Geroge Danezis—Ed. (UCL)
Ania Piotrowska (UCL)
Helger Lipmaa (UT)
Michal Zajac (UT)
Claudia Diaz (KUL)
Tariq Elahi (KUL)
Benjamin Weggenmann (SAP)
Aggelos Kiayias (UEDIN)

Design, modelling and analysis

Deliverable D3.1

31st October 2016
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	2016.06.22	AP (UCL)	Intial draft
0.2	2016.06.22	GD (UCL)	Initial review, edits and comments
0.3	2016.06.22	GD, AP (UCL)	Reconstruction
0.4	2016.06.26	VM (UCL)	Review and comments
1.0	2016.06.29	GD (UCL)	Final review - submission to the EC
1.1	2016.10.24	GD (UCL)	Revision after first periodic review
1.2	2016.10.24	GD, AP (UCL)	Restructure of the document after consultation with project partners
1.3	2016.10.25	GD, AP (UCL)	Unifieing the report and describing the results of research outputs
1.4	2016.10.25	GD, AP (UCL)	Added surveys about existing anonymous communication systems and shuffle protocols
1.5	2016.10.25	GD, AP (UCL)	Describing relevance between research outputs and project tasks
1.6	2016.10.25	VM (UCL)	Review and comments
1.7	2016.10.28	HH (GH)	Review, edits and feedback
1.8	2016.10.31	AP (UCL), TZ, AK (UEDIN)	Final editing after consultation with project partner
2.0	2016.10.31	AK (UEDIN), TZ (UEDIN), GD (UCL)	Revisioned final version and submission to the EC

Executive Summary

Deliverable 3.1 presents the report of activities and outputs of PANORAMIX WP3, which aims to investigate and propose technology options for building PANORAMIX mix-networks. In deliverable D3.1 we survey the existing shuffle based techniques and compare their functionalities, properties and limitations as well as discuss how those techniques may support PANORAMIX project. D3.1 presents also a report of the first year research outputs from WP3 and outlines their relation to the commitments described in the project proposal. We present the new design options investigated by research partners in WP3 and outline how they can support PANORAMIX. This deliverable presents also how the novel techniques proposed by WP3 support other work packages in PANORAMIX project.

Contents

1. Preface to Deliverable D3.1

The aim of the PANORAMIX project is to develop a wide-spread infrastructure based on robust mix-networks (WP4), which guarantees privacy and anonymity properties for a number of high-impact applications. The PANORAMIX project targets three main use cases: (1) privacy-preserving and anonymous messaging systems (WP7), (2) private electronic voting protocols (WP5), (3) privacy-friendly surveying, statistics and big data gathering protocols (WP6).

PANORAMIX WP3, and the deliverable D3.1, aims to investigate and propose technology options for building mix-networks. It provides the necessary background and design options for our collaboration partners in WP4, as well as in support of in terms of new research findings for our partners and their use-cases in WP5, WP6 and WP7. WP3 focuses on the design of the secure and efficient mix network protocols and conducts the theoretical and experimental security analysistasks that require original research and advanced development. Therefore, WP3 presents *new* research provoked by the PANORAMIX project and the requirements of the partners. First, note that as WP3 is focused on research and is so experimental and risky, it is important that components of this deliverable by published as papers, as peer review may catch errors and problems that the PANORAMIX partners could not catch by themselves. Second, this deliverable is the most difficult to read by a non-expert of any deliverable, but the mathematical machinery presented is necessary in order to verify that these new designs provably satisfy the security and properties, as well as properties around verifiability, latency, and anonymity as discussed in D4.1. Therefore, readers who do not have a mathematical background may want to focus on Chapter 1 as well as the surveys of Chapter 2 and 3, but for the rest of the papers may feel free to skim the proofs and focus on the conclusions of these chapters. It is not expected that each partner understand the research fully, but instead that the academic partners will provide options in the design for each partner and the core PANORAMIX infrastructure, as well as support in writing the initial code. The industrial partners can chose between the design options in WP3 and then they can, with the help of the academic partners, make sure the code they have developed can be matured for industrial use-cases.

As discussed in WP4 as well as WP7 and WP5, the requirements listed by the partners in terms of a "real world" mix networking infrastructure have opened a number of novel research questions. For example, both e-voting and messaging require lower latency and scalability than the existing mix networking solutions can provide, leading to work investigating paradigms to improve scalability in terms of secure multi-party computation. Furthermore, e-voting has need for verifiability. In this deliverable we present designs for both low latency shuffling that also satisfies the verifiability requirements in Part 2 by designing new shuffle protocols needed for e-voting that are not based on unrealistic "random oracle" assumptions and that are succinct (i.e. do not take up too much space). Due to the difficulty and time consuming nature of cryptographic work, we did not capture all the requirements. For example, in the secure messaging use-case from WP7, a system is needed where clients can go offline, and this requirement will lead to future work in D3.2.

This document presents a report of the research and other outcomes from WP3 in the first year of the project, and in line with the deliverable as outlined in the PANORAMIX project

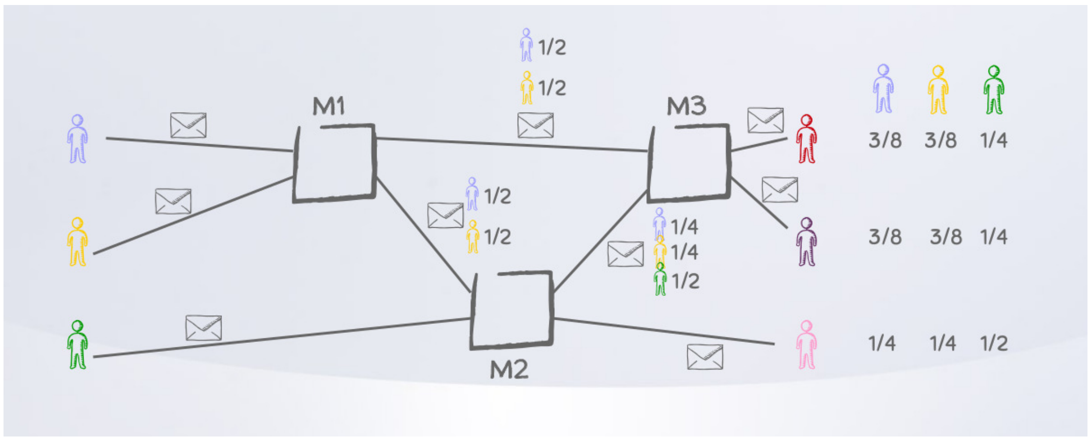


Figure 1.1: A mix net example

proposal. First, we overview the relevancy of the research to mix networking in section 1.1. In section 1.2 we outline the structure of this deliverable in relation to the commitments in project proposal; in section 1.3, we map each chapter of this deliverable to the objectives and tasks of WP3 as described in the original project work plan.

1.1 A brief introduction to mix networking

While encryption can make the encrypted message itself unlinkable to the plaintext message (i.e. unlinkability in terms of bits), it is much harder to eliminate what has been termed *metadata* of the message, i.e. the patterns such as timing, length, and network-level identifiers that can be used to identify who is communicating to whom via techniques from traffic analysis. Although cryptography itself has firm mathematical foundations, it deals with only a small, if crucial, component of anonymous communications. Other techniques such re-scheduling (i.e. sending a group of messages at the same time), re-packetizing (i.e. sending a messages that are all the same length), and re-routing (i.e. destroying patterns in the underlying network itself) are needed in order to defeat traffic analysis and so have *anonymous communications.*

The first practical method for anonymous communicating proposed by David Chaum (a member of the PANORAMIX Advisory Board) is *mix networking.* In mix-networking, a message is sent to different nodes. Each node collects the messages, and outputs them, re-encrypting them and making sure they are uniform in size and re-sending them at the same time in a random order as shown in Figure 1.1.[1] In this figure, three senders are trying to send messages to three receivers via a small mix network of only three mix nodes (M). In the figure is also given the probability that a receiver has received a message that can be determined by a global adversary who is observing the traffic flows.

A number of problems should be apparent. First, a determined global adversary can still de-anonymise people using the network if, for example, during a given time period no message is sent to a person via an output node that no-one else is using (see Figure 1.2 for an example. In this case, *dummy traffic*, i.e. fake packets that are the same size as the other messages, may be needed, which increases the amount of bandwidth. More importantly, sending a message through a mix net requires the messages to be mixed using a *shuffle* in order to determine which message should be sent to which other mix node. Ideally, we should be able to prove that a mix node is not malicious by verifying their shuffle, and zero knowledge proofs are one technique for doing this. Also, the messages must all be sent at the same time, which requires work (to be

[1] Thanks to Carmela Troncoso (IMDEA) for sharing the following three illustrations from her "Traffic Analysis: or... encryption is not enough" slides at https://software.imdea.org/~carmela.troncoso/talks/CTroncoso_TrafficAnalysis_Croatia2016.pdf.

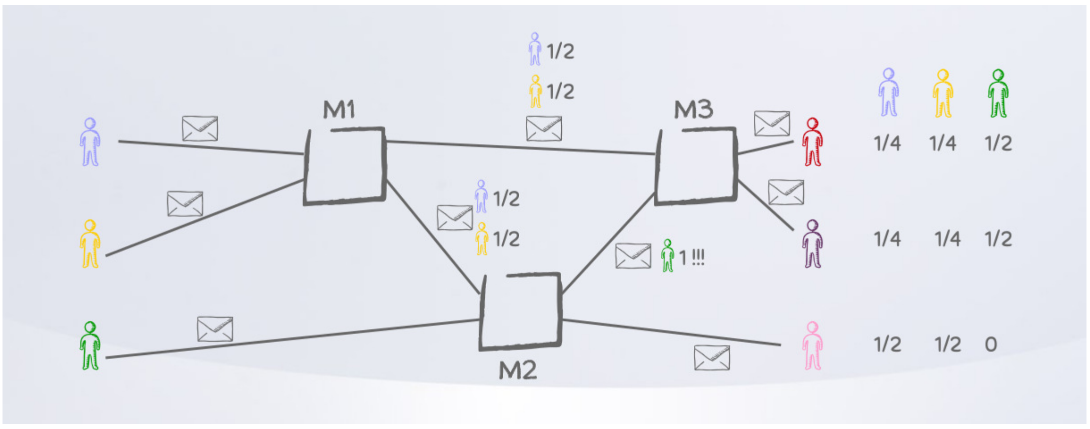


Figure 1.2: A mix net example with one route missing

done in the future in either D3.2 or D3.3) on *flushing* (batching) the messages. Lastly, as we need to send the message via several nodes, we have higher and higher latency, and this requires work to be one to make the technique more efficient.

Although the examples have so far been simple, in reality a mix network like PANORAMIX will have many, many nodes in order to send messages between a realistic number of senders and receivers of messages, as shown in Figure 1.3.

Due to the number of mix nodes, *efficiency* is very important and a single inefficient component can make the entire system unusable for real-world applications. Also, historically different use-cases such as messaging have had each mix net nodes that decrypt, shuffle, and store are called *decryption* mix-nets while e-voting tends to use *re-encryption* mix nets that blinds inputs. All the work in this deliverable creates state-of-the art solutions for efficient shuffling with zero-knowledge proofs as well as state of the art techniques for helping the privacy for the input and output nodes. We also compare mix networking with more popular techniques such as onion-routing that are not resistant to a global passive adversary as they do not shuffle, avoid timing attacks, or use dummy traffic and show how techniques like secure multi-party computation can also provide anonymity that can complement mix networking. Therefore, in addition to surveys of state of the art, the rest of this deliverable focuses on improving shuffling and other efficiency guarantees in terms of scalability that would be needed for real-world deployment of mix networking for the PANORAMIX use-cases. Until these hard research problems are tackled, we will not see a real-world mix networking system like PANORAMIX reach the usage of alternative onion-routing systems like Tor.

1.2 Outline of the deliverable

The deliverable D3.1 is described in the project proposal as comprising:

Deliverable D3.1 (Initial report) [M10] Modelling and Design elements.
Describes

- *some of the existing shuffle protocols* (WP3.2),
- *initial design options for mix-nets* (WP3.1),
- *definitions of privacy* (WP3.3)

Subsequent chapters in this deliverable are organised to closely match the description of deliverable D3.1 above, and are divided in 3 parts to clarify the distinctions given in the deliv-

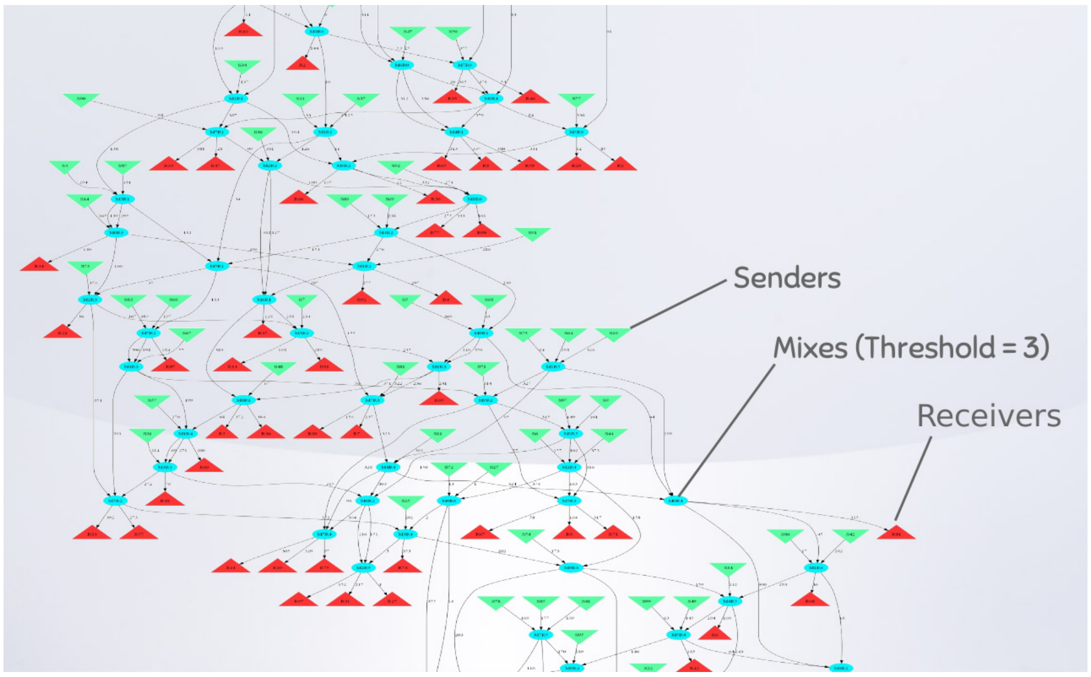


Figure 1.3: A more realistic mix networking case

erable's legally binding Description of Work.

Part I provides a survey of the existing anonymous messaging and shuffle protocols in two chapters: Chapter 2 presents previous work on re-encryption mix-nets and Chapter 3 describes decryption mix-nets that use shuffling.

Part II presents the initial designs options for building anonymous communication network and more efficient yet secure non-interactive Zero-Knowledge shuffle protocols in 3 chapters: Chapter 4 describes an improved shuffle protocol; Chapter 5 better techniques for proving the correctness of shuffles; and Chapter 6 a proposed shuffle based on multi-party computation.

Finally, Part III presents the privacy definitions and methodologies: Chapter 7 presents differential privacy definitions relating to location privacy supporting WP6; and Chapter 8 describes an empirical evaluation methodology for evaluating the security of low-latency anonymity systems to support WP7 designs.

1.3 WP3 objectives and mapping to D3.1 deliverable

In this section we relate each chapter of this deliverable to each of the tasks of WP3, and summarise their key contributions to the PANORAMIX project in terms of what other WPs and requirements from the partners they address:

T3.1 Mix-nets. We also performed research supporting Task 3.1. First off, we present a survey, that thoroughly studies all key mix-net designs, shuffle protocols and anonymity systems and categorises them in terms of their path selection procedures and other structural and security characteristics. This directly informs the design options for the PANORAMIX WP4 mix-net.

- **Chapter 2** - Existing shuffle protocols: A Survey of Anonymous Communication Protocols for Messaging

  This chapter surveys the existing designs and solutions for anonymous communication, including re-encryption mix-nets, and their performance, as well as technologies relating to decryption mix networks. The survey delivered by PANORAMIX partners compares the existing solutions and introduces a taxonomy which classifies the existing anonymous protocols to allow compare them in terms of routing characteristics, performance and scalability. The presented summarization serves as a background information for our partners in PANORAMIX WP6 and WP7.

The introduced taxonomy extends the previous routing characteristics defined by Feeney, which were not supporting several anonymous communication networks. This novel definition of different criteria groups allows to widely investigate existing solutions and find the thresholds between the security, scalability and performance as well as to support the future designs.

This chapter is led to a partner technical report on the topic and is currently undergoing peer review at scientific venue:

[SSA+16] Fatemeh Shirazi, Milivoj Simeonovski, Muhammad Rizwan Asghar, Michael Backes, and Claudia Diaz. A survey on routing in anonymous communication protocols. Technical report, KU Leuven Technical Report, 2016

T3.2 Zero-Knowledge proofs of correct shuffle / mix verifiability. We deliver work supporting PANORAMIX WP3.2. We present designs of efficient yet secure non-interactive Zero-Knowledge shuffle protocols; each of the works also presents the state of the art of shuffle protocols on which the new proposed designs build, and a comparison of their performance and characteristics. Closer to the messaging use-case, we also present a new design for anonymous messaging that uses secure multi-party computation for shuffling messages. This option is based on requirements in terms of scalability to support the PANORAMIX WP7 messaging use-case and may work well not only by itself, but as a system for each messaging server to run in WP7.

- **Chapter 3** - Existing shuffle protocols: A Survey of Shuffle protocols

  This chapter describes the existing shuffle protocols, compare the interactive and non-interactive shuffle arguments and discuss their efficiency. It provides a wide overview of cryptographic shuffles, namely those that come with a proof that the shuffling was correcti.e., no elements were added or removed. This supports and informs directly the design of the PANORAMIX WP4 mix-net, as well as the election use-case in PANORAMIX WP5.

- **Chapter 4** - Initial design options for mix-nets: Efficient Culpably Sound NIZK Shuffle Argument without Random Oracles

  Zero-knowledge shuffle arguments enable the *prover* to prove, that she mixed the ciphertexts correctly, without revealing how they were shuffled or any other secrets. As such, shuffle arguments are crucial in the design of mix-nets for e-voting where one has high security requirements for ballot secrecy and unlinkabilitya key requirement of PANORAMIX WP5 (Election use-case). Thus, in order to develop secure yet efficient and practical mix-net implementations PANORAMIX partners need to develop and deploy a provable and secure high-performance non-interactive ZK shuffle proofs, with significantly lower overheads than previous approachesas the one suggested in this chapter.

  Most of the well known efficient non-interactive shuffle arguments are constructed in the random oracle model — that is, by assuming the existence of an hypothetical "totally random function" that everybody has access to. Since such functions cannot be efficiently implemented, random oracle model arguments only offer heuristic security guarantees. Thus, the partners studied the existing non-interactive zero knowledge proofs and proposed in the chapter the most efficient known zero knowledge shuffle argument that does not use random oracles. This is therefore much more likely to be a secure basis for real-world usage of PANORAMIX than other existing shuffle techniques, and this shuffling approach will likely be adopted by the core PANORAMIX to be used across all the use-cases.

This chapter resulted in a partner peer-reviewed publication on the topic:
[FL16] Prastudy Fauzi and Helger Lipmaa. *Efficient Culpably Sound NIZK Shuffle Argument Without Random Oracles*, pages 200–216. Springer International Publishing, Cham, 2016

- **Chapter 5** - Initial design options for mix-nets: Prover-Efficient Commit-And-Prove Zero-Knowledge SNARKs

  SNARKs are "succinct non-interactive arguments of knowledge". By using a SNARK, one can efficiently prove (i.e. verify) in zero knowledge that some property holds, without the *verifier* getting extra information. This allows to develop a e-voting applications based on mix-nets (relevant to the PANORAMIX WP5 election use-case) which obtain almost ideal security, universal verifiability but at the same time being efficient in terms of performance.
  Two crucial properties of SNARKs are non-interactiveness (the same proof can be generated once and then verified by many different verifiers without each one interacting with the verifier) and succinctness (the proof should be short and efficient to verify). This design proposes the most prover-efficient known SNARKs for several interesting problems, including solving NP-complete problems like Subset-Sum but also a new range proof. Range proofs are in particular needed in e-voting. The techniques developed in this chapter advice the partners in WP5 how to develop and construct efficient zero knowledge proofs for secure and private e-voting applications and so extends the previous designs in PANORAMIX on new shuffling techniques for some of the more stringent privacy requirements of e-voting in terms of verifiability.

