \in M$ can provide δ-sender anonymity, for any*

$$\delta < \begin{cases} 1 - [1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}] f_\beta(\ell) & \mathsf{c} \geq \ell \\ 1 - [1 - 1/\binom{\mathsf{K}}{\mathsf{c}}] f_\beta(\mathsf{c}) - f_\beta(\ell - \mathsf{c}) & \mathsf{c} < \ell \end{cases}$$

where $f_\beta(x) = \min(1, ((x + \beta \mathsf{N} x)/(\mathsf{N} - 1)))$.

Proof. Let u_0, u_1 be the challenge users and let b be the challenge bit. Moreover, let t_0 be the time the challenge message is sent by u_b and let $r = t_0 + t$ be the time it is received by the recipient, where t is the delivery time decided by the oracle O. Similar to Section 8.5, we now calculate the advantage of $\mathcal{A}_{paths}$ against Π_{ideal}.

We distinguish two cases, depending on ℓ and c: 1. First, where the number of compromised parties c is at least as large as the maximal latency ℓ. In this case, all parties on the path of the challenge message could be compromised. 2. Second, where not all parties on the path of the challenge message can be compromised. And hence, the analysis focuses on the arrival times of messages from u_{1-b}. For a graphical depiction of the relationship between the rounds a message from u_{1-b} arrives and it satisfying Invariant 1 we refer to Figure 8.5.

1) Case $\mathsf{c} \geq \ell$. We know, $\ell \geq t$ holds by definition. The invariant is true only if u_{1-b} sends at least one message in one of the rounds between $(r-\ell)$ and $(r-1)$. Additionally, if u_{1-b} sends at least

one message in $\{r-\ell,\ldots,t_0\}$, the invariant holds only if there is at least one non-compromised party on the path between t_0 and $(r-1)$. Whereas, if u_{1-b} does not send any message in $\{r-\ell,\ldots,t_0\}$, and the first message from u_{1-b} in the interval $\{t_0+1, r-1\}$ arrives at t_1, the invariant holds only if there is at least one non-compromised party on the path between t_1 and $(r-1)$.

Note that $\mathsf{K} > \mathsf{c} \geq \ell$. Also recall from Section 8.4 that $\mathcal{A}_{paths}$ picks the c parties uniformly at random from K parties. Hence,

$$
\begin{aligned}
&\Pr[\text{Invariant 1 is true}] \\
&\leq \sum_{j=r-\ell}^{t_0} \Pr[u_{1-b}.\text{sent}(j) \wedge \neg u_{1-b}.\text{sent}(j+1, t_0)] \\
&\qquad\qquad \cdot \Pr[\neg\text{Cmpr}(t)] \\
&+ \sum_{j=t_0+1}^{r} \Pr[u_{1-b}.\text{sent}(j) \wedge \neg u_{1-b}.\text{sent}(r-\ell, j-1)] \\
&\qquad\qquad \cdot \Pr[\neg\text{Cmpr}(r-j)] \\
&\leq \Pr[\neg\text{Cmpr}(\ell)] \cdot \Pr[u_{1-b}.\text{sent}(r-\ell, r-1)] \\
&\leq [1 - \binom{\mathsf{c}}{\ell} / \binom{\mathsf{K}}{\ell}] \cdot \min(1, ((\ell + \beta\mathsf{N}\ell)/(\mathsf{N}-1))).
\end{aligned}
$$

By Claim 1 the adversary wins whenever Invariant 1 is not true, and by Claim 4 $\mathcal{A}_{paths}$ has zero advantage whenever Π_{ideal} satisfies the invariant. Hence, we know that the probability that the adversary guesses incorrectly is bounded by:
$\Pr[0 = \mathcal{A}_{paths} | b = 1] = \Pr[1 = \mathcal{A}_{paths} | b = 0]$
$\leq \frac{1}{2}\Pr[\text{Invariant 1 is true}] \leq \frac{1}{2}[1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}] \cdot \min(1, (\frac{\ell+\beta\mathsf{N}\ell}{\mathsf{N}-1}))$.
Thus, $\delta \geq 1 - [1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}] \cdot \min(1, (\frac{\ell+\beta\mathsf{N}\ell}{\mathsf{N}-1}))$.

2) Case $\mathsf{c} \leq \ell$: The probability that all parties on the mutual path of the challenge message and a message from the alternative sender u_{1-b} are compromised now mainly depends on the arrival time of the messages from u_{1-b}. We find two sub-cases depending on the oracle's choice for t.

2a) Case $\mathsf{c} \leq t$:

$$
\begin{aligned}
&\Pr[\text{Invariant 1 is true}] \\
&\leq \Pr[u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})] + \Pr[\neg u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})] \\
&\qquad\qquad \cdot \Pr[u_{1-b}.\text{sent}(r-\mathsf{c}, r)] \cdot \Pr[\neg\text{Cmpr}(\mathsf{c})] \\
&\leq \min(1, (\tfrac{(\ell-\mathsf{c})+\beta\mathsf{N}(\ell-\mathsf{c})}{\mathsf{N}-1})) \\
&\qquad\qquad + \min(1, (\tfrac{\mathsf{N}-(\ell-\mathsf{c})-\beta\mathsf{N}(\ell-\mathsf{c})}{\mathsf{N}-1})(\tfrac{\mathsf{c}+\beta\mathsf{N}\mathsf{c}}{\mathsf{N}-(\ell-\mathsf{c})-\beta\mathsf{N}(\ell-\mathsf{c})}))[1 - \tfrac{1}{\binom{\mathsf{K}}{\mathsf{c}}}] \\
&\leq f_\beta(\ell-\mathsf{c}) + f_\beta(\mathsf{c})[1 - 1/\tbinom{\mathsf{K}}{\mathsf{c}}].
\end{aligned}
$$

Note that the probability that there are no messages from u_{1-b} in $[(r-\ell),(r-\mathsf{c})]$ and that there is at least one message from u_{1-b} in $[(r-\mathsf{c}), r]$ are not independent from each other. The best thing a protocol can do with the noise messages is to have $\mathsf{N}\beta\ell$ unique users, different from the ℓ users who send their actual message, send the noise messages. Thus, if a user sends a message in $[(r-\ell),(r-\mathsf{c})]$, he can not send a message in $[(r-\mathsf{c}), r]$. The above calculations are done considering that best scenario. Also note that the value of K may be larger or smaller than ℓ and t, but as long as $\mathsf{c} \leq \mathsf{K}$, the bound given above holds. Hence, $\delta \geq 1 - f_\beta(\ell-\mathsf{c}) - [1 - 1/\binom{\mathsf{K}}{\mathsf{c}}] \cdot f_\beta(\mathsf{c})$.

2b) Case $t < \mathsf{c}$:

$$
\begin{aligned}
&\Pr\left[\text{Invariant 1 is true}\right] \\
&\leq \Pr\left[u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})\right] \cdot \Pr\left[\neg\text{Cmpr}(t)\right] \\
&\quad + \Pr\left[\neg u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})\right] \\
&\qquad \cdot \Pr\left[u_{1-b}.\text{sent}(r-\mathsf{c}), r)\right] \cdot \Pr\left[\neg\text{Cmpr}(t)\right] \\
&\leq \Pr\left[u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})\right] + \Pr\left[\neg u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})\right] \\
&\qquad\qquad \cdot \Pr\left[u_{1-b}.\text{sent}(r-\mathsf{c}, r)\right] \cdot \Pr\left[\neg\text{Cmpr}(t)\right]
\end{aligned}
$$

The event expression above is the same as in the previous case ($t > \mathsf{c}$). The bound on δ thus follows analogously. □

8.6.2 Impossibility for Strong Anonymity

Theorem 4. *For user distribution U_B with $\mathsf{K} \in poly(\eta)$, $\mathsf{K} > \mathsf{c} \geq \ell$, $\ell < \mathsf{N}$ and $\beta\mathsf{N} \geq 1$, no protocol $\Pi \in M$ can achieve strong anonymity if $2\ell\beta < 1 - \epsilon(\eta)$ or $\ell \in \mathcal{O}(1)$, where $\epsilon(\eta) = 1/\eta^d$ for a positive constant d.*

We refer to Appendix 8.12 for the proof. To achieve strong anonymity against $\mathcal{A}_{paths}$, we need $\ell \in \omega(1)$, additional to the constraint of $2\ell\beta > 1 - neg(\eta)$. We now focus on the constraint $\ell \in \omega(1)$ and refer to Section 8.5.2 for a comprehensive case study on the other constraint.

Interesting Cases Now we are going to discuss a few interesting cases for different values of $\ell < \mathsf{c}$, and K.

1. $\ell = \eta$ and $\mathsf{K}/\mathsf{c} = constant$: In this case we have, $\binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell} = \frac{\mathsf{c}(\mathsf{c}-1)\ldots(\mathsf{c}-\ell+1)}{\mathsf{K}(\mathsf{K}-1)\ldots(\mathsf{K}-\ell+1)} < (\mathsf{c}/\mathsf{K})^{\ell} = (\mathsf{c}/\mathsf{K})^{\eta}$. Hence, $\binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}$ becomes negligible and strong anonymity is possible. Even though c has a high value, because of the high value of ℓ it is highly likely that the challenge message will meet a message from u_{1-b} at some honest node, given a high value of β such that $2\ell\beta > 1 - neg(\eta)$.

2. $\ell = \mathcal{O}(1), \mathsf{c} = \mathcal{O}(1)$: Now we have, $\binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell} = \frac{\mathsf{c}(\mathsf{c}-1)\ldots(\mathsf{c}-\ell+1)}{\mathsf{K}(\mathsf{K}-1)\ldots(\mathsf{K}-\ell+1)} > ((\mathsf{c}-\ell)/(\mathsf{K}-\ell))^{\ell}$. But $\mathsf{K} \in poly(\eta)$, and c and ℓ can only have integer values. Hence $((\mathsf{c}-\ell)/(\mathsf{K}-\ell))^{\ell}$ is non-negligible, and hence $\binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}$ is also non-negligible. Even though c has a small value, ℓ is also small. Hence, it is unlikely that the challenge message will mix with a message from u_{1-b} at some honest node. Thus, strong anonymity cannot be achieved.

Theorem 5. *For user distribution U_B with $\mathsf{K} \in poly(\eta)$, $\mathsf{c} \in \mathcal{O}(1)$, $\mathsf{K} > \ell > \mathsf{c}$, $\ell < \mathsf{N}$ and $\beta\mathsf{N} \geq 1$, no protocol $\Pi \in M$ can achieve strong anonymity if $2(\ell - \mathsf{c})\beta < 1 - \epsilon(\eta)$, where $\epsilon(\eta) = \frac{1}{\eta^d}$ for a positive constant d.*

We refer to Appendix 8.12 for the proof. The analysis in this case is exactly same as Section 8.5.2, except that here we need to consider the slice of $(\ell - \mathsf{c})$ rounds instead of ℓ rounds.

It is worth repeating here, all the constraints we have derived in Section 8.5 and Section 8.6 are necessary for anonymity, but they are not sufficient conditions for anonymity.

8.7 Unsynchronized Users with Non-compromising Adversaries

In this and the subsequent section we use an unsynchronised user distribution U_P: In each round, independent of other users and other rounds, each client tosses a biased coin with success probability $p \in (0, 1]$. On a success the client sends a message in that round, otherwise it does not send a message. Consequently, the number of messages per round follows Binomial distribution $Binom(\mathsf{N}, p)$

if the number of users N is large and p sufficiently small, the resulting binomial distribution reduces to a Poisson distribution, which is a close approximation of real-life traffic patterns.

For a protocol with bandwidth overhead β, we distinguish between the actual probability that users want to send messages p' and the value for p that we use in our analysis, i.e., we set $p = p' + \beta$. In this unsynchronised scenario the bandwidth of genuine messages contributes to the anonymity bound. As in Section 8.5 we consider a *non-compromising* adversary.

8.7.1 Lower Bound on Adversarial Advantage

Theorem 6. *For user distribution U_P, no protocol $\Pi \in M$ can provide δ-sender anonymity, for any $\delta < 1 - \left(\frac{1}{2} + f_p(\ell)\right)$, where $f_p(x) = \min(1/2,\ 1-(1-p)^x)$ for a positive integer x.*

Proof. Since we consider a non-compromising adversary, $\Pr[\text{Invariant 1 is True}] = \Pr[\text{T is not empty}]$, where T is defined as in Invariant 1.

Let us consider the random variables $X^{(1)}, X^{(2)}, \ldots, X^{(\mathsf{N})}$, where $X^{(i)}$ denotes the event of the i^{th} user sending her own message within a given interval of ℓ rounds $[a, b]$, with $(b-a) = \ell$. All $X^{(i)}$s are mutually independent and we have,

$$X^{(i)} = \begin{cases} 0 & \text{with probability } (1-p)^{\ell} \\ 1 & \text{with probability } (1-(1-p)^{\ell}). \end{cases}$$

Next, let $X = \sum_{i=1}^{\mathsf{N}} X^{(i)}$ be a random variable representing the number of users that send messages in an interval of ℓ rounds. We calculate for the expected value $\mathbb{E}[X]$ of X,

$\mathbb{E}[X] = \sum_{i=1}^{\mathsf{N}} \mathbb{E}[X^{(i)}] = \mathsf{N}(1-(1-p)^{\ell}) = \mu$.

Using the Chernoff Bound on the random variable X we derive $\Pr[X - \mu \geq \mathsf{N}a] \leq \exp(-2a^2\mathsf{N})$, which for $a = \frac{\mu}{\mathsf{N}}$ lets us estimate, $\Pr[X \geq 2\mu] \leq \exp\left(-2(\mu^2/\mathsf{N}^2)\mathsf{N}\right)$. For brevity in the following calculation we denote, $Pr[X \geq 2\mu]$ by E and the event that T is non-empty by Y and since all users are acting independently from each other we get for $j \in \{0, \ldots, \mathsf{N}\}$, $\Pr[Y|X=j] = 1 - \Pr[\neg Y|X=j] = \frac{j}{\mathsf{N}}$. **For** $2\mu \leq \mathsf{N}$, we have,

$$\begin{aligned}
&\Pr[Y] \\
=&\Pr[X \geq 2\mu] \cdot \Pr[Y|X \geq 2\mu] + \Pr[X < 2\mu] \cdot \Pr[Y|X < 2\mu] \\
\leq&\Pr[X \geq 2\mu] \cdot \Pr[Y|X = \mathsf{N}] + \Pr[X < 2\mu] \cdot \Pr[Y|X = 2\mu] \\
=&E \cdot \Pr[Y|X = \mathsf{N}] + (1-E) \cdot \Pr[Y|X = 2\mu] \\
=&E \cdot \frac{\mathsf{N}}{\mathsf{N}} + (1-E) \cdot \frac{2\mu}{\mathsf{N}} \quad = 1 - (1-E)\,(1 - 2f_p(\ell))\,.
\end{aligned}$$

If $2\mu > \mathsf{N}$, we get with $f(\ell) = \min\left(\frac{1}{2},\ 1-(1-p)^{\ell}\right)$, $\Pr[Y] \leq E + (1-E)\,1 \leq 1 \leq 1 - (1-E)\,(1-2f_p(\ell))$.

Thus, $\delta \geq 1 - \Pr[Y] \geq (1-E)\,(1-2f_p(\ell))$. We now use Markov's Inequality on X and derive $E = \Pr[X \geq 2\mu] \leq \frac{1}{2}$, which means, $\delta \geq \frac{1}{2}\,(1-2f_p(\ell)) \geq \frac{1}{2} - f_p(\ell)$. □

Note that in the proof of Theorem 6, in case p is a constant and N is a very high value, then E goes towards zero and instead of using Markov's inequality, we can derive $\delta \geq 1 - 2f_p(\ell)$.

8.7.2 Impossibility for Strong Anonymity

Theorem 7. *For user distribution U_P and $p > 0$, no protocol $\Pi \in M$ can achieve strong anonymity if $2\ell p < 1 - \epsilon(\eta)$, where $\epsilon(\eta) = 1/\eta^d$ for a positive constant d.*

We refer to Appendix 8.12 for the proof. Similar to the constraints in Section 8.5 and Section 8.6, this is also a necessary constraint for anonymity, not a sufficient condition. There can exist ℓ and p such that $2\ell p > 1 - neg(\eta)$, but still no protocol can achieve strong anonymity.

Interesting Cases Now we are going to discuss a few interesting cases for different values of ℓ, p, and N.

1. $p = \frac{1}{\eta}$, $\ell = \eta$: Here, $f_p(\ell) = 1 - (1-p)^\ell > 1 - 1/e > \frac{1}{2}$. Hence, $\delta \geq \frac{1}{2} - f_p(\ell) = 0$. Since $p\ell = 1$, in ℓ rounds the protocol has 1 message per user on an average. So, the protocol has a high chance of winning. Whereas in Section 8.5.2, we saw that Π_{ideal} can win with absolute certainty in this case.

2. $p = \frac{1}{2\tau}$, $\ell = \tau$, τ is a positive integer: even for $\tau > 2$, $f_p(\ell) = 1 - (1-p)^\ell < 0.45$. Hence, $\delta \geq \frac{1}{2} - f_p(\ell) > 0.05$. Even though $2\ell p = 1$, strong anonymity can not be achieved. In an expected scenario, in a slice of ℓ rounds only $p\ell = \frac{1}{2}$ portion of the total users send messages, and hence there is a significant chance that u_{1-b} is in the other half. Note that this is different from the scenario with synchronized users where Π_{ideal} could achieve strong anonymity in this case (c.f. Section 8.5.2).

3. p = $\frac{1}{9}$, $\ell = 3$: Here, $f_p(\ell) = 1 - (1-p)^\ell = 1 - \left(\frac{8}{9}\right)^3 < 0.29$, and $\delta \geq \frac{1}{2} - f_p(\ell) > 0.21$; because of low values of both p and ℓ only a few users send messages within the interval of ℓ rounds, and hence the protocol has a small chance to win. As in Section 8.5.2, Π_{ideal} can not achieve strong anonymity in this case, since the necessary constraints are not satisfied.

8.8 Unsynchronized Users with Partially Compromising Adversaries

Finally, we consider partially compromising adversaries that can compromise a set of c parties $\mathsf{P_c} \subset \mathsf{P}$ for the user distribution U_P defined in Section 8.7.

8.8.1 Lower Bound on Adversarial Advantage

Theorem 8. *For user distribution U_P, no protocol $\Pi \in M$ can provide δ-sender anonymity, for any*

$$\delta < \begin{cases} 1 - [1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}][\frac{1}{2} + f_p(\ell)] & \mathsf{c} \geq \ell \\ \left(1 - [1 - 1/\binom{\mathsf{K}}{\mathsf{c}}][\frac{1}{2} + f_p(\mathsf{c})]\right) & \\ \quad \cdot \left(1 - [1/2 + f_p(\ell - \mathsf{c})]\right) & \mathsf{c} < \ell \end{cases}$$

where $f_p(x) = \min(1/2,\ 1 - (1-p)^x)$ for a positive integer x.

We derive the bound in Theorem 8 by combining the techniques presented in Section 8.6 and Section 8.7. Since the proof does not introduce novel techniques, we omit it and instead refer the interested reader to Appendix 8.12 for the proof.

8.8.2 Impossibility for Strong Anonymity

To analyze the negligibility condition of δ in this scenario, we heavily borrow the analyses that we already have conducted in Section 8.7.2 and Section 8.6.2. We are going to analyze this scenario in two parts:

Case $\mathsf{c} \geq \ell$: We have, $\delta \geq 1 - [1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}]\,[\frac{1}{2} + f_p(\ell)]$.

To make δ negligible, both the factors $[1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}]$ and $[1/2 + f_p(\ell)]$ have to become overwhelming. From Theorem 4, we know that we need $\ell \in \omega(1)$ to make $[1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}]$ overwhelming. This is a necessary condition, but not sufficient. For a detailed discussion, we refer to Section 8.6.2. From Section 8.7.2 we know that the necessary condition for $[1/2 + f_p(\ell)]$ to be overwhelming is $2\ell p > 1 - neg(\eta)$. Hence, both conditions are necessary to achieve strong anonymity.