  This chapter resulted in a partner peer-reviewed publication on the topic:
  [Lip16] Helger Lipmaa. *Prover-Efficient Commit-and-Prove Zero-Knowledge SNARKs*, pages 185–206. Springer International Publishing, Cham, 2016

- **Chapter 6** - Perfectly Anonymous Messaging via Secure Multiparty Computation

  Going beyond zero-knowledge shuffling for PANORAMIX, this chapter presents 'XYZ', a new design of an anonymous messaging system that provides *perfect anonymity* and can scale in the order of hundreds of thousands of users, via a shuffle based on an efficient formulation of secure multi-party computation. It is possible that this XYZ design could be used for the Greenhost case of messaging in WP7 if mix networking by itself and the other shuffles cannot handle the requirements of its messaging use-case. Although this is hard research problem and work will continue in D3.2 to deal with the problems of churn in messaging clients and the details of the needed latency, this initial design has many remarkable properties by virtue of building on a different research framework than traditional mix networking systems, i.e., secure multi-party computation.
  In brief, it isolates two suitable *ideal functionalities*, called dialing and conversation, that when used in succession realise anonymous messaging. With this as a starting point, we apply secure multi-party computation (SMC) to instantiate them with *information theoretic* security in the semi-honest model. Using a parallelization technique scales them to a large number of users, without sacrificing privacy, and provides

a degree of forward security on the client side and can be instantiated in a variety of different ways with different SMC implementations overall, illustrating how SMC is a competitive with traditional mix-nets and DC-nets for anonymous communication. Although PANORAMIX is focused on mix networking for the backbone, Greenhost and WP7 will consider a SMC-based solution for their problem in terms of client-server anonymity if a design closer to the mix networking system used by the core PANORAMIX infrastructure cannot be found.

T3.3 Differential Privacy mechanisms and mix-net applications. Finally, we studied supporting WP3.3. Our work on private statistics collection uses novel definitions of privacy inspired from differential privacy and traditional game based cryptographic definitions; the same work evaluates the utility that can be achieved despite different levels of privacy protection. Similar privacy metrics have been successful in measuring attacks on websites and Tor, and thus we believe these metrics will be useful to use with PANORAMIX's core infrastructure and messaging clients.

- **Chapter 7** - Definitions of privacy: AWAREAnonymization with Guaranteed Privacy

  In this chapter SAP presents the assumptions and goals of the SAP Product Security Research project AWARE "Anonymization With guARantEed privacy", relevant to PANORAMIX WP6 use-case on private statistics and telemetry . The main goal is to provide a framework for the data protection officer to apply anonymization with measurable and reliable guarantees. Many previous existing anonymization methods fail at providing these goals since they do not provide any formal privacy guarantee and are vulnerable to attacks that re-identify the originators of the anonymised data. This report investigates a differential privacy definition, that does provide a formal privacy guarantee, and examines how it performs at simultaneously protecting the privacy of the users and providing good utility for analysis. This chapter provides the results of experiments in which the differentially private anonymization mechanisms was applied to protect different types of sensitive data. The results indicate that when applied properly, differentially private mechanisms can protect privacy while still providing utility with sufficient accuracy for further analysis.

  This chapter is based on a partner (SAP) technical report that was created in response to their requirements for surveys and statistics using PANORAMIX:

  [KKHB16] Florian Kerschbaum, Mathias Kohler, Florian Hahn, and Daniel Bernau. Aware: Anonymization with guaranteed privacy. Technical report, SAP Internal Product Security Research Technical Report, 2016

- **Chapter 8** - Definitions of privacy: Empirical Evaluation of Privacy via Website Fingerprinting

  This chapter is focused on defining privacy in a manner relevant to PANORAMIX. The presented taxonomy and analysis supports the partners in PANORAMIX WP6 and WP7 in development of secure protocols for gathering statistics and messaging. While mix networking has well-understood privacy on the mix nodes, attacks will generally happen on the messaging client, which is usually a client accessed via a website or a native application that calls a Web-enabled API, as done in WP4. This chapter presents new web fingerprinting attacks and their empirical evaluation. These attacks should be applied to the PANORAMIX low latency mixing system and are very relevant to PANORAMIX statistics and messaging use-cases, as we show these attacks can be prevented with much better performance than the current state of the art solutions. Given that PANORAMIX core infrastructure is still being developed, this attacks were done via an analysis of ordinary websites, although attacks on web-enabled messaging clients or input nodes will have the same properties.

Website fingerprinting has emerged as a serious threat to anonymity of internet users. Even despite using the privacy preserving technologies, website fingerprinting attacks may enable an adversary to infer what website a user visits. It has been shown, that an adversary by passively observing the size and timing of packets can infer with varying degrees of certainty what websites a user is visiting.

The existing state-of-the art solutions apply some supervised or semi-supervised learning techniques to track when and if a user visits a small number of websites, however this technique is unrealistic as most users can visit any website they choose, and are not restricted in a small fixed set. As a result, current research solutions achieve worst results, in an open world environment.

In this chapter, the designed fingerprinting solution collects data and uses ML classifiers that do not degrade in accuracy when the number of websites we wish to fingerprint is scaled up. We used hashing techniques that are most often employed in computer vision and image processing research to train on a much larger scale of websites. This will allow to fingerprint websites in an open world environment with a much higher confidence of success than current existing techniques. Proper investigation of fingerprinting attacks has a crucial role in designing and building secure and anonymous communication in PANORAMIX. Thus, thanks to investigating this type of attacks we can understand the security problems which the low-latency anonymity systems developed in PANORAMIX WP4 will face and develop a better solutions resistant to this type of attacks.

This chapter resulted in a partner peer-reviewed publication on the topic:

[HD16] Jamie Hayes and George Danezis. k-fingerprinting: a robust scalable website fingerprinting technique. *USENIX Security Symposium 2016*, August 2016

Part I

Existing mix and shuffle protocols

2. Exisiting shuffle protocols: A Survey on Routing in Anonymous Communication Protocols

In this chapter, we survey previous research on designing, developing, and deploying systems for anonymous communication, comparing their security, performance and scalability properties. We provide a taxonomy for clustering all prevalently considered approaches (including Mixnets, DC-nets, onion routing, and DHT-based protocols) with respect to their unique routing characteristics, deployability, and performance. The presented taxonomy and comparative assessment provide important insights about the differences between the existing classes of anonymous communication protocols, and to clarify the relationship between the routing characteristics of these protocols, and their performance and scalability, in order to deliver neccassery background information about design options to support partners in WP5, WP6 and WP7 of PANORAMIX project.

2.1 Introduction

The Internet has evolved from a mere communication network used by millions of users to a global platform for social networking, communication, education, entertainment, trade, and political activism used by billions of users. In addition to the indisputable societal benefits of this transformation, the mass reach of the Internet has created new powerful threats to online privacy.

The widespread dissemination of personal information that we witness today in social media platforms and applications is certainly a source of concern. The disclosure of potentially sensitive data, however, not only happens when people deliberately post content online, but also inadvertently by merely engaging in any sort of online activities. This inadvertent data disclosure is particularly worrisome because non-expert end-users cannot be expected to understand the dimensions of the collection taking place and its corresponding privacy implications.

Widely deployed communication protocols only protect, if at all, the content of conversations, but do not conceal from network observers who is communicating with whom, when, from where, and for how long. Network eavesdroppers can silently monitor users' online behavior and build up comprehensive profiles based on the aggregation of user communications' metadata. Today, users are constantly tracked, monitored, and profiled, both with the intent of monetizing their personal information through targeted advertisements, and by nearly omnipotent governmental agencies that rely on the mass collection of metadata for conducting dragnet surveillance at a planetary scale.

Anonymous Communication (AC) systems have been proposed as a technical countermeasure to mitigate the threats of communications surveillance. The concept of AC systems was introduced by Chaum [1] in 1981, with his proposal for implementing an anonymous email service that aimed at concealing who sent emails to whom. The further development of this concept in the last decades has seen it applied to a variety of problems and scenarios, such as anonymous voting [2,3], Private Information Retrieval (PIR) [4], censorship-resistance [5,6], anonymous web browsing [7], hidden web services [8], and many others.

Public interest in AC systems has strikingly increased in the last few years. This could be explained as a response to recently revealed dragnet surveillance programs, the fact that deployed AC networks seem to become (according to leaked documents[1]) a major hurdle for communications surveillance, and to somewhat increased public awareness on the threats to privacy posed by modern information and communication technologies.

The literature offers a broad variety of proposals for anonymity network designs. Several of these designs have been implemented, and some are successfully deployed in the wild. Of the deployed systems, the most successful example to date is the Tor network, which is used daily by about two million people [9].

Existing designs take a variety of approaches to anonymous routing for implementing the AC network. Routing determines how data is sent through the network, and it as such constitutes the central element of the AC design, determining to a large extent both security and performance of the system. These approaches rely on different threat models and sets of assumptions, and they provide different guarantees to their users. Even though survey articles on AC systems exist [10–18], we still lack a systematic understanding, classification, and comparison of the routing characteristics of the plurality of existing AC approaches.

The purpose of this survey is to provide a detailed overview of the routing characteristics of current AC systems, and to examine how their features determine the anonymity guarantees offered by those systems, as well as its overall performance. To this end, we first *identify the routing characteristics* that are relevant for AC protocols and provide a *taxonomy* for clustering the systems with respect to their routing characteristics, deployability, and performance. Then, we apply the taxonomy to the extensive scope of existing AC systems, in particular including Mixnets, DC-nets, onion routing systems, and DHT-based protocols. Finally, we discuss the relationship between the different routing decisions, and how they affect performance and scalability.

Section

2.2 Anonymous Routing Protocol Characteristics

This section first introduces the routing characteristics considered in our taxonomy, and then discusses deployability, and performance metrics for AC networks.

2.2.1 Routing Characteristics

Generally, routing in a communication network refers to the selection of nodes for relaying communication through the network. Routing schemes, however, require some essential design components. For anonymous communication, we consider four building blocks that are relevant to routing in AC networks. These building blocks are node management, transfer/retrieval of node information to/by the routing decision maker, path selection, and forwarding or relaying; where path selection is the main design component of routing schemes for AC protocols.

Several taxonomies and classifications for routing protocols have been proposed in the literature [19–21]. However, AC networks aim to conceal the metadata of communications and thus have security requirements that make them fundamentally different from other networks.

[1]https://wikileaks.org/

In this section, we present a classification for anonymous routing protocols. Our classification (see Tables

1. *Communication model* describes whether the communication is based on single-channels or multi-channels.

2. *Structure* describes whether or not nodes are treated equally.

3. *State information* describes where the topology information is maintained.

4. *Scheduling* describes whether the information about routes is maintained at the source or is instead computed on-demand.

This taxonomy does not address several relevant design features of AC networks, such as probabilistic node selection for constructing circuits, and security considerations for protecting routing information from different network adversaries. In addition, not all the characteristics identified by Feeney are relevant to AC routing. For example, the distinction between single- and multi-channel features is not relevant in overlay networks, which constitutes a standard design choice for many AC networks.

We redefine Feeney's criteria to account for design choices that are relevant to anonymous routing protocols. We distinguish three groups of features inspired by Feeney's categories: *network structure*, *routing information*, and *communication model*:

1. *Network structure* describes the characteristics of the anonymous relays, the connections between them, and the underlying network topology.

2. *Routing information* describes the network information available to entities deciding on the route of an anonymous connection.

3. *Communication model* defines the entities that make the routing decisions and describes how these decisions are made.

In what follows, we describe these features in more details, including their various sub-features and corresponding notation symbols used to denote individual feature instantiations. We refer to Table

Network Structure

We consider first the network features that are relevant to anonymous routing. These are, specifically, features relating to: (a) the ***topology*** of the network, which describes how nodes are connected; (b) the ***connection type***, describing the characteristics of the connections between nodes; and (c) ***symmetry***, describing whether the entities participating in the network are all similar, or if they can take on different roles and responsibilities for routing data through the network.

a) **Topology.** The topology describes the arrangement of various elements of the network, such as routers and communication links between those routers. We only take the logical topology of the network into account, which determines how data flows within it. We note that physical topology characteristics, such as the geographical location of computers, sometimes matters in anonymous routing decisions, for example when considering adversaries that control an Autonomous System (AS) [22, 23].

Table 2.1: Overview of the Protocol Routing Characteristics

Feature Name			**Description**	**Instantiation and Symbols**
Network Structure	Network topology		Degree of node connectivity in the network	☒ (fully) □ (mostly) ⊏ (partially)
	Connection type	Direction	Data flow in connections	→ (unidirectional) ↔ (bidirectional)
		Synchronization	Timing model for connection establishment and data sending	≠ (asynchronous) ≅ (synchronous)
	Symmetry	Roles	Users operating as relays	•··•··• (peer-to-peer) •··• (client-server) •··○··• (hybrid)
		Topology	Node topology for routing	··· (flat) ✤ (hierarchical)
		Decentralization	Degree of decentralization for non-routing services	⊙ (semi decentralized) ○ (fully decentralized)
Routing Info	Network view		Network view necessary for making routing decisions	● (complete) ◐ (partial)
	Updating		Triggers for routing information updates	◷ (periodic) ↯ (event-based)
Communication Model	Routing type		Node selection per route	•··· (source-routed) ··•·· (hop-by-hop)
	Scheduling		Prioritization of traffic	≡ (fair) (prioritized)
	Node selection	Determinism	Determinism of node selection	✓ (deterministic) ✗ (non-deterministic)
		Selection set	Permissible set of nodes per route	Ⓐ (all) 🛑 (restricted, security) 🌍 (restricted, network) ☺ (user-based)
		Selection probability	Node selection probability per route	⊛ (uniform) ◎ (weighted, static) ❋ (weighted, dynamic)
Performance, Deployability	Latency		Protocol latency	L (low-latency) H (high-latency) M (mid-latency)
	Communication mode		Longevity of connections	(connection-based) ✉ (message-based)
	Implementation		Implemented	✓ (yes) ✗ (no)
	Code availability		Open source	✓ (yes) ✗ (no)

We consider the network as a graph in which the routers are represented by graph nodes. An edge between two nodes exists if the routing strategy allows those two nodes to be directly connected as part of the same anonymous circuit.

The connectivity of nodes varies widely across AC network designs, and the advantages and disadvantages of high and low levels of connectivity have been the subject of debate for over a decade [24].

Restricted routing proposals [25] have shown that for high-latency applications, partially connected networks with certain topological characteristics (*e.g.,* based on expander graphs) provide optimal anonymity and latency trade-offs and mitigate certain attacks. These results further emphasize the impact of network connectivity features for anonymous routing.

We classify anonymity networks into three categories according to their connectivity: *fully connected*, *mostly connected*, and *partially connected* networks.

- We consider a network to be *fully connected* (☒)[2]

[2] In parenthesis, we define the symbol or the keyword that is used in the comparative Tables

when nodes can potentially connect to most (or all) other nodes (our rule of thumb is that a node on average should be able to connect to at least 95% of the other nodes; this allows us to include systems that only exclude a small number of connections in order to prevent certain special cases from occurring).

- We call a network *mostly connected* (□) if its nodes can potentially connect to at least half the other nodes.
- Finally, in *partially connected* (⊏) networks nodes only connect to a relatively small subset of the whole network.

Higher connectivity in the network topology leads to better resilience (availability) against node failure, such as Denial of Service (DoS) attacks, such resilience might have in turn a positive influence on anonymity [24].
On the other hand, eliminating connections that might induce security problems, such as the connection between two nodes from the same IP family that may be easier to control by an adversary, but can be beneficial to anonymity. The same holds for eliminating connections that would induce higher latency, which would, in turn, improve the performance of the system.

- **Connection Type.** Here, we consider the *direction* and *synchronization* of connections. As far as the direction is concerned, we consider the following options:

  – A connection is *unidirectional* ($\rightarrow$) if the data flow between two entities can only be in one direction.

  – A connection between two entities is *bidirectional* ($\leftrightarrow$) if data can flow in both directions and the same connection is used for sending back the response to a received message.

  Typically, interactive applications, such as web browsing, require bidirectional channels, while non-interactive applications, such as email, can just close the connection as soon as the message has been forwarded. In the first case, short-lived session keys can be setup to achieve forward secrecy properties; however, in non-interactive applications, such as email, forward secrecy is harder to achieve.
  Bidirectional circuits have the advantage that they induce less overhead in terms of circuit construction. Unidirectional connections have the advantage that they are less vulnerable to timing attacks, as a malicious node can only observe data flowing in one direction, which is less informative than bidirectional connections in which patterns of requests and response are visible to all nodes in the path. However, note that in unidirectional connection, a larger number of nodes are going to be involved in relaying the communication between a sender and a receiver.
  Further, we consider whether the anonymity system involves connection *synchronization*:

  – A connection is *asynchronous* ($\neq$) if the establishment of connections and relaying of messages is initiated by a user without any timing coordination with other participants.

  – Connections are *synchronous* ($\cong$) if they begin and end at specific timings and messages are also relayed at specific moments in time, based on some timing coordination between network entities.

  Asynchronous systems are conceptually simpler as they impose fewer constraints on the activity of network participants. However, the distinct timing of actions leaks information valuable to perform traffic analysis and, for example, reveals long-term communication patterns [26] or perform end-to-end correlation attacks [27–29].
  Synchronous systems are often more difficult to engineer and come with a performance or usability penalty; moreover, secure and reliable time becomes an additional dependency of the system, and a possible point of failure or vulnerability to attack. However, synchronization constitutes a very powerful design feature to offer robust anonymity guarantees in the presence of powerful adversaries because it disables trivial end-to-end correlation attacks based on start and end times of connections [30], and other timing data that synchronization makes less granular, enabling the aggregation of participants, connections, and events in *anonymity sets.* Synchronous anonymity systems were proposed in the early 1990s by Pfitzmann *et al.* to anonymize ISDN telephony calls [31]. These proposals were both feasible from an engineering perspective (compatible with the network requirements and introducing a low-efficiency cost), and clearly spelled-out anonymity guarantees as well as full unobservability for local calls.

- **Symmetry.** We consider symmetry in the roles of the network entities. An anonymity system is intuitively "more symmetric" when all the participating entities have similar roles and responsibilities, and "less symmetric" if there are different roles, capabilities, and trust assumptions among the entities that participate in the routing.
  We thus first examine the overlap between the *roles* of end-users who initiate communications and relaying nodes. We distinguish three types of systems.

  – We classify a system as *peer-to-peer* (●· · ●· · ●), when end-users are expected (often even obliged) to operate as relaying nodes in order to use the AC network.

  – At the other end of the spectrum, in *client-server* (●· · ●) systems, users are not expected (often even forbidden) to operate as relaying nodes on order to use the system.

  – We call a system *hybrid* (●· · ○· · ●) if it combines characteristics of both *peer-to-peer* and *client-server* systems, *i.e.,* end-users may or may not operate as relaying nodes.

  These different levels of symmetry come with advantages and disadvantages [24]. Peer-to-peer systems can better scale as the number of users grows, because new users also increase the capacity of the network. Further, peer-to-peer networks are more resilient to node failures and have better availability properties. In client-server architectures, however, it is possible to run nodes more reliably and securely (as nodes are not necessarily run by laymen end-users), which in particular helps in handling liability issues with respect to complaints. Having end users run just client software has a lower cost for end-users in terms of resources, and offers opportunities for simpler, and thus often more usable, client software.

Routing Information

We now consider the information available to the entity (or entities) that decides on the route of a connection, and how that information is made available.

a) **Network View**. This determines the completeness of information available to establish a route.

- The routing decision-maker has a *complete view* (●) of the system if routing information about all nodes is available to her.
- The decision maker has a *partial view* (◐) of the system if the routing information available to her only covers a subset of the nodes that form the AC network.

A complete view allows the decision maker to choose among the full set of nodes. However, a partial view improves the scalability of the network, as the distribution of routing information for the full network may consume significant bandwidth and network resources. There are also some attacks that become possible when the routing decision makers only have a partial view of the network. For example, route fingerprinting attacks [32, 33] are possible if each user knows different subsets of routers. In these attacks, the initiator of a connection can be identified by the nodes that make up the route, since typically a very small number of users will know a certain combination of network nodes.

b) **Updating**. This determines how frequently routing information is updated.

- Routing information is updated *periodically* (🕒) if it is updated in predefined time intervals.
- Routing information is updated *event-based* ($\lightning$) if the updates are triggered by events in the network other than timeouts.
- No updating mechanism is in place (✗).

Second, we distinguish whether nodes are organized in a flat or a hierarchical structure with respect to routing. We call the resulting feature the *topology*:

– A network has a *flat* ($\cdots$) structure if every node has the same importance and rank when making routing decisions.

– A network has a *hierarchical* (✤) structure if nodes have different capabilities and priorities towards the routing algorithm.

Hierarchical structures are often introduced to improve efficiency and performance. However, a non-flat hierarchy can make the network less resilient to attacks, as the failure of a node that is placed high in the hierarchy has a severe impact on the performance of the network.
The third and last dimension of symmetry addresses the degree of *decentralization* of network services other than (but auxiliary to) the routing itself. Note that we are not considering *centralized* models because they are a single point of failure for surveillance and insecure by design.

– A network is *semi decentralized* ($\odot$) if it includes one or a small number of entities performing a service critical to routing (*e.g.,* compiling and distributing network directory information). This accounts for the fact that especially high levels of trust placed on these entities, which constitute more of a point of failure than a simple relay.

– A network is *fully decentralized* ($\bigcirc$) if the system design does not include entities that have to be especially trusted for the provision of functionalities that enable the routing. Fully decentralized systems have a better distribution of trust.

Communication Model

We finally consider features that describe the creation of anonymous routes.

a) **Routing Type.** This refers to the selection of nodes to determine a route.

- The routing decision is *source-routed* (●· · ·) if the initiator of the communication selects the set of nodes that will form the anonymous route.
- The routing decision is *hop-by-hop* (· · ● · ·) (also called "random routing") if the initiator only selects the first relay node, which in turn picks the second, and so on, until the message reaches its final destination.

Source-routing enables the initiator to pick nodes she trusts, and prevents adversaries from biasing the node selection towards compromised nodes. A variation of the basic source-routed model is found in some systems that provide receiver anonymity. In these systems, the initiator and the receiver select, respectively, the first and second halves of the route, which are joined in the middle at a rendezvous point. An advantage of hop-by-hop routing is that even if the initiator only knows a subset of nodes, her connections might be routed throughout the whole network, mitigating route fingerprinting attacks [32]. In literature, other node selection strategies have been proposed, which we have not taken into consideration such as dynamic routing schemes using distance vector routing (*i.e.,* [**?**]) and link-state routing (*i.e.,* [**?**]). Such algorithms are often disregarded for AC networks because of the predictability they offer, which is in conflict with anonymity.

b) **Scheduling.** This refers to the way a node serves incoming scheduling requests.

- *Fair* (≡) scheduling means that all types of connection are treated same.
- *Prioritized* () scheduling means that certain connections are given priority over others.

Prioritized scheduling can improve performance and reduce congestion. However, differential treatment of traffic may undermine anonymity as the traffic of different priorities would be distinguishable and thus not conform a single (larger) anonymity set. An example of prioritized scheduling is when the scheduling follows an economic model, which might mitigate flooding attacks [34].

c) **Node Selection.** This refers to the protocol features that determine which nodes are selected to be part of an anonymous route. The number of nodes that are selected to form the anonymous connection can either be fixed (deterministically) or be computed probabilistically according to some distribution.

- Node selection can either be *deterministic* (✓) or non-deterministic (probabilistic) (✗).

To characterize node selection, we consider the *selection set* that determines which nodes are eligible for being on the route, and the *selection (probability) distribution* that describes the likelihood of each of the nodes in the selection set being chosen for a route.

- The selection set may contain *all nodes* (Ⓐ) of the network.
- It may contain a *security-restricted subset* (🛑) of all network nodes, *i.e.,* a subset that is selected according to some *security-restrictions*, for example establishing that all the nodes in a route must be in different /16 IP subnets.

- It may contain a *network-restricted subset* (🌍) of all network nodes, *e.g.,* a subset aimed at guaranteeing the quality of the communication, by for example avoiding congested links and nodes.
- And finally, the selection set may be user-specific, considering *user preferences and trust assumptions* (☺).

We are left to define the selection probability with which individual nodes are chosen.

- The probability distribution that describes how nodes are selected may be *uniform* (⊛).
- The probability distribution is *statically weighted*, *i.e.,* weighted based on *general, static parameters* (◎), for example the bandwidth of the nodes.
- The probability distribution is *dynamically weighted* based on *state-specific dependencies* (❋), for example the nodes' response time.

Even for general parameters, weighted selection often requires frequent updates so they reflect the current state of the network. In other words, we consider parameters that are calculated in real-time to be *dynamic* biases, and parameters based on routing information that is unchanged until the next periodic update to be *static*. Uniform selection typically offers better anonymity levels, while weighted selection often improves performance.

2.2.2 Performance and Deployability

In addition to the routing characteristics identified before, we finally identify the following list of metrics that can be used to evaluate performance and deployability characteristics of AC protocols.

(a) **Latency.** In the literature, AC protocols are usually classified into two performance categories:

- Protocols with *low-latency* (L) incorporate no latency to the communication and typically support applications that require real-time communication (*e.g.,* web browsing).
- Protocols with *high latency* (H) do not require real-time communications and support applications that can tolerate a certain delay between requests and responses (*e.g.,* email communication).
- Protocols with *mid latency* (M) introduce a random delay and may induce a restricted latency; hence, these protocols support applications that can tolerate a restricted delay between requests and responses (*e.g.,* file sharing).

(b) **Communication Mode.** We distinguish two kinds of communication modes, depending on the longevity of individual connections.

- We classify protocols as *connection-based* () if routes between senders and receivers are maintained for a certain amount of time and used for exchanging multiple data transfers.
- If routes are created just to send a message and no state is maintained for further exchanges, then we classify a protocol as *message-based* (🖂).

(c) **Implementation and Code Availability.** This indicates whether or not a prototype of the protocol has been implemented, and if the code is publicly available, respectively. In both cases, the answer is either yes (✓) or no (✗).

2.3 Routing Classification of AC Protocols

In this section, we present a categorization of AC protocols. We have classified these protocols into four main families: (1) Mixnet-based protocols, (2) Onion Routing-based protocols, (3) Random Walk and Distributed Hash Table (DHT)-based protocols, and (4) DCNet-based protocols (5) Miscellaneous, containing a few protocols that do not fit into the aforementioned categories. A few protocols are presented in the most representative category, albeit they can technically fall under other categories as well, *e.g.,* Octopus and Torsk are DHT-based, but they also use onion routing. We summarize our classification of the routing aspects in two comparative tables (namely Table

We now discuss the AC protocols individually, starting with Mixnet-based protocols (from Section

2.3.1 Mixes

The idea of anonymous communication was originally proposed by David Chaum in 1981 [1] and initiated a new field of privacy research. The central concept proposed by Chaum is the use of *mix nodes*, or *mixes* in short. Mix nodes cryptographically transform messages so that they cannot be traced based on their content. Further, mixes shuffle ("mix") input messages and output them in a reshuffled form. Thereby, they hide the input-output relation between individual messages, such that an adversary is not able to establish a correlation between input and output messages. In Chaumian mixes, the mix node does not output the messages immediately upon arrival, but instead collects a certain number of messages (up to a threshold) into a so-called *batch*, which introduces a delay in message transmission. The mix shuffles input messages within a batch and flushes them out ordered lexicographically.

2.3.2 Mix Selection Strategies

In order to distribute trust, Chaum proposed to relay messages through a fixed sequence of mix nodes[3] called a *mix cascade.* Chaum proposes a deterministic node selection without specifying how the nodes are selected (node selection strategy) for mix cascades. He only suggests that certain factors such as the networks topology and user's trust can be used for mix node selection. In a mix cascade, messages are successively encrypted (in a layered fashion) with the public key of each mix in the cascade (see Figure

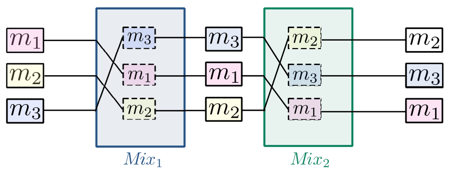


Figure 2.1: A mix cascade with two mixes

As the message is transferred from one mix to the next, the current mix peels off (decrypts) the corresponding layer (*i.e.,* remove one layer of encryption with its private key), obtains the inner layer together with the corresponding address of the next destination, and sends the

[3]In the literature, a sequence of mixes is usually referred to as *path* or *route*.

message to that destination. This procedure is repeated until the last mix delivers the data to its final destination. In order to receive replies for messages while staying untraceable (to obtain recipient anonymity [80]), return addresses are used. Chaum proposed to encrypt the address of the recipient of replies separately so that the respondent only needs to append the untraceable return address to her replies. The anonymous replies are also sent similarly in a layered fashion to the respondent. From now on, we refer to the encrypted return address block as the reply block. Note that in the case of the anonymous replies, the recipient of the reply is the routing decision maker.

In order to overcome a single point of failure in availability of mix cascades, *free-route* mix networks have been proposed. In free-route mix networks, the route is not fixed and any sequence of nodes from the network can be used for relaying messages. An important aspect in mix cascades and free-route mix networks design is how mixes are selected. Selecting mixes for a mix cascade or for a path in a free-route mix network may follow different strategies. Namely, a deterministic strategy, a uniformly random selection, or a variation such as random selection biased by network state, or reputation/reliability scores. When multiple mix cascades are available for the users to choose from, node selection has two dimensions: selecting a set of mixes for building the cascades, and selecting a particular mix cascade for relaying the messages. Moreover, predefined probability distributions and topological restrictions can also be taken into account for mix selection. Danezis [25] proposed the *restricted routes* mix networks that leverage the mix cascade model (*i.e.,* being less vulnerable to intersection attacks and being secure against global adversaries) and free-route mix networks (*i.e.,* being scalable). He proposes a mix network topology that is based on constant degree graphs (sparse expander graphs), where each mix only communicates with a few neighboring nodes based on a predefined probability distribution. Next, we review two variants of mix selection, one for free-route mix networks and one for mix cascades.

Mixes that fail, lead to further delays in mix networks, thus selecting reliable mix nodes can lead to better performance. Dingledine *et al.* [40] proposed to identify mixes that fail and use a reputation system for mix selection leading to more reliability and efficiency for the mix network. In their proposed system, mixes issue receipts for each received message. After a mix has sent a message to the next mix, if it is not receiving a receipt within a restricted time, it asks a set of witnesses to resend the message and receive the receipt and forward it to the original mix. The system establishes routing paths following the free-route node selection strategy, where the mixes are selected based on their past behavior (reputation score). Such a strategy suggests use of a non-deterministic node selection, biased towards mix nodes with high reputation scores. Mixes that have no positive ratings at all are avoided for mix selection. The main weakness of their scheme is that the reliability depends on the witnesses that need to be trusted, or at least a core group of trusted witnesses.

Unlike the previous system, which relies upon trusted global witnesses, Dingledine and Syverson [41] proposed a mix cascade protocol with distributed trust. The system they propose uses a reputation mechanism for rearranging mix cascades in order to obtain more reliable cascades. The construction of such cascade utilizes communal randomness and reputation scores provided by all of the mixes; therefore, there is no need of a trusted central authority. To mitigate the weakness of the previous work, mix nodes of a cascade act as witnesses for the reliability of their own cascade. All mixes submit random values to the configuration servers, which order mixes based on their reputation score and pick the top mix nodes to create a pool of mixes. From this pool, the mixes are selected randomly of mix cascade rearrangement. For each cascade, routing relevant information such as available bandwidth and expected waiting time are published.

Based on this information and the reputation score of the mixes, users choose mix cascade for their messages. Note that if the mix network is large, the network view might not be complete for the users.

2.3.3 Variations of Flushing Strategies

Flushing algorithm (or batching strategies) specifies the precise timing at when a batch of collected messages is flushed out of the mix in order to be simultaneously delivered to the respective recipients. Flushing strategies are analogous to the forwarding component of the routing and they highly influence the scheduling routing characteristic defined in Section

Mixes that delay messages individually, for example based on a certain probability distribution, and lead to continuous flushing are called continuous mixes. One example of continuous mixes is the *Stop-and-Go* mixes (*SG-mix*) [37] system. The initiator of a message assigns for each mix in the path a randomly selected delay (from an exponential distribution). The independent random delays that are assigned to each message make the performance and anonymity of each message independent of the other users in the system. However, a drawback of their system is that SG-mixes are vulnerable when incoming traffic is low [81]. Another type of flushing algorithms is pool mixes that only flush out a fraction of messages of a batch at each round, and keep the remainder in the memory of the mix (pool) for next flushing rounds. In pool mixes, the number of messages that are forwarded may be determined by deterministic or non-deterministic functions, and the message selection may be a uniformly random or weighted based on dynamic conditions (*e.g.,* based on incoming traffic). When the average delay of the messages is equal, pool mixes offer better anonymity since the anonymity set is bigger. Another advantage of pool mixes is that they are suitable for networks with fluctuating traffic load. Pool mixes, however, still need to specify *when* messages are flushed out and therefore combined with other flushing techniques such as threshold (described above) or time restrictions. Timed mixes enforce a time restriction for flushing out messages. The anonymity of timed mixes is vulnerable to low traffic since if only one message arrives before the time restriction is met, the mix provides no anonymity measure for that message. Moreover, a combination of the aforementioned flushing strategies can also be used by mixes [17,81]. For example, the two prominent *remailers*, namely Mixmaster [42] and Mixminion [43], use *timed dynamic pool mixes* as flushing strategies [82], which are a combination of timed and threshold pool flushing techniques, where the parameters depend on the network traffic. The flushing algorithm of Mixmaster has been characterized by generalized mixes [83]. We review these remailer protocols in Section

Next, we review some mix protocols from the literature that have been suggested for applications such as ISDN telephone, web browsing, and anonymous emails. In order to anonymize ISDN telephone communication with its intrinsic requirements on low-latency, Pfitzmann *et al.* [31] introduced the concept of *ISDN mixes*. An important feature of ISDN mixes is to maintain constant traffic in the network to avoid traffic analysis. ISDN mixes use threshold mixes. To obtain sender and receiver anonymity, ISDN mixes use two mix cascades, each built by the sender and receiver, respectively, which are connected either by a connecting mix; when used in long distance communications by the long distance network operators. Initially, a broadcast takes place to exchange the connecting details and the time where the communication takes place. To achieve constant traffic, a number of ISDN channels, with an equal amount of messages, need to start and end their communication at the same time (in a so-called *time-slice*). However, this is time-consuming and would lead to blocking the connection, which is not suitable since ISDN mixes use narrow-banded channels and were designed for low-latency communication. In Table

A real-world realization built on ISDN mixes are *Webmixes* (also known as JAP) [38,39] designed for real-time Internet applications, passing the traffic to several available mix cascades. In Webmixes, the mixes transform the messages cryptographically and re-shuffle their order before flushing them out. However, messages are not delayed by flushing strategies. Webmixes use an adaptation of the time-slice method introduced by ISDN mixes. Routes in Webmixes consist of *JAP proxies*, which are local software at the users, one (or several) mix cascade(s) consisting of reliable and high capacity mix nodes, and a cache-server. Web requests are sent from the users JAP proxy through the mix cascade and the cache-server, and furthermore delivered to the destination server. The web replies are sent back the same route and a copy of the reply is saved at the *cache-server*. Hourly mix cascade information is published by so-called *Info Servers*. Users can choose among the published mix cascades by the info servers. ISDN mixes, real-time mixes, and Webmixes have a deterministic node selection to build the mix cascade, where nodes selection for the cascades relies on the network state.

2.3.4 Prominent Applications of Mixes: Remailers

The original concept of mixes has an immediate application to *high-latency* remailer systems for providing anonymous e-mail service.

Babel [36] aims at mitigating traffic analysis attacks by delaying only some messages of the batches. Babel uses independent forward routes and return routes. Forward routes may include a reply block (where the return route mix addresses are encrypted in a layered fashion) that may be used by recipients for anonymous replies. Forward routes are considered to have better anonymity; one of the reasons for this is that reply blocks enable replay attacks on anonymous replies [84]. Babel introduces *intermix detours*, where mix nodes choose a random sequence of mixes and relay the message through them before forwarding the message further to the next mix of the original route. In Babel, the flushing algorithm uses time restrictions (intervals) and thresholds for flushing out messages. Another technique Babel proposes to use is *probabilistic deferment*, where a number of messages (determined by a biased coin) are delayed at each mix (this is similar to pool mixes). Babel proposes to use of free-route mix networks, where mixes are chosen uniformly random for each route by the user. However, there were no details given how routing information is communicated to users.

Mixmaster [42] is an anonymous remailer, where mixes transform messages cryptographically into uniform sizes by adding random data at the end of each data packet. If a message is too large, Mixmaster splits up the message to achieve uniform sized packets and sends these packets independently of each other through a series of mixes, which do not necessarily need to be all the same. Only the last mix needs to be the same for all packets of one email message, which has been split up before. Mixmaster adopts a free-route path selection, the node selection is not specified by the protocol, though statistics on the reliability of mixes can be used to bias node selection [25]. Though the Mixmaster protocol did not specify details about maintaining mix information, later implementations of Mixmaster adopted an ad hoc scheme for distributing routing information [43]. One the main weaknesses of Mixmaster is that it only guarantees sender anonymity, since reply blocks are not used in Mixmaster.

Mixminion (or Type III remailer) [43] are widely considered as the state-of-the-art remailer. To guarantee equal routing information for all senders, Mixminion deploys a group of redundant and a synchronized system of *directory servers*, which was not considered in the Mixmaster design. Note that we disregard the directory servers synchronization for our classification in Table

2.3.5 Onion Routing

Onion routing [7] [85] is designed for anonymizing connections for applications with low-latency constraints, such as web browsing.

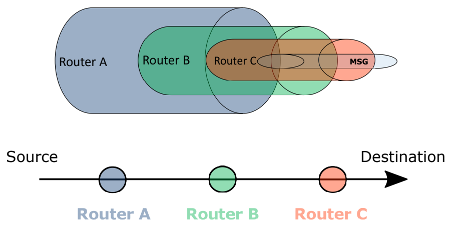


Figure 2.2: The concept of onion routing

An onion routing network consists of a set of nodes so-called *Onion Routers (ORs).* Users choose an ordered sequence of ORs to establish a bidirectional channel, so-called *circuit*, for relaying their data through the onion routing network. The communication is encrypted in a layered fashion and the ORs in the circuit each can decrypt their corresponding layer. When the communication is relayed by an OR in the circuit, the OR removes the corresponding layer of encryption and forwards the data to the next OR in the circuit (see Figure

2.3.6 Onion Routing-based Protocols

Onion routing is used in Tor [8], which constitutes an extension of the original onion routing design, with some modifications to achieve better security, efficiency and deployability. The Tor network, an open source and free to use the framework, consists of a large set of volunteering routers (at the time of writing, there exist more than 7000 routers [9]). The network is mostly connected because routers can connect to any router from the Tor network, except for connections between routers located in the same IP /16 subnet space, which are not possible. Tor's services are used daily by approximately 2,000,000 users [9]. Each user runs a piece of software called Onion Proxy (OP) that manages all Tor related processes, *e.g.,* establishing circuits or handling connections from user applications. Tor deploys a group of well-known and trusted authoritative servers that publish on a regular basis (typically, every hour) a list of all active Tor nodes with their characteristics, *e.g.,* estimated bandwidth, IP addresses, and cryptographic keys. This list is called a *consensus*. After the user has obtained the consensus, the OP of the user chooses an ordered set of usually three ORs to build a circuit. The first node in a circuit is called the *entry* node, the second node is the *middle* node, and the last node in the circuit is the *exit* node. The first node that is selected is the exit node, then the entry node of the circuit is selected, and last the middle node of the circuit is selected. After selecting a set of ORs, the OP contacts the entry node and builds a circuit with it. This newly created circuit is used to contact the middle OR to extend the circuit and similarly through the middle node the exit node is contacted to extend the circuit. The established circuit can now be used to anonymously relay data.