Case $\mathsf{c} < \ell$: We have,
$\delta \geq (1 - [1/2 + f_p(\ell - \mathsf{c})])(1 - [1 - 1/\binom{\mathsf{K}}{\mathsf{c}}][1/2 + f_p(\mathsf{c})])$.
In the above expression, we can see two factors:
(i) $F_1 = (1 - [\frac{1}{2} + f_p(\ell - \mathsf{c})])$, (ii) $F_2 = (1 - [1 - 1/\binom{\mathsf{K}}{\mathsf{c}}][\frac{1}{2} + f_p(\mathsf{c})])$.

To make δ negligible, it suffices that F_1 or F_2 become negligible. Unlike Section 8.6, here $f_p(\ell - \mathsf{c})$ and $f_p(\mathsf{c})$ are independent, which allows us to analyze F_1 and F_2 independently. First, F_1 is similar to the δ-bound in Section 8.7, except that we consider $f_p(\ell - \mathsf{c})$ instead of $f_p(\ell)$. Hence, the analysis of F_1 is analogous to Section 8.7.2. Second, F_2 is negligible if both $[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}]$ and $[1/2 + f_p(\mathsf{c})]$ are overwhelming. From Section 8.6.2 we know that $[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}]$ can not be overwhelming for a constant c. Moreover, $f_p(\mathsf{c})$ can be analyzed exactly as $f_p(\ell)$ in Section 8.7.2.

8.9 Recipient Anonymity

We derive impossibility results for recipient anonymity analogous to our results for sender anonymity via the same strategy we employed in the previous sections. In this case, since we are considering recipient anonymity, we assume only one sender in $\mathcal{S}$, and N' users in $\mathcal{R}$. Here, the adversary is naturally not informed about the delivery of the challenge message by a recipient, but of the sending of the challenge message by the sender. Moreover, instead of ignoring all internally generated messages in Claim 2 we ignore all internally terminating messages. Note that this gives β a slightly different flavor.

Synchronized Users We slightly tweak the user distribution to suit the definition of *recipient anonymity*. We assume that all the input messages come within N' rounds, exactly one message per round, following a random permutation that assigns one round to each recipient. In a given round, the sender sends a message to the assigned recipient. Then, the protocol decides when to deliver the message to the recipient, but not delaying more than ℓ rounds. Let $f_\beta^{RA}(x) = \min\left(1, \left(\frac{(x+\ell)+(x+\ell)\beta\mathsf{N}'}{\mathsf{N}'}\right)\right)$. Then we get that no protocol $\Pi \in M$ can provide δ-recipient anonymity in the following cases:

- Without compromisation: $\delta < 1 - f_\beta^{RA}(\ell)$.
- For adversaries that compromise up to c parties:
  - if $\mathsf{c} \geq \ell$: $\delta < 1 - [1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}] f_\beta^{RA}(\ell)$.
  - if $\mathsf{c} < \ell$: $\delta < 1 - [1 - 1/\binom{\mathsf{K}}{\mathsf{c}}] f_\beta^{RA}(\mathsf{c}) - f_\beta^{RA}(\ell - \mathsf{c})$.

Moreover, no protocol M with $\mathsf{K} \in poly(\eta)$ can achieve strong recipient anonymity when $\ell < \mathsf{N}'$ and $\beta\mathsf{N}' \geq 1$ in the following cases, where $\epsilon(\eta)$ is a non-negligible function.

- Without compromisation: if $4\ell\beta < 1 - \epsilon(\eta)$,
- For adversaries that compromise up to c parties:
  - if $\mathsf{K} > \mathsf{c} \geq \ell$: $4\ell\beta < 1 - \epsilon(\eta)$ OR $\ell \in \mathcal{O}(1)$.
  - if $\mathsf{K} > \ell > \mathsf{c}$: $4(\ell - \mathsf{c})\beta < 1 - \epsilon(\eta)$.

Unsynchronized Users Similar to the previous case, here also we borrow the definition of user distribution from Section 8.7, with minor modifications. The biased coins are now associated with recipients instead of senders — in each round the sender sends a message **for a recipient**, with probability p. Let $f_p^{RA}(x) = \min(1/2,\ 1-(1-p)^{\ell+x})$. Then we get that no protocol $\Pi \in M$ can provide δ-recipient anonymity in the following cases:

- Without compromisation: $\delta < 1 - (1/2 + f_p^{RA}(\ell))$.
- For adversaries that compromise up to c parties:
  - If $\mathsf{c} \geq \ell$: $\delta < [1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}][1/2 + f_p^{RA}(\ell)]$.
  - If $\mathsf{c} < \ell$: $\delta < \Big(1 - [1/2 + f_p^{RA}(\ell - \mathsf{c})]\Big) \cdot \Big(1 - [1/2 + f_p^{RA}(\mathsf{c})][1 - 1/\binom{\mathsf{K}}{\mathsf{c}}]\Big)$.

Moreover, for $p > 0$, no protocol can achieve strong recipient anonymity if $2\ell p < 1 - \epsilon(\eta)$, where $\epsilon(\eta)$ is a non-negligible function. For a detailed recipient-anonymity analysis, we refer the readers to the extended version [15].

8.10 Implications

To put our result into perspective, we discuss whether our trilemma excludes strong anonymity for a few AC protocols from the literature. More precisely, this section exemplarily applies the results from Theorem 2 and Theorem 7, i.e., with synchronized and unsynchronized user distributions and a global network-level, non-compromising adversary. We use both results since for some AC protocols (e.g., DC-nets [7]) the synchronized user distribution is more accurate and for other protocols (e.g., Tor [16]) the unsynchronized user distribution is more accurate. Our constraints mark an area on a 2D graph (see Figure 8.6) with latency overhead (x-axis) versus bandwidth overhead (y-axis) where strong anonymity is impossible. As the latency of some AC protocols depends on system parameters and we want to place the protocols in a 2D graph, we carefully choose system parameters and make a few simplifying assumptions, which are subsequently described.

This section is solely intended to put our impossibility result into perspective. It is not meant and not qualified to be a performance and scalability comparison of the discussed AC protocols. Table 8.1 in the appendix summarizes bounds on the bandwidth β and latency overhead ℓ (in the sense of this work).

Technically, this section considers translations of AC protocols into our protocol model. As these translations do not provide any additional insights, we do not present the full translated protocols but only the abstraction steps. We abstract away the cryptographic instantiation of messages including the bandwidth overhead they introduce over the plaintext. We assume an upper bound on the latency of the protocol and are oblivious to server-side noise (see Claim 2). Moreover, recall that we are only interested in the question whether our trilemma excludes strong anonymity for the ten AC protocols from the literature; hence, we consider the upper bound on the latency and bandwidth overhead for deterministic latency. For randomized latency, such as Loopix [41], we list for simplicity the expected delay as the latency bound.

Low-latency protocols such as Tor [16], Hornet [8], and Herd [33] are low-latency AC protocols, i.e., they immediately forward messages. While Tor and Hornet do not produce asymptotically more than a constant amount of both bandwidth overhead and latency overhead and thus cannot provide strong anonymity, Herd produces dummy traffic linearly proportional to the number of

users (bandwidth overhead $\beta \in \theta(\mathsf{N}/\mathsf{N})$), thus the trilemma does *not* exclude strong anonymity for Herd.

Riposte [9] uses secure multiparty computation and a variant of PIR to implement an anonymous bulletin board. Riposte operates in epochs and for each epoch the set of users is public. Hence, Riposte is expected to be run with long epochs to maximize the number of users that participate in an epoch, which leads us to estimating the latency overhead to be $\ell \in \theta(N)$. To counter traffic analysis attacks, Riposte clients send constant dummy traffic, resulting in a bandwidth overhead of $\beta \in \theta(\mathsf{N}/\mathsf{N})$. Thus, the trilemma does not exclude strong anonymity for Riposte.

Vuvuzela [49] is a mix-net that is tailored towards messengers. Clients communicate by deposing their encrypted messages in one of the mix net nodes. To achieve strong resistance against compromised servers, Vuvuzela takes a path through all servers, resulting in a latency overhead of $\ell \in \theta(\mathsf{K})$ (for K servers). Additionally, Vuvuzela utilizes constant traffic, leading to a bandwidth overhead of $\beta \in \theta(\mathsf{N}/\mathsf{N})$, and has the potential for strong anonymity.

Riffle [32] uses a verifiable mix-net. Just as Vuvuzela, Riffle also chooses paths that traverse all K servers, leading to $\ell \in \theta(\mathsf{K})$ and if we assume $\mathsf{K} \in \theta(\log(\eta))$, we get $\ell \in \theta(\log(\eta))$. We assume that the clients send dummy traffic up to a constant rate (depending on the user's sending rate p'), so we have $\beta \in \theta(\mathsf{N}/\mathsf{N})$ and the potential for strong anonymity.

In a **threshold mix net**, each of the K mix servers waits until it received up to a threshold T many messages before relaying the messages to the next mix, resulting in $\ell \in \theta(T \cdot K)$. Threshold mixes [44] do not provide strong anonymity unless their threshold T is of the order of the number of users N. As such a large threshold are impractical for a large number of users, we judge it impossible to achieve strong anonymity for practical of Threshold mixes.

Loopix [41] is a mix net that combines exponentially distributed delays at each mix-node and dummy messages from each user. Ignoring so-called loop messages (meant to counter active attacks), Loopix naturally enforces our unsynchronised user distribution: the rate at which Loopix clients send messages is the sum of a dummy-message rate (β) and a payload message rate (p'), which are system parameters. We assume that the path lengths in Loopix' stratified topology is $\sqrt{K}$ with the number of nodes $K \in \theta(\log(\eta))$. If $\beta + p' \geq 1/\sqrt{\eta}$, and if every hop introduces an expected delay of $\ell' \geq \frac{\sqrt{\eta}}{\sqrt{K}}$, the expected latency overhead is $\ell = \sqrt{K} \cdot \ell'$, in particular $\ell \in \theta(\sqrt(\eta))$. We get $(p' + \beta)\ell = \frac{1}{\sqrt{\eta}} \cdot \sqrt{\eta} = 1$ and the trilemma does not exclude strong anonymity for Loopix.

In **AC protocols based on DC-nets** [7, 24] each party broadcasts either a dummy or real message in every round to every other party. As our bandwidth overhead only counts dummy-message rates, it does not capture the broadcast, thus $\beta \in \theta(\mathsf{N}/\mathsf{N})$. DC-nets use a combination operation (e.g., an XOR) that causes dummy messages to cancel out. Then, all parties output the resulting bitstring. If only one real message is sent, the bitstring equals this message. As Theorem 7 assumes a synchronized user distribution, in each round only one party sends a message, thus our model treats ℓ as $\ell \in \theta(1)$.

The **Dissent-AT** [51] scheme (the AnyTrust-variant of Dissent) improves on the performance of DC-nets by relying on dedicated servers. Instead of broadcasting to every other client, clients in Dissent-AT send these messages to at least one of these dedicated servers. These servers then perform a DC-net communication round. Abstracting from an initial set-up phase and only counting the client-messages, Dissent-AT has $\beta \in \theta(\mathsf{N}/\mathsf{N})$ for the clients (assuming that each client communicates to one server), and $\ell \in \theta(1)$.

Dicemix [43] is a peer-to-peer AC protocol that is based on the DC-net approach. While Dicemix includes a self-healing mechanism that leads to $4 + 2f$ communication rounds for one message if f peers are malicious, this mechanism does not kick in if all peers are honest, leading to only 4 communication rounds, resutling in $\ell \in \theta(1)$. As every party sends a message in every

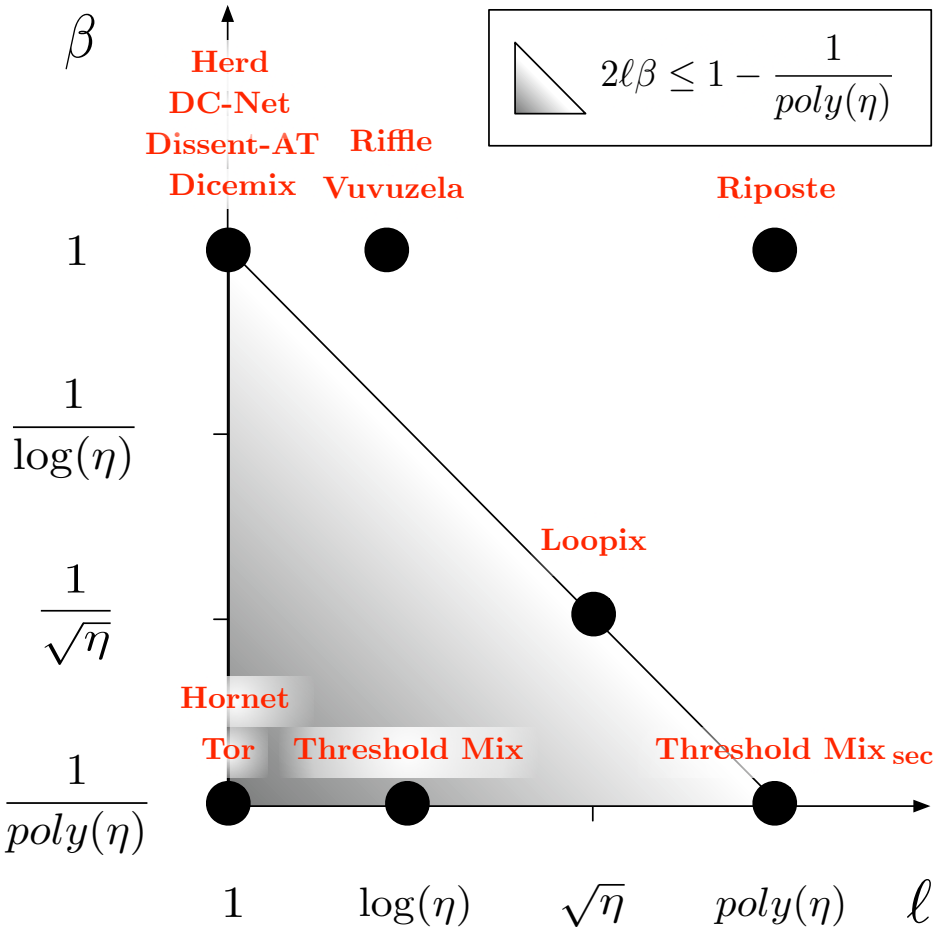


Figure 8.6: Asymptotic latency overhead (ℓ) and bandwidth overhead (β) together with the "area of impossibility" where $2\ell\beta \leq 1 - \epsilon(\eta)$. We portray protocols as dots depending on their choices for ℓ and β. Technically, if we use Theorem 7, we β is replaced by $p = \beta + p'$, where p' is the rate at which users send messages. This graph assumes N is ca. $poly(\eta)$, the number of nodes K is ca. $\log \eta$. The threshold for Threshold Mix $T = 1$ and for Threshold Mix_{sec} $T = \mathsf{N} = poly(\eta)$. In the graph, both the axes are approximately in logarithmic scale. (For a more accurate visual representation we refer the readers to Appendix 8.13 and [1].)

round $\beta \in \theta(\mathsf{N}/\mathsf{N})$.

Table 8.1: Latency vs. bandwidth vs. strong anonymity of AC protocols, with the number of protocol-nodes K, number of clients N, and message-threshold T, expected latency ℓ' per node, dummy-message rate β.

Protocol	Latency	Bandwidth	Strong Anonymity
Tor [16]	$\theta(1)$	$\theta(1/\mathsf{N})$	impossible
Hornet [8]	$\theta(1)$	$\theta(1/\mathsf{N})$	impossible
Herd [33]	$\theta(1)$	$\theta(\mathsf{N}/\mathsf{N})$	possible
Riposte [9]	$\theta(N)$	$\theta(\mathsf{N}/\mathsf{N})$	possible
Vuvuzula [49]	$\theta(\mathsf{K})$	$\theta(\mathsf{N}/\mathsf{N})$	possible
Riffle [32]	$\theta(\mathsf{K})$	$\theta(\mathsf{N}/\mathsf{N})$	possible
Threshold mix [44]	$\theta(T\mathsf{K})$	$\theta(1/\mathsf{N})$	impossible*
Loopix [41]	$\theta(\sqrt{K}\ell')$	$\theta(\beta)$	possible
DC-Net [7, 24]	$\theta(1)$	$\theta(\mathsf{N}/\mathsf{N})$	possible
Dissent-AT [51]	$\theta(1)$	$\theta(\mathsf{N}/\mathsf{N})$	possible
DiceMix [43]	$\theta(1)$	$\theta(\mathsf{N}/\mathsf{N})$	possible

* if T in $o(poly(\eta))$

8.11 Protocol Model Revisited

8.11.1 Validity of the Protocol Model (Contd.)

Lemma 2. *Let* Π *be a protocol* $\in M$ *with* K *parties with parameters* β *and* ℓ*. Then: 1. Messages are delivered within* ℓ *steps. 2. The protocol adds (for the* unsynchronised *case on average) a maximum of* β *noise messages per user per round. 3. Whenever a party in* $\mathcal{S} \cup \mathsf{P}$ *sends a message to another party in* $\mathsf{P} \cup \mathcal{R}$*, the adversary learns that and in which round this happens. 4. For every message that leaves the network (received by R), the adversary additionally learns whether the message is the target message. 5. For every compromised party, the adversary learns the mapping between the input messages and the output messages.*

Proof. Let Π be a protocol $\in M$ with K parties with parameters β and ℓ. Part (2) of the Lemma holds, since we restrict the user distributions accordingly and since the none of the transitions in the petri-net can create more tokens within the network than it consumes from its input place.

We show the part (1) of the lemma via structural induction over fired transitions of the petri net. We additionally add to the induction invariant that all tokens that are not in $\mathcal{S}$ have a timestamp for their next transition of $\mathsf{ts} = 1$ and a remaining time of $\mathsf{t_r} > 0$ and there are at least $\mathsf{t_r}$ rounds left in which the token can be delivered.

Induction base: The protocol is initialized and no transitions have happened. Thus, no messages have been sent so far, i.e., there is no message that has not been delivered within ℓ steps. The only transition that can fire is $T_{\mathcal{S}}$ and for $\ell > 0$, the message introduced into the network in this way does not need to be delivered already ($0 < \mathsf{t_r} = \ell$). Moreover, $T_{\mathcal{S}}$ sets the timestamp of this message token to $\mathsf{ts} = 1$

Induction step: Let tr be any execution trace s.t. the induction invariant is satisfied and let t be an arbitrary possible transition that extends tr to $tr :: t$.

We distinguish two cases for t: In case t is $T_{\mathcal{S}}$, it consumes a token from $P_{\mathcal{S}}$ and puts this token into a place P_i and, by definition we have $\mathsf{t_r} > 0$ and $\mathsf{ts} = 1$. Otherwise, the transition is T_{P_i} for

some i and consumes a token from P_i accordingly. By the induction invariant, the token has $\mathsf{t_r} > 0$. If this token has $\mathsf{t_r} - 1 = 0$, the transition delivers the token to R. Otherwise, t decreases $\mathsf{t_r}$ by one (thus fulfilling the condition that there are at least $\mathsf{t_r}$ rounds left in which the token can be delivered) and sets $\mathsf{ts} = 1$. Since every token in any place P_i needs to be consumed in every round, the protocol delivers every message in at most ℓ steps.

Other parts of the lemma: By definition of our petri net, whenever a transition fires, an element (t, r) is placed into Tokens, containing the public fields of t, such as $t.\mathsf{prev}$ and $t.\mathsf{next}$, as well as the current round number r, which fulfills part (3). Moreover, whenever the transition places the token in R, the adversary can additionally see the field $t.\mathsf{msg}$ and no transition can change the field msg, which allows the adversary to effectively tag and recognize the challenge message and thus fulfills part (4). Finally, if any party P_i is compromised, P_i does not modify the unique (and otherwise freshly sampled) field $t.\mathsf{ID_t}$, which allows the adversary to map incoming and outgoing messages.

Since the transitions discussed here are the only way for messages to be sent to a recipient, the model correctly enforces the conditions from the lemma. □

8.11.2 Expressing Protocols in the petri net model

Modeling DC net Here we show how to model an actual DC net type protocol using our petri net model M as defined in Section 8.4. Specifically we pick up the *short DC net* protocol proposed by *Golle and Juels* [24], and present M_{DC} which models the aforementioned protocol.