In 2002, Wright *et al.* introduced the predecessor attack [86] on onion routing. To defend against this and related attacks, selecting a small set of nodes was introduced for Tor [87]. Previously, each user maintained a list of 3 randomly pre-selected (so-called *guard*) nodes with high bandwidth and uptime. This list was updated every 30/60 days and the user could choose uniformly random an entry node from this list for each path construction. This has changed recently because Tor is starting to let each user select only one fixed entry guard node for 9 months [88].

In the early onion routing design, it was suggested to select the nodes uniformly random [89]. Due to performance considerations, Tor's routing policy does not select nodes with the same probability, but rather preference is given to high-bandwidth nodes. The likelihood that nodes are chosen for certain positions in a given route depends on the ratios of overall node bandwidths and node such as the IP addresses and whether they can be selected as entry node or as exit node. Moreover, some additional bandwidth weights are used to balance off the node selection. As mentioned before, a further development in the routing policy is to disallow a communication to pass through two nodes within the same /16 subnet IP address. The implications of these changes with respect to structural node corruption have been recently explored by Backes *et al.* [49, 90].

Next, we review two prominent attacks on Tor's routing. Murdoch *et al.* have proposed a traffic-analysis attack using timing information to identify Tor nodes and to infer traffic load to a specific initiator. Their investigation shows a degradation of Tor's anonymity against such attacks. They furthermore propose some strategies to prevent the risk of such attacks, mainly by increasing communication latency [91]. Bauer *et al.* have proposed a traffic analysis attack aim at decreasing the anonymity of Tor [28]. Their attack investigates the load balancing that is performed by Tor, where high bandwidth nodes are preferred in the node selection strategy. They show that performance optimization impairs the anonymity of Tor against end-to-end traffic analysis attacks.

Since Tor has been proposed, there has been a great deal of research on extending Tor's routing strategy. The proposed extensions to the Tor routing protocol aim mostly at improving either the achieved anonymity of Tor, or the performance that Tor users experience.

Improvements to Tor's anonymity have been often realized by aiming at an improved node selection. For example, improving anonymity by using better weighting at the node selection phase has been proposed in [48] and [49]. Involving AS-level information in the node selection has been proposed by [23] and [44]. Moreover, offering the user a tuneup option between uniformly random node selection (for high anonymity) and weighted random node selection with a bias towards high bandwidth nodes (for better performance) has been suggested by Snader and Borisov [46].

Tor's performance problems have several causes, and hence suggested improvements aim at different aspects of the Tor routing protocol. One cause of Tor performance is high congestion [13, 92], often caused by bulk traffic, which induces high latency for interactive/web traffic. Several solutions to solve the problem of high waiting times for interactive traffic have been proposed. One possible solution is to increase the number of connections between two nodes [50–53], which can be used to separate interactive and bulk traffic into different connections. Another solution is to prioritize interactive traffic in the scheduling phase [54] [55]. An alternative solution is to improve how Tor's resources are used by improving node selection with a more realistic estimation of the available bandwidth of nodes [48]. Furthermore, another solution to Tor's congestion problem is to enforce avoiding congested nodes at the node selection phase [47]. Another reason for Tor's high latency is circuitous paths [44]. To solve this problem, node selection strategies have been proposed that take the destination between chosen nodes into account [44, 45, 48].

The scalability of Tor has also been subject to new proposals for the Tor routing protocol in the literature. One proposal to tackle scalability issues is to give the user only the information about the necessary nodes for path construction and to hide the complete view of the system from the user by either managing Tor nodes as a DHT table and using Kademlia for node retrieval [60], or by using private node retrieval [56].

2.3.7 Random Walks, Structured and Unstructured DHT-based Protocols

In this section, we review *random walk protocols*, where the communication is relayed randomly through the network. We consider a protocol a random walk protocol if node selection is hop-by-hop routed and a random selection. Random walk protocols are often combined with peer-to-peer network structures.

Crowds [57] is one of the early AC systems designed for anonymous web browsing. The key design feature of Crowds is a random peer selection. In Crowds, all nodes are grouped into so-called *crowds*; all nodes within a crowd might connect to each other for relaying a communication. Each node in the crowd is called a *jondo*. A so-called *blender* is responsible for managing and administrating nodes. Crowds has a peer-to-peer structure since all users of the system are nodes themselves. The user randomly selects a node and sends her message (*i.e.,* website request). Upon receiving the request, this node flips a biased coin to decide whether to send the request directly to the receiver or to forward it to another node selected uniform at random. This continues until the message arrives at the destination. The server replies are relayed through the same nodes in reverse order. Wright *et al.* showed that Crowds is vulnerable to so-called predecessor attacks [86, 93]. In order to prevent such type of attacks, Crowds suggested to employ static route (a user keeps the route for a while) such that an attacker does not have multiple routes to link to the same jondo [57]. However, even keeping routes static for a day is not enough to prevent predecessor attacks [84].

MorphMix [58, 59] is a dynamic peer-to-peer AC network. Technically, MorphMix establishes circuit-based connections using layered encryption, where the anonymous route is established iteratively by the nodes on the route. Each node is typically only aware of a set of network nodes, which is not necessarily covering all nodes. In order to avoid repeated connections with the same set of nodes, a node has to forget about nodes it has not been connected and constantly require new node information. After an initiator selects the first node, she selects randomly a witness for each hop thereafter, randomly chosen from the nodes in her local database. She asks the next hop to extend the route with the assistance of the witness she has chosen, where nodes propose a set of candidate nodes for the next hop and the witness chooses one of them. To prevent path compromise, nodes can only propose nodes with different IP prefix to her own IP address to the witness. The witness should not be selected from the nodes to which the initiator is connected currently to avoid initiators being identified by witness nodes. In order to mitigate guessing whether a node was initiator by the next hop, the initiator adds random delays to her communication before forwarding them in the tunnel establishment phase.

Efficiency is one of the main problems in random walk protocols. In the next section, we review DHT-based protocols, which aim at efficient node lookup and selection. Random walk protocols can employ DHT lookups to gain better efficiency (*e.g.,* AP3 protocol [62]).

DHT-based Protocols

In distributed systems, where there are network administrators, a challenge is to locate a node. One solution is to use Distributed Hash Tables (DHTs) to manage the distributed nature of the data (relaying nodes or distributed storage). Generally, DHT refers to a trust-distributing, structured-data management model for storing (value, key) pairs and is accompanied with key-based lookups for locating the corresponding stored value (see Figure

DHT structures enable efficient routing even when the peers of a DHT structure keep only information (key-value pairs) of a partial subset of all the other peers of the DHT structure; this, in turn, leads also to improved scalability of such systems. Another important feature of DHT-based structures is having better load balancing. For systems, where nodes have only a partial view of the structure, hop-by-hop routing is preferable. Some AC protocols use randomness in the routing strategy besides the classical lookup method. For example, node selection is carried out by selecting a random key and by then using a classical lookup method (an adaptation of Chord, Kademlia, or Pastry) to find that key. Next, we review AC protocols that use an adaptation from Kademlia, Chord, Pastry for their node lookup (considered as structured DHT-based protocols). We proceed by reviewing independent DHT-based routing proposals for AC that are considered unstructured DHT-based protocols. We start with *AP3* [62], a random walk protocol aiming at providing anonymity when a large part of the nodes is compromised. AP3 uses the same routing strategy as Crowds, with the difference that the node information is retrieved using Pastry and that the node does not have a complete view of the system.

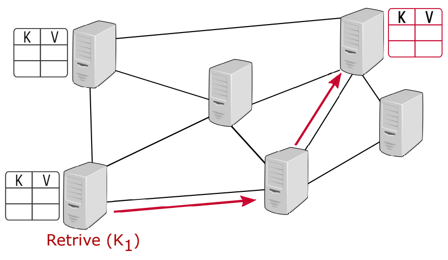


Figure 2.3: The concept of distributed hash tables

Next, we review two protocols that aim at replacing node selection of source-routed protocols such as onion routing with structured DHT systems making the suitable to be combined with onion routing. *Salsa* [63], proposed by Nambiar *et al.*, aims at providing scalability and preventing malicious colluding nodes to be able to bias routing. Salsa virtually divides nodes into groups, which are organized in a binary tree form. For routing, simultaneous redundant lookups and bound checking are used in order to avoid malicious nodes returning wrong addresses. The lookup queries are carried out similar to the Chord lookup in a recursive fashion. qIn Salsa, the routing information that is available to each node is partial; however, the tree structure allows nodes to carry out source-routing.

McLachlan *et al.* have proposed *Torsk* [60], a peer-to-peer AC protocol, replacing Tor's node selection and directory service with a DHT design. It aims at providing better scalability for Tor. Their design uses DHT tables for node selection by using a randomly chosen key that is looked up in the table using Kademlia. To secure lookups, Torsk uses the "root certification" approach proposed by Myrmic [97] and randomly selected secret "secret buddies."

Panchenko *et al.* proposed *NISAN* [61], an AC protocol that aims at achieving high scalability and preventing adversaries to bias routing. NISAN uses iterative search to select nodes randomly for constructing anonymous paths. It uses an adaptation of Chord, where the node lookups are aggregated. Moreover, NISAN hides the node that it is looking up, by requiring the complete routing table and enforcing bound checking to further prevent selecting nodes from routing tables, which were manipulated by malicious nodes.

Octopus [64] aims at providing security by preventing malicious nodes to be able to bias routing. It also aims at providing anonymity by hiding which nodes have been looked up for anonymous paths. For routing, Octopus uses iterative lookups by sending the query to the closest node to the searched key in the local routing table and then retrieving the routing table from that node until the node containing the corresponding value to the key is found. Node selection is carried out in two phases. In the first phase, nodes are selected by the path initiator (user). In the second phase, the last node selected in the first phase chooses the remaining nodes. Therefore, Octopus is not purely a random walk protocol. After establishing anonymous paths, Octopus splits queries to different paths and adds dummy traffic to hide the real queries among them. Furthermore, as security measures, Octopus enforces bound checking on the received routing tables to prevent using manipulated routing tables, and it proactively tries to identify and remove malicious nodes.

Next, we review two file sharing protocols that use DHT for routing file requests and their responses. They, however, use unstructured routing. Clarke *et al.* proposed *Freenet* [65], a peer-to-peer censorship-resistant system for sharing storage space. Freenet offers strong decentralization in order to provide privacy and robustness against attacks. The key design feature of Freenet is based on storage replication and plausible deniability. Files are stored multiple times at the nodes, are indexed by binary file keys, and can be looked up by their corresponding key. Each node has a dynamic routing table including the node information with the stored keys. The original design uses a heuristic deterministic routing using potentially all Freenet participating nodes choosing mostly neighborhood nodes (currently called Opennet mode). Freenet uses an adaptive routing using DHTs with keys that are location-independent. Three methods are used for key construction: keyword-signed key, signed-subspace key, and content-hash key (for more details see [65]). The routing table is updated periodically to achieve better performance. The replication of files provides resilience against node failure and node overloads. In the Opennet mode, a heuristic-based deterministic routing approach (a distance-directed depth-first search with backtracking) is used [66, 98]. When a file request arrives at a node, including a key and a value for hops-to-live, if the file is not stored locally, the node looks up the node with the nearest key in the routing table and forwards the file request to the corresponding node. The node receiving the request repeats the process until either the file is found or the hops-to-live is reached. If the requested file is found, the node forwards the file to the node from which it has received the request, stores a copy of the file locally and updates its routing table in order to optimize routing for future requests. If the node that is contacted is not responding, the node sends the request to the node with the second-nearest key. If that node is also unresponsive, it contacts the third-nearest one, and so on. If the file is not retrieved within the hops-to-live number of hops then the search is aborted and the file requester is informed. The nodes that are forwarding the requested file back to the file requester change randomly the sender address, providing reasonable deniability for the node that has stored the file [65]. The Opennet mode was vulnerable to various attacks. In particular, nodes participating in Freenet were not protected, and an attacker could easily find out whether a router is a participating Freenet node. In the Darknet mode, such shortcomings are addressed. In 2010, Freenet has been extended by a membership-concealing Darknet mode, where trusted connection are used for routing [66]. In the Darknet mode, the user chooses the nodes from her trusted nodes [66]. The routing table is consisting of nodes derived from a fixed graph, which is the social graph of the node. In the Darknet mode, the routing table is not optimized during time and cannot include nodes that are not derived from the social graph of the node. Since the Darknet mode is based on the trusted network of a user, the structure of the network is following Kleinberg's small world model [99]. The relaying nodes only know their predecessor and the successor in order provide privacy. In Freenet, the data is encrypted using symmetric encryption. The key is transferred either with the address or in the header of the file request [65].

GNUnet [67] was originally designed as a peer-to-peer censorship-resistant content sharing system, but has been expanded into other applications such as anonymous file sharing using the *GAP* protocol [68]. GAP aims at providing requester and responder anonymity for file search and file sharing. In GAP, a node that is relaying a message in the forward route has the option to "drop out" from the reply route (for example due to network state and its own heavy load) and when the reply is sent back, the node is over-jumped. Moreover, when queries arrive at the nodes, they can be dropped if the node has already too much load. Routing in GAP uses credit rating scheme, where relaying requests and replies increases the credit and sending uses the credit. The credit score is local at each node. In GAP, the file request can either be sent to newly selected nodes or to a node where there is already a connection established. This is decided based on the node's current CPU and load, the credit rating and a random factor. The node selection is random with a bias towards nodes, which have a closer identifier to the hash value of the file that is queried. Moreover, the network activity is also taken into account in node selection (giving preference to "hot paths"). Unlike Freenet, GAP uses a time-live restriction to avoid routing loops; when time-to-live is reached, the query is forwarded directly to the destination with a certain probability. For flushing in GAP, nodes use a combination of timed and threshold mixes for flushing batches of messages, where the time restriction is selected randomly.

2.3.8 DC Networks

The idea of *DCnets* (Dining Cryptographers Networks) was first proposed by Chaum [69] and later revisited [70, 71]. DCnets are an important alternative to mix-based schemes and their extensions due to their resistance against traffic analysis attacks. DCnets offer non-interactive anonymous communication using secure multi-party computation with information-theoretically secure anonymity, guaranteeing sender anonymity while enabling all participants to verify the final outcome. The key concept is that every participant outputs a message that is disguised by XORing them with the keys the participants are sharing pairwise with other participants. The participants combine their outputs and share the output with each other (*i.e.,* they broadcast their output). When the encrypted messages are combined, the keys cancel each other out, and the message is revealed; however, the sender remains unknown (see Figure

The DCnet concept can be generalized, to transmit large messages simply by repeating the protocol as desired [71]. DCnet expects all participants to be involved in every run of the protocol and requires pairwise shared keys between the participants. Moreover, every participant needs to disclose the same number of bits in each round. The participants can share the keys for every round, or they can repeatedly use the same key; this makes DCnet unconditionally or computationally secure, under the assumption that the protocol is executed correctly. Moreover, DCnets also have practical challenges, such as the message transmission or avoiding collisions (unintentional) and disruptions (intentional collisions). Since a collision invalidates the message (bit), when only one-bit messages are sent, just one of the participants may transmit at a time (although all participants are involved in each round). If multiple participants want to send messages within a block of communication, they need to occupy different positions within the block. One proposed solution is to randomly pick a position (slot) in the block that is going to transmit and reserve the position in earlier rounds (pre-transmission round). However, this

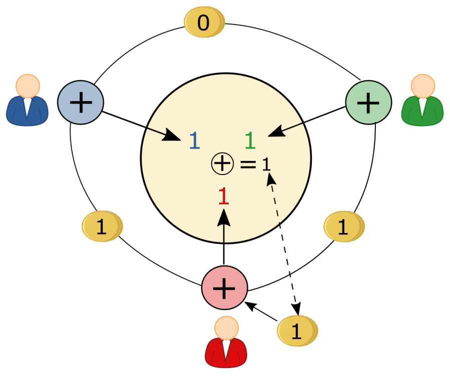


Figure 2.4: The concept of DC network

might only shift problem and again in the reservation round collisions might occur. The basic DCnet does not prevent any disruption, such as actively blocking participants from sending the message; hence, it is susceptible to anonymous DoS attacks. To partially address this problem, some solutions to detect disrupters in DCnets have been proposed in the literature [100, 101]. Furthermore, recovering from a fault is only possible by re-broadcasting the messages.

Chaum proposed in his DCnet to either use a ring topology for sharing the messages or use broadcast to transmit messages to all participants at once. The ring topology solution has a the problem of detecting the disruptions because malicious participants can adapt their answers to other participants answers to avoid being detected. Basically, if two users submit reverse bits, they cancel each other out and the disruptions remain undetected. Other topologies that have been proposed for DCnets are tree [102] or star topologies [103]. The broadcast solution has the problem of being expensive and introduces the problem of collision. The major limitations of DCnet are the strong assumptions that they require: first, participants follow the protocol honestly and are expected not to collude; second, unconditional sender anonymity is guaranteed only if there is an unconditional secure channel between every pair of participants. Furthermore, DCnets are vulnerable to Sybil attacks [104].

Herbivore [72] is built on top of DCnets aiming at better efficiency and scalability and managing churn. To improve scalability, Herbivore breaks down the participants into smaller groups called *cliques*, a message can only be traced to a clique but not to the corresponding sender/receiver within their clique. Within a clique, participants are organized in a star topology, where the central node relays all messages between members of a clique. The central node is changed for each new round of communication. For inter-clique communication, the cliques are connected to each other in a ring topology. For locating cliques, Herbivore employs the Chord protocol [95]. In order to mitigate intersection attacks, nodes departure from a clique can be vetoed by the node that is in the middle of a long-run transmission.Although authors claim low-latency, we decided to classify the protocols as being *high latency* since it contains a central node that has to wait for messages from all other nodes in the clique. One of the main weaknesses of Herbivore is that smaller anonymity sets are achieved and the applications have a time restriction based on the cliques lifetime. Moreover, the star topology makes the design vulnerable to DoS attacks.

Dissent (Dinning-cryptographers Shuffled-Send Network) [73] is a latency-tolerant protocol for AC. It is the first protocol that provides accountability for a small-size group, and also maintains integrity. Dissent is built on top of DCnets, but relaxes the aforementioned assumption that all participants follow the protocol correctly. In Dissent, anonymous communication is guaranteed for members of a group. Apart from the multi-party computation and layered encryption to hide the sender of the messages, to solve the collision problem, each group member influences the position of the messages of other group members in the final transmission block. Dissent consists of two sub-protocols: a *shuffle protocol* and a *bulk protocol*, In the bulk protocol, each member creates an assignment table for each of the other member, so-call *message descriptors*. The shuffle protocol is used to shuffle these messages descriptors. Based on these message descriptors, each participant inserts her messages to a cipher stream, which is a slice of the message block that needs to be transmitted. The shuffle protocol functions similar to mix cascades, where each participant receives the set of message descriptors (which were encrypted in a layered fashion) and shuffles them and passes them over to the next participant. Thereafter, each member transmits one cipher stream. When these cipher-streams are combined, a vector of concatenated messages is obtained. Dissent uses broadcasting for intermediate runs of its protocols such as sharing keys. However, the final cipher streams are not necessarily broadcasted, and can be sent to a single group member or a non-member node. Hence, Dissent primarily only guarantees sender anonymity and further protocol setup details determine whether recipient anonymity is also achieved. To mitigate untraceable DoS attacks (disruptions), go/no-go messages and blame phases are used in Dissent, which identify collisions and malicious participants and enables accountability.

Wolinsky *et al.* have extended Dissent to improve scalability and efficiency [74]. They propose to group participants and use designated servers, where the group members share keys with these servers instead of each other (the network consists of server nodes and participant nodes). In the basic version of Dissent, the group size was restricted; however, in the extended version, the participants may form larger groups, though the servers consist of a significantly smaller group, while still being not completely centralized to avoid the single point of failure. Hence, the extended Dissent builds an asymmetric topology for key sharing. At least one of the servers needs to be honest to prevent compromises. While latency introduced at the shuffle protocol made the basic version of Dissent unsuitable for interactive and low-latency applications, the extended Dissent, if used in a local-area setting, can be suitable for low-latency communication.