We model a DC net protocol with N participants, where $\mathcal{S} = \mathsf{P}$, $|\mathcal{S}| = |\mathsf{P}| = \mathsf{N}$. We denote the parties with $P_1, \ldots, P_N$. The protocol can be denoted by Π_{DC} ={paramgen, keydist, post, verify, extract}[7] - as described below.

• *paramgen*: In $prot_{DC}$, *paramgen* is executed by a trusted entity and the output is published. Since we are mainly interested in the anonymity game, we consider that *paramgen* step is executed by our honest challenger and happens outside the protocol run, and the output is globally known (to all the transitions T_{P_i}).

• *keydist*: using the output of *paramgen*, this step yields for each party P_i a private key x_i and a corresponding public key y_i. In $prot_{DC}$, the above key generation part is done by a trusted entity, and hence we consider that it is done by our honest challenger and for each party P_i the public-private keypair x_i, y_i is already known to the corresponding transition function T_{P_i}. As part of protocol each party P_i publishes its public key y_i. Additionally, each party P_j receives from P_i a share of private key $x_{i,j}$ and a share of public key $y_{i,j}$, where the keys are shared in a (k, N) threshold manner for a parameter $k \leq \mathsf{N}$.

• *post*: Each player P_i generates a vector of random pads $W_i = \{W_i(1), W_i(2), \ldots, W_i(N)\}$[8] using x_i. Π_{DC} does not handle *collisions*, instead assumes that the players decide their positions by a consensus protocol. Similarly our model assumes that each party P_i knows its position, and assume the position is q_i (but not known to the adversary). Then each player P_i computes the vector V_i such that $V_i(w) = W_i(w)$ for all $w \neq i$ and $V_i(w) = W_i(w) \oplus m_i$ for $w = q_i$, where m_i is the message of P_i. Also, each player P_i computes $\sigma_i = \{\sigma_i(1), \sigma_i(2), \ldots, \sigma_i(\mathsf{N})\}$, where σ_i includes the identity of player P_i and a proof of valid formatting of V_i. Then P_i publishes both the vectors V_i and σ_i. Our model assumes the pair $(V_i(w), \sigma_i(w))$ for each position w as a single message, where

[7] Since we are mainly interested in the anonymity property, we don't need to model the part of the protocol where the protocol parties reconstructs the keys in case of a failure. But it is easy to extend M_{DC} to include that step by adding one more round to the current model.

[8] The anonymity game does not include multiple sessions. Also, in our model all the N players participate in a protocol run.

$V_i(w)$ is a message content and $\sigma_i(w)$ becomes a part of *meta* field. For each position w player P_i generates one such message, and publishes the message to all other players.

• *verify* and *extract* are local computations after a party P_i receives messages from all other parties.

Although the protocol model assumes that the adversary can not read the contents of any message, here we shall model Π_{DC} along with its cryptographic primitives to demonstrate the expressiveness of our model. Alternatively, to get rid of all the cryptographic primitives, the parties can send a dummy message ($= 0$) whenever $V_i(w) = W_i(w)$, and the actual message m_i whenever $V_i(w) \neq W_i(w)$.

As per our anonymity definition in Section 8.3, we assume that up to $(\mathsf{N} - 2)$ users can be compromised, which necessarily makes up to $(\mathsf{N} - 2)$ protocol parties compromised. The adversary chooses two challenge users, and one of them sends the challenge message depending on the challenge bit b. All other $(N - 1)$ users send dummy messages.

In M_{DC} we model Π_{DC} as a two round protocol. The challenger sets the initial configuration of the petri-net with the messages to be sent by each party. In the first round, each party P_i sends two kinds messages: (1) publishes the public key message y_i and (2) sends share of the public-private keypair $(x_{i,j}, y_{i,j})$ to P_j for all $j \neq i$. Here, one party can publish a message to $(N-1)$ other parties by sending $(N - 1)$ separate messages. In the second round, each party P_i publishes N messages: one message for each position, only one of them contains his own message. After second round, every party receives messages from every other party, and then does local computations to verify and extract the original messages.

For Π_{DC}, we do not actually need a separate recipient R in Π_{DC}, if we make $\mathcal{R} = \mathsf{P}$. But, to be consistent with M, in M_{DC} we keep a separate recipient. In the second round whenever a party P_i publishes a message, P_i also sends a copy to R. This easily models the fact that the adversary knows whenever a message is published, but avoids the complication of modeling a subset of compromised recipients.

The *meta* fields of the tokens contains the following subfields: (1) stage, (2) position, (3) sigma. *stage* can have three possible values identifying three possible cases: (1) public key distribution, (2) share of the public-private keypair, (3) message. Using *stage* subfield, any party in the protocol recognizes if the message is part of *keydist* messages, or part of *post* messages. When the value of *stage* is *message*, the user posts $V_i(w)$, and *position* takes the value of w. *sigma* includes the identity of the sender and a proof of computation whenever necessary. *sigma* fields helps in the *verify* stage, we avoid the details here.

If we want to analyze the user distribution for Π_{DC}, we do not count the first round since it is used only for key exchange. Note that, if we get rid of the cryptographic primitives, we do not require the first round.

Modeling Tor Since Tor does not operate in rounds, embedding it into our model is not straight forward. Suppose, a Tor node takes at least x milliseconds to process a message when it receives a message, and it takes at least y milliseconds for a message to travel from one node to the next node over a network link. Then we define *one round* as $x + y$ milliseconds. We assume a perfect condition where each node takes exactly equal computation time for one message, and each link has exactly same delay. In the real world, delays and computation times are less stable, but can be estimated by an adversary. Instead of analyzing this, we instead allow the messages to remain within the node for the respective time.

Tor nodes and recipients are separate entities and hence, $\mathcal{S}$, P and R are mutually exclusive. Whenever a Tor node receives a message, the node immediately processes and forwards that message

to the next node or recipient. We can either model the latency overhead ℓ of Tor by estimating the time messages spend within the network that exceeds the (minimal) round length $x+y$ from above, or we set it to the number of hops, i.e., $\ell = 3$. In either case, we assume that ℓ does not increase with η and thus get a latency overhead $\ell \in O(1)$. For analyzing Tor with a variable number h of hops, we can instead set $\ell = h$. When a party P_i receives a message, T_{P_i} can retrieve the next hop from the meta field of the message. Since Tor does not add any noise messages, the bandwidth overhead is $\beta = 0$.

8.12 Delayed Proofs

Proof of Claim 1. If the set T is empty, then S_{1-b} is empty as well. However, by construction of our protocol mode, the set S_b is always non-empty. Consequently, the adversary $\mathcal{A}_{paths}$ will output b and thus win with probability 1. If T is not empty, the following cases can occur:

1. The challenge message never passes through an honest node: In this case, the field $\mathsf{ID_t}$ of the message never changes for the tokens. Thus, S_b will have exactly one element, and S_{1-b} will be an empty set, and consequently $\mathcal{A}_{paths}$ wins.

2. The challenge message passes through one or more honest nodes at times t', such that $t' < \min(T)$, but not afterwards. Following the same reasoning as above, we see that paths before $\min(T)$ can be ambiguous, but none of them leads to u_{1-b}. Hence, S_b can have multiple elements, but S_{1-b} will still be an empty set. Thus, $\mathcal{A}_{paths}$ wins.

3. The challenge message passes through an honest node at time t' with $t' \geq min(T)$. In this case, the invariant is true.

In all of the above mentioned cases either the invariant is true, or the adversary wins with probability 1. □

Proof of Claim 2. Let u_0, u_1 be the challenge users and let b be the challenge bit and let r be the round in which the challenge message is delivered to the recipient. We discuss both parts of the invariant separately:

(i) The set T is not empty. Since by definition, T is the set of messages sent by u_{1-b}, messages originating in any party not in $\mathcal{S}$ do not influence T. Moreover, any message sent by u_{1-b} in a round previous to $r - \ell$ does not influence T either. Thus, noise messages staying in the protocol for more than ℓ rounds, do not improve the probability of T being not empty.

(ii) The challenge message passes through at least one honest node at some time t' such that, $t' \in \{\min(T), \ldots, r-1\}$. Obviously this second part of the invariant does not depend on any noise message. □

Proof of Claim 3. We want to prove our claim by contradiction. Suppose, Π_{ideal} is not the best protocol. That means, there exists a protocol Π_{new}, which satisfies Invariant 1 with a higher probability than Π_{ideal}, against the adversary $\mathcal{A}_{paths}$.

Now we construct a new protocol Π_{hybrid}, which exactly follows the strategy of Π_{ideal} with one exception: for a given message Π_{hybrid} selects the time delay t same as Π_{new}, instead of querying it from oracle O. Suppose, the challenge message is delivered to the recipient at round r. Given the set $\{\min(T), \ldots, r-1\}$, the ideal strategy for ensuring that at least one honest party is on the path of the challenge message is to ensure that as many distinct parties as possible are on this path. Also, given the time delay t, the value of $min(T)$ is independent of the protocol, since protocols in M are oblivious to the challenge users and the challenge message. Hence, Π_{hybrid} has a probability of satisfying Invariant 1 at least as high as Π_{new}.

Now, if we compare Π_{hybrid} and Π_{ideal}: they follow the same strategy. But Π_{ideal} picks the time delay t for any message from oracle O (except for messages from u_{1-b}) such that t is *optimal*. The time delay t can be picked for each message independent of the time delays of other messages. Hence, the value of t received from oracle O for the challenge message is optimal. Hence, Π_{ideal} satisfies Invariant 1 with probability at least as high as Π_{hybrid}. Thus, Π_{new} does not satisfy Invariant 1 with a higher probability than Π_{ideal}. □

Proof of Claim 4. If the Invariant is true, the challenge message passes through an honest party at t', such that $t' > min(T)$. Hence, there is at least one message (noise or original message) from u_{1-i} which visits the same honest party together with the challenge message (Π_{ideal} ensures that all messages are always kept together until they are delivered). That ensures that in addition to $S_b \neq \emptyset$, we also have $S_{1-b} \neq \emptyset$ and thus $\mathcal{A}_{paths}$ outputs a random bit (and has an advantage of zero). □

Proof of Theorem 2. For strong anonymity, we require: $\delta(\eta) = neg(\eta)$, and we know that for Π_{ideal} we have: $\delta(\eta) \geq 1 - f_\beta(\ell) = \left(\frac{\mathsf{N}-\ell-\beta\mathsf{N}\ell}{\mathsf{N}-1}\right) \geq \left(\frac{\mathsf{N}-\ell-\beta\mathsf{N}\ell}{\mathsf{N}}\right) \geq 1 - \frac{\ell}{\mathsf{N}} - \beta\ell$.
We assume for contradiction that there is a protocol limited by ℓ and β such that $2\ell\beta < 1 - \epsilon(\eta)$ that still achieves strong anonymity. Since $\delta(\eta) = neg(\eta)$, we know that $\epsilon(\eta) > \delta(\eta)$.

$$\begin{aligned}
&\epsilon(\eta) > \delta(\eta) \implies \epsilon(\eta) > 1 - \frac{\ell}{\mathsf{N}} - \beta\ell \\
\implies &\epsilon(\eta) > 1 - \frac{\ell}{\mathsf{N}} - \frac{1}{2}\left(1 - \epsilon(\eta)\right) \\
\iff &2\ell > \mathsf{N}\left(1 - \epsilon(\eta)\right) \overset{\mathsf{N}\beta\geq 1}{\implies} 2\ell\beta > 1 - \epsilon(\eta)
\end{aligned}$$

The above contradicts the assumption that $2\ell\beta < 1 - \epsilon(\eta)$.

Note: In case $\beta\mathsf{N} < 1$, no noise messages are allowed per round (i.e., $\beta = 0$) and thus $\delta(\eta) \geq 1 - \ell/\mathsf{N}$, which is not negligible unless $\ell = \mathsf{N}$, since $\mathsf{N} = poly(\eta)$. □

Proof of Theorem 4. When $\mathsf{c} > \ell$: $\delta \geq 1 - \left[1 - \frac{\binom{\mathsf{c}}{\ell}}{\binom{\mathsf{K}}{\ell}}\right] f_\beta(\ell)$.

For δ to become $neg(\eta)$, we need both $[1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}]$ and $f_\beta(\ell)$ to become overwhelming. From Theorem 2 and Theorem 1, we know that $2\ell\beta > 1 - neg(\eta)$ is a necessary condition for $f_\beta(\ell)$ to become overwhelming. Now, we are left with the factor $[1-\binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}]$. This can become overwhelming iff $[\binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}]$ becomes negligible. We know that $\mathsf{K} > \mathsf{c} \geq \ell$ and $\mathsf{K} \in poly(\eta)$. Hence, for some constant x,

$$\begin{aligned}
&\frac{\mathsf{c}-\ell}{\mathsf{K}-\ell} > \frac{1}{\eta^x} \iff \left(\frac{\mathsf{c}-\ell}{\mathsf{K}-\ell}\right)^\ell > \left(\frac{1}{\eta^x}\right)^\ell \\
\implies &\frac{\mathsf{c}(\mathsf{c}-1)\ldots(\mathsf{c}-\ell)}{\mathsf{K}(\mathsf{K}-1)\ldots(\mathsf{K}-\ell)} > \left(\frac{\mathsf{c}-\ell}{\mathsf{K}-\ell}\right)^\ell > \left(\frac{1}{\eta^x}\right)^\ell \\
\iff &\frac{\binom{\mathsf{c}}{\ell}}{\binom{\mathsf{K}}{\ell}} > \left(\frac{1}{\eta^x}\right)^\ell.
\end{aligned}$$

For any $\ell \in \mathcal{O}(1)$, $(1/\eta^x)^\ell$ is non-negligible. □

Proof of Theorem 5. When $\mathsf{c} < \ell$:
$\delta \geq 1 - \left[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}\right] f_\beta(\mathsf{c}) - f_\beta(\ell - \mathsf{c})$.

First consider the factor $[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}]$. Since $\mathsf{K} = poly(\eta)$ and $\mathsf{c} = constant$, $[1/\binom{\mathsf{K}}{\mathsf{c}}]$ can never be negligible. And thus, $[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}]$ can never be overwhelming. So, $[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}]f_\beta(\mathsf{c})$ can never be overwhelming as well, since $f_\beta(\mathsf{c}) \leq 1$.

Now, let's consider $f_\beta(\ell - \mathsf{c})$ and $f_\beta(\mathsf{c})$. Note that, these two factors represent the probabilities of two dependent but mutually exclusive events, and hence $f_\beta(\mathsf{c}) + f_\beta(\ell - \mathsf{c}) \leq 1$. And we already know that $[1 - 1/\binom{\mathsf{K}}{\mathsf{c}}]$ can never be overwhelming. Thus, the only way δ can become negligible is if $f_\beta(\ell - \mathsf{c})$ becomes overwhelming. Note that, if $a + b \leq 1$ and $c < 1$, the only way $ac + b = 1$ is possible if $b = 1$.

Now we can follow exactly the same procedure as in the proof of Theorem 2 to say: $f_\beta(\ell - \mathsf{c})$ can not become overwhelming if $2(\ell - \mathsf{c})\beta < 1 - \epsilon(\eta)$. □

Proof of Theorem 7. We know $0 \leq E \leq 1/2$. When $2\mu \leq \mathsf{N}$,

$$\begin{aligned}\delta \geq & (1 - E)\,(1 - 2f_p(\ell)) \geq 1/2\left(2\,(1 - p)^\ell - 1\right)\\ \geq & 1/2\,(2\,(1 - \ell p) - 1) = 1/2\,(1 - 2\ell p)\,.\end{aligned}$$

Thus, if $2\ell p < 1 - \epsilon(\eta)$,

$$\begin{aligned}2\ell p < 1 - \epsilon(\eta) &\iff 1 - 2\ell p > \epsilon(\eta)\\ &\implies \delta > 1/2 \cdot \epsilon(\eta) = \text{non-negligible.}\end{aligned}$$

Thus, when $2\mu \leq \mathsf{N}$, a necessary condition for δ to become negligible is $2\ell p > 1 - neg(\eta)$.

When $2\mu > \mathsf{N}$, using $\mu = \mathsf{N}(1 - (1 - p)^\ell)$ we get:

$$\begin{aligned}2\mathsf{N}(1 - (1 - p)^\ell) > \mathsf{N} &\implies (1 - p)^\ell < 1/2\\ &\implies 1 - p\ell < 1/2 \iff 2p\ell > 1.\end{aligned}$$

□

Proof of Theorem 8. Let $X^{(i)}(x)$ and $X(x)$ be defined as in the proof for Theorem 6, where we replace the fixed length ℓ of the slice by a variable x. Using the Chernoff Bound on the random variable $X(x)$ calculate $\Pr[X(x) - \mu(x) \geq \mathsf{N}a] \leq \exp(-2a^2\mathsf{N})$, and for $a = \frac{\mu(x)}{\mathsf{N}}$, we define $E(x)$ as :

$$\begin{aligned}E(x) = \Pr[X(x) \geq 2\mu(x)] &\leq \exp\left(-2\mu(x)^2/\mathsf{N}^2 \cdot \mathsf{N}\right)\\ &\leq \exp\left(-2(1 - (1 - p)^x)^2\mathsf{N}\right).\end{aligned}$$

Note that, similar to $X^{(i)}(x)$ and $X(x)$, $\mu(x)$ is also defined as in the proof for Theorem 6, but for a slice of variable length x. We denote the event that sender u_{1-b} sends at least one message in an interval of size x by $Y(x)$ and since all users are acting independently from each other we get for $j \in \{0, \ldots, \mathsf{N}\}$, $\Pr[Y(x)|X(x) = j] = 1 - \Pr[\neg Y|X(x) = j] = \frac{j}{\mathsf{N}}$. Moreover, for any value of x with $2\mu(x) \leq \mathsf{N}$,

$$\begin{aligned}\Pr[Y(x)] = & \Pr[X(x) \geq 2\mu(x)] \cdot \Pr[Y(x)|X(x) \geq 2\mu(x)]\\ & + \Pr[X(x) < 2\mu(x)] \cdot \Pr[Y(x)|X(x) < 2\mu(x)]\\ \leq & \Pr[X(x) \geq 2\mu(x)] \cdot \Pr[Y(x)|X(x) = \mathsf{N}]\\ & + \Pr[X(x) < 2\mu(x)] \cdot \Pr[Y(x)|X(x) = 2\mu(x)]\\ = & E(x)\Pr[Y|X(x) = \mathsf{N}]\\ & + (1 - E(x))\Pr[Y|X(x) = 2\mu(x)]\\ = & E(x)\,(\mathsf{N}/\mathsf{N}) + (1 - E(x))\,(2\mu(x)/\mathsf{N})\\ = & 1 - (1 - E(x))\left(1 - 2\left(1 - (1 - p)^x\right)\right).\end{aligned}$$

If $2\mu(x) > \mathsf{N}$, we get with $f(x) = \min\left(\frac{1}{2},\ 1-(1-p)^x\right)$:

$$\begin{aligned}\Pr[Y(x)] &\leq E(x) + (1-E(x)) \cdot 1 \leq 1 \\ &\leq 1 - (1-E(x))\,(1-2f(x))\,.\end{aligned}$$

Now, we calculate the probability of Invariant 1 being true, under our protocol Π_{ideal} and as in the proof for Theorem 3. We distinguish two cases depending on c and ℓ:

Case 1): $\mathsf{c} > \ell$

$$\begin{aligned}&\Pr[\text{Invariant 1 is true}] \\ &\leq \Pr[\neg\text{Cmpr}(\ell)] \cdot \Pr[u_{1-b}.\text{sent}(r-\ell, r-1)] \\ &= \Pr[\neg\text{Cmpr}(\ell)] \cdot \Pr[Y(\ell)] \\ &\leq \left[1 - \tbinom{\mathsf{c}}{\ell}/\tbinom{\mathsf{K}}{\ell}\right]\left[1 - \Big(1-E(\ell)\Big)\Big(1-2f_p(\ell)\Big)\right].\end{aligned}$$

By applying Markov's inequality on the random variable $X(x)$, we get $E(x) = \Pr[X(x) \geq 2\mu(x)] \leq \frac{1}{2}$. Thus, we derive for δ: $\delta \geq 1 - \left[1 - \tbinom{\mathsf{c}}{\ell}/\tbinom{\mathsf{K}}{\ell}\right]\left[\frac{1}{2} + f_p(\ell)\right]$.

Case 2): $\mathsf{c} < \ell$**.** As for the proof of Theorem 3 we split this case into two sub-cases, depending on t and c.

Case 2a): $\mathsf{c} < t$

$$\begin{aligned}&\Pr[\text{Invariant 1 is true}] \\ &\leq \Pr[u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})] + \Pr[\neg u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})] \\ &\quad \cdot \Pr[u_{1-b}.\text{sent}(r-\mathsf{c}, r)] \cdot \Pr[\neg\text{Cmpr}(\mathsf{c})]\end{aligned}$$

$$\begin{aligned}&= \Pr[Y(\ell-\mathsf{c})] + [1 - \Pr[Y(\ell-\mathsf{c})]]\Pr[Y(\mathsf{c})]\Pr[\neg\text{Cmpr}(c)] \\ &\leq [1-(1-E(\ell-\mathsf{c}))\,(1-2f_p(\ell-\mathsf{c}))] \\ &\quad + [(1-E(\ell-\mathsf{c}))\,(1-2f_p(\ell-\mathsf{c}))] \\ &\quad\quad \cdot [1-(1-E(\mathsf{c}))\,(1-2f_p(\mathsf{c}))]\left[1 - {}^1/\tbinom{\mathsf{K}}{\mathsf{c}}\right].\end{aligned}$$

Thus, for the adversarial advantage δ we derive,

$$\begin{aligned}\delta &\geq 1 - \Pr[\text{Invariant 1 is true}] \\ &\geq 1 - [1-(1-E(\ell-\mathsf{c}))\,(1-2f_p(\ell-\mathsf{c}))] \\ &\quad - [(1-E(\ell-\mathsf{c}))\,(1-2f_p(\ell-\mathsf{c}))] \\ &\quad\quad \cdot [1-(1-E(\mathsf{c}))\,(1-2f_p(\mathsf{c}))]\left[1 - \tbinom{\mathsf{c}}{\mathsf{c}}/\tbinom{\mathsf{K}}{\mathsf{c}}\right] \\ &= [(1-E(\ell-\mathsf{c}))\,(1-2f_p(\ell-\mathsf{c}))] \\ &\quad \cdot \left(1 - [1-(1-E(\mathsf{c}))\,(1-2f_p(\mathsf{c}))]\left[1 - 1/\tbinom{\mathsf{K}}{\mathsf{c}}\right]\right) \\ &\geq \left(1 - \left[\tfrac{1}{2} + f_p(\ell-\mathsf{c})\right]\right)\left(1 - \left[\tfrac{1}{2} + f_p(\mathsf{c})\right]\left[1 - 1/\tbinom{\mathsf{K}}{\mathsf{c}}\right]\right).\end{aligned}$$

We again use Markov's inequality to replace $E(x)$ by $1/2$.

Case 2b): $t \leq \mathsf{c}$

$$\begin{aligned}&\Pr[\text{Invariant 1 is true}] \\ &\leq \Pr[u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})] \cdot \Pr[\neg\text{Cmpr}(t)] \\ &\quad + \Pr[\neg u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})] \\ &\quad\quad \cdot \Pr[u_{1-b}.\text{sent}(r-\mathsf{c}, r)] \cdot \Pr[\neg\text{Cmpr}(\mathsf{c})] \\ &\leq \Pr[u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})] + \Pr[\neg u_{1-b}.\text{sent}(r-\ell, r-\mathsf{c})] \\ &\quad\quad \cdot \Pr[u_{1-b}.\text{sent}(r-\mathsf{c}, r)]\Pr[\neg\text{Cmpr}(\mathsf{c})]\end{aligned}$$

The above event expression is exactly same as the expression we had in the previous case ($t > \mathsf{c}$). Thus, the rest of the calculations and bounds are exactly same as the previous case. □

8.13 Visual 3D representations of the results

In the paper, we focus on lower-bound results for strong anonymity (or negligible δ values). However, our key Theorems 1, 3, 6 and 8 also offer lower bounds for non-negligilable δ values, which can be of interest to several AC protocols.

On our project webpage [1], we visualize these lower bounds using interactive 3D surface plots. In particular, we plot the adversarial advantage $\delta \in [0, 1]$ as a function of β and ℓ. We encourage the readers to interact with these plots to better understand our results for non-negligilbe δ values.

Here, in Figures 8.7 to 8.10, we present and analyze four snapshots of those lower bound plots for the number of users $N = 10000$. The x-axis represents latency ℓ (ranging from 0 to 200), and the y-axis bandwidth overhead β (ranging from 0.0 to 0.04). But in Figure 8.9 and Figure 8.10, the y-axis actually represents total bandwidth $p = p' + \beta$ as in Theorem 7.

A derived δ lower bound for the non-compromising adversary is also a valid lower bound for a (partially) compromising adversary. For some edge cases (e.g., when ℓ is close to N and β is close to 0), due to some approximations employed in the compromising adversaries scenario, the non-compromising adversary lower bound is actually tighter than the compromising adversaries lower bound. Therefore, in Figure 8.10, while plotting the 3D graph for a partially compromising adversary scenario, we have used the maximum of the lower bounds on δ for compromising adversary and non-compromising adversary.

In each plot, the dark blue region indicates the possibility of obtaining strong anonymity. For any point (x, y) outside those regions, strong anonymity is not possible. For example, as shown in Figure 8.7, for $\ell = 100$ the bandwidth overhead β has to be at least 0.01 to expect strong anonymity.

For the chosen c and K, the plots in Figures 8.7 and 8.8 are almost identical as the ℓ and β factors contribute more to anonymity than the compromised parties can affect it. If we instead compare Figure 8.9 with Figure 8.10, the effect of compromisation is noticeable: the dark blue region in Figure 8.10 is much smaller than that in Figure 8.9. Also, we can see a steep wall in Figure 8.10 for $\ell \leq \mathsf{c} = 20$, demonstrating that providing anonymity becomes difficult when $\ell < \mathsf{c}$; however, for $\ell > \mathsf{c}$, the effect of compromisation is less noticeable.

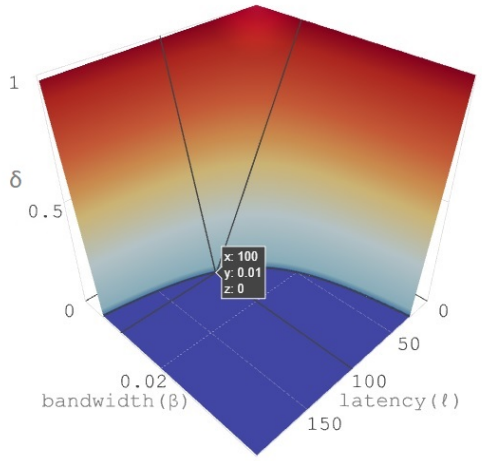


Figure 8.7: Synchronized User Distribution with Non-compromising Adversaries. $z = 1 - f_\beta(\ell)$, where $f_\beta(x) = \min(1, ((x + \beta \mathsf{N} x)/(\mathsf{N} - 1)))$.

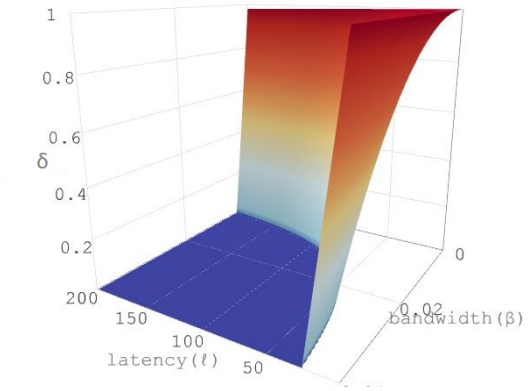


Figure 8.8: Synchronized User Distribution with Partially compromising Adversaries. Total protocol parties $\mathsf{K} = 100$, number of compromised parties $\mathsf{c} = 20$. $z = 1 - [1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}] f_\beta(\ell)$ for $\ell \leq \mathsf{c}$, $z = 1 - [1 - 1/\binom{\mathsf{K}}{\mathsf{c}}] f_\beta(\mathsf{c}) - f_\beta(\ell - \mathsf{c})$ otherwise.

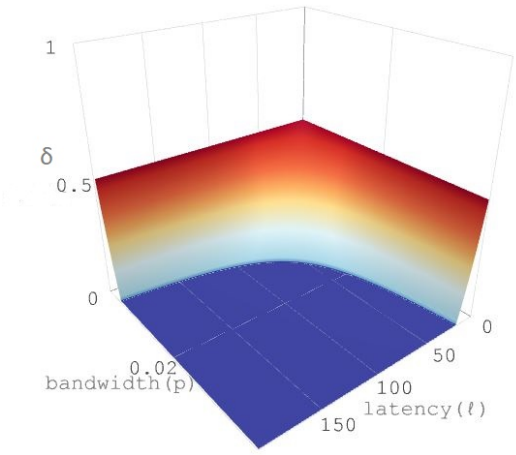


Figure 8.9: Unsynchronized User Distribution with Non-compromising Adversaries. $z = 1 - \left(\frac{1}{2} + f_p(\ell)\right)$, where $f_p(x) = \min(1/2,\ 1 - (1 - p)^x)$.

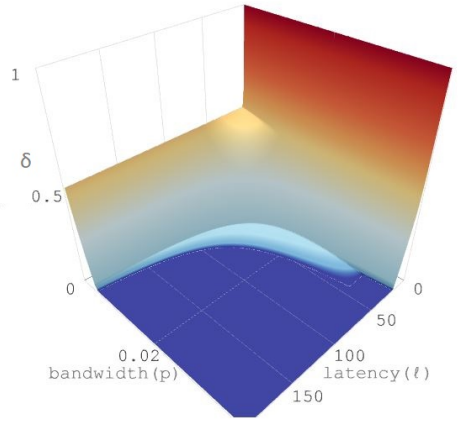


Figure 8.10: Unsynchronized User Distribution with Partially compromising Adversaries. Total number of protocol parties $\mathsf{K} = 100$, number of compromised parties $\mathsf{c} = 20$. $z' = 1 - [1 - \binom{\mathsf{c}}{\ell}/\binom{\mathsf{K}}{\ell}][\frac{1}{2} + f_p(\ell)]$ for $\ell \leq \mathsf{c}$, $z' = (1 - [1 - 1/\binom{\mathsf{K}}{\mathsf{c}}][1/2 + f_p(\mathsf{c})]) \cdot (1 - [1/2 + f_p(\ell - \mathsf{c})])$ otherwise. We set $z = \max(z', 1 - (1/2 + f_p(\ell)))$

Bibliography

[1] Anonymity Trilemma Project Webpage. Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency overhead—choose two. https://freedom.cs.purdue.edu/projects/anonymity/trilemma/.

[2] M. Backes, A. Kate, P. Manoharan, S. Meiser, and E. Mohammadi. AnoA: A Framework For Analyzing Anonymous Communication Protocols. In *Proc. 26th IEEE Computer Security Foundations Symposium (CSF 2013)*, pages 163–178, 2013.

[3] M. Backes, P. Manoharan, and E. Mohammadi. TUC: Time-sensitive and Modular Analysis of Anonymous Communication. IACR ePrint Archive Report 2013/664, 2013. http://www.infsec.cs.uni-saarland.de/ mohammadi/paper/tuc.pdf.

[4] K. S. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. C. Sicker. Low-resource routing attacks against tor. In *Proc. 6th ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 11–20, 2007.

[5] J. Camenisch and A. Lysyanskaya. A formal treatment of onion routing. In *Proc. CRYPTO 2005*, pages 169–187, 2005.

[6] S. Chakravarty, M. V. Barbera, G. Portokalidis, M. Polychronakis, and A. D. Keromytis. On the effectiveness of traffic analysis against anonymity networks using flow records. In *Proc. 15th International Conference on Passive and Active Measurement*, pages 247–257, 2014.

[7] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[8] C. Chen, D. E. Asoni, D. Barrera, G. Danezis, and A. Perrig. HORNET: High-speed onion routing at the network layer. In *Proc. ACM Conference on Computer and Communications Security (CCS)*, pages 1441–1454, 2015.

[9] H. Corrigan-Gibbs, D. Boneh, and D. Mazières. Riposte: An anonymous messaging system handling millions of users. In *Proc. 36th IEEE Symposium on Security and Privacy (S&P 2015)*, pages 321–338, 2015.

[10] H. Corrigan-Gibbs and B. Ford. Dissent: Accountable Anonymous Group Messaging. In *Proc. 17th ACM Conference on Computer and Communication Security (CCS)*, pages 340–350, 2010.

[11] H. Corrigan-Gibbs, D. I. Wolinsky, and B. Ford. Proactively Accountable Anonymous Messaging in Verdict. In *Proc. 22nd USENIX Security Symposium*, pages 147–162, 2013.

[12] G. Danezis. Statistical disclosure attacks: Traffic confirmation in open environments. In *Proc. Security and Privacy in the Age of Uncertainty, (SEC2003)*, pages 421–426, 2003.

[13] G. Danezis, C. Diaz, C. Troncoso, and B. Laurie. Drac: An architecture for anonymous low-volume communications. In *Proc. 10th Privacy Enhancing Technologies Symposium (PETS 2010)*, 2010.

[14] G. Danezis and A. Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *Proc. 6th Information Hiding Workshop (IH 2004)*, 2004.

[15] D. Das, S. Meiser, E. Mohammadi, and A. Kate. Anonymity trilemma: Strong anonymity, low bandwidth, low latency—choose two. Cryptology ePrint Archive, Report 2017/954, 2017. https://eprint.iacr.org/2017/954.

[16] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proc. 13th USENIX Security Symposium (USENIX)*, pages 303–320, 2004.

[17] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proc. 13th USENIX Security Symposium*, 2004.

[18] D. Dolev, R. Reischuk, and H. R. Strong. Early stopping in byzantine agreement. *J. ACM*, 37(4):720–741, 1990.

[19] J. Feigenbaum, A. Johnson, and P. Syverson. A probabilistic analysis of onion routing in a black-box model. In *Proc. Workshop on Privacy in the Electronic Society (WPES 2007)*, 2007.

[20] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Proc. EUROCRYPT 2004*, 2004.

[21] N. Gelernter and A. Herzberg. On the limits of provable anonymity. In *Proc. Workshop on Privacy in the Electronic Society (WPES 2013)*, pages 225–236, 2013.

[22] R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and fact-track multiparty computations with applications to threshold cryptography. In *Proc. ACM PODC*, pages 101–111, 1998.

[23] Y. Gilad and A. Herzberg. Spying in the Dark: TCP and Tor Traffic Analysis. In *Proc. 12th Privacy Enhancing Technologies Symposium (PETS 2012)*, 2012.

[24] P. Golle and A. Juels. Dining cryptographers revisited. In *Proc. of Eurocrypt 2004*, 2004.

[25] A. Hevia and D. Micciancio. An indistinguishability-based characterization of anonymous channels. In N. Borisov and I. Goldberg, editors, *Proc. Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 24–43, 2008.

[26] R. Jansen, F. Tschorsch, A. Johnson, and B. Scheuermann. The sniper attack: Anonymously deanonymizing and disabling the Tor network. In *Proc. Network and Distributed Security Symposium - NDSS '14*, 2014.

[27] K. Jensen. *Colored Petri Nets. Basic Concepts, Analysis Methods and Practical Use.*, volume 3. 1997.

[28] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson. Users get routed: Traffic correlation on tor by realistic adversaries. In *Proc. ACM SIGSAC conference on Computer & communications security 2013*, pages 337–348, 2013.

[29] D. Kesdogan, J. Egner, and R. Büschkes. Stop-and-go MIXes: Providing probabilistic anonymity in an open system. In *Proc. Information Hiding Workshop (IH 1998)*, 1998.

[30] N. Kiyavash, A. Houmansadr, and N. Borisov. Multi-flow Attacks Against Network Flow Watermarking Schemes. In *Proc. 17th USENIX Security Symposium*, 2008.

[31] L. M. Kristensen, S. Christensen, and K. Jensen. The practitioners guide to coloured petri nets. *International Journal on Software Tools for Technology Transfer (STTT)*, 2(2):98–132, 1998.

[32] A. Kwon, D. Lazar, S. Devadas, and B. Ford. Riffle: An Efficient Communication System With Strong Anonymity. In *Proc. Privacy Enhancing Technologies Symposium (PETS 2016)*, pages 115–134, 2016.

[33] S. Le Blond, D. Choffnes, W. Caldwell, P. Druschel, and N. Merritt. Herd: A Scalable, Traffic Analysis Resistant Anonymity Network for VoIP Systems. In *Proc. ACM Conference on Special Interest Group on Data Communication (SIGCOMM 2015)*, pages 639–652, 2015.

[34] S. Le Blond, D. Choffnes, W. Zhou, P. Druschel, H. Ballani, and P. Francis. Towards Efficient Traffic-analysis Resistant Anonymity Networks. In *Proc. ACM SIGCOMM 2013*, pages 303–314, 2013.

[35] M. Backes, A. Kate, P. Manoharan, S. Meiser, and E. Mohammadi. AnoA: A Framework For Analyzing Anonymous Communication Protocols. *Journal of Privacy and Confidentiality (JPC)*, 7(2)(5), 2016.

[36] P. Mittal, M. Wright, and N. Borisov. Pisces: Anonymous communication using social networks. In *Proc. 20th Network and Distributed System Security Symposium (NDSS 2013)*, 2013.

[37] S. J. Murdoch and G. Danezis. Low-cost traffic analysis of Tor. In *Proc. IEEE Symposium on Security and Privacy 2005*, 2005.

[38] L. Øverlier and P. F. Syverson. Locating Hidden Servers. In *Proc. 27th IEEE Symposium on Security and Privacy*, pages 100–114, 2006.

[39] S. Oya, C. Troncoso, and F. Pérez-González. Do dummies pay off? limits of dummy traffic protection in anonymous communications. In *Proc. 14th Privacy Enhancing Technologies Symposium (PETS 2014)*, 2014.

[40] F. Pérez-González and C. Troncoso. Understanding statistical disclosure: A least squares approach. In *Proc. 12th Privacy Enhancing Technologies Symposium (PETS 2012)*, pages 38–57, 2012.

[41] A. Piotrowska, J. Hayes, T. Elahi, S. Meiser, and G. Danezis. The loopix anonymity system. In *Proc. 26th USENIX Security Symposium*, 2017.

[42] W. Reisig. *Primer in Petri Net Design.* 1st edition, 1992.

[43] T. Ruffing, P. Moreno-Sanchez, and A. Kate. P2P Mixing and Unlinkable Bitcoin Transactions. In *Proc. 25th Annual Network & Distributed System Security Symposium (NDSS)*, 2017.

[44] A. Serjantov, R. Dingledine, and P. Syverson. From a trickle to a flood: Active attacks on several mix types. In *5th Information Hiding Workshop (IH 2002)*, pages 36–52, 2003.

[45] T. K. Srikanth and S. Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Distributed Computing*, 2(2):80–94, 1987.

[46] Y. Sun, A. Edmundson, L. Vanbever, O. Li, J. Rexford, M. Chiang, and P. Mittal. RAPTOR: Routing attacks on privacy in Tor. In *Proc. 24th USENIX Security Symposium*, 2015.

[47] The Tor Blog. One cell is enough to break Tor's anonymity. https://blog.torproject.org/blog/one-cell-enough. Accessed Nov 2017.

[48] The Tor Project. https://www.torproject.org/. Accessed in Nov 2017.

[49] J. van den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proc. 25th ACM Symposium on Operating Systems Principles (SOSP 2015)*, 2015.

[50] D. Wikström. A Universally Composable Mix-Net. In *Proc. 1st Theory of Cryptography Conference (TCC)*, pages 317–335, 2004.

[51] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson. Dissent in Numbers: Making Strong Anonymity Scale. In *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 179–182, 2012.

9. A cryptographic framework for the privacy of email ecosystems

9.1 Contribution to the PANORAMIX project

The main contribution of the University of Edinburgh to the third year for WP3 is the introduction of a security framework for the formal treatment of privacy in e-mail ecosystems. By introducing our cryptographic framework, we aim at the following objectives:

1. Enable security analysis of email ecosystems according to the standards of state-of-the art cryptography.

2. Support the embedding and taxonomy of the well-known notions of privacy (e.g. unlinkability, anonymity, unobservability, differential privacy etc., see [2, 4, 3]), by parameterising the said framework with respect to different cases of privacy leakage.