2.3.9 Miscellaneous Protocols

Tarzan [76, 77] is a peer-to-peer anonymous fully decentralized IP-level network overlay. All participants are peers; they are all potential originators of traffic, and also potential relays. Tarzan nodes do not implement any mixing strategies and simply forward incoming traffic. After the initiator node selects a set of nodes (preferably from existing connections from previous communication rounds) to form a route through the overlay network, a tunnel via these nodes is established for relaying communication. Unlike the early design of the protocol [76], where the peers only needed to know about a random subset of nodes, the final version [77] introduces a *gossip-based* protocol based on the Name-Dropper protocol [105], where more node information is requested from randomly chosen nodes. The aim is to gain information about all available servers in the network to avoid attacks that are facilitated due to churn, such as fingerprinting attacks [32]. Node information is stored in a ring model and lookups are carried out using the Chord algorithm [95]. The initiator only selects nodes randomly from distinct IP subnets, a three layer hierarchy selection is used to make sure nodes are from distinct subnets.

I2P [106] is a distributed overlay network, originally aimed at enabling anonymous communication between two nodes within the I2P network. Note that currently there is a service built on top of I2P to allow getting connected to web servers [107]. Currently, the number of I2P routers is estimated to be between 40,000 and 50,000 [108].

The network metadata (containing router contact information and destination contact information) is distributed among a subset of all nodes so-called *floodfill* nodes, and is managed using DHT structure by employing Kademlia for node lookups. At bootstrapping, users obtain a list of I2P peers from websites and then contact two floodfill routers from the list and requests router information that is available to that floodfill node. In order to mitigate that malicious floodfill nodes are not biasing node selection by providing manipulated router information, router information is stored at eight floodfill nodes [109].

Nodes are categorized into tiers (called *peer profiling*) based on the previous performance (response times) and reliability (uptime) of nodes. Three main types of tiers are defined in I2P: high capacity, fast, and standard. The routing protocol of I2P, so-called *Garlic Routing*, is source-routed with a randomized node selection biased towards faster nodes [79].

In I2P, communication channels are unidirectional and called tunnels; tunnels for outgoing traffic are called *outbound* and tunnels for incoming traffic are called *inbound*. Each user maintains a number of inbound and outbound tunnels; outbound inbound tunnels of other users can be retrieved from the floodfill nodes. When users want to relay communication to each other, the nodes in the chosen inbound and outbound tunnels shape the relaying route. Moreover, there are two types of tunnels in I2P – client tunnels and exploratory tunnels – for which different peer selection strategies are used. Client tunnels are used for application traffic, and exploratory tunnels are used to send administrative information. For client tunnels, peers are selected randomly from the nodes that are categorized as fast-tier nodes, which is done locally by the client using previous measurements. For exploratory tunnels, peers are selected randomly from the set of nodes that are categorized as standard tier.

The communication through I2P is protected using *garlic encryption*. Garlic encryption is very similar to onion encryption, with the difference that multiple data messages may be contained in a single *garlic message*.

2.4 Discussion

2.4.1 Routing Features: Commonalities and Differences

In this section, we discuss commonalities and differences between the investigated classes of AC protocols with respect to their routing characteristics. The discussion is grounded on our classification presented in Tables

Mixnet-based protocols, as classified in Table

Generally speaking, mix cascade networks employ rather synchronized connection because messages are sent in batches and mostly depend on their flushing algorithms in a timely schedule. For example, timed mixes lead to synchronized message transmission. Recall that the flushing algorithm in Mixmaster and Mixminion partially uses time restrictions. However, we consider these two protocols with asynchronous message transmissions due to the possibility that low traffic might lead to a threshold restriction instead of a time restriction. As for free-route systems, in SG-mixes, message transmission is also synchronized due to assigned time ranges by the routing initiator. Nevertheless, these timing ranges are not coordinated with other users or mix nodes. Dingledine *et al.*'s proposal for a reputation system for mixnets [40] also uses a synchronized message relaying to enable verifying the correctness of the routing process.

In the mix protocols, node management has not been always specified in the protocol description. For example, in Chaumian mixes, the view of the routing decision maker is not discussed; however, it can be implicitly deduced that it is complete. The anonymous remailer Mixmaster does not discuss node management either; however, the later implementation uses ad hoc systems, which suggests a partial view [43]. The remailer Mixminion defines a node management strategy to insure a complete view for the routing decision maker.

Source-routing is one of the inherent routing features of mix cascade protocols because the routing paths are fixed beforehand. Choosing the mixes for the mix cascade might be either deterministic such as in the case of Webmixes or non-deterministic such as in the case of Reliable mix cascades.

Flushing algorithms do apparently impact scheduling. Note that some protocols in Tables

As mentioned in Section

All mix cascade protocols are high latency AC networks and have a message-based communication mode; exceptions are ISDNs, Real-time mixes, and Webmixes, which are designed for low-latency applications, such as web browsing. Note that the latencies might be restricted, for instance in case of Stop-and-go mixes, where the delays are randomly selected from a restricted time range.

Onion routing protocols, as classified in Table

One inherent routing feature of onion routing protocols, due to preventing additional latency, is to have no synchronization, which makes such protocols sensitive to timing attacks and global adversaries. Another inherent feature is that all onion routing protocols have a client-server model, which improves their usability and leads to a higher number of users, thus contributing to better anonymity for onion routing protocols [111]. They are characterized as complete network view due to a central authority, which distributes the list of Tor routers. One exception is [56], which realizes private node retrieval and thereby constrains the decision maker's view of the network. A complete view helps against adversary biasing node selection and is preferred in source-routing in order to prevent the decision maker to choose from a smaller set of nodes. Further inherent routing features concerning the communication model include routing type, scheduling, determinism in the node selection, and the selection set. The exceptions here are [54, 55], where they suggest a prioritization at the scheduling phase in favor of interactive traffic in order to reduce delays that interactive users might experience.

Node selection in all onion routing-based protocols is non-deterministic. This is important since the Tor network consists of volunteers and it is very likely to have a fraction of malicious nodes among them. A non-deterministic node selection reduces the chances of consistently selecting malicious nodes. Since the adversary is assumed to be local, a non-deterministic node selection makes targeted surveillance harder.
Furthermore, the node selection probability is generally weighted using static parameters, except for a few approaches that dynamically adjust weights, *e.g.,* for balancing security versus performance [46] and for avoiding congestion [47, 53]. Onion routing protocols are low-latency and have circuit-based communication mode, which are both inherent routing features of these protocols. Although we classify Tor as a protocol where the routing decision maker has a complete view, it is worth mentioning that the unlisted relays, known as *bridges*, are not part of this view.

Next, we discuss random walk protocols and DHT-based protocols. Crowds are Morphmix are two of the early random walk protocols that were proposed for anonymous communication. However, they present conceptual differences in terms of routing features. Both *Crowds* and *Morphmix* have fully connected topologies since every node may build a connection with every other node, resulting in better availability of the system, which leads to a bigger attack surface for timing attacks.

The path length in Crowds may vary and is determined in a non-deterministic manner to make simple timing attacks harder for external, local, and passive adversaries. Still, this does not necessarily hold for the case that at least one of the nodes in the path is malicious. In Morphmix, the initiator does not select the nodes of the route herself, rather decides on the number of nodes and establishes the connection.

Crowds is semi-decentralized because routing information of nodes is distributed by a central entity (the blender), which introduces a single point of failure with respect to node administration. Morphmix, however, has a fully decentralized structure. The network view is complete in Crowds, which, on the one hand, protects Crowds from eclipse attacks and on the other hand, is important since Crowds has a hop-by-hop routing type that makes the node selection sensitive to be biased by adversaries. In Morphmix, the network view is partial, and therefore, witnesses were introduced to prevent the biased node selection. Moreover, an inherent feature of random walk protocols is that the node selection is non-deterministic. In Crowds, each node is chosen from the set of all nodes based on a geometric distribution [112]; whereas, in Morphmix, the initiator knows a subset of nodes.

An inherent routing feature of DHT-based protocols is partially connected topology and a partial network view. The routing information is distributed among nodes and no single node has the complete list. Such a design increases the scalability of the protocols. A partially connected network topology makes DHT-based protocols less resilient against DoS attacks, which aim at disconnecting the network as much as possible compared to onion routing protocols. The connection direction is bidirectional for the majority of protocols with two exceptions. The exceptions are the file sharing applications Gnunet and Freenet Opennet mode.

Generally, DHT-based protocols are fully peer-to-peer protocols. There are two exceptions in this category: namely, Torsk and Salsa, where the first one has a hybrid role structure while the latter one allows both hybrid and fully peer-to-peer role structures. For being partially connected, DHT-based protocols provide a partial view of the network to the routing decision maker. Note that this may introduce a series of attacks. Examples of attacks against protocols that provide only a partial view of the network to the routing decision maker are route fingerprinting attacks [32], and route bridging attacks [33]. Another series of attacks, which might be possible due to a partial network view, are attacks that aim at disconnecting target nodes from the rest of the network, such as eclipse attacks [113].

Most of the DHT-based protocols are characterized with a hop-by-hop routing type. Exceptions are NISAN, Salsa, and Octopus, with source-routing. In Octopus, there are two decision makers for node selection; the path initiator who decides only about a segment of the path and the last node of that segment, which initiates the rest of the path. In our study, we could not find much information about the scheduling of DHT-based protocols, in particular for protocols that have not been deployed. Most of the DHT-based protocols have non-deterministic node selection, again here exceptions are the file sharing applications, where the routing path does not need to be anonymous.

The set selection for DHT-based protocols is, in most cases, all nodes within the routing table (*i.e.,* all nodes available to the decision maker). However, there are two exceptions: Torsk, where the set selection is restricted by security and network restrictions, and Freenet in the Darknet mode, where the set selection is based on trust assumptions of the user. For most of DHT-based protocols, the selection probability is uniform, exceptions are Freenet and Gnunet. Both protocols do not aim at providing unlinkability [80] nor they hide that a user is participating in the network. Nevertheless, they hide the role of the peer in the network. Most of the DHT-based protocols are message-based except Torsk, AP3, and Salsa.

Next, we discuss the DCNets protocols. *DCNet-based protocols*, as classified in Table

In order to improve efficiency and performance, some DCNet-based protocols [72, 74, 75] have been proposed, which vary in their routing features. Unlike the first group, in these protocols, the network structure is partially connected. For example, in Herbivore, participants are organized in star topologies, which are then connected in a ring topology. The organization of the nodes yields a hierarchical structure for the second group of DCnet protocols. Moreover, in the extended version of Dissent, users do not share keys with each other but rather with designated servers. Furthermore, the new versions of DCnet-based protocols enforce network restrictions to the selection set in order to increase efficiency and performance.

We conclude this part of the discussion with miscellaneous protocols. *Tarzan* protocol originally had a partially connected topology that was due to its partial network view of the route initiator. However, in the later version of Tarzan, a gossip-based strategy has been proposed to have a complete view for the route initiator, which leads to a fully connected topology as marked in Table

The connectivity of *I2P* is similar to onion-routing protocols due to the same restriction for node selection. However, note that since the network view is not necessarily complete in I2P, the connectivity might be slightly less than onion routing-protocols. I2P uses a unidirectional connection direction, which reduces the timing data that a single relay can have, however, more relays are going to be involved in the communication between a sender and receiver. The routing information of I2P is managed in DHT-like fashion, and each database node (floodfill peer) has a slice of the information [79], which might enable adversaries to carry out eclipse attacks targeting floodfill nodes [109].

The connectivity of *I2P* is similar to onion routing protocols due to the similarities for the node selection. I2P is characterized with unidirectional connection, which reduces the timing data that a single relay can have. However, multiple relays participate in the communication between a sender and receiver. The routing information of I2P is managed in a DHT-like fashion. Each database node (floodfill peer) has a slice of the information [79], which could enable adversaries to carry out eclipse attacks targeting floodfill nodes [109].

Since a user obtains node information from more than one floodfill node (up to eight), the union of this information might cover most of the I2P network and give the decision maker an almost complete view. I2P uses a source-routing approach, allowing the users to choose nodes that are faster. The selection probability in I2P is non-deterministic with a bias towards nodes that are profiled as fast responding nodes. Response times of these nodes differ among users; hence, timing attacks are more difficult to mount compared to Tor, where the node selection is biased using publicly known information [114]. Since response times are continuously measured, we have marked the selection probability with a bias based on dynamic restrictions. At the node level, I2P nodes use a prioritized scheduling mechanism, where each task has "bid", and the task with the lowest (best) bid is served first [115].

2.4.2 Correlation, Conflicts, Trade-offs, and Applications

In this section, we address correlation (*i.e.,* dependencies and conflicts), and trade-offs between routing characteristics of AC networks. First, we review direct and indirect correlations of routing features by comparing them with each other. We conclude this section with a discussion about the relevance of specific routing characteristics for certain applications.

Table 2.2: Overview of the adversary definitions, focus of routing feature, and challenges that our four routing classes face

Routing Class	Adversary Type	Routing Feature in Focus	Challenges
Mixnet	Global & active	Forwarding (scheduling) & node management (topology)	Traffic analysis attacks, such as flooding attacks
Onion routing	Local & active	Node selection	Traffic analysis attacks, *i.e.,* timing attacks
Random Walks (DHT)	Local & active	Node selection & transfer of routing information	Partitioning attacks & biasing node selection
DCnet	Global & passive	Forwarding	Collision and disruption

We have defined the topology only based on connectivity of relaying routers (see Section

There is an evident correlation between hierarchy and topology of AC networks. A hierarchical AC network does not have a fully connected network structure. For example, Herbivore, which has a hierarchical routing strategy, has a partially connected topology. Moreover, the network view of the routing decision maker can have an influence on the topology of the AC network. Generally speaking, a partial network view might lead to a partially connected network topology for the AC network because the routing decision maker might have difficulties accessing routing information of certain nodes. This holds for random walk and DHT-based protocols. One exception is demonstrated by PIR-TOR, which uses PIR to keep the network view minimal, albeit the topology is fully connected. Therefore, the correlation between topology and the network view depends on further factors. For example, if the topology is partially connected, it might be that the routing decision maker has a partial view, but it also might be due to some other routing restrictions.

We also observe a correlation between topology and selection set. Namely, restrictions in the selection set lead to reduced connectivity of the network topology. For example, in Restricted Route mix networks, the network view is complete; however, connectivity is restricted due to restrictions in the selection set, which leads to a partially connected network topology.

Although the synchronization of connections is not directly correlated to scheduling, it depends on the forwarding strategy of the particular nodes. As mentioned in Section

AC networks with a hierarchical structure have partially connected network structure (*i.e.,* Herbivore and the extended version of Dissent). By definition, hierarchical organization of nodes restricts the selection set.

Node management is more challenging in fully decentralized AC networks. Therefore, obtaining a complete view and a periodic updating of routing information is more difficult. When the network view of the routing decision maker is partial, often source-routing has the advantage to prevent the bias of malicious nodes and partitioning attacks. Thus, AC protocols that use this combination need to employ a secure node selection policy in their protocol. Examples of such protocols are Octopus and Morphmix. Octopus uses bound checking and proactively identifies malicious nodes; while, the latter one randomly selects witnesses to prevent bias in node selection. A partial network view also restricts the selection set because the routing decision maker can select only nodes that it is aware of.

Clearly, flushing algorithms also influence scheduling. For example, pool mixes can be defined to induce prioritized scheduling. There is also a correlation between scheduling and latency because in a prioritized scheduling algorithm, some type of traffic is delayed.

Flushing algorithms also influence latency. Timed mixes by themselves do not necessarily influence latency. However, they might induce latency if long time restrictions have been selected. Same with the threshold mixes, when the incoming traffic is low compared to the threshold that has been set up. There is also a correlation between latency and communication mode. High latency AC networks usually use a message based communication mode and vice versa. This is because connections are not going to be used further (*e.g.,* replies are not going to be sent in a short time); therefore, setting up a circuit is unnecessary.

Next, we compare our four main groups by discussing their applications. Mixnets are designed to be secure against traffic analysis and global adversaries by aggregating messages into batches. However, they are vulnerable to the collusion of mixnets and flooding attacks [82], in case if there are not enough (honest) users. Moreover, Mixnets' resilience against traffic analysis comes with a price and makes them more appropriate for high latency applications, such as emails and electronic voting.

Onion routing protocols, such as Tor, are more efficient (in particular faster) and have little computational overhead, making them suitable for low-latency applications, such as web browsing. Tor also leverages a large number of volunteer nodes. Almost all of these nodes are known to the routing decision maker. However, the complete network structure for the routing decision maker can limit scalability. Moreover, Tor is considered to be only secure against local adversaries and it is vulnerable to traffic analysis attacks [91, 116–120], in particular if the adversary can access both ends of the communication.

Random walk protocols and protocols using DHT are designed rather for fully peer-to-peer networks, where the network view is incomplete. Having a fully peer-to-peer network motivates the growth of the network and helps scalability. Therefore, they are suitable, for instance, for anonymous file sharing, where the nodes have to dedicate a considerable amount of resources. However, being fully peer-to-peer is considered to affect the usability of the protocol. Unfortunately, this might lead to a decrease in the number of users of such systems and in turn reduce anonymity. Last but not least, classic DCnets provide information-theoretic anonymity but some of them require a restricted setting, where all users or nodes need to be honest. The classic DCnets were also not resilient against DoS attacks. Moreover, DCnets tend do have a large communication overhead and do not scale well. Even Dissent, which employs a client-server approach for better scalability, can only scale up to a few thousand clients [74]. Therefore, they are more suitable for applications, such as micro-blogging, but at a small scale.

In Table

2.5 Concluding Remarks

In this work, we classified anonymous routing characteristics. We identified main criteria groups, each with several routing features and dimensions tackling various aspects routing decisions in AC protocols. Moreover, we shortly described and then carefully evaluated the bulk of existing AC protocols under our classification. Furthermore, we discussed the relevance between routing decisions that are made in such networks and their influence on anonymity and security. We have learned several lessons from conducting our survey. On the one hand, security, anonymity, scalability, and performance goals that are favored for anonymous communication are very hard to reach altogether, simply because the routing decisions, which support each of these goals, often contradict each other. This is especially true for achieving strong anonymity and good performance, which is still an open problem. On the other hand, routing aspects are related to each other, for example, a partial view of the system (in the routing information) often supports the hop-by-hop routing. Therefore, it is very hard to separate the various routing aspects from one to another protocol. We observe that making certain routing decisions leads often to a trade-off between security, anonymity, scalability, and performance goals. Finally, our classification uncovers which routing decisions have to be tailored to the security, anonymity, scalability, and performance goals that are necessary for a specific use case of a given AC protocol.

Bibliography

[1] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–88, 1981.

[2] K. Sako and J. Kilian, "Receipt-free Mix-Type voting scheme - A practical solution to the implementation of a voting booth," in *Advances in Cryptology - EUROCRYPT '95, International Conference on the Theory and Application of Cryptographic Techniques, Saint-Malo, France, May 21-25, 1995, Proceeding*, pp. 393–403, 1995.

[3] M. Jakobsson, A. Juels, and R. L. Rivest, "Making mix nets robust for electronic voting by randomized partial checking," in *Proceedings of the 11th USENIX Security Symposium, San Francisco, CA, USA, August 5-9, 2002*, pp. 339–353, 2002.

[4] R. Dingledine, M. J. Freedman, and D. Molnar, "The free haven project: Distributed anonymous storage service," in *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings*, pp. 67–95, 2000.

[5] M. Waldman, A. D. Rubin, and L. F. Cranor, "Publius: A robust, tamper-evident, censorship-resistant, and source-anonymous web publishing system," in *9th USENIX Security Symposium, Denver, Colorado, USA, August 14-17, 2000.*

[6] M. Waldman and D. Mazières, "Tangler: a censorship-resistant publishing system based on document entanglements," in *CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, Pennsylvania, USA, November 6-8, 2001.*, pp. 126–135, 2001.

[7] D. Goldschlag, M. Reed, and P. Syverson, "Hiding routing information," in *Information Hiding* (R. Anderson, ed.), vol. 1174 of *Lecture Notes in Computer Science*, pp. 137–150, Springer Berlin Heidelberg, 1996.

[8] 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, pp. 303–320, USENIX Association, 2004.

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

[10] E. Erdin, C. Zachor, and M. Gunes, "How to find hidden users: A survey of attacks on anonymity networks," *Communications Surveys Tutorials, IEEE*, vol. PP, no. 99, pp. 1–1, 2015.

[11] K. Sampigethaya and R. Poovendran, “A survey on mix networks and their secure applications,” *Proceedings of the IEEE*, vol. 94, pp. 2142–2181, December 2006.

[12] B. Conrad and F. Shirazi, “Survey on Tor and I2P,” in *ICIMP*, pp. 22–28, July 2014.

[13] M. AlSabah and I. Goldberg, “Performance and security improvements for Tor: A survey.” Cryptology ePrint Archive, Report 2015/235, 2015.

[14] J. Ren and J. Wu, “Survey on anonymous communications in computer networks,” *Computer Communications*, vol. 33, no. 4, pp. 420–431, 2010.

[15] M. Edman and B. Yener, “On anonymity in an electronic society: A survey of anonymous communication systems,” *ACM Computing Surveys (CSUR)*, vol. 42, pp. 5:1–5:35, December 2009.

[16] G. Danezis and C. Díaz, “A survey of anonymous communication channels,” tech. rep., Microsoft Research, 2008.

[17] A. Serjantov, “On the anonymity of anonymity systems,” tech. rep., University of Cambridge, Computer Laboratory, October 2004.

[18] J. Raymond, “Traffic analysis: Protocols, attacks, design issues, and open problems,” in *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings*, pp. 10–29, 2000.

[19] P. Bell and K. Jabbour, “Review of point-to-point network routing algorithms,” *Communications Magazine, IEEE*, vol. 24, pp. 34–38, January 1986.

[20] L. M. Feeney, “A Taxonomy for Routing Protocols in Mobile Ad Hoc Networks,” 1999.

[21] X. Zou, B. Ramamurthy, and S. Magliveras, “Routing techniques in wireless ad hoc networks - classification and comparison,” in *Proceedings of the Sixth World Multiconference on Systemics, Cybernetics, and Informatics (SCI 2002*, 2002.

[22] N. Feamster and R. Dingledine, “Location diversity in anonymity networks,” in *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, WPES ’04, (New York, NY, USA), pp. 66–76, ACM, 2004.

[23] M. Edman and P. Syverson, “As-awareness in Tor path selection,” in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS ’09, (New York, NY, USA), pp. 380–389, ACM, 2009.

[24] R. Böhme, G. Danezis, C. Díaz, S. Köpsell, and A. Pfitzmann, “On the PET workshop panel mix cascades versus peer-to-peer: Is one concept superior?,” in *Privacy Enhancing Technologies* (D. Martin and A. Serjantov, eds.), vol. 3424 of *Lecture Notes in Computer Science*, pp. 243–255, Springer Berlin Heidelberg, 2005.

[25] G. Danezis, “Mix-networks with restricted routes,” in *Privacy Enhancing Technologies, Third International Workshop, PET 2003, Dresden, Germany, March 26-28, 2003, Revised Papers*, pp. 1–17, 2003.

[26] G. Danezis, "Statistical disclosure attacks," in *Security and Privacy in the Age of Uncertainty, IFIP TC11 18th International Conference on Information Security (SEC '03), May 26-28, 2003, Athens, Greece*, pp. 421–426, 2003.

[27] B. Levine, M. Reiter, C. Wang, and M. Wright, "Timing attacks in low-latency mix systems," in *Financial Cryptography* (A. Juels, ed.), vol. 3110 of *Lecture Notes in Computer Science*, pp. 251–265, Springer Berlin Heidelberg, 2004.

[28] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-resource routing attacks against Tor," in *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*, WPES '07, (New York, NY, USA), pp. 11–20, ACM, 2007.

[29] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "Correlation-based traffic analysis attacks on anonymity networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 7, pp. 954–967, 2010.

[30] S. Murdoch and P. Zielinski, "Sampled traffic analysis by internet-exchange-level adversaries," in *Privacy Enhancing Technologies* (N. Borisov and P. Golle, eds.), vol. 4776 of *Lecture Notes in Computer Science*, pp. 167–183, 2007.

[31] A. Pfitzmann, B. Pfitzmann, and M. Waidner, "Isdn-mixes: Untraceable communication with very small bandwidth overhead," in *Kommunikation in verteilten Systemen*, vol. 267 of *Informatik-Fachberichte*, pp. 451–463, Springer Berlin Heidelberg, 1991.

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

[33] 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, (Berlin, Heidelberg), pp. 151–166, Springer-Verlag, 2008.

[34] C. Grothoff, "An excess-based economic model for resource allocation in peer-to-peer networks," *Wirtschaftsinformatik*, vol. 3-2003, June 2003.

[35] A. Jerichow, J. Müller, A. Pfitzmann, B. Pfitzmann, and M. Waidner, "Real-time mixes: a bandwidth-efficient anonymity protocol," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 495–509, 1998.

[36] C. Gülcü and G. Tsudik, "Mixing email with babel," in *1996 Symposium on Network and Distributed System Security, NDSS '96, San Diego, CA, February 22-23, 1996*, pp. 2–16, 1996.

[37] 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*, pp. 83–98, 1998.

[38] 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*, pp. 115–129, 2000.

[39] O. Berthold, H. Federrath, and M. Köhntopp, "Project anonymity and unobservability in the internet," in *Proceedings of the Tenth Conference on Computers, Freedom and Privacy: Challenging the Assumptions*, CFP '00, (New York, NY, USA), pp. 57–65, ACM, 2000.

[40] R. Dingledine, M. Freedman, D. Hopwood, and D. Molnar, "A reputation system to increase MIX-Net reliability," in *Information Hiding* (I. Moskowitz, ed.), vol. 2137 of *Lecture Notes in Computer Science*, pp. 126–141, Springer Berlin Heidelberg, 2001.

[41] R. Dingledine and P. Syverson, "Reliable MIX cascade networks through reputation," in *Financial Cryptography* (M. Blaze, ed.), vol. 2357 of *Lecture Notes in Computer Science*, pp. 253–268, Springer Berlin Heidelberg, 2002.

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

[43] 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*, pp. 2–15, 2003.

[44] M. Akhoondi, C. Yu, and H. Madhyastha, "Lastor: A low-latency as-aware Tor client," in *Security and Privacy (SP), 2012 IEEE Symposium on*, pp. 476–490, May 2012.

[45] M. Sherr, M. Blaze, and B. T. Loo, "Scalable link-based relay selection for anonymous routing," in *Proceedings of Privacy Enhancing Technologies, 9th International Symposium (PETS 2009)* (I. Goldberg and M. J. Atallah, eds.), vol. 5672 of *Lecture Notes in Computer Science*, pp. 73–93, Springer, August 2009.

[46] R. Snader and N. Borisov, "Improving security and performance in the Tor network through tunable path selection," *Dependable and Secure Computing, IEEE Transactions on*, vol. 8, pp. 728–741, September 2011.

[47] T. Wang, K. Bauer, C. Forero, and I. Goldberg, "Congestion-aware path selection for Tor," in *Financial Cryptography and Data Security* (A. Keromytis, ed.), vol. 7397 of *Lecture Notes in Computer Science*, pp. 98–113, Springer Berlin Heidelberg, 2012.

[48] A. Panchenko, F. Lanze, and T. Engel, "Improving performance and anonymity in the Tor network," in *31st IEEE International Performance Computing and Communications Conference, IPCCC 2012, Austin, TX, USA, December 1-3, 2012*, pp. 1–10, 2012.

[49] M. Backes, A. Kate, S. Meiser, and E. Mohammadi, "(Nothing else) MATor(s): Monitoring the Anonymity of Tor's Path Selection," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, (New York, NY, USA), pp. 513–524, ACM, 2014.

[50] D. Gopal and N. Heninger, "Torchestra: Reducing interactive traffic delays over Tor," in *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, WPES '12, (New York, NY, USA), pp. 31–42, ACM, 2012.

[51] M. AlSabah and I. Goldberg, "PCTCP: per-circuit tcp-over-ipsec transport for anonymous communication overlay networks," in *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pp. 349–360, 2013.

[52] J. Geddes, R. Jansen, and N. Hopper, "IMUX: managing Tor connections from two to infinity, and beyond," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society, WPES 2014, Scottsdale, AZ, USA, November 3, 2014*, pp. 181–190, 2014.

[53] M. AlSabah, K. S. Bauer, T. Elahi, and I. Goldberg, "The path less travelled: Overcoming Tor's bottlenecks with traffic splitting," in *Privacy Enhancing Technologies - 13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013. Proceedings*, pp. 143–163, 2013.

[54] C. Tang and I. Goldberg, "An improved algorithm for Tor circuit scheduling," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, (New York, NY, USA), pp. 329–339, ACM, 2010.

[55] M. AlSabah, K. S. Bauer, and I. Goldberg, "Enhancing Tor's performance using real-time traffic classification," in *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pp. 73–84, 2012.

[56] P. Mittal, F. Olumofin, C. Troncoso, N. Borisov, and I. Goldberg, "PIR-Tor: Scalable anonymous communication using private information retrieval," in *Proceedings of the 20th USENIX Conference on Security*, SEC '11, (Berkeley, CA, USA), pp. 31–31, USENIX Association, 2011.

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

[58] 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, (New York, NY, USA), pp. 91–102, ACM, 2002.

[59] M. Rennhard and B. Plattner, "Practical anonymity for the masses with morphmix," in *Financial Cryptography* (A. Juels, ed.), vol. 3110 of *Lecture Notes in Computer Science*, pp. 233–250, Springer Berlin Heidelberg, 2004.

[60] J. McLachlan, A. Tran, N. Hopper, and Y. Kim, "Scalable onion routing with Torsk," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, (New York, NY, USA), pp. 590–599, ACM, 2009.

[61] A. Panchenko, S. Richter, and A. Rache, "NISAN: network information service for anonymization networks," in *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*, pp. 141–150, 2009.

[62] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. S. Wallach, "AP3: cooperative, decentralized anonymous communication," in *Proceedings of the 11st ACM SIGOPS European Workshop, Leuven, Belgium, September 19-22, 2004*, p. 30, 2004.

[63] A. Nambiar and M. Wright, "Salsa: A structured approach to large-scale anonymity," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, (New York, NY, USA), pp. 17–26, ACM, 2006.

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

[65] 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*, pp. 46–66, Springer-Verlag New York, Inc., 2001.

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

[67] K. Bennett, T. Stef, C. Grothoff, T. Horozov, and I. Patrascu, "The gnet whitepaper," June 2002.

[68] K. Bennett and C. Grothoff, "gap practical anonymous networking," in *Privacy Enhancing Technologies* (R. Dingledine, ed.), vol. 2760 of *Lecture Notes in Computer Science*, pp. 141–160, Springer Berlin Heidelberg, 2003.

[69] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, no. 1, pp. 65–75, 1988.

[70] M. Waidner and B. Pfitzmann, "The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability," in *Advances in Cryptology - EUROCRYPT '89* (J.-J. Quisquater and J. Vandewalle, eds.), vol. 434 of *Lecture Notes in Computer Science*, pp. 690–690, Springer Berlin Heidelberg, 1990.

[71] P. Golle and A. Juels, "Dining cryptographers revisited," in *Advances in Cryptology - EUROCRYPT '04* (C. Cachin and J. Camenisch, eds.), vol. 3027 of *Lecture Notes in Computer Science*, pp. 456–473, Springer Berlin Heidelberg, 2004.

[72] S. Goel, M. Robson, M. Polte, and E. Sirer, "Herbivore: A scalable and efficient protocol for anonymous communication," tech. rep., Cornell University, 2003.

[73] 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, pp. 340–350, 2010.

[74] 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, pp. 179–192, USENIX Association, 2012.

[75] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Scalable anonymous group communication in the anytrust model," in *European Workshop on System Security (EuroSec)*, vol. 4, 2012.

[76] M. J. Freedman, E. Sit, J. Cates, and R. Morris, "Introducing tarzan, a peer-to-peer anonymizing network layer," in *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers*, pp. 121–129, 2002.

[77] M. J. Freedman and R. Morris, "Tarzan: A peer-to-peer anonymizing network layer," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pp. 193–206, ACM, 2002.

[78] "I2P documentation." https://geti2p.net/en/docs. Last accessed: August 05, 2014.

[79] L. Schimmer, “Peer profiling and selection in the I2P anonymous network,” in *Proceedings of PET-CON 2009.1*, pp. 59–70, March 2009.

[80] 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*, pp. 1–9, 2000.

[81] C. Díaz and B. Preneel, “Taxonomy of mixes and dummy traffic,” in *Information Security Management, Education and Privacy, IFIP 18th World Computer Congress, TC11 19th International Information Security Workshops, 22-27 August 2004, Toulouse, France*, pp. 215–230, 2004.

[82] A. Serjantov, R. Dingledine, and P. Syverson, “From a trickle to a flood: Active attacks on several mix types,” in *Information Hiding* (F. Petitcolas, ed.), vol. 2578 of *Lecture Notes in Computer Science*, pp. 36–52, Springer Berlin Heidelberg, 2003.

[83] C. Díaz and A. Serjantov, “Generalising mixes,” in *Privacy Enhancing Technologies* (R. Dingledine, ed.), vol. 2760 of *Lecture Notes in Computer Science*, pp. 18–31, Springer Berlin Heidelberg, 2003.

[84] G. Danezis, C. Diaz, and P. F. Syverson, “Systems for Anonymous Communication,” in *CRC Handbook of Financial Cryptography and Security* (B. Rosenberg and D. Stinson, eds.), CRC Cryptography and Network Security Series, pp. 341–390, Chapman & Hall, August 2010.

[85] M. Reed, P. Syverson, and D. Goldschlag, “Anonymous connections and onion routing,” *Selected Areas in Communications, IEEE Journal on*, vol. 16, pp. 482–494, May 1998.

[86] M. Wright, M. Adler, B. N. Levine, and C. Shields, “An analysis of the degradation of anonymous protocols,” in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2002, San Diego, California, USA*, The Internet Society, 2002.

[87] M. Wright, M. Adler, B. Levine, and C. Shields, “Defending anonymous communications against passive logging attacks,” in *Security and Privacy (SP), 2003. Proceedings. 2003 Symposium on*, pp. 28–41, May 2003.

[88] R. Dingledine, N. Hopper, G. Kadianakis, and N. Mathewson, “One fast guard for life (or 9 months),” *7th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2014)*, 2014.

[89] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, “Towards an analysis of onion routing security,” in *Designing Privacy Enhancing Technologies* (H. Federrath, ed.), vol. 2009 of *Lecture Notes in Computer Science*, pp. 96–114, Springer Berlin Heidelberg, 2001.

[90] M. Backes, A. Kate, P. Manoharan, S. Meiser, and E. Mohammadi, “AnoA: A framework for analyzing anonymous communication protocols,” in *Computer Security Foundations Symposium (CSF), 2013 IEEE 26th*, pp. 163–178, June 2013.

[91] S. Murdoch and G. Danezis, “Low-cost traffic analysis of Tor,” in *Security and Privacy, 2005 IEEE Symposium on*, pp. 183–195, May 2005.

[92] R. Dingledine and S. J. Murdoch, "Performance improvements on Tor or, why Tor is slow and what we're going to do about it," tech. rep., The Tor Project, November 2009.

[93] M. K. Wright, M. Adler, B. N. Levine, and C. Shields, "The predecessor attack: An analysis of a threat to anonymous communications systems," *ACM Trans. Inf. Syst. Secur.*, vol. 7, pp. 489–522, Novmeber 2004.

[94] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, (London, UK, UK), pp. 53–65, Springer-Verlag, 2002.

[95] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '01, pp. 149–160, ACM, 2001.

[96] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, Middleware '01, (London, UK, UK), pp. 329–350, Springer-Verlag, 2001.

[97] P. Wang, I. Osipkov, N. Hopper, and Y. Kim, "Myrmic: Provably secure and efficient DHT routing," tech. rep., DTC, 2006.

[98] S. Roos, B. Schiller, S. Hacker, and T. Strufe, "Measuring freenet in the wild: Censorship-resilience under observation," in *Privacy Enhancing Technologies - 14th International Symposium, PETS 2014, Amsterdam, The Netherlands, July 16-18, 2014. Proceedings*, pp. 263–282, 2014.

[99] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00, pp. 163–170, ACM, 2000.

[100] J. Bos and B. den Boer, "Detection of disrupters in the dc protocol," in *Advances in Cryptology - EUROCRYPT '89* (J.-J. Quisquater and J. Vandewalle, eds.), vol. 434 of *Lecture Notes in Computer Science*, pp. 320–327, Springer Berlin Heidelberg, 1990.

[101] M. Waidner, "Unconditional sender and recipient untraceability in spite of active attacks," in *Advances in Cryptology - EUROCRYPT '89* (J.-J. Quisquater and J. Vandewalle, eds.), vol. 434 of *Lecture Notes in Computer Science*, pp. 302–319, Springer Berlin Heidelberg, 1990.

[102] S. Dolev and R. Ostrobsky, "Xor-trees for efficient anonymous multicast and reception," *ACM Trans. Inf. Syst. Secur.*, vol. 3, pp. 63–84, May 2000.

[103] A. Pfitzmann and M. Waidner, "Networks without user observability - design options," in *Advances in Cryptology - EUROCRYPT '85* (F. Pichler, ed.), vol. 219 of *Lecture Notes in Computer Science*, pp. 245–253, Springer Berlin Heidelberg, 1986.

[104] J. Douceur, "The sybil attack," in *Peer-to-Peer Systems* (P. Druschel, F. Kaashoek, and A. Rowstron, eds.), vol. 2429 of *Lecture Notes in Computer Science*, pp. 251–260, Springer Berlin Heidelberg, 2002.

[105] M. Harchol-Balter, F. T. Leighton, and D. Lewin, "Resource discovery in distributed networks," in *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '99, pp. 229–237, 1999.

[106] J. P. Timpanaro, I. Chrisment, and O. Festor, "I2P's usage characterization," in *Traffic Monitoring and Analysis* (A. Pescapè, L. Salgarelli, and X. Dimitropoulos, eds.), vol. 7189 of *Lecture Notes in Computer Science*, pp. 48–51, Springer Berlin Heidelberg, 2012.

[107] J. P. Timpanaro, C. Isabelle, and F. Olivier, "Monitoring the I2P network," tech. rep., October 2011.

[108] "I2P statistics." http://stats.i2p.re/. Last accessed: January 25, 2016.

[109] C. Egger, J. Schlumberger, C. Kruegel, and G. Vigna, "Practical attacks against the i2p network," in *Proceedings of the 16th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2013)*, October 2013.

[110] M. AlSabah, K. Bauer, I. Goldberg, D. Grunwald, D. McCoy, S. Savage, and G. Voelker, "Defenestrator: Throwing out windows in Tor," in *Privacy Enhancing Technologies* (S. Fischer-Hübner and N. Hopper, eds.), vol. 6794 of *Lecture Notes in Computer Science*, pp. 134–154, Springer Berlin Heidelberg, 2011.

[111] R. Dingledine and N. Mathewson, "Anonymity loves company: Usability and the network effect," in *Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS 2006)* (R. Anderson, ed.), (Cambridge, UK), June 2006.

[112] G. Danezis, C. Diaz, E. Ksper, and C. Troncoso, "The wisdom of crowds: Attacks and optimal constructions," in *Computer Security ESORICS 2009* (M. Backes and P. Ning, eds.), vol. 5789 of *Lecture Notes in Computer Science*, pp. 406–423, Springer Berlin Heidelberg, 2009.

[113] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 299–314, Dec. 2002.

[114] "I2P peer profiling and selection." https://geti2p.net/en/docs/how/peer-selection. Last accessed: January 25, 2016.

[115] "I2P transport overview." https://geti2p.net/en/docs/transport. Last accessed: January 25, 2016.

[116] S. Chakravarty, A. Stavrou, and A. Keromytis, "Traffic analysis against low-latency anonymity networks using available bandwidth estimation," in *Computer Security - ESORICS 2010* (D. Gritzalis, B. Preneel, and M. Theoharidou, eds.), vol. 6345 of *Lecture Notes in Computer Science*, pp. 249–267, Springer Berlin Heidelberg, 2010.

[117] "A practical congestion attack on Tor using long paths," in *Presented as part of the 18th USENIX Security Symposium (USENIX Security 09)*, (Montreal, Canada), USENIX, 2009.

[118] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. F. Syverson, "Users get routed: traffic correlation on tor by realistic adversaries," in *2013 ACM SIGSAC Conference on*

Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013, pp. 337–348, 2013.