3. Formally analyse the anonymity of the PANORAMIX email ecosystem and compare with the anonymity of other existing solutions.

Our framework follows the simulation-based paradigm according to the *Universal Composability (UC)* standards put forth by Canetti [1]. In the UC model, the execution flow among interacting entities is rigorously defined while the security of the studied protocols is preserved under arbitrary compositions of protocol executions. The UC model is currently the best way to formally capture security of entities that interact over internet communications, therefore greatly suitable in the case of an email ecosystem, where clients, service providers and (potentially) mixing nodes, conctinuously interact in a complicated manner.

Building upon the UC model, we conceptualise an "ideal" email ecosystem and we enhance it by parameterising its privacy according to some *leakage function* over the execution transcript that specifies the type of information that is leaked to the (global passive) adversary. This ideal ecosystem serves as a privacy reference point that a real-world ecosystem (e.g., the PANORAMIX email ecosystem) should approximate security-wise by appearing a similar behaviour regarding execution transcripts. If this happens, then the real-world ecosystem is essentially as secure as the ideal one, with respect to this leakage function. For example, if the leakage function provides the adversary with the sender and recipient identities, then the level of privacy matches the standard notion of anonymity.

Towards this developing our cryptographic framework, modelling the ecosystem's network parameters (synchronization, message delay, channel throughput/bandwidth/capacity, etc.) is essential. We view this step, along with the part of grading various well-known notions of privacy via the leakage function above, as two interesting research contributions that may be of independent interest.

9.2 A UC framework for the privacy of email ecosystems

9.2.1 Entities and protocols of an email ecosystem

The entities that are involved in a private email "(eco)system" **E** are the following:

- The *service providers (SPs)* $\mathsf{SP}_1, \ldots, \mathsf{SP}_N$ that register users and are responsible for mailbox management and email transfer. The SPs are aware of each other and also connect to other external services.
- The *clients* $C_1, \ldots, C_n$ that wish to exchange email messages and are registered with the SPs. For simplicity, we assume that each client is linked with only one SP that manages the client's mailbox. We write $C_\ell@\mathsf{SP}_i$ to denote that C_ℓ is registered to SP_i We define the set of all clients whose mailboxes SP_i is managing as $\mathcal{C}_i := \left\{ C_\ell \;\middle|\; C_\ell@\mathsf{SP}_i \right\}$.
- The *mix-node* subsystem **MX** consisting of the mix-nodes $\mathsf{MX}_1, \ldots, \mathsf{MX}_m$ that is the core of the anonymous email routing mechanism.

An execution of an email ecosystem **E** comprises the following protocols:

- The Send protocol between client C_s and its service provider SP_i. In particular, C_s that wishes to send a message M to some client address $C_r@\mathsf{SP}_j$ authenticates to SP_i and provides it with an encoding $\mathsf{Encode}(M, C_r@\mathsf{SP}_j)$. At the end of the protocol, $\mathsf{Encode}(M, C_r@\mathsf{SP}_j)$ is at the outbox of $C_s@\mathsf{SP}_i$ managed by SP_i.
- The Route protocol that is executed among $\mathsf{SP}_1, \ldots, \mathsf{SP}_N$ and $\mathsf{MX}_1, \ldots, \mathsf{MX}_m$. Namely, the encoded message $\mathsf{Encode}(M, C_r@\mathsf{SP}_j)$ is forwarded to the **MX** subsystem, which in turn eventually delivers it to SP_j that manages the inbox of C_r.
- The $\mathsf{Receive}$ protocol between client C_r and SP_j, where C_r can retrieve (a subset of) the messages from the inbox of $C_r@\mathsf{SP}_j$ via a combination of fetch requests and push operations.

Correctness. The correctness of an email ecosystem **E** dictates that in every honest execution of $\mathsf{Send}, \mathsf{Route}$ and $\mathsf{Receive}$ protocols, C_r will obtain the message M composed by C_s encoded as $\mathsf{Encode}(M, C_r@\mathsf{SP}_j)$.

Remark 9.1. In this work, we restrict to email solutions where the client-side operations are simple and do not include complex interaction with the SPs for the execution of heavy cryptographic primitives (e.g. secure MPC). As we will explain shortly, the client-friendly approach poses some limitations on the security level that the email ecosystem can achieve.

9.2.2 Input format

We formally express an input of the ecosystem for some time moment. We write $[\![\cdot]\!]$ to denote a multiset.

Definition 1 (Email Input)**.** *Let* ***E*** *be an email ecosystem with service providers* $\mathsf{SP}_1, \ldots, \mathsf{SP}_N$ *and clients* $C_1, \ldots, C_n$*. We define the* input I_T *of* ***E*** *at global time* $\mathsf{Cl} = T$

$$I_T := \left[\!\!\left[(\mathsf{Send}, C_{s_i}@\mathsf{SP}_{S_i}, M_i, C_{r_i}@\mathsf{SP}_{R_i}) \right]\!\!\right]_{i \in K_T} \cup \left[\!\!\left[(\mathsf{Fetch}, C_{\hat{r}_j}@\mathsf{SP}_{\hat{R}_j}) \right]\!\!\right]_{j \in \hat{K}_T},$$

where $S_i, R_i, \hat{R}_j \in [N]$ *and* $s_i, r_i, \hat{r}_j, K_T, \hat{K}_T \in [n]$*. The quadruple* $(\mathsf{Send}, C_{s_i}@\mathsf{SP}_{S_i}, M_i, C_{r_i}@\mathsf{SP}_{R_i})$ *denotes that sender client* $C_{s_i}@\mathsf{SP}_{S_i}$ *composed a message* M_i *for recipient client* $C_{r_i}@\mathsf{SP}_{R_i}$*, and the pair* $(\mathsf{Fetch}, C_{\hat{r}_j}@\mathsf{SP}_{\hat{R}_j})$ *denotes that* $C_{\hat{r}_j}@\mathsf{SP}_{\hat{R}_j}$ *has made a fetch request to* $\mathsf{SP}_{\hat{R}_j}$*.*

9.2.3 A global clock functionality

The global clock functionality is parameterised by a set of parties $\mathbf{P}$, a set of functionalities $\mathbf{F}$, the UC environment $\mathcal{Z}$ and the adversary $\mathcal{A}$.

The global clock functionality $\mathcal{G}_{\mathsf{clock}}(\mathbf{P}, \mathbf{F})$.

Initialisation.

- Upon receiving $(\mathsf{sid}, \text{INIT\_CLOCK})$ from $\mathcal{Z}$, if its status is not 'execute', then
  1. It initialises the global clock variable as $\mathsf{Cl} \leftarrow 0$.
  2. It initialises the set of advanced parties as $L_{\mathsf{adv}} \leftarrow \emptyset$.
  3. It sets its status to 'execute' and sends the message '$(\mathsf{sid}, \mathsf{ready})$' to $\mathcal{Z}$.

Execution on status 'execute'.

- Upon receiving $(\mathsf{sid}, \text{ADVANCE\_CLOCK})$ from $P \in \mathbf{P} \setminus \{\mathcal{Z}, \mathcal{A}\}$, it adds P in L_{adv} and sends the message $(\mathsf{sid}, \text{ADVANCE\_CLOCK}, P)$ to $\mathcal{A}$. If $L_{\mathsf{adv}} = \mathbf{P} \cup \mathbf{F}$, then it updates as $\mathsf{Cl} \leftarrow \mathsf{Cl} + 1$ and resets $L_{\mathsf{adv}} \leftarrow \emptyset$.
- Upon receiving $(\mathsf{sid}, \text{ADVANCE\_CLOCK})$ from $F \in \mathbf{F}$, it adds F in L_{adv} and sends the message $(\mathsf{sid}, \text{ADVANCE\_CLOCK}, \mathcal{F})$ to $\mathcal{F}$. If $L_{\mathsf{adv}} = \mathbf{P} \cup \mathbf{F}$, then it updates as $\mathsf{Cl} \leftarrow \mathsf{Cl} + 1$ and resets $L_{\mathsf{adv}} \leftarrow \emptyset$.
- Upon receiving $(\mathsf{sid}, \text{READ\_CLOCK})$ from $P \in \mathbf{P} \cup \mathbf{F} \cup \{\mathcal{Z}, \mathcal{A}\}$, then it sends $(\mathsf{sid}, \text{READ\_CLOCK}, \mathsf{Cl})$ to P.

Figure 9.1: The global clock functionality $\mathcal{G}_{\mathsf{clock}}(\mathbf{P}, \mathbf{F})$ interacting with the environment $\mathcal{Z}$ and the adversary $\mathcal{A}$.

9.2.4 An ideal-world e-mail privacy functionality

Let $\mathbf{Ad}$ be the set of all valid email addresses linking the set of clients $\mathbf{C} = \{C_1, \ldots, C_n\}$ and the set of SPs $\mathbf{SP} = \{\mathsf{SP}_1, \ldots, \mathsf{SP}_N\}$. Let $\mathbf{MX} = \mathsf{MX}_1, \ldots, \mathsf{MX}_m$ be the set of mix servers. The history of an email ecosystem execution that involves the entities in $\mathbf{C}, \mathbf{SP}$ and $\mathbf{MX}$ is expressed as a list H, where each entry of H is associated with a unique pointer ptr. The leakage in each step of the execution, is expressed via a *leakage function* Leak defined as follows: (a) when given as input an execution history sequence H, it outputs some leakage string $\mathsf{Leak}(H)$; (b) when given as input a pointer ptr to some history entry of H and $\mathsf{Leak}(H)$, then $\mathsf{Leak}(\mathsf{ptr}, \mathsf{Leak}(H)$ is a bit value $1/0$ that determines whether the adversary will obtain the leakage $\mathsf{Leak}(H)$ or not. We require that during a time slot, the environment sends a message for every party, even when the party is 'sleeping', so that the clock can be advanced as described in Fig. 9.1.

The *email privacy functionality* $\mathcal{F}_{\mathsf{priv}}^{\mathsf{Leak}, \Delta_{\mathsf{net}}}(\mathbf{C}, \mathbf{SP}, \mathbf{Ad}, \mathbf{MX})$ is parameterised by the message delivery delay bound Δ_{net} and the leakage function Leak.

Initialisation

- $\mathcal{F}_{\mathsf{priv}}^{\mathsf{Leak}, \Delta_{\mathsf{net}}}(\mathbf{C}, \mathbf{SP}, \mathbf{Ad}, \mathbf{MX})$ sets its status to 'init'. It initialises the set of active entities L_{act}, the set of clock-advanced entited L_{adv}, the "history" list H, and the set of leaked entries L_{leak} as empty. For every valid address $C_r@\mathsf{SP}_j \in \mathbf{Ad}$, it initializes a list $\mathsf{Inbox}[C_r@\mathsf{SP}_j]$ as empty.

– Upon receiving $(\mathsf{sid}, \textsc{Init})$ from a party $P \in (\mathbf{C} \cup \mathbf{SP} \cup \mathbf{MX})$, if $\mathcal{F}_{\mathsf{priv}}^{\mathsf{Leak},\Delta_{\mathsf{net}}}(\mathbf{C}, \mathbf{SP}, \mathbf{Ad}, \mathbf{MX})$'s status is '$\mathsf{init}$', then it adds P to L_{act}. If $L_{\mathsf{act}} \subsetneq \mathbf{C} \cup \mathbf{SP} \cup \mathbf{MX}$, then it sends the message $(\mathsf{sid}, \mathsf{init}, P)$ to Sim. If $L_{\mathsf{act}} = \mathbf{C} \cup \mathbf{SP} \cup \mathbf{MX}$, then it sets its status to '$\mathsf{execute}$' and sends the message $(\mathsf{sid}, \mathsf{init}, P, \mathsf{ready})$ to Sim.

Execution on status '$\mathsf{execute}$'

– Upon receiving $\big(\mathsf{sid}, \textsc{Active}, @\mathsf{SP}_i\big)$ from C_ℓ, if $C_\ell @\mathsf{SP}_i \in \mathbf{Ad} \setminus L_{\mathsf{act}}$, then

1. $\mathcal{F}_{\mathsf{priv}}^{\mathsf{Leak},\Delta_{\mathsf{net}}}(\mathbf{C}, \mathbf{SP}, \mathbf{Ad}, \mathbf{MX})$ adds $\big(\mathsf{ptr}, \big(\mathsf{sid}, \mathsf{CI}, \textsc{Active}, C_\ell @\mathsf{SP}_i\big)\big)$ to H and $C_\ell @\mathsf{SP}_i$ to L_{act}.
2. It sends the message $\big(\mathsf{sid}, \textsc{Leak}(\mathsf{ptr}, H, \mathsf{Leak})\big)$ to Sim.

– Upon receiving $\big(\mathsf{sid}, \textsc{Inactive}, @\mathsf{SP}_i\big)$ from C_ℓ, if $C_\ell @\mathsf{SP}_i \in \mathbf{Ad}$, then

1. It adds $\big(\mathsf{ptr}, \big(\mathsf{sid}, \mathsf{CI}, \textsc{Inactive}, C_\ell @\mathsf{SP}_i\big)\big)$ to H and if $C_\ell @\mathsf{SP}_i \in L_{\mathsf{act}}$, then it deletes $C_\ell @\mathsf{SP}_i$ from L_{act}.
2. It sends the message $\big(\mathsf{sid}, \textsc{Leak}(\mathsf{ptr}, H, \mathsf{Leak})\big)$ to Sim.

– Upon receiving $\big(\mathsf{sid}, \textsc{Send}, \langle C_s @\mathsf{SP}_i, M, C_r @\mathsf{SP}_j\rangle\big)$ from C_s, if $C_s @\mathsf{SP}_i \in \mathbf{Ad} \cap L_{\mathsf{act}}$ and $C_r @\mathsf{SP}_j \in \mathbf{Ad}$, then

1. It adds $\big(\mathsf{ptr}, \big(\mathsf{sid}, \mathsf{CI}, \textsc{Send}, \langle C_s @\mathsf{SP}_i, M, C_r @\mathsf{SP}_j\rangle\big), \text{'}\mathsf{pending}\text{'}\big)$ to H.
2. It sends the message $\big(\mathsf{sid}, \textsc{Leak}(\mathsf{ptr}, H, \mathsf{Leak})\big)$ to Sim.

– Upon receiving $\big(\mathsf{sid}, \textsc{Allow\_Send}, \mathsf{ptr}'\big)$ from Sim, if $\mathsf{ptr}' \in L_{\mathsf{leak}}$ and ptr' refers to a history entry of the form $\big(\mathsf{sid}, \mathsf{CI}', \textsc{Send}, \langle C_{s'} @\mathsf{SP}_{i'}, M', C_{r'} @\mathsf{SP}_{j'}\rangle\big)$ with status '$\mathsf{pending}$', then it adds this entry to $\mathsf{Inbox}[C_r @\mathsf{SP}_j]$ and updates the entry's status to '$(\mathsf{sent}, \mathsf{CI})$'.

– Upon receiving $\big(\textsc{Fetch}, \mathsf{sid}, C_r @\mathsf{SP}_j\big)$ from C_r, if $C_r \in L_{\mathsf{act}}$ and $C_r @\mathsf{SP}_j \in \mathbf{Ad}$, then

1. It adds $\big(\mathsf{ptr}, \big(\mathsf{sid}, \mathsf{CI}, \textsc{Fetch}, C_r @\mathsf{SP}_j\big), \text{'}\mathsf{pending}\text{'}\big)$ to H.
2. It sends the message $\big(\mathsf{sid}, \textsc{Leak}(\mathsf{ptr}, H, \mathsf{Leak})\big)$ to Sim.

– Upon receiving $\big(\mathsf{sid}, \textsc{Allow\_Fetch}, \mathsf{ptr}', n\big)$ from Sim, if $\mathsf{ptr}' \in L_{\mathsf{leak}}$ and ptr' refers to a history entry of the form $\big(\mathsf{sid}, \mathsf{CI}', \textsc{Fetch}, C_{r'} @\mathsf{SP}_{j'}\big)$ with status '$\mathsf{pending}$', then it defines the list $\mathsf{Inbox}_n[C_r @\mathsf{SP}_j]$ that contains the first n entries of (or all if $n \geq \mathsf{Inbox}[C_r @\mathsf{SP}_j]$) and sends the message $(\mathsf{sid}, \mathsf{Inbox}_n[C_r @\mathsf{SP}_j])$ to C_r and updates the entry's status to '$(\mathsf{fetched}, n, \mathsf{CI})$'. In addition, it removes the first n entries of $\mathsf{Inbox}[C_r @\mathsf{SP}_j]$ from the list.

– Upon receiving $\big(\mathsf{sid}, \textsc{Read\_Clock}\big)$ from a party $P \in (\mathbf{C} \cup \mathbf{SP} \cup \mathbf{MX})$, then

1. It adds $\big(\mathsf{ptr}, \big(\mathsf{sid}, \mathsf{CI}, \textsc{Read\_Clock}, P\big)\big)$ to H and sends the message $\big(\mathsf{sid}, \textsc{Read\_Clock}\big)$ to $\mathcal{G}_{\mathsf{clock}}$. Upon receiving $\big(\mathsf{sid}, \textsc{Read\_Clock}, \mathsf{CI}\big)$ from $\mathcal{G}_{\mathsf{clock}}$, it sends the message $\big(\mathsf{sid}, \textsc{Read\_Clock}, \mathsf{CI}\big)$ to P.
2. It sends the message $\big(\mathsf{sid}, \textsc{Leak}(\mathsf{ptr}, H, \mathsf{Leak})\big)$ to Sim.

– Upon receiving $\big(\mathsf{sid}, \textsc{Advance\_Clock}\big)$ from a party $P \in (\mathbf{C} \cup \mathbf{SP} \cup \mathbf{MX}) \setminus L_{\mathsf{adv}}$, then

1. It adds $\big(\mathsf{ptr}, \big(\mathsf{sid}, \mathsf{CI}, \textsc{Advance\_Clock}, P\big)\big)$ to H and P in L_{adv}.
2. It sends the message $\big(\mathsf{sid}, \textsc{Leak}(\mathsf{ptr}, H, \mathsf{Leak})\big)$ to Sim.
3. If $L_{\mathsf{act}} = \mathbf{C} \cup \mathbf{SP}$, then it sets its status to '$\mathsf{advance}$' and sends the message $\big(\mathsf{sid}, \textsc{Advance\_Clock}\big)$ to $\mathcal{G}_{\mathsf{clock}}$.

Clock advancement on status 'advance'.

– Upon receiving $\big(\mathsf{sid}, \textsc{Advance\_Clock}\big)$ from $\mathcal{G}_{\mathsf{clock}}$,

1. For every history entry of the form $\big(\mathsf{sid}, \mathsf{Cl}', \textsc{Send}, \langle C_{s'}@\mathsf{SP}_{i'}, M', C_{r'}@\mathsf{SP}_{j'}\rangle\big)$ with status 'pending' such that $\mathsf{Cl} - \mathsf{Cl}' = \Delta_{\mathsf{net}}$, $\mathcal{F}_{\mathsf{priv}}^{\mathsf{Leak},\Delta_{\mathsf{net}}}(\mathbf{C}, \mathbf{SP}, \mathbf{Ad}, \mathbf{MX})$ adds this entry to $\mathsf{Inbox}[C_r@\mathsf{SP}_j]$ and updates the entry's status to '$(\mathsf{finished}, \mathsf{Cl})$'.

2. For every history entry of the form $\big(\mathsf{sid}, \mathsf{Cl}', \textsc{Fetch}, C_{r'}@\mathsf{SP}_{j'}\big)$ with status 'pending' such that $\mathsf{Cl} - \mathsf{Cl}' = \Delta_{\mathsf{net}}$, it sends the message $(\mathsf{sid}, \mathsf{Inbox}[C_r@\mathsf{SP}_j])$ to C_r, resets the list $\mathsf{Inbox}[C_r@\mathsf{SP}_j] \leftarrow \varepsilon$ and updates the entry's status to '$(\mathsf{fetched}, |\mathsf{Inbox}[C_r@\mathsf{SP}_j]|, \mathsf{Cl})$'.