[119] P. Mittal, A. Khurshid, J. Juen, M. Caesar, and N. Borisov, "Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, (New York, NY, USA), pp. 215–226, ACM, 2011.

[120] G. O'Gorman and S. Blott, "Improving stream correlation attacks on anonymous networks," in *Proceedings of the 2009 ACM Symposium on Applied Computing (SAC), Honolulu, Hawaii, USA, March 9-12, 2009*, pp. 2024–2028, 2009.

3. Existing shuffle protocols: A Survey of Shuffle protocols

A crucial part of any mix-net is a secure, private and efficient shuffle argument. A zero-knowledge shuffle argument enables a prover to convince a verifier given two lists of ciphertexts, that one lists of ciphertexts is a permutation of other lists of ciphertexts, without revealing any additional information except the truth of the statement; that is, a shuffle argument should be zero knowledge [23]. In this chapter, we present a survey of existing shuffle protocols and their comparison. The solid understanding of the security and performance properties, as well as possible threads and issues allows to design and develop the secure yet efficient e-voting applications; neccessary to support WP5 of PANORAMIX project.

3.1 Efficiency

One of the things that makes the construction of efficient shuffle arguments difficult is the fact that the prover may not know any of the corresponding plaintexts. Due to this, while contemporary shuffle arguments are relatively efficient, they are at the same time conceptually quite complicated and rely (say) on novel characterization of permutation matrices. In particular, computationally most efficient shuffle arguments tend to rely either

- on the CRS-model [7] and require a larger number of rounds (unless one relies on the random oracle model [5] to make the argument non-interactive by using the Fiat-Shamir heuristic [18]), or
- offer less security (for example, the argument of [21] is not zero knowledge).

On the other hand, existing random oracle-less CRS-model non-interactive shuffle arguments [29, 35] are considerably less efficient. While the random oracle model is dubious from the security viewpoint [9], there are no known practical attacks on random oracle-model shuffle arguments.

3.2 Interactive and non-interactive shuffle arguments

Interactive shuffle arguments

We recall three main paradigms that are used in known *computationally* efficient interactive shuffle arguments. Other approaches are known, but they have up to now resulted in significantly less computation-efficient shuffle arguments. An additional direction has been to minimize the communication and the verifier's computation at the cost of possibly larger prover's computation; see [4].

Table 3.1: Interactive shuffles comparison. In this table shuffles based on Elgamal has been included and analysed. Here N, the number of ciphertexts to shuffle equals $N = nm$.

	[1]	[21]	[20]	[19]	[44]
rounds	3	3	5	3	5
pro comp.	$O(\log(N))N$	$8N$	$9N$	$7N$	$9N$
ver comp.	$O(\log N)$	$10N$	$10N$	$8N$	$11N$
size (in Kbits)	$O(\log N)N$	$5.3N$	$5.3N$	$1.5N$	$3.7N$

	[38]	[25]	[28]	[4]	[4]
rounds	7	7	7	9	$\log(m)$
pro comp.	$8N$	$6N$	$3mN$	$2\log(m)N$	$O(N)$
ver comp.	$12N$	$6N$	$4N$	$4N$	$4N$
size (in Kbits)	$7.7N$	$3N$	$3m^2 + 3n$	$11m + 5n$	$11m + 5n$

The approach of Furukawa and Sako [21] uses permutation matrices, relying a specific characterization of permutation matrices. Namely, a matrix M is a permutation matrix if $\mathbf{M}^{(i)} \cdot \mathbf{M}^{(j)} = \delta_{ij}$ and $\mathbf{M}^{(i)} \cdot \mathbf{M}^{(j)} \odot \mathbf{M}^{(k)} = \delta_{ijk}$, where δ_{ij} is the Kronecker delta, $\delta_{ijk} = \delta_{ij}\delta_{ik}$, $\odot$ denotes element-wise multiplication, and $\cdot$ denotes scalar product. The Furukawa-Sako argument satisfies a privacy requirement that is weaker than zero knowledge. Later, it has been made more efficient — and zero knowledge — by Furukawa [19]. Importantly, arguments of this approach have only 3 messages.

The approach of Neff [38] uses the fact that permuting the roots of a polynomial results in the same polynomial; Neff's argument has been made efficient by Groth [26]. While Neff's approach results in computationally more efficient arguments, the resulting arguments require 7 messages.

The approach by Terelius and Wikström [43] uses permutation matrices together with the fact that $\mathbb{Z}_q[\mathbf{X}]$ is a unique factorization domain. It is based on an alternative characterization of permutation matrices: $M \in \mathbb{Z}_q^{N\times N}$ is a permutation matrix iff (a) $\prod_{i=1}^{N} \mathbf{M}^{(i)} \cdot X_i = \prod_{i=1}^{N} X_i$ for independent random variables X_i, and (b) $M \cdot \mathbf{1}_N = \mathbf{1}_N$. The Terelius-Wikström approach results of shuffle arguments of intermediate number of messages (namely, 5). However, up to now it has required somewhat higher computational complexity than the first two approaches.

Efficiency comparison between different interactive shuffle arguments were collected in Table 3.1.

Non-interactive shuffle arguments

Advantages of non-interactive setting Although interactive shuffle arguments are usually more efficient than they non-interactive counterparts, the later are considered much more practical and their scope of use seems much broader. Especially, non-interactive setting allows various users to verify correctness of the protocol after it was proceeded and after, e.g., mix-servers were shut down.

Furthermore, this approach makes computational effort on behalf of mix-servers (that have to prove the fairness of the execution) independent from the number of potential verifiers. The later plays a great role when a system prepared for a potentially millions of verifiers, what is demanded for applications like electronic voting. Thus, although for a small amount of verifiers interactive arguments lead in terms of efficiency, non-interactivity makes system truly scalable.

Table 3.2: A comparison of different NIZK shuffle arguments compared with the computationally most efficient known shuffle argument in the random oracle model [26]. If not stated otherwise pro–prover's computational complexity is described in exponentiation, ver–verifier's computational complexity is described in the number of bilinear pairings, and both CRS and communication sizes are in the number of group elements.

	[29]	[36]	[16]	[26]
$\lvert CRS \rvert$	$2N + 8$	$7N + 6$	$8N + 17$	$N + 1$
Communication	$18N + 120$	$12N + 11$	$9N + 2$	$480N$ bits
pro's computation	$54N + 246$	$28N + 11$	$18N + 3$	$6N$ $(+2N)$
ver's computation	$75N + 282$	$28N + 18$	$18N + 6$	$6N$ exp.
Knowledge assumptions	No	Yes	Yes	No
Relying on GBGM	PP, SP	Knowledge	Knowl., PSP	No
Random oracle	No	No	No	Yes
Soundness	Culpable	Full	Culpable	Full

NI shuffle argument, state of the art Up to now, only three NIZK shuffle arguments in the CRS model have been proposed, by Groth and Lu [29], Lipmaa and Zhang [36], Fauzi and Lipmaa [16] all of which are significantly slower than the fastest arguments in the random oracle model (see Tbl. 3.2). The Groth-Lu shuffle argument only provides culpable soundness [29, 31] in the sense that if a malicious prover can create an accepting shuffle argument for an incorrect statement, then this prover together with a party that knows the secret key can break the underlying security assumptions.

Relaxation of the soundness property is unavoidable, since [2] showed that only languages in **P/poly** can have direct black-box adaptive perfect NIZK arguments under a (polynomial) cryptographic hardness assumption. If the underlying cryptosystem is IND-CPA secure, then the shuffle language is not in **P/poly**, and thus it is necessary to use knowledge assumptions [13] to prove its adaptive soundness. Moreover, [29] argued that culpable soundness is a sufficient security notion for shuffles, since in any real-life application of the shuffle argument there exists some coalition of parties who knows the secret key.

Table 3.2 provides a brief comparison between known NIZK shuffle arguments in the CRS model and the most computationally efficient known shuffle argument in the random oracle model [26]. We emphasize that the values in parentheses show the cost of computing and communicating the shuffled ciphertexts themselves, and must be added to the rest. Moreover, the cost of the shuffle argument from [36] should include the cost of a range argument. Unless written otherwise, the communication and the CRS length are given in group elements, the prover's computational complexity is given in exponentiations, and the verifier's computational complexity is given in bilinear pairings. In each row, highlighted cells denote the best efficiency or best security (e.g., not requiring the PKE assumption) among arguments in the CRS model. Of course, a full efficiency comparison can only be made after implementing the different shuffle arguments.

3.3 Description of existing shuffle protocols

Neff [38] The paper published by Neff in 2001 proposed one of the first efficient interactive shuffle arguments. The arguments works in any group where Diffie-Hellman problem is intractable. Thus, it allows to implement the scheme in the elliptic curves, but is not achievable for a bilinear setting.

Argument presented in the paper makes use of Schwartz-Zippel lemma, what result in security upper bounded by some fraction $1 - N/q$ for N being the number of shuffled elements and q the size of the group where operations are performed. Author points out that both parameters N and q should be fitted to a setting used to perform protocol. E.g. if protocol is to be executed in an interactive way N and q can guarantee less security than in a case when a malicious party, a cheater, is allowed to perform an exhaustive computations off-line.

From the efficiency point of view, the argument is much more efficient compared to the previous. Prover computation is limited by $8N+5$ while for Furukawa-Sako [21] it is $18N+18$ and Sako-Kilian [40] $642N$. The proof size has also been optimized and is limited by $8N + 5$ group elements.

Neff's argument differs from other arguments in a way it proves that some elements (ciphertexts) were permuted. Despite of proving that a set of elements were transformed accordingly to a matrix that is a permutation matrix, it maps ciphertexts into roots of some polynomial, say P, and permuted ciphertexts into roots of some other polynomial P' and shows that both polynomials are equal (with overwhelming probability) using property of identity of polynomials under root permutation.

Groth [26] In 2003 Groth proposed a shuffle that was based on approach used previously by Neff [38] (that polynomials are identical under permutation of their roots), but with a great complexity optimization.

Argument proposed in [26] is a 7-move public coin HVZK that, unlike [38], makes use of a CRS that contains of a public key for a homomorphic commitment scheme. The choice of commitment scheme is crucial for the security of the argument. If commitment scheme is statistically binding then argument is unconditionally sound. On the other hand, if the scheme is statistically hiding, then the argument is statistically HVZK.

One of the strong points of this argument is fact that it is well suited to use techniques like batching and multi-exponentiation what can have a great impact on the complexity of the whole protocol.

Authors use as a building block a new argument for a shuffle of *known* contents. This argument takes as input a sequence of messages $m_1, \ldots, m_N$ and outputs a commitment $c \leftarrow \mathsf{com}(m_{\pi(1)}, \ldots, m_{\pi(N)})$ for some permutation π along with a proof Π showing that c indeed consists of a permutation of messages $m_1, \ldots, m_N$. This building block is used in a full shuffle argument as follows: prover commits to a permutation of known values $1, \ldots, N$, i.e. $c \leftarrow \mathsf{com}(\pi(1), \ldots, \pi(N))$ and then shows that for given two sequences of ciphertexts $C_1, \ldots, C_N$ and $C'_1, \ldots, C'_N$ holds $C_{\pi(i)} = C'_i$ for all $i \in 1, \ldots, N$.

Furthermore authors show how to modify presented argument to work with decryption mix-nets by creating an argument of shuffle-and-decrypt operation.

Argument proposed by Groth [26] is the most efficient known non-interactive argument for shuffle. However, this argument is made non-interactive by using Fiat-Shamir heuristics [18], thus it works is the Random Oracle Model which is impossible to achieve in a real world.

Terelius-Wikström [43] The paper by Terelius and Wikström provides a shuffle argument for restricted shuffles, that is shuffles that permutation included is chosen from a public yet limited subset of all permutations. This is done by showing that a permutation π is contained in a group of automorphism of a publicly known polynomial, i.e. permutations such that $F(x_1, \ldots, x_N) = F(x_{\pi(1)}, \ldots, x_{\pi(N)})$ for some publicly known F.

Furthermore, authors show how the basic principle behind proposed techniques can be used in an efficient shuffle argument for unrestricted shuffle.

In this paper, permutation of N elements is defined by an $N \times N$ permutation matrix that contains in every row and column exactly one entry different than zero. Proof of such property goes as follows: let $(x_i)_{i=1}^N$ denote the list of variables and $\mathbf{m}_i$ i-th row of matrix M then, if the matrix has more than one non-zero entry in a row or column then $\prod_{i=1}^N \langle \mathbf{m}_i, x_i \rangle \neq \prod_{i=1}^N x_i$. What can be easily checked by using Schwartz-Zippel lemma.

Having this proven it is enough to show that sum of elements in every column and row is one.

Furukawa [19] The Furukawa shuffle protocol is a three round zero-knowledge protocol for Elgamal ciphertext shuffling, proposed in [19]. Using this protocol a mixer can prove that Elgamal ciphertexts where shuffled correctly without leaking any other information. Furukawa shuffle protocol is the most efficient three round shuffle argument currently known, any other efficient interactive shuffle arguments need more than three rounds.

This shuffle argument is based on common approach that represents a permutation as a permutation matrix.

Loosely the protocol works as follows. The prover (mixer) commits to the columns of a permutation matrix $A = (A_{i,j})$ that corresponds to the permutation that it used for shuffling the ciphertexts. The prover sends the commitments to the verifier. The verifier responds by sending N challenge values $c_1, \ldots, c_N$ where N is the number of ciphertexts. The prover sends a response

$$r_i = \sum_{j=1}^{N} A_{i,j} c_j$$

for every $i \in \{1, 2, \ldots, N\}$.

Verifier checks five equations to conclude whether the shuffling was done correctly or not. First it checks an equation that tells if r_i is computed in a correct form. Then it checks two equations to verify that A is a permutation matrix. Furukawa uses a novel description of a permutation matrix to make these two checks efficient. Namely A is a permutation matrix if the following two properties hold

$$\sum_{h=1}^{n} A_{h,i} A_{h,j} A_{h,k} = \delta_{i,j,k}$$

$$\sum_{h=1}^{n} A_{h,i} A_{h,j} = \delta_{i,j}$$

for any $i, j, k \in \{1, 2, \ldots, N\}$. Here $\delta_{i,j,k}$ and $\delta_{i,j}$ denote the Kronecker delta. Finally the verifier checks two equation that tell if the permutation matrix A was used for shuffling the ciphertexts.

Prover's computation complexity is $8N$ exponentiations, although there is a simple modification that reduces it to $7N$ exponentiations. Verifier's computation is $6N$ exponentiations. Communication complexity is $3N \log q + N \log p$ where p and q are security parameters with the property that $q|(p-1)$.

Bayer-Groth [4] Bayer-Groth paper introduces the first efficient non-interactive shuffle argument that has sublinear communication complexity. To shuffle $N = mn$ elements argument transmits only $O(m + n)$ elements (that optimizes for $m = n$) what is as little as $O(\sqrt{N})$. Furthermore the prover computation is efficient almost as in the protocols with linear communication.

To compare this result with the first sublinear argument [28] one has to mention that the paper by Groth and Ishai was inefficient from the prover point of view who was supposed to compute up to $O(Nm)$ exponentiations. Thus, the protocol was limited by the small m.

High-level description of the argument proposed in the paper goes as follows. The prover who proves the correctness of shuffle for some permutation π of N ciphertexts commits to $\pi(1), \pi(2), ..., \pi(N)$. Then, after receiving a challenge x, commits to $x^{\pi(1)}, x^{\pi(2)}, ..., x^{\pi(N)}$. Now, the prover gives an argument of opening of the commitments to permutation of respectively $1, 2, ..., N$ and $x^1, x^2, ..., x^N$ and shows that the same permutation has been used in both cases.

To check that the same permutation has been used in both commitments the verifier sends random challenges y and z. Then by homomorphic properties of the commitment, the prover shows that,

$$\prod_{i=1}^{N} (y\pi(i) + x^{\pi(i)} - z) = \prod_{i=1}^{N} (yi + x^i - z) \tag{3.1}$$

Both expressions from the left and the right are two identical degree N polynomials in z. The only difference is that the roots have been permuted [38]. The verifier does not know a priori that the the two polynomials are identical but using the Schwartz-Zippel lemma she can deduce that the prover has negligible chance (over the choice of z) to make a convincing argument unless there is a permutation π. Furthermore, there is negligible probability over the choice of y of this being true unless the first commitment contains $\pi(1), \pi(2), \ldots, \pi(N)$ and the second contains $x^{\pi(1)}, x^{\pi(2)}, \ldots, x^{\pi(N)}$.

In order to show that a sequence $(C_i')_{i=1}^N$ is in fact a sequence $(C_i)_{i=1}^N$ but with entries permuted, that is $C_i{}' = \varepsilon_{pk}(1; \rho_i)C_{\pi(i)}$ for $i = 1, 2, \ldots, N$, prover uses commitments $x^{\pi(1)}, x^{\pi(2)}, \ldots, x^{\pi(N)}$ and so-called *multiexponentiation argument* to show that there exist a randomness ρ such that

$$\prod_{i=1}^{N} C_i{}^{x^i} = \varepsilon_{pk}(1; \rho) \prod_{i=1}^{N} (C_i{}')^{x^{\pi(i)}}. \tag{3.2}$$

Since the efficiency of the argument strongly depends from the efficiency of the multiplication operation, authors propose a number of speed-ups by substituting standard multiplication algorithm by Toom-Cook [45, 11] and Fast Fourier Transform [12].

Groth-Lu [29] The argument presented by Groth and Lu in [29] is considered first efficient non-interactive shuffle argument that works in the CRS model and does not rely on the random oracle assumption (unlike [26]). Authors claim that zero-knowledge of the protocol is perfect.

To achieve non-interactivity without using random oracle authors used techniques from [32, 30], making non-interactive witness-indistinguishable proofs by using bilinear groups. This approach has become a standard technique for a oracle-less non-interactive proofs and was recently used, e.g. in [16]. However, this approach usually demands verifier to compute a number of bilinear pairings instead of exponentiation. As was shown in [3] a single exponentiation can be performed up to 7 times faster than a pairing.

Furthermore, the scheme proposed by Groth and Lu relies on BBS cryptosystem [8] where every ciphertext consists of 3 group elements (Elgamal ciphertexts needs only 2 group elements).

Although it may not look as a big difference, on average using 3 elements for a ciphertext instead of 2 multiplies the number of necessary pairings computed by a verifier by 3/2.

As claimed by authors, proposed argument consists of $15N$ group elements, while the statement needs $6N$ elements.

Lipmaa-Zhang [36] Lipmaa and Zhang [36] proposed a more efficient NIZK shuffle argument by using knowledge assumptions under which they also bypassed the impossibility result of [2] and proved that their shuffle argument is sound. However, their shuffle argument is sound only under the assumption that there is an extractor that has access to the random coins of all encrypters, e.g., all voters, allowing her to extract all plaintexts and randomizers. Authors say in this case that the argument is white-box sound. White-box soundness is clearly a weaker security notion than culpable soundness of [29], and it would be good to avoid it.

In addition, the use of knowledge assumptions in [36] forces the underlying BBS [8] cryptosystem to include knowledge components (so ciphertexts are twice as long) and be lifted (meaning that one has to solve discrete logarithm to decrypt, so plaintexts must be small). Thus, one has to use a random oracle-less range argument [39, 10, 17, 34] to guarantee that the plaintexts are small and thus to guarantee the soundness of the shuffle argument (see [36] for a discussion). While range proofs only have to be verified once (e.g., by only one mix-server), this still means that the shuffle argument of [36] is somewhat slower than what is given in Tbl. 3.2. Moreover, in the case of e-voting, using only small plaintexts restricts the applicability of a shuffle argument to only certain voting mechanisms like majority. On the other hand, a mechanism such as Single Transferable Vote would likely be unusable due to the length of the ballots.

Panoramix impact on the shuffle arguments state of the art

Although during Panoramix project no interactive protocols have been proposed yet, two papers have been delivered so far in the non-interactive setting.

Fauzi-Lipmaa [16] The paper provides non-interactive zero-knowledge shuffle argument that is more efficient than previous ones. The authors created an argument that at cost of slightly longer crs and weaker security model (culpable soundness instead of full soundness) makes both prover and verifier computation more efficient: prover's computation has been reduced by $10N$ exponentiations ($28N$ to $18N$) and verifier's by $10N$ pairings ($28N$ to $18N$).