3. It sends the message $\big(\mathsf{sid}, \textsc{Leak}(\mathsf{ptr}, H, \mathsf{Leak})\big)$ to Sim.

4. It sets its status to 'execute' and L_{adv} as empty, and sends the message $(\mathsf{sid}, \mathsf{clock\_advanced})$ to Sim.

> **Procedure** LEAK on input $(\mathsf{ptr}, H, \mathsf{Leak})$.
> – If $\mathsf{Leak}(\mathsf{ptr}, \mathsf{Leak}(H)) = 1$, then return $\big(\mathsf{next}, \mathsf{Leak}(H)\big)$; otherwise, return next.

We denote by $\mathrm{EXEC}^{\mathcal{F}_{\mathsf{priv}}^{\mathsf{Leak},\Delta_{\mathsf{net}}}(\mathbf{C},\mathbf{SP},\mathbf{Ad},\mathbf{MX})}_{\mathsf{Sim},\mathcal{Z},\mathcal{G}_{\mathsf{clock}}(\mathbf{C}\cup\mathbf{SP}\cup\mathbf{MX},\mathcal{Z},\mathsf{Sim})}(\lambda)$, the output of the environment $\mathcal{Z}$ in an ideal-world execution of $\mathcal{F}_{\mathsf{priv}}^{\mathsf{Leak},\Delta_{\mathsf{net}}}(\mathbf{C}, \mathbf{SP}, \mathbf{Ad}, \mathbf{MX})$ under the presence of Sim.

9.2.5 The $\big(\mathcal{G}_{\mathsf{clock}}, \mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}\big)$-hybrid world

In a real world execution of the email ecosystem $\mathbb{E}$, the clients, the SPs and the mix servers interact according to the protocol guidelines, the environment's instructions, and advance their local time by making calls to $\mathcal{G}_{\mathsf{clock}}$, which plays the role of a hybrid functionality. The message delivery is execute via the $\mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}$ described in Section. 9.1.4, the functionality that captures the notion of an authenticated channel upon which a maximum delivery delay can be imposed. the functionality is parameterised by $\mathbf{G}[\mathbb{E}]$ that denotes the network topology of $\mathbb{E}$ formalised as a graph with node set $\mathbf{C} \cup \mathbf{SP} \cup \mathbf{MX}$.

The authenticated channel functionality $\mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}(\mathbf{G}[\mathbb{E}])$.

The functionality initialises a list of pending messages L_{pend} as empty.

- Upon receiving $(\mathsf{sid}, \textsc{Channel}, M, P')$ from $P \in \mathbf{C} \cup \mathbf{SP} \cup \mathbf{MX}$, if $(P, P') \in \mathbf{G}[\mathbb{E}]$, then
  1. It sends the message $(\mathsf{sid}, \textsc{Read\_Clock})$ to $\mathcal{G}_{\mathsf{clock}}(\mathbf{C} \cup \mathbf{SP} \cup \mathbf{MX})$.
  2. Upon receiving $(\mathsf{sid}, \textsc{Read\_Clock}, \mathsf{Cl})$ to $\mathcal{G}_{\mathsf{clock}}(\mathbf{C} \cup \mathbf{SP} \cup \mathbf{MX})$, it picks a unique pointer ptr and stores the entry $\big(\mathsf{ptr}, (\mathsf{sid}, \mathsf{Cl}, \textsc{Channel}, P, M, P')\big)$ to L_{pend}.
  3. It sends the message $\big(\mathsf{ptr}, (\mathsf{sid}, \textsc{Channel}, P, M, P')\big)$ to $\mathcal{A}$.
- Upon receiving $(\mathsf{sid}, \textsc{Allow\_Channel}, \mathsf{ptr}')$ from $\mathcal{A}$, if there is an entry $\big(\mathsf{ptr}', (\mathsf{sid}, \mathsf{Cl}', \textsc{Channel}, P, M, P')\big)$ in L_{pend}, then it sends the message (sid, M, P) to P' and deletes $\big(\mathsf{ptr}', (\mathsf{sid}, \mathsf{Cl}', \textsc{Channel}, P, M, P')\big)$ from L_{pend}.

- Upon any activation from a party $P \in \mathbf{P}$ or $\mathcal{A}$ as above,
  1. It sends the message $(\mathsf{sid}, \textsc{Read\_Clock})$ to $\mathcal{G}_{\mathsf{clock}}(\mathbf{C} \cup \mathbf{SP} \cup \mathbf{MX})$.
  2. Upon receiving $(\mathsf{sid}, \textsc{Read\_Clock}, \mathsf{Cl})$ to $\mathcal{G}_{\mathsf{clock}}(\mathbf{C} \cup \mathbf{SP} \cup \mathbf{MX})$, it parses L_{pend}. For every entry $\big(\mathsf{ptr}', (\mathsf{sid}, \mathsf{Cl}', \textsc{Channel}, P, M, P')\big)$ such that $\mathsf{Cl} - \mathsf{Cl}' = \Delta_{\mathsf{net}}$, it sends the message (sid, M, P) to P' and deletes $\big(\mathsf{ptr}', (\mathsf{sid}, \mathsf{Cl}', \textsc{Channel}, P, M, P')\big)$ from L_{pend}.

Figure 9.2: The authenticated channel functionality $\mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}(\mathbf{G}[\mathbb{E}])$ interacting with the adversary $\mathcal{A}$.

We denote by $\mathrm{EXEC}^{\mathbb{E}^{\mathcal{G}_{\mathsf{clock}}}, \mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}}_{\mathcal{A}, \mathcal{Z}, \mathcal{G}_{\mathsf{clock}}(\mathbf{C} \cup \mathbf{SP} \cup \mathbf{MX}, \mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}, \mathcal{Z}, \mathcal{A})}(\lambda)$ the output of the environment $\mathcal{Z}$ in a real-world execution of $\mathbb{E}^{\mathcal{G}_{\mathsf{clock}}, \mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}}$ under the presence of $\mathcal{A}$.

9.2.6 A UC definition of e-mail privacy

The definition of a private email ecosystem is provided below.

Definition 2 (UC Email Privacy)**.** *Let λ be the security parameter and $\Delta_{\mathsf{net}}, \epsilon$ be non-negative values. Let $\mathbb{E}$ be an email ecosystem with client set $\mathbf{C} = C_1, \ldots, C_n$, service provider set $\mathbf{SP} = \mathsf{SP}_1, \ldots, \mathsf{SP}_N$, mix server set $\mathbf{MX} = \mathsf{MX}_1, \ldots, \mathsf{MX}_m$ and $\mathbf{Ad}$ be the list of the client's identities and all valid addresses. We say that $\mathbb{E}^{\mathcal{G}_{\mathsf{clock}}, \mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}}$* achieves statistical (resp. computational) ϵ-privacy with respect to leakage function $\mathsf{Leak}(\cdot)$ and message delay Δ_{net}*, if for every unbounded (resp. PPT) global passive adversary $\mathcal{A}$, there is a PPT simulator Sim such that for every PPT environment $\mathcal{Z}$, it holds that*

$$\mathrm{EXEC}^{\mathcal{F}_{\mathsf{priv}}^{\mathsf{Leak}, \Delta_{\mathsf{net}}}(\mathbf{C}, \mathbf{SP}, \mathbf{Ad}, \mathbf{MX})}_{\mathsf{Sim}, \mathcal{Z}, \mathcal{G}_{\mathsf{clock}}(\mathbf{C} \cup \mathbf{SP} \cup \mathbf{MX}, \mathcal{Z}, \mathsf{Sim})}(\lambda) \approx_{\epsilon} \mathrm{EXEC}^{\mathbb{E}^{\mathcal{G}_{\mathsf{clock}}}, \mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}}_{\mathcal{A}, \mathcal{Z}, \mathcal{G}_{\mathsf{clock}}(\mathbf{C} \cup \mathbf{SP} \cup \mathbf{MX}, \mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}, \mathcal{Z}, \mathcal{A})}(\lambda) \ .$$

Asymptotically, we say that $\mathbb{E}^{\mathcal{G}_{\mathsf{clock}}, \mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}}$ achieves statistical (resp. computational) privacy with respect to leakage function $\mathsf{Leak}(\cdot)$ and message delay Δ_{net}*, if for every unbounded (resp. PPT) global passive adversary $\mathcal{A}$, there is a PPT simulator Sim such that for every PPT environment $\mathcal{Z}$,* $\mathrm{EXEC}^{\mathcal{F}_{\mathsf{priv}}^{\mathsf{Leak}, \Delta_{\mathsf{net}}}(\mathbf{C}, \mathbf{SP}, \mathbf{Ad}, \mathbf{MX})}_{\mathsf{Sim}, \mathcal{Z}, \mathcal{G}_{\mathsf{clock}}(\mathbf{C} \cup \mathbf{SP}, \mathcal{Z}, \mathsf{Sim})}(\lambda)$ *and* $\mathrm{EXEC}^{\mathbb{E}^{\mathcal{G}_{\mathsf{clock}}}, \mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}}_{\mathcal{A}, \mathcal{Z}, \mathcal{G}_{\mathsf{clock}}(\mathbf{C} \cup \mathbf{SP} \cup \mathbf{MX}, \mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}, \mathcal{Z}, \mathcal{A})}(\lambda)$ *are statistically (resp. computationally) indistinguishable.*

9.2.7 Types of leakage functions

According to the description of $\mathcal{F}^{\mathsf{Leak},\Delta_{\mathsf{net}}}$, the history H of the execution of the email ecosystem $\mathbb{E}$, tagged by sid is a sequence of messages in one of the following formats

$$\begin{array}{cc}
\big(\mathsf{ptr},\big(\mathsf{sid},T,\textsc{Active},C_\ell@\mathsf{SP}_i\big)\big) & \big(\mathsf{ptr},\big(\mathsf{sid},T,\textsc{Inactive},C_\ell@\mathsf{SP}_i\big)\big)\\
\big(\mathsf{ptr},\big(\mathsf{sid},T,\textsc{Send},\langle C_s@\mathsf{SP}_i,M,C_r@\mathsf{SP}_j\rangle\big),\text{`pending'}\big) & \big(\mathsf{ptr},\big(\mathsf{sid},T,\textsc{Send},\langle C_s@\mathsf{SP}_i,M,C_r@\mathsf{SP}_j\rangle\big),(\text{`sent'},T')\big)\\
\big(\mathsf{ptr},\big(\mathsf{sid},T,\textsc{Fetch},C_r@\mathsf{SP}_j\big),\text{`pending'}\big) & \big(\mathsf{ptr},\big(\mathsf{sid},T,\textsc{Fetch},C_r@\mathsf{SP}_j\big),(\text{`fetched'},n,T')\big)\\
\big(\mathsf{ptr},\big(\mathsf{sid},\mathsf{Cl},\textsc{Advance_Clock},P\big)\big) & \big(\mathsf{ptr},\big(\mathsf{sid},\mathsf{Cl},\textsc{Read_Clock},P\big)\big)
\end{array}$$

where T,T' are values of the global clock Cl. Any leakage function that captures an informal notion of privacy, e.g. anonymity, unlinkability, or unobservability, should be well-defined over the set of history sequences for execution tagged by sid, denoted as $\mathcal{H}_{\mathsf{sid}}$. Before describing examples of leakage functions we define the following (multi)sets for some history $H\in\mathcal{H}_{\mathsf{sid}}$:

– The *active address set for H at time slot T*

$$\begin{aligned}\mathsf{Act}_T[H] =:\Big\{(C_\ell,\mathsf{SP}_i)\in\mathbf{C}\times\mathbf{SP}\;\Big|\;&\exists(\mathsf{ptr}',T'):\big[T'\le T\big]\wedge\\&\wedge\Big[\big(\mathsf{ptr}',\big(\mathsf{sid},T',\textsc{Active},C_\ell@\mathsf{SP}_i\big)\big)\in H\Big]\wedge\\&\wedge\Big[\forall(\mathsf{ptr}'',T''):T'\le T''\le T\Rightarrow\big(\mathsf{ptr}'',\big(\mathsf{sid},T'',\textsc{Inactive},C_\ell@\mathsf{SP}_i\big)\big)\notin H\Big]\Big\}\,.\end{aligned}$$

– The *sender set for H at time slot T*

$$\mathcal{S}_T[H]:=\Big\{C_s\in\mathcal{C}\;\Big|\;\exists(\mathsf{ptr},\mathsf{SP}_i):\big(\mathsf{ptr},\big(\mathsf{sid},T,\textsc{Send},\langle C_s@\mathsf{SP}_i,*,*\rangle\big),*\big)\in H\Big\}\,.$$

– The *sender multiset for H at time slot T*, denoted by $[\![\mathcal{S}_T]\!][H]$, is defined analogously. The difference with $\mathcal{S}_T[H]$ is that the cardinality of the SEND messages provided by C_s is attached.

– The *fetching recipient set for H at time slot T*

$$\mathcal{FR}_T[H]:=\Big\{C_r\in\mathcal{C}\;\Big|\;\exists(\mathsf{ptr},\mathsf{SP}_j):\big(\mathsf{ptr},\big(\mathsf{sid},\textsc{Fetch},\langle C_r@\mathsf{SP}_j\rangle\big),*\big)\in H\Big\}\,.$$

– The *fetching recipient multiset for H at time slot T*, denoted by $[\![\mathcal{R}]\!][H]$, is defined analogously. The difference with $\mathcal{R}_T[H]$ is that the cardinality of the FETCH messages provided by C_r is attached.

$$[\![\mathcal{FR}_T]\!][H]:=\Big\{\Big(C_r\in\mathcal{C}\big|\big\{(\mathsf{ptr},\mathsf{SP}_j)\big|\big(\mathsf{ptr},\big(\mathsf{sid},T,\textsc{Fetch},\langle C_r@\mathsf{SP}_j\rangle\big),*\big)\in H\big\}\big|\Big)\in\mathcal{C}\times\mathbb{N}\Big\}\,.$$

– The *message recipient set for H at time slot T*

$$\mathcal{MR}_T[H]:=\Big\{C_r\in\mathcal{C}\;\Big|\;\exists(\mathsf{ptr},\mathsf{SP}_j):\big(\mathsf{ptr},\big(\mathsf{sid},T,\textsc{Send},\langle *,*,C_r@\mathsf{SP}_j\rangle\big),*\big)\in H\Big\}\,.$$

– The *message recipient multiset for H at time slot T*, denoted by $[\![\mathcal{S}_T]\!][H]$, is defined analogously. The difference with $\mathcal{S}_T[H]$ is that the cardinality of the SEND messages sent to C_r is attached.

Time Intervals. Given the above notation, we can define the respective (multi)set for H and time frame $F = [T_1, T_2]$ as the union of all sets for H at intermediate time slots. For instance, the *sender set for H and time frame $F = [T_1, T_2]$*, is defined as

$$\mathcal{S}_F[H] := \bigcup_{T_1 \leq T \leq T_2} \mathcal{S}_T[H] \ .$$

Our framework enables us to cover a large number of privacy concepts by tailoring the corresponding leakage function. To demonstrate this, we will use our framework to express the leakage functions from the previous Section.

Unobservability. The unobservability propoerty suggests that nothing is leaked besides the "activity bit" of the clients. In our framework, this is formally captured as follows:

$$\mathsf{Leak}(H) = \mathsf{Act}(H)$$
$$\mathsf{Leak}(\mathsf{ptr}, H) = \begin{cases} 1, & \text{if } \big(\mathsf{ptr}, \big(\mathsf{sid}, \textsc{Advance\_Clock}\big)\big) \in H \\ 0, & \text{otherwise} \end{cases}$$

Anonymity. Our framework gives away as "minimum leakage" the identities of users who attempt to send and fetch messages. The corresponding function $\mathsf{Leak}_{\mathsf{Anon}}(H)$ operates as follows: If the last entry of H is $\big(\mathsf{ptr}, \big(\mathsf{sid}, \mathsf{Cl}, \textsc{Advance\_Clock}\big)\big)$, then output: $\mathcal{S}[H]_{\mathsf{Cl}-1,\mathsf{Cl}}, \mathcal{R}[H]_{\mathsf{Cl}-1,\mathsf{Cl}}$, otherwise output $\perp$. $\mathsf{Leak}_{\mathsf{Anon}}(\mathsf{ptr}, \mathsf{Leak}_{\mathsf{Anon}}(H))$ returns 0 if $\mathsf{Leak}_{\mathsf{Anon}}(H)$ is $\perp$, and 1 otherwise.

Weak Anonymity. Correspondingly, weak anonymity reveals message cardinalities in addition to identities of message senders and recipients. The corresponding function $\mathsf{Leak}_{\mathsf{WAnon}}(H)$ operates as follows: If the last entry of H is $\big(\mathsf{ptr}, \big(\mathsf{sid}, \mathsf{Cl}, \textsc{Advance\_Clock}\big)\big)$, then output: $[\![\mathcal{S}]\!][H]_{\mathsf{Cl}-1,\mathsf{Cl}}, [\![\mathcal{MR}]\!][H]_{\mathsf{Cl}-1,\mathsf{Cl}}$, otherwise output $\perp$. $\mathsf{Leak}_{\mathsf{WAnon}}(\mathsf{ptr}, \mathsf{Leak}_{\mathsf{WAnon}}(H))$ returns 0 if $\mathsf{Leak}_{\mathsf{WAnon}}(H)$ is $\perp$, and 1 otherwise.

9.2.8 Example case study: The complete communication solution

The complete communication email ecosystem $\mathbb{E}_{\mathsf{comp}}$ operates in rounds and under a known message delivery delay bound Δ_{net}. The ecosystem utilises a public key encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. For the system's security and liveness, we require that at the k-th round, (i) clients provide messages to their SPs during the time interval $[(k-1)T + 2(k-1)\Delta_{\mathsf{net}}), kT + 2(k-1)\Delta_{\mathsf{net}}]$, and (ii) the round is finalised during the time interval $[kT + (2k-1)\Delta_{\mathsf{net}}), kT + 2k\Delta_{\mathsf{net}}]$, where the SPs interact with each other and send messages to their assigned clients.

Furthermore, throughout the description we assume that the following hold:

(a). The execution begins at global time $\mathsf{Cl} = 0$.

(b). All communication is executed via $\mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}(\mathbf{G}[\mathbb{E}])$ as described in Fig. 9.2.

(c). All entities are aware of each others' public keys. By, $\mathsf{Enc}_{[X]}(Y)$ we denote the encryption of Y under X's public key.

(d). All ciphertexts are of the same length.

(e). All computations require one unit slot.

(f). Each client $C_1, \ldots, C_n$ is assigned to exactly one address, so there are exactly n valid addresses in total.

(g). Each client sends at most one message per round.

Description

The email ecosystem $\mathbb{E}_{\mathsf{comp}}^{\mathcal{G}_{\mathsf{clock}}, \mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}}$ with client set $\mathbf{C} = C_1, \ldots, C_n$, service provider set $\mathbf{SP} = \mathsf{SP}_1, \ldots, \mathsf{SP}_N$ and valid address set $\mathbf{Ad}$ (and no mix servers) has network topology

$$\mathbf{G}[\mathbb{E}] := \big(\mathbf{C} \cup \mathbf{SP}, \{(C_\ell, \mathsf{SP}_i) \mid C_\ell @ \mathsf{SP}_i \in \mathbf{Ad}\} \cup \{(\mathsf{SP}_i, \mathsf{SP}_j) \mid (i, j \in [N]) \wedge (i \neq j)\big) \ .$$

The real-world execution for $\mathbb{E}_{\mathsf{comp}}$ is done according the following steps:

- *Prior to the execution start* (e.g. at least Δ_{net} time before $\mathsf{Cl} = 0$),
  - On input $\big(\mathsf{sid}, \textsc{Init}\big)$, a party $P \in \mathbf{C} \cup \mathbf{SP}$ that is not yet initialised, runs $\mathsf{Gen}(1^\lambda)$ to generate a pair of a private and a public key pair $(\mathsf{sk}_P, \mathsf{pk}_P)$. Then, it broadcasts the message $\big(\mathsf{sid}, (\mathsf{init}, \mathsf{pk}_P), P\big)$ to all clients and SPs by sending $\big(\mathsf{sid}, (\mathsf{init}, \mathsf{pk}_P), P'\big)$ to $\mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}(\mathbf{G}[\mathbb{E}])$, for every $P' \in \mathbf{C} \cup \mathbf{SP} \setminus \{P\}$.
  - When SP_i has received $\big(\mathsf{sid}, (\mathsf{init}, \mathsf{pk}_{\mathsf{SP}_j}, \mathsf{SP}_j)$ for every $i \in [N] \backslash \{j\}$, then begins the engagement in the email message exchange with its assigned clients and the other SPs.
- *During round k and for every* $\mathsf{Cl} \in [(k-1)T + 2(k-1)\Delta_{\mathsf{net}}), kT + 2(k-1)\Delta_{\mathsf{net}}]$,
  - On input $\big(\mathsf{sid}, \textsc{Active}, \mathsf{SP}_i\big)$, if C_ℓ is not logged in, then she logs in by sending the message $\big(\mathsf{sid}, \mathsf{Enc}_{[\mathsf{SP}_i]}(\textsc{Active}, C_\ell)\big)$ to SP_i, where $C_\ell @ \mathsf{SP}_i$ is her valid address (i.e. in $\mathbf{Ad}$). Upon receiving and decrypting as $\big(\mathsf{sid}, \textsc{Active}, C_\ell\big)$, SP_i checks that $C_\ell @ \mathsf{SP}_i \in \mathbf{Ad}$ and if so, then it adds C_ℓ to its set of active users L_{act}^i.
  - On input $\big(\mathsf{sid}, \textsc{Inactive}, \mathsf{SP}_i\big)$, if C_ℓ is logged in, then she logs out by sending the message $\big(\mathsf{sid}, \mathsf{Enc}_{[\mathsf{SP}_i]}(\textsc{Inactive}, C_\ell)\big)$ to SP_i, where $C_\ell @ \mathsf{SP}_i \in \mathbf{Ad}$. Upon receiving and decrypting as $\big(\mathsf{sid}, \textsc{Active}, C_\ell\big)$, SP_i checks that $C_\ell @ \mathsf{SP}_i \in \mathbf{Ad}$ and if so, then it removes C_ℓ from its set of active users L_{act}^i.
  - On input $\big(\mathsf{sid}, \textsc{Send}, \langle C_s @ \mathsf{SP}_i, M, C_r @ \mathsf{SP}_j \rangle\big)$, if C_s is logged in to SP_i, then she sends the message $\big(\mathsf{sid}, \mathsf{Enc}_{[\mathsf{SP}_i]}\big(C_s @ \mathsf{SP}_i, \mathsf{Enc}_{[\mathsf{SP}_j]}\big(C_r @ \mathsf{SP}_j, \mathsf{Enc}_{[C_r]}(M)\big)\big)\big)$ to SP_i which checks that $C_s @ \mathsf{SP}_i \in \mathbf{Ad}$ and if so, then it decrypts and adds $\big(\mathsf{sid}, C_s @ \mathsf{SP}_i, \mathsf{Enc}_{[\mathsf{SP}_j]}\big(C_r @ \mathsf{SP}_j, \mathsf{Enc}_{[C_r]}(M)\big)\big)$ to its set of messages pending to be sent, denoted by L_{send}^i.
  - On input $\big(\mathsf{sid}, \textsc{Fetch}, C_r @ \mathsf{SP}_j\big)$, if C_r is logged in to SP_j, it sends the message $\big(\mathsf{sid}, C_r @ \mathsf{SP}_j, \mathsf{Enc}_{[\mathsf{SP}_j]}(\textsc{Fetch})\big)$ to SP_j which, if $C_r @ \mathsf{SP}_j$ is a valid address, it decrypts and adds $\mathsf{Inbox}[C_r @ \mathsf{SP}_j]$ to its set of inboxes which messages are pending to be pushed, denoted by L_{push}^i.
  - On input $\big(\mathsf{sid}, \textsc{Advance\_Clock}\big)$, the entity $P \in \mathbf{C} \cup \mathbf{SP}$ sends the message $\big(\mathsf{sid}, \textsc{Advance\_Clock}\big)$ to $\mathcal{G}_{\mathsf{clock}}$. In addition, if P is a client C_ℓ that is logged in to SP_i but has not received an input at global time Cl, then it sends a dummy message $\big(\mathsf{sid}, \mathsf{Enc}_{[\mathsf{SP}_i]}(0)\big)$ to SP_i.
  - On input $\big(\mathsf{sid}, \textsc{Read\_Clock}\big)$, the entity $P \in \mathbf{C} \cup \mathbf{SP}$ sends the message $\big(\mathsf{sid}, \textsc{Read\_Clock}\big)$ to $\mathcal{G}_{\mathsf{clock}}$. Upon receiving $\big(\mathsf{sid}, \textsc{Read\_Clock}, \mathsf{Cl}\big)$ from $\mathcal{G}_{\mathsf{clock}}$, P stores Cl as its local time and forwards the message $\big(\mathsf{sid}, \textsc{Read\_Clock}, \mathsf{Cl}\big)$ to the environment.

- *During round k and for every $T' \in [kT + (2k-1)\Delta_{\mathsf{net}}), kT + 2k\Delta_{\mathsf{net}}]$,*

  1. For every address $C_s@\mathsf{SP}_i$, if there is a message $\big(\mathsf{sid}, C_s@\mathsf{SP}_i, \mathsf{Enc}_{[\mathsf{SP}_j]}\big(C_r@\mathsf{SP}_j, \mathsf{Enc}_{[C_r]}(M)\big)\big)$ in L^i_{send}, then SP_i broadcasts $\big(\mathsf{sid}, C_s@\mathsf{SP}_i, \mathsf{Enc}_{[\mathsf{SP}_j]}\big(C_r@\mathsf{SP}_j, \mathsf{Enc}_{[C_r]}(M)\big)\big)$ to all SPs and removes the message from L^i_{send}. If there is no such message for $C_s@\mathsf{SP}_i$ but $C_s \in L^i_{\mathsf{act}}$, then SP_i broadcasts a dummy message $\big(\mathsf{sid}, \mathsf{Enc}_{[\mathsf{SP}_i]}\big(0, \mathsf{Enc}_{[\mathsf{SP}_i]}(0)\big)\big)$ under its own key.

  2. Upon receiving a message $(\mathsf{sid}, \mathsf{Enc}_{[\mathsf{SP}_j]}(\cdot,\cdot))$ from some SP, SP_j checks whether E_2 is a ciphertext under its public key that decrypts as a valid address $C_r@\mathsf{SP}_j$ along with a ciphertext E. If so, then it adds E to $\mathsf{Inbox}[C_r@\mathsf{SP}_j]$.

  3. For every address $C_r@\mathsf{SP}_j$, if $\mathsf{Inbox}[C_r@\mathsf{SP}_j] \in L^i_{\mathsf{push}}$, then SP_j forwards all messages $E_{r,1}, \ldots E_{r,n_r}$ in $\mathsf{Inbox}[C_r@\mathsf{SP}_j]$ to C_r along with $n - n_j$ dummy ciphertexts under C_r's public key, empties $\mathsf{Inbox}[C_r@\mathsf{SP}_j]$ and removes it from L^i_{push}. If $\mathsf{Inbox}[C_r@\mathsf{SP}_j] \notin L^i_{\mathsf{push}}$ but $C_r \in L^i_{\mathsf{act}}$, then SP_j forwards n dummy ciphertexts to C_r. In any case, SP_j sends a message of the form $(\mathsf{sid}, E_{r,1}, \ldots, E_{r,n})$ to C_r, if C_r is active.

  4. Upon receiving $(\mathsf{sid}, E_{r,1}, \ldots, E_{r,n})$ from SP_j and if C_r has sent a $\big(\mathsf{sid}, \textsc{Fetch}, C_r@\mathsf{SP}_j\big)$ request, C_r decrypts all ciphertexts and stores the ones that are not dummy, i.e. the correspond to actual mail messages.

Extensions. The aforementioned conditions (a)-(f), where assumed to simplify analysis. We note that this does not harm generality; namely, (a),(b) and (c) guarantee secure end-to-end communication between entities, (d) can be generalised to a computation upper bound after which the next round begins, whereas removing (e) and (f) could be addressed by adding a suitable amount of dummy traffic, specified by the upper bound on the number of addresses per client and messages per client in each round.

Security

We prove that $\mathbb{E}_{\mathsf{comp}}$ achieves unobservability as formally expressed in Subsection 9.1.7. We consider that leakage happens in a "clock-based" manner, i.e. the ideal adversary obtains leakage only during clock advancement stage. This assumption is natural for global passive adversaries, as whatever happens within the discretised time slot can be consider beyond the adversary's observational sensitivity. For the privacy of $\mathbb{E}_{\mathsf{comp}}$, we require that the underlying public key encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ satisfies the properties of (i) *multiple challenge IND-CCA (m-IND-CCA) security*, which is equivalent to standard IND-CCA security (up to negligible error). For completeness, we recall the definitions of m-IND-CCA below.

Definition 3 (m-IND-CCA security)**.** *We say that the public key encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ achieves m-IND-CCA security, if for every PPT adversary $\mathcal{B}$, the probability that $\mathcal{B}$ wins the following game is no more than $1/2 + \mathsf{negl}(\lambda)$.*

The m-IND-CCA game $\mathcal{G}^{\mathcal{E},\mathcal{B}}_{\mathrm{m-IND-CCA}}(1^\lambda)$.

- The challenger runs $\mathsf{Gen}(1^\lambda)$ and obtains a pair of a secret key sk and a public key pk. It chooses a random bit $b \in \{0,1\}$. It provides $\mathcal{B}$ with pk.
- The adversary $\mathcal{B}$ may send (polynomially many) challenge queries of the form (M_0, M_1) to the challenger which responds with $\mathsf{Enc}(\mathsf{pk}, M_b)$.
- The adversary $\mathcal{B}$ may make (polynomially many) ciphertext decryption queries of the form C to the challenger which responds with $\mathsf{Dec}(\mathsf{sk}, C)$, under the restriction that C is not a response to some previous challenge query.
- The game returns 1 iff $\mathcal{B}$ outputs a bit b^* that matches b.

We prove the privacy of $\mathbb{E}_{\mathsf{comp}}$ in the following theorem.

Theorem 1. Let $\mathbb{E}_{\mathsf{comp}}$ with clients $\mathbf{C} = \{C_1, \ldots, C_n\}$ and service providers $\mathbf{SP} = \mathsf{SP}_1, \ldots, \mathsf{SP}_N$ be implemented over an m-IND-CCA secure encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Then, there is a negligible function $\epsilon(\cdot)$ such that $\mathbb{E}^{\mathcal{G}_{\mathsf{clock}}, \mathcal{F}^{\Delta_{\mathsf{net}}}_{\mathsf{auth}}}_{\mathsf{comp}}$ achieves computational ϵ-privacy with respect to leakage function

$$\mathsf{Leak}(H) = \mathsf{Act}(H)$$
$$\mathsf{Leak}(\mathsf{ptr}, H) = \begin{cases} 1, & \text{if } \big(\mathsf{ptr}, \big(\mathsf{sid}, \textsc{Advance\_Clock}\big)\big) \in H \\ 0, & \text{otherwise} \end{cases}$$

and message delay Δ_{net}, against any global passive PPT adversary $\mathcal{A}$.

Proof. Let $\mathcal{A}$ be a global passive PPT adversary against $\mathbb{E}^{\mathcal{G}_{\mathsf{clock}}, \mathcal{F}^{\Delta_{\mathsf{net}}}_{\mathsf{auth}}}_{\mathsf{comp}}$. We construct an ideal adversary Sim for $\mathcal{A}$ such that given $\mathbf{C}, \mathbf{SP}, \mathbf{Ad}$ and for any environment $\mathcal{Z}$ operates as follows:

– Upon receiving a message $\big(\mathsf{sid}, X\big)$ from $\mathcal{Z}$, it forwards $\big(\mathsf{sid}, X\big)$ playing the role of a simulated environment.

– Upon receiving a message $\big(\mathsf{sid}, M\big)$ from $\mathcal{A}$ on behalf of the environment, it forwards $\big(\mathsf{sid}, M\big)$ to $\mathcal{Z}$.

– Upon receiving $\big(\mathsf{sid}, \mathsf{init}, P\big)$ or $\big(\mathsf{sid}, \mathsf{init}, P, \mathsf{ready}\big)$ from $\mathcal{F}^{\mathsf{Leak}, \Delta_{\mathsf{net}}}_{\mathsf{priv}}(\mathbf{C}, \mathbf{SP}, \mathbf{Ad})$, it runs $\mathsf{Gen}(1^\lambda)$ on behalf of $P \in \mathbf{C} \cup \mathbf{SP}$, to generate a pair of a private and a public key pair $(\mathsf{sk}_P, \mathsf{pk}_P)$. Then, it broadcasts the message $\big(\mathsf{sid}, (\mathsf{init}, \mathsf{pk}_P), P\big)$, playing the role of $\mathcal{F}^{\Delta_{\mathsf{net}}}_{\mathsf{auth}}(\mathbf{G}[\mathbb{E}_{\mathsf{comp}}])$.Observe that since $\mathcal{A}$ is global and passive, the execution will always initiate upon $\mathcal{Z}$'s request.

– Upon receiving $\big(\mathsf{sid}, \textsc{Advance\_Clock}, P\big)$ from some entity $P \in \mathbf{C} \cup \mathbf{SP}$ from $\mathcal{G}_{\mathsf{clock}}$, it sends $\big(\mathsf{sid}, \textsc{Read\_Clock}\big)$ to $\mathcal{G}_{\mathsf{clock}}$. Upon receiving $\big(\mathsf{sid}, \textsc{Read\_Clock}, \mathsf{Cl}\big)$ from $\mathcal{G}_{\mathsf{clock}}$, it stores Cl as its local time.

– Upon receiving $\big(\mathsf{sid}, \textsc{Read\_Clock}\big)$ from $\mathcal{A}$, it forwards $\big(\mathsf{sid}, \textsc{Read\_Clock}\big)$ to $\mathcal{G}_{\mathsf{clock}}$ and returns to $\mathcal{A}$ the response $\big(\mathsf{sid}, \textsc{Read\_Clock}, \mathsf{Cl}\big)$ it received from $\mathcal{G}_{\mathsf{clock}}$.

– Upon receiving $(\mathsf{sid}, \mathsf{next}, \mathsf{Act}[H])$ during time Cl from $\mathcal{F}_{\mathsf{priv}}^{\mathsf{Leak},\Delta_{\mathsf{net}}}(\mathbf{C}, \mathbf{SP}, \mathbf{Ad})$, which according to the definition of Leak happens only after $\mathcal{G}_{\mathsf{clock}}$ has forwarded $\big(\mathsf{sid}, \textsc{Advance\_Clock}\big)$ to $\mathcal{F}_{\mathsf{priv}}^{\mathsf{Leak},\Delta_{\mathsf{net}}}(\mathbf{C}, \mathbf{SP}, \mathbf{Ad})$, it (i) sets of the simulated active clients/addresses for H at time Cl as $\mathsf{Act}_{\mathsf{Cl}}[\tilde{H}] \leftarrow \mathsf{Act}[H]$, and (ii) sends $\big(\mathsf{sid}, \textsc{Read\_Clock}\big)$ to $\mathcal{G}_{\mathsf{clock}}$. Upon receiving $\big(\mathsf{sid}, \textsc{Read\_Clock}, \mathsf{Cl}\big)$ from $\mathcal{G}_{\mathsf{clock}}$, it stores Cl as its local time. Then, for every round k it operates as follows:

  ◦ If $\mathsf{Cl} \in [(k-1)T + 2(k-1)\Delta_{\mathsf{net}}, kT + 2(k-1)\Delta_{\mathsf{net}}]$, then Sim simulates the messages sent from the clients to their SPs. Namely, for every client C_ℓ that has either remained or turned active at Cl (i.e. $C_\ell \in \mathsf{Act}_{\mathsf{Cl}}[\tilde{H}]$) and is logged in as $C_\ell@\mathsf{SP}_i$, Sim sends a dummy message $\big(\mathsf{sid}, \textsc{Channel}, \mathsf{Enc}_{[\mathsf{SP}_i]}(0), \mathsf{SP}_i\big)$ to the simulated $\mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}$ that forwards $\big(\mathsf{ptr}, \big(\mathsf{sid}, \textsc{Channel}, C_\ell, \mathsf{Enc}_{[\mathsf{SP}_i]}(0), \mathsf{SP}_i\big)\big)$ to $\mathcal{A}$, where ptr is a unique pointer.

  ◦ If $\mathsf{Cl} = (k-1)T + (2k-1)\Delta_{\mathsf{net}}$, then Sim simulates the messages broadcast among the SPs. Observe that since

$$\big(kT + (2k-1)\Delta_{\mathsf{net}}\big) - \big(kT + 2(k-1)\Delta_{\mathsf{net}}\big) = \Delta_{\mathsf{net}},$$

  for each C_ℓ assigned to SP_i that has sent a message $\big(\mathsf{sid}, \textsc{Channel}, \mathsf{Enc}_{[\mathsf{SP}_i]}(0), \mathsf{SP}_i\big)$ to $\mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}$ at time $\mathsf{Cl}' \in [(k-1)T+2(k-1)\Delta_{\mathsf{net}}), kT+2(k-1)\Delta_{\mathsf{net}}]$, it holds that SP_i will receive the message $\big(\mathsf{sid}, \mathsf{Enc}_{[\mathsf{SP}_i]}(0), C_\ell\big)$ before Cl. Therefore, Sim can simulate the broadcast among the SPs as follows: on behalf of each SP_i and for every C_ℓ s.t. SP_i has obtained a message $\big(\mathsf{sid}, \mathsf{Enc}_{[\mathsf{SP}_i]}(0), C_\ell\big)$ (equivalently, $C_\ell@\mathsf{SP}_i \in \mathsf{Act}_{\mathsf{Cl}'}[\tilde{H}]$ for $\mathsf{Cl}' = kT + 2(k-1)\Delta_{\mathsf{net}}$) and for every $\mathsf{SP}_j \neq \mathsf{SP}_i$, it sends the message $\big(\mathsf{sid}, \textsc{Channel}, \mathsf{Enc}_{[\mathsf{SP}_i]}(0), \mathsf{SP}_j\big)$ to $\mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}$ at Cl.

  ◦ If $\mathsf{Cl} = (k-1)T + 2k\Delta_{\mathsf{net}}$, then Sim simulates the messages sent by the SPs to their clients. Similarly as above, the delay bound Δ_{net} guarantees that during $[(k-1)T+(2k-1)\Delta_{\mathsf{net}}, kT + 2k\Delta_{\mathsf{net}}]$, for every active address $C_s@\mathsf{SP}_i$, every $\mathsf{SP}_j \neq \mathsf{SP}_i$ will receive a dummy message $\big(\mathsf{sid}, \mathsf{Enc}_{[\mathsf{SP}_i]}(0), \mathsf{SP}_i\big)$. Therefore, Sim can simulate what SP_j sends to its active assigned clients by sending a dummy vector of n ciphertexts $(\mathsf{sid}, \mathsf{Enc}_{[C_r]}(0), \ldots, \mathsf{Enc}_{[C_r]}(0))$ to every C_r s.t. $C_r@\mathsf{SP}_j \in \mathsf{Act}_{\mathsf{Cl}'}[\tilde{H}]$ for $\mathsf{Cl}' = kT + 2(k-1)\Delta_{\mathsf{net}}$.

We prove the privacy of $\mathbb{E}_{\mathsf{comp}}^{\mathcal{G}_{\mathsf{clock}},\mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}}$ via a reduction to the m-IND-CCA security of the underlying public key encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, which are assumed in the theorem's statement. Our reduction works as follows: Let $\mathcal{A}$ be a real-world adversary and $\mathcal{Z}$ be an environment. First, we order the clients and servers as parties $P_1, \ldots, P_{n+N}$. Then, we construct a sequence of "hybrid" m-IND-CCA adversaries $\mathcal{B}_1, \ldots, \mathcal{B}_{n+N}$, where $\mathcal{B}_{j^*}$ executes the following steps:

1. It receives a public key pk from the m-IND-CCA challenger.

2. It generates the parties $P_1, \ldots, P_{n+N}$ and simulates an execution of $\mathbb{E}_{\mathsf{comp}}^{\mathcal{G}_{\mathsf{clock}},\mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}}$ conducted by $\mathcal{Z}$ and under the presence of $\mathcal{A}$, also playing the role of $\mathcal{G}_{\mathsf{clock}}, \mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}$. The simulation differs from an actual execution as shown below:

   (a) Upon initialization of a party P_j: if $P_j \neq P_{j^*}$, then $\mathcal{B}_{j^*}$ honestly generates a fresh private-public key pair $(\mathsf{sk}_j, \mathsf{pk}_j)$. If $P_j = P_{j^*}$, then it sets $\mathsf{pk}_{j^*} := \mathsf{pk}$.

   (b) When a party P_i must send an encrypted message M to P_j: if $j < j^*$, then $\mathcal{B}_{j^*}$ sends an encryption of M under pk_j; if $j = j^*$, then it sends a challenge pair $(M_0, M_1) := (0, M)$ to the m-IND-CCA challenger; if $j > j^*$, then it sends an encryption of 0 under pk_j.

(c) Since all parties are honest, $\mathcal{B}_{j^*}$ is completely aware of the plaintext-ciphertext correspondence. Therefore, when P_i encrypts M under P_j's public key to a ciphertext C, $\mathcal{B}_{j^*}$ runs P_j as if the latter had decrypted C as M.

3. It returns the output of $\mathcal{Z}$.

Given the description of $\mathcal{B}_{j^*}$, $j^* = 1, \ldots, n+N$, we make the following observations:

- The hybrid step: for every $1 \leq j^* < n+N$, the adversaries $\mathcal{B}_{j^*}$ and $\mathcal{B}_{j^*+1}$ have the same behaviour regarding the parties P_j, where $j \neq j^*, j^*+1$. In addition, if the m-IND-CCA challenge bit b is 1 then $\mathcal{B}_{j^*}$ respects encryption of P_j^* and replaces with 0 any plaintext intended for P_{j^*+1}, which is extactly the behaviour of $\mathcal{B}_{j^*+1}$, if $b = 0$. Therefore, it holds that

$$\Pr\left[\mathcal{B}_j^* = 1 \mid b = 1\right] = \Pr\left[\mathcal{B}_{j^*+1} = 1 \mid b = 0\right] . \tag{9.1}$$

- For the limit case of $j^* = n+N$, we observe that if $b = 1$, then $\mathcal{B}_{n+N}$ executes real-world communication respecting the environments' instructions and inputs. Thus, we have that

$$\Pr\left[\mathcal{B}_{n+N} = 1 \mid b = 1\right] = \mathrm{EXEC}^{\mathbb{E}^{\mathcal{G}_{\mathsf{clock}}}, \mathcal{F}_{\mathsf{auth}}^{\Delta_{\mathsf{net}}}}_{\mathcal{A},\mathcal{Z},\mathcal{G}_{\mathsf{clock}}(\mathbf{C} \cup \mathbf{SP},\mathcal{Z},\mathcal{A})}(\lambda) . \tag{9.2}$$

- For the limit case of $j^* = 1$, we observe that if $b = 0$, then $\mathcal{B}_1$ replaces all real-world communication with encryptions of 0, exactly as Sim does in its simulation. Thus, we have that

$$\Pr\left[\mathcal{B}_1 = 1 \mid b = 0\right] = \mathrm{EXEC}^{\mathcal{F}_{\mathsf{priv}}^{\mathsf{Leak},\Delta_{\mathsf{net}}}(\mathbf{C},\mathbf{SP},\mathbf{Ad})}_{\mathsf{Sim},\mathcal{Z},\mathcal{G}_{\mathsf{clock}}(\mathbf{C} \cup \mathbf{SP},\mathcal{Z},\mathsf{Sim})}(\lambda) . \tag{9.3}$$

Next, by the m-IND-CCA security of $\mathcal{E}$, we have that for every $j^* \in [n+N]$, it holds that

$$\begin{aligned}
&\Big|\Pr\left[\mathcal{B}_{j^*} = 1 \mid b = 1\right] - \Pr\left[\mathcal{B}_{j^*} = 1 \mid b = 0\right]\Big| = \\
=&\Big|\Pr\left[\mathcal{B}_{j^*} = 1 \mid b = 1\right] - \big(1 - \Pr\left[\mathcal{B}_{j^*} = 0 \mid b = 0\right]\big)\Big| \leq \\
\leq&\Big|2 \cdot \Pr\left[(\mathcal{B}_{j^*} = 1) \wedge (b = 1)\right] + 2 \cdot \Pr\left[(\mathcal{B}_{j^*} = 0) \wedge (b = 0)\right] - 1\Big| = \\
=&\Big|2 \cdot \Pr\left[\mathcal{G}^{\mathcal{E},\mathcal{B}_{j^*}}_{\mathrm{m-IND-CCA}}(1^\lambda) = 1\right] - 1\Big| \leq \Big|2 \cdot \big(1/2 + \mathsf{negl}(\lambda)\big) - 1\Big| = \mathsf{negl}(\lambda) .
\end{aligned} \tag{9.4}$$

Finally, by Eq. (9.1),(9.2),(9.3), and (9.4), we get that

$$\begin{aligned}
&\Big|\mathrm{EXEC}^{\mathcal{F}_{\mathsf{priv}}^{\mathsf{Leak},\Delta_{\mathsf{net}}}(\mathbf{C},\mathbf{SP},\mathbf{Ad})}_{\mathsf{Sim},\mathcal{Z},\mathcal{G}_{\mathsf{clock}}(\mathbf{C} \cup \mathbf{SP},\mathcal{Z},\mathsf{Sim})}(\lambda) - \mathrm{EXEC}^{\mathcal{F}_{\mathsf{priv}}^{\mathsf{Leak},\Delta_{\mathsf{net}}}(\mathbf{C},\mathbf{SP},\mathbf{Ad})}_{\mathsf{Sim},\mathcal{Z},\mathcal{G}_{\mathsf{clock}}(\mathbf{C} \cup \mathbf{SP},\mathcal{Z},\mathsf{Sim})}(\lambda)\Big| = \\
=&\Big|\Pr\left[\mathcal{B}_1 = 1 \mid b = 0\right] - \Pr\left[\mathcal{B}_{n+N} = 1 \mid b = 1\right]\Big| = \\
=&\Big|\Pr\left[\mathcal{B}_1 = 1 \mid b = 0\right] - \sum_{j^*=1}^{n+N-1} \Pr\left[\mathcal{B}_{j^*} = 1 \mid b = 1\right] + \sum_{j^*=2}^{n+N} \Pr\left[\mathcal{B}_{j^*} = 1 \mid b = 0\right] - \Pr\left[\mathcal{B}_{n+N} = 1 \mid b = 1\right]\Big| = \\
=&\Big|\sum_{j^*=1}^{n+N} \Big(\Pr\left[\mathcal{B}_{j^*} = 1 \mid b = 1\right] - \Pr\left[\mathcal{B}_{j^*} = 1 \mid b = 0\right]\Big)\Big| \leq \\
\leq&\sum_{j^*=1}^{n+N} \Big|\Pr\left[\mathcal{B}_{j^*} = 1 \mid b = 1\right] - \Pr\left[\mathcal{B}_{j^*} = 1 \mid b = 0\right]\Big| = \mathsf{negl}(\lambda)
\end{aligned}$$

which completes the proof.

□

9.2.9 Framework extension: towards a complete ideal network functionality

As ongoing work of our framework and supplementary material to this deliverable, we propose a network functionality that captures a big number of network parameters, such as message delay, channel bandwidth and throughput and synchronisation loss. By formalising communication via this functionality, we will be able to capture a wide spectrum of privacy countermeasures that exist in the literature.

Let $\mathcal{E}$ be the set of entities (SPs, clients and mix-nodes) of $\mathbf{E}$. We assume that the email ecosystem $\mathbf{E}$ runs continuously in time units $T = 0, 1, 2, 3, \ldots$ called *slots* under the existence of a *global clock* variable Cl over the integers. Each entity (network node) $X \in \mathcal{E}$ is equipped with an internal clock $\mathsf{Cl}[X]$ and interacts with the other nodes via an incoming and an outgoing tape.

For each $X, Y \in \mathcal{E}$, we denote by $[X \mapsto Y]$ the (simplex) communication channel from X to Y. Each channel $[X \mapsto Y]$ has a *bandwidth* and a *throughput* value denoted by $\mathsf{BW}[X, Y]$ and $\mathsf{Through}[X, Y]$, respectively, where naturally $\mathsf{BW}[X, Y] \geq \mathsf{Through}[X, Y]$. We set $\mathsf{BW}[X, Y] = \mathsf{Through}[X, Y] = 0$, if channel $[X \mapsto Y]$ does not exist. Additionally, we define the list $\mathsf{Data}[X \mapsto Y]$ that contains the pending data transferred via $[X \mapsto Y]$.

We consider a *global passive* adversary $\mathcal{A}$ that observes the network traffic. Our setting is *semi-synchronous*, i.e. $\mathcal{A}$ can drift the entities' internal clocks w.r.t. Cl up to a *synchronisation loss bound* Δ_{sync}. Moreover, we allow the $\mathcal{A}$ schedule the traffic at will, restricted only by some fixed *bounded delay parameter* Δ_{net} and the specified channel throughputs. In particular, at any given slot T and channel $[X \mapsto Y]$, $\mathcal{A}$ acts as a rushing adversary that can obtain $\mathsf{Data}[X \mapsto Y]$ and permute it according to its strategy. At the end of T, $\mathcal{A}$ must forward the first $\mathsf{Through}[X, Y]$ messages in list $\mathsf{Data}[X \mapsto Y]$ to Y. The forwarded messages must comprise all messages which delivery has delayed for Δ_{net} slots unless these exceed the thresholds $\mathsf{Through}[X, Y]$, where network congestion enforces message drop.

The threat model under the presence of an adversary $\mathcal{A}$ is captured by the *networking functionality* $\mathcal{F}_{\mathsf{net}}$ described below. The functionality takes as input the *network parameters* denoted by

$$\mathsf{net\_params} := \left(\mathcal{E}, \{\mathsf{BW}[X, Y], \mathsf{Through}[X, Y]\}_{X,Y \in \mathcal{E}}, \Delta_{\mathsf{sync}}, \Delta_{\mathsf{net}}\right) .$$

We denote by $L_1 \leftarrow L_2 \circ L_1$ the appending of list L_2 to list L_1.

Initialisation $(T = 0)$

– Upon receiving $\mathsf{initialize}$ from $\mathcal{A}$, if its status is neither set to '$\mathsf{execute}$' nor '$\mathsf{finalise}$', then it executes the following steps:

1. It sets the global clock as $\mathsf{Cl} \leftarrow 0$.
2. It initialises the set of corrupted entities as $S_{\mathsf{corr}} \leftarrow \emptyset$.
3. It initialises the set of entities that completed their activity as $S_{\mathsf{comp}} \leftarrow \emptyset$.
4. For each $X, Y \in \mathcal{E}$
   – It sets the clock of X as $\mathsf{Cl}[X] \leftarrow 0$.
   – It initializes the list $\mathsf{Data}[X, Y] \leftarrow \varepsilon$.
   – It initializes the list $\mathsf{Receive}[X] \leftarrow \varepsilon$.
   – It initializes the "delayed data lists" $\mathsf{Delay}[i, X, Y] \leftarrow \varepsilon$, where $i = 0, \ldots, \Delta_{\mathsf{net}}$.
5. It sets its status to '$\mathsf{execute}$' and sends the message '$\mathsf{slot\_start}$' to $\mathcal{A}$.

Process execution during slot $\mathsf{Cl} = T$ under 'execute' status

- Upon receiving (push, M, Y) from X, where M is a message and Y is the intended recipient, if the size of $\mathsf{Delay}[0, X, Y]$ does not exceed $\mathsf{BW}[X, Y]$, then it selects a new unique message ID $\mathsf{id}(M)$ for M, appends $(\mathsf{id}(M), X, M, Y)$ to $\mathsf{Delay}[0, X, Y]$, and provides $\mathcal{A}$ with $(\mathsf{id}(M), X, M, Y)$.
- Upon receiving $\mathsf{complete\_push}$ from X, it adds X to S_{comp} and sends $(\mathsf{complete\_push}, X)$ to $\mathcal{A}$.
- Upon receiving $(\mathsf{corrupt}, X)$ from $\mathcal{A}$, it adds X to S_{corr}.
- Upon receiving $(\mathsf{forward}, \mathsf{id}(M))$ from $\mathcal{A}$, if there is a triple (X, M, Y) and some $i = 0, \ldots, \Delta_{\mathsf{net}}$ such that $(\mathsf{id}(M), X, M, Y)$ is an element of $\mathsf{Delay}[i, X, Y]$, then it appends (X, M, Y) to $\mathsf{Data}[X, Y]$, also removing $(\mathsf{id}(M), X, M, Y)$ from $\mathsf{Delay}[i, X, Y]$.
- Upon receiving $(\mathsf{reorder}, \mathsf{Data}[X, Y], L)$ from $\mathcal{A}$, if L is a permutation of $\mathsf{Data}[X, Y]$, then it sets $\mathsf{Data}[X, Y] \leftarrow L$.
- Upon receiving $(\mathsf{set}, \mathsf{Cl}[X], T')$ from $\mathcal{A}$, if $X \in S_{\mathsf{corr}}$ and $|T' - \mathsf{Cl}| \leq \Delta_{\mathsf{sync}}$, then it sets $\mathsf{Cl}[X] \leftarrow T'$.
- Upon receiving $\mathsf{complete}$ from $\mathcal{A}$, if $S_{\mathsf{comp}} = \mathcal{E}$, then it sets its status to '$\mathsf{finalise}$' and $S_{\mathsf{comp}} \leftarrow \emptyset$. Next, it sends the message '$\mathsf{slot\_end}$' to $\mathcal{A}$.

Finalising slot $\mathsf{Cl} = T$ and time advancement under 'finalise' status

- Upon receiving $\mathsf{arrange\_messages}$ from $\mathcal{A}$, it executes the following steps:
  1. For each $X, Y \in \mathcal{E}$,
     - It appends the first up to $\mathsf{Through}[X, Y]$ elements of $\mathsf{Delay}[\Delta_{\mathsf{net}}, X, Y]$ to the first positions of $\mathsf{Data}[X, Y]$, i.e. it sets $\mathsf{Data}[X, Y] \leftarrow \mathsf{Delay}[\Delta_{\mathsf{net}}, X, Y] \circ \mathsf{Data}[X, Y]$.
     - For $i = \Delta_{\mathsf{net}}, \ldots, 1$, it overwrites the data in $\mathsf{Delay}[i, X, Y]$ with the data in $\mathsf{Delay}[i - 1, X, Y]$.
     - It sets $\mathsf{Delay}[0, X, Y] \leftarrow \varepsilon$.
     - It appends the first up to $\mathsf{Through}[X, Y]$ elements of $\mathsf{Data}[X, Y]$ to $\mathsf{Receive}[Y]$, by setting $\mathsf{Receive}[Y] \leftarrow \mathsf{Data}[X, Y] \circ \mathsf{Receive}[Y]$. Next, it removes the appended elements from $\mathsf{Data}[X, Y]$.
  2. It sends the list $\mathsf{Receive}[X]$ of every $X \in \mathcal{E}$ to $\mathcal{A}$.
- Upon receiving $(\mathsf{reorder}, \mathsf{Receive}[X], L)$ from $\mathcal{A}$, if L is a permutation of $\mathsf{Receive}[X]$, then it sets $\mathsf{Receive}[X] \leftarrow L$.
- Upon receiving (pull, b) from X, where $b \in \{0, 1\}$, if $b = 1$, then it writes the contents of $\mathsf{Receive}[X]$ on the incoming tape of X. Next, it adds X to S_{comp} and sends the message (pull, b, X) to $\mathcal{A}$.
- Upon receiving $\mathsf{advance\_clock}$ from $\mathcal{A}$, if $S_{\mathsf{pull}} = \mathcal{E}$, then it executes the following steps:
  1. For each $X \notin S_{\mathsf{corr}}$, it sets $\mathsf{Cl}[X] \leftarrow \mathsf{Cl}[X] + 1$ and $\mathsf{Receive}[X] \leftarrow \varepsilon$.
  2. It sets $S_{\mathsf{comp}} \leftarrow \emptyset$ and advances the global clock as $\mathsf{Cl} \leftarrow \mathsf{Cl} + 1$.
  3. It sets its status to '$\mathsf{execute}$' and sends the message '$\mathsf{slot\_start}$' to $\mathcal{A}$.

9.2.10 Email protocol execution over $\mathcal{F}_{\mathsf{net}}$

Via $\mathcal{F}_{\mathsf{net}}(\mathsf{net\_params}$, we can describe the execution of an email ecosystem, as shown in Fig. 9.3.

Email protocol execution over $\mathcal{F}_{\mathsf{net}}$.

Send **Protocol**.

- Upon receiving $(C_s@SP_i, M, C_r@SP_j)$ from $\mathcal{Z}$, C_s computes the encoding as $(\mathsf{Encode}(M), C_r@\mathsf{SP}_j)$ and sends the message $\big(\mathsf{push}, C_s, (\mathsf{Encode}(M), C_r@\mathsf{SP}_j), SP_i\big)$ to $\mathcal{F}_{\mathsf{net}}$.

Route **Protocol**.

- Upon receiving (pull, b) from $\mathcal{Z}$, SP_i sends the message $(\mathsf{pull}, b, \mathsf{SP}_i)$ to $\mathcal{F}_{\mathsf{net}}$.
- When reading a message $\big(C_s, (\mathsf{Encode}(M), C_r@\mathsf{SP}_j), \mathsf{SP}_i\big)$ from its incoming tape, SP_i chooses an entry mix-node MX_k according to the protocol description.

Receive **Protocol**.

- Upon receiving $(\mathsf{Fetch}, C_r@SP_j)$ from $\mathcal{Z}$, C_r sends the message $(\mathsf{push}, C_r, \mathsf{Fetch}, \mathsf{SP}_j)$ to $\mathcal{F}_{\mathsf{net}}$.
- Upon receiving (pull, b) from $\mathcal{Z}$, SP_j sends the message $(\mathsf{pull}, b, \mathsf{SP}_j)$ to $\mathcal{F}_{\mathsf{net}}$.
- When reading a message $(C_r, \mathsf{Fetch}, \mathsf{SP}_j)$ from its incoming tape, it forwards the message $(C_r, \mathsf{Fetch}, \mathsf{SP}_j)$ to $\mathcal{Z}$.
- Upon receiving $(\mathsf{push}, C_r@SP_j)$, if $\mathsf{Inbox}[C_r@SP_j]$ is non-empty, SP_j retrieves the first message of $\mathsf{Inbox}[C_r@SP_j]$ denoted by $\mathsf{Inbox}[C_r@SP_j][1]$. Then, it sends $(\mathsf{push}, \mathsf{SP}_j, \mathsf{Inbox}[C_r@SP_j][1], C_r)$ to $\mathcal{F}_{\mathsf{net}}$, removing $\mathsf{Inbox}[C_r@SP_j][1]$ from $\mathsf{Inbox}[C_r@SP_j]$.
- Upon receiving (pull, b) from $\mathcal{Z}$, C_r sends the message (pull, b, C_r) to $\mathcal{F}_{\mathsf{net}}$.
- When reading a message $\mathsf{Inbox}[C_r@SP_j][1]$ from its incoming tape, it parses $\mathsf{Inbox}[C_r@SP_j][1]$ as $(C_s@\mathsf{SP}_i, \mathsf{Encode}(M, C_r@\mathsf{SP}_j))$ and decodes it as $(C_s@\mathsf{SP}_i, M)$. Then, it forwards $(C_s@\mathsf{SP}_i, M)$ to $\mathcal{Z}$.

Figure 9.3: Email protocol execution over $\mathcal{F}_{\mathsf{net}}$.

Bibliography

[1] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145. IEEE Computer Society, 2001.

[2] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In R. Dingledine and P. F. Syverson, editors, *Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers*, volume 2482 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 2002.

[3] C. Dwork. Differential privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, pages 1–12, 2006.

[4] A. Pfitzmann and M. Hansen. Anonymity, Unlinkability, Unobservability, Pseudonymity, and Identity management – a consolidated proposal for terminology. Version v0.25, December 2005.