The security of the Fauzi-Lipmaa shuffle argument is proven under a knowledge assumption [13] (PKE, [27]) and three computational assumptions (PCDH, TSDH, PSP). Knowledge assumptions are non-falsifiable [37], and their validity has to be very carefully checked in each application [6]. Moreover, the PSP assumption of [16] is novel, and its security is proven in the Generic Group Model [41].

This Fauzi-Lipmaa shuffle differs from the shuffle of [35] also in its security model. Briefly, in the security proof of the shuffle argument of [35] it is assumed that the adversary obtains — by using knowledge assumptions — not only the secrets of the possibly malicious mix-server, but also the plaintexts and randomizers computed by all voters. This model was called *white-box soundness* in [16], where it was also criticized. Moreover, in the shuffle argument [35], the plaintexts have to be small for the soundness proof to go through; for this, all voters should use efficient CRS-model range proofs [34].

On the other hand, the shuffle of [16] is proven culpably sound [29] though also under knowledge assumptions. Intuitively, this means that if a cheating adversary produces an invalid (yet accept-able) shuffle together with the secret key, then one can break one of the underlying knowledge or computational assumptions.

Compared to [29], which also achieves culpable soundness, the new argument has 3 times faster proving and more than 4 times faster verification. Compared to [29, 35], it is based on a more standard cryptosystem (Elgamal). While the new shuffle argument is still at least 2 times slower than the most efficient known random oracle based shuffle arguments, it has almost optimal *online* prover's computation. Of course, a full efficiency comparison can only be made after implementing the different shuffle arguments.

The construction of the shuffle in [16] goes as follows. First commit to the permutation ψ (by committing separately to first $n-1$ rows of the corresponding permutation matrix $\mathbf{\Psi}$) and to the vector $\mathbf{t}$ of blinding randomizers. Here, authors use the *polynomial commitment scheme* with $\mathsf{com}(\mathsf{ck}; \mathbf{m}; r) = (g_1, g_2^{\gamma})^{rP_0(\chi)+\sum_{i=1}^{n} m_i P_i(\chi)} \in \mathfrak{G}_1 \times \mathfrak{G}_2$, in pairing-based setting, where $\hat{e} : \mathfrak{G}_1 \times \mathfrak{G}_2 \to \mathfrak{G}_T$ is a bilinear pairing, g_i is a generator of $\mathfrak{G}_i$ for $i \in \{1, 2\}$, $(P_i(X))_{i=0}^{n}$ is a tuple of linearly independent polynomials, χ is a trapdoor, γ is a knowledge secret, and $\mathsf{ck} = ((g_1, g_2^{\gamma})^{P_i(\chi)})_{i=0}^{n}$ is the CRS. For different values of $P_i(X)$, variants of this commitment scheme have been proposed before [24, 27, 33].

The authors show that $\mathbf{\Psi}$ is a correct permutation matrix by constructing n witness-indistinguishable succinct *unit vector arguments*, each of which guarantees that a row of $\mathbf{\Psi}$ is a unit vector, for implicitly constructed $\mathbf{\Psi}_n = \mathbf{1_n} - \sum_{i=1}^{n-1} \mathbf{\Psi}_i$. Then authors use the recent square span programs (SSP, [14]) approach to choose the polynomials $P_i(X) = y_i(X)$ so that the unit vector argument is efficient.

After that, the authors postulate a natural concrete verification equation for shuffles, and construct the shuffle argument from this. If privacy were not an issue (and thus $\mathfrak{v}'_i = \mathfrak{v}_{\psi(i)}$ for every i), the verification equation would just be the tautology $\prod_{i=1}^{n} \hat{e}(\mathfrak{v}'_i, g_2^{y_i(\chi)}) =^? \prod_{i=1}^{n} \hat{e}(\mathfrak{v}_i, g_2^{y_{\psi^{-1}(i)}(\chi)})$. Clearly, if the prover is honest, this equation holds. However, it does not yet guarantee soundness, since an adversary can use $g_1^{y_j(\chi)}$ (given in the CRS) to create $(\mathfrak{v}'_i)_{i=1}^{n}$ in a malicious way. To eliminate this possibility, by roughly following an idea from [29], authors also verify that $\prod_{i=1}^{n} \hat{e}(\mathfrak{v}'_i, g_2^{\hat{y}_i(\chi)}) =^? \prod_{i=1}^{n} \hat{e}(\mathfrak{v}_i, g_2^{\hat{y}_{\psi^{-1}(i)}(\chi)})$ for some well-chosen polynomials $\hat{y}_i(X)$. (Note that instead of n univariate polynomials, [29] used n random variables χ_i, increasing the size of the secret key to $\Omega(n)$ bits.)

To show that the verifications are instantiated correctly, authors also need a *same-message argument* that shows that commitments w.r.t. two tuples of polynomials $(y_i(X))_{i=1}^{n}$ and $(\hat{y}_i(X))_{i=1}^{n}$ commit to the same plaintext vectors. Authors construct an efficient same-message argument by using an approach that is (again, roughly) motivated by the QAP-based approach of [22]. This argument is an argument of knowledge, given that the polynomials $\hat{y}_i(X)$ satisfy an additional restriction.

Since also privacy is required, the actual verification equations are more complicated. In particular, $\mathfrak{v}'_i = \mathfrak{v}_{\psi(i)} \cdot \mathsf{enc}_{\mathsf{pk}}(1; t_i)$, and (say) $g_2^{y_{\psi^{-1}(i)}(\chi)}$ is replaced by the second element $g_2^{\gamma(r_i y_0(\chi)+y_{\psi^{-1}(i)}(\chi))}$ of a commitment to $\mathbf{\Psi}_i$. The resulting complication is minor (it requires one to include into the shuffle argument a single ciphertext $U \in \mathfrak{G}_1^2$ that compensates for the added randomness). The full shuffle argument consists of commitments to $\mathbf{\Psi}$ and to $\mathbf{t}$ (both committed twice, w.r.t. the polynomials $(y_i(X))_{i=0}^{n}$ and $(\hat{y}_i(X))_{i=0}^{n}$), n unit vector arguments (one for each row of $\mathbf{\Psi}$), $n-1$ same-message arguments, and finally U.

If $\hat{y}_i(X)$ are well-chosen, then from the two verification equations and the soundness of the unit vector and same-message arguments it follows, under a new computational assumption PSP (*Power Simultaneous Product*), related to an assumption from [29]), that $\mathfrak{v}'_i = \mathfrak{v}_{\psi(i)}$ for every i.

Authors prove culpable soundness [29, 31] of the new argument. Since the security of the new shuffle argument does not depend on the cryptosystem either having knowledge components or being lifted, one can use Elgamal encryption [15] instead of the non-standard knowledge BBS encryption introduced in [35]. Since the cryptosystem does not have to be lifted, one can use more complex voting mechanisms with more complex ballots. The use of knowledge assumptions means that the new argument is an argument of knowledge.

The new shuffle argument can be largely precomputed by the prover and forwarded to the verifier even before the common input (i.e., ciphertexts) arrive. Similarly, the verifier can perform a large part of verification before receiving the ciphertexts. (See [47] for motivation for precomputation.) The prover's computation in the online phase is dominated by just two $(n + 1)$-wide multi-exponentiations (the computation of U). The multi-exponentiations can be parallelized; this is important in practice due to the wide availability of highly parallel graphics processors.

Bibliography

[1] Masayuki Abe. Mix-Networks on Permutation Networks. In Kwok-Yan Lam, Eiji Okamoto, and Chaoping Xing, editors, *ASIACRYPT 1999*, volume 1716 of *LNCS*, pages 258–273, Singapore, 14–18 November 1999. Springer, Heidelberg.

[2] Masayuki Abe and Serge Fehr. Perfect NIZK with Adaptive Soundness. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 118–136, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg.

[3] Miguel Ambrona, Gilles Barthe, and Benedikt Schmidt. Automated Unbounded Analysis of Cryptographic Constructions in the Generic Group Model. In Marc Fischlin and Jean-Sebastien Coron, editors, *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 822–851, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg.

[4] Stephanie Bayer and Jens Groth. Efficient Zero-Knowledge Argument for Correctness of a Shuffle. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg.

[5] Mihir Bellare and Phillip Rogaway. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In Victoria Ashby, editor, *ACM CCS 1993*, pages 62–73, Fairfax, Virginia, 3–5 November 1993. ACM Press.

[6] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the Existence of Extractable One-Way Functions. In David Shmoys, editor, *STOC 2014*, pages 505–514, New York, NY, USA, May 31 – Jun 3, 2014. ACM Press.

[7] Manuel Blum, Paul Feldman, and Silvio Micali. Non-Interactive Zero-Knowledge and Its Applications. In *STOC 1988*, pages 103–112, Chicago, Illinois, USA, May 2–4, 1988. ACM Press.

[8] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. In Matthew K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55, Santa Barbara, USA, August 15–19, 2004. Springer, Heidelberg.

[9] Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited. In Jeffrey Scott Vitter, editor, *STOC 1998*, pages 209–218, Dallas, Texas, USA, May 23–26, 1998.

[10] Rafik Chaabouni, Helger Lipmaa, and Bingsheng Zhang. A Non-Interactive Range Proof with Constant Communication. In Angelos Keromytis, editor, *FC 2012*, volume 7397 of *LNCS*, pages 179–199, Bonaire, The Netherlands, Feb 27–Mar 2, 2012. Springer, Heidelberg.

[11] Stephen A Cook and Stål O Aanderaa. On the minimum computation time of functions. *Transactions of the American Mathematical Society*, 142:291–314, 1969.

[12] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

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

[14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square Span Programs with Applications to Succinct NIZK Arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014 (1)*, volume 8873 of *LNCS*, pages 532–550, Kaohsiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg.

[15] Taher Elgamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Trans. on Inf. Theory*, 31(4):469–472, 1985.

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

[17] Prastudy Fauzi, Helger Lipmaa, and Bingsheng Zhang. Efficient Non-Interactive Zero Knowledge Arguments for Set Operations. In Nicolas Christin and Rei Safavi-Naini, editors, *FC 2014*, volume ? of *LNCS*, pages 216–233, Bridgetown, Barbados, March 3–7, 2014. Springer, Heidelberg.

[18] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *CRYPTO 1986*, volume 263 of *LNCS*, pages 186–194, Santa Barbara, California, USA, 11–15 August 1986. Springer, Heidelberg, 1987.

[19] Jun Furukawa. Efficient and Verifiable Shuffling and Shuffle-Decryption. *IEICE Transactions*, 88-A(1):172–188, 2005.

[20] Jun Furukawa, Hiroshi Miyauchi, Kengo Mori, Satoshi Obana, and Kazue Sako. An implementation of a universally verifiable electronic voting scheme based on shuffling. In Matt Blaze, editor, *Financial Cryptography, 6th International Conference, FC 2002, Southampton, Bermuda, March 11-14, 2002, Revised Papers*, volume 2357 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 2002.

[21] Jun Furukawa and Kazue Sako. An Efficient Scheme for Proving a Shuffle. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 368–387, Santa Barbara, USA, August 19–23, 2001. Springer, Heidelberg.

[22] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic Span Programs and NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645, Athens, Greece, April 26–30, 2013. Springer, Heidelberg.

[23] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof-Systems. In Robert Sedgewick, editor, *STOC 1985*, pages 291–304, Providence, Rhode Island, USA, May 6–8, 1985. ACM Press.

[24] Philippe Golle, Stanislaw Jarecki, and Ilya Mironov. Cryptographic Primitives Enforcing Communication and Storage Complexity. In Matt Blaze, editor, *FC 2002*, volume 2357 of *LNCS*, pages 120–135, Southhampton Beach, Bermuda, March 11–14, 2002. Springer, Heidelberg.

[25] Jens Groth. A Verifiable Secret Shuffle of Homomorphic Encryptions. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 145–160, Miami, Florida, USA, January 6–8, 2003. Springer, Heidelberg.

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

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

[28] Jens Groth and Yuval Ishai. Sub-linear Zero-Knowledge Argument for Correctness of a Shuffle. In Smart [42], pages 379–396.

[29] Jens Groth and Steve Lu. A Non-interactive Shuffle with Pairing Based Verifiability. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 51–67, Kuching, Malaysia, December 2–6, 2007. Springer, Heidelberg.

[30] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect Non-Interactive Zero-Knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 338–359, St. Petersburg, Russia, May 28–June 1, 2006. Springer, Heidelberg.

[31] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New Techniques for Noninteractive Zero-Knowledge. *Journal of the ACM*, 59(3), 2012.

[32] Jens Groth and Amit Sahai. Efficient Non-interactive Proof Systems for Bilinear Groups. In Smart [42], pages 415–432.

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

[34] Helger Lipmaa. Prover-Efficient Commit-And-Prove Zero-Knowledge SNARKs. In David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 2016*, volume 9646 of *LNCS*, pages 185–206, Fes, Morocco, April 13–15, 2016. Springer, Heidelberg.

[35] Helger Lipmaa and Bingsheng Zhang. A More Efficient Computationally Sound Non-Interactive Zero-Knowledge Shuffle Argument. In Visconti and Prisco [46], pages 477–502.

[36] Helger Lipmaa and Bingsheng Zhang. A More Efficient Computationally Sound Non-Interactive Zero-Knowledge Shuffle Argument. *Journal of Computer Security*, 21(5):685–719, 2013.

[37] Moni Naor. On Cryptographic Assumptions and Challenges. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109, Santa Barbara, USA, August 17–21, 2003. Springer, Heidelberg.

[38] C. Andrew Neff. A Verifiable Secret Shuffle and Its Application to E-Voting. In *ACM CCS 2001*, pages 116–125, Philadelphia, Pennsylvania, USA, November 6–8 2001. ACM Press.

[39] Alfredo Rial, Markulf Kohlweiss, and Bart Preneel. Universally Composable Adaptive Priced Oblivious Transfer. In Hovav Shacham and Brent Waters, editors, *Pairing 2009*, volume 5671 of *LNCS*, pages 231–247, Palo Alto, CA, USA, August 12–14, 2009. Springer, Heidelberg.

[40] Kazue Sako and Joe Kilian. Receipt-Free Mix-Type Voting Scheme - A Practical Solution to the Implementation of a Voting Booth. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *EUROCRYPT 1995*, volume 921 of *LNCS*, pages 393–403, Saint-Malo, France, 21–25 May 1995. Springer, Heidelberg.

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

[42] Nigel Smart, editor. *EUROCRYPT 2008*, volume 4965 of *LNCS*, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg.

[43] Björn Terelius and Douglas Wikström. Proofs of Restricted Shuffles. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 2010*, volume 6055 of *LNCS*, pages 100–113, Stellenbosch, South Africa, May 3–6, 2010. Springer, Heidelberg.

[44] Björn Terelius and Douglas Wikström. Efficiency Limitations of Σ-Protocols for Group Homomorphisms Revisited. In Visconti and Prisco [46], pages 461–476.

[45] Andrei L Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. In *Soviet Mathematics Doklady*, volume 3, pages 714–716, 1963.

[46] Ivan Visconti and Roberto De Prisco, editors. *SCN 2012*, volume 7485 of *LNCS*, Amalfi, Italy, September 5–7, 2012. Springer, Heidelberg.

[47] Douglas Wikström. A Commitment-Consistent Proof of a Shuffle. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP 2009*, volume 5594 of *LNCS*, pages 4007–421, Brisbane, Australia, July 1–3, 2009. Springer, Heidelberg.

Part II

Initial design options for mix-nets

4. Initial design options for mix-nets: Efficient Culpably Sound NIZK Shuffle Argument without Random Oracles

One way to guarantee security against malicious voting servers, implementing mix-nets, is to use non-interactive zero-knowledge (NIZK) shuffle arguments. Up to now, only two NIZK shuffle arguments in the CRS model have been proposed. Both arguments are relatively inefficient compared to known random oracle based arguments. We propose a new, more efficient, shuffle argument in the CRS model. Importantly, its online prover's computational complexity is 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, it satisfies a stronger notion of soundness. This chapter presents a new efficient NIZK shuffle argument, which serves as a design option for WP5 of PANORAMIX project.

4.1 Introduction

A mix network, or mix-net, is a network of mix-servers designed to remove the link between ciphertexts and their senders. To achieve this goal, a mix-server of a mix-net initially obtains a list of ciphertexts $(z_i)_{i=1}^n$. It then re-randomizes and permutes this list, and outputs the new list $(z_i')_{i=1}^n$ together with a non-interactive zero knowledge (NIZK, [2]) shuffle argument [22] that proves the re-randomization and permutation was done correctly, without leaking any side information. If enc is a multiplicatively homomorphic public-key cryptosystem like Elgamal [7], a shuffle argument convinces the verifier that there exists a permutation ψ and a vector $\boldsymbol{t}$ of randomizers such that $z_i' = z_{\psi(i)} \cdot \mathsf{enc}_{\mathsf{pk}}(1; t_i)$, without revealing any information about ψ or $\boldsymbol{t}$. Mix-nets improve security against malicious voting servers in e-voting. Other applications of mix-nets include anonymous web browsing, payment systems, and secure multiparty computation.

It is important to have a *non-interactive* shuffle argument outputting a short bit string that can be verified by anybody (possibly years later) without interacting with the prover. Many NIZK shuffle arguments are known in the random oracle model, see for example [10, 20, 9, 23, 13]. Since the random oracle model is only a heuristic, it is strongly recommended to construct NIZK arguments in the common reference string (CRS) model [2], without using random oracles. [1] We note that the most efficient shuffle arguments in the random oracle model like [13] also require a CRS.

Up to now, only two NIZK shuffle arguments in the CRS model have been proposed, by

[1] In a practical implementation of a mix-net, one can use the random oracle model also for other purposes, such as to construct a pseudo-number generator or a public-key cryptosystem. In most of such cases, it is known how to avoid the random oracle model, although this almost always incurs some additional cost.

Groth and Lu [15] and Lipmaa and Zhang [18, 19], both of which are significantly slower than the fastest arguments in the random oracle model (see Tbl. 4.1). The Groth-Lu shuffle argument only provides culpable soundness [15, 16] in the sense that if a malicious prover can create an accepting shuffle argument for an incorrect statement, then this prover *together* with a party that knows the secret key can break the underlying security assumptions. Relaxation of the soundness property is unavoidable, since [1] showed that only languages in **P/poly** can have direct black-box adaptive perfect NIZK arguments under a (polynomial) cryptographic hardness assumption. If the underlying cryptosystem is IND-CPA secure, then the shuffle language is *not* in **P/poly**, and thus it is necessary to use knowledge assumptions [5] to prove its adaptive soundness. Moreover, [15] argued that culpable soundness is a sufficient security notion for shuffles, since in any real-life application of the shuffle argument there exists some coalition of parties who knows the secret key.

Lipmaa and Zhang [18] proposed a more efficient NIZK shuffle argument by using knowledge assumptions under which they also bypassed the impossibility result of [1] and proved that their shuffle argument is sound. However, their shuffle argument is sound only under the assumption that there is an extractor that has access to the random coins of all encrypters, e.g., all voters, allowing her to extract all plaintexts and randomizers. We say in this case that the argument is *white-box sound*. White-box soundness is clearly a weaker security notion than culpable soundness of [15], and it would be good to avoid it.

In addition, the use of knowledge assumptions in [18] forces the underlying BBS [4] cryptosystem to include knowledge components (so ciphertexts are twice as long) and be lifted (meaning that one has to solve discrete logarithm to decrypt, so plaintexts must be small). Thus, one has to use a random oracle-less range argumentto guarantee that the plaintexts are small and thus to guarantee the soundness of the *shuffle* argument (see [18] for a discussion). While range proofs only have to be verified once (e.g., by only one mix-server), this still means that the shuffle argument of [18] is somewhat slower than what is given in Tbl. 4.1. Moreover, in the case of e-voting, using only small plaintexts restricts the applicability of a shuffle argument to only certain voting mechanisms like majority. On the other hand, a mechanism such as Single Transferable Vote would likely be unusable due to the length of the ballots.

Tbl. 4.1 provides a brief comparison between known NIZK shuffle arguments in the CRS model and the most computationally efficient known shuffle argument in the random oracle model [13]. We emphasize that the values in parentheses show the cost of computing and communicating the shuffled ciphertexts themselves, and must be added to the rest. Moreover, the cost of the shuffle argument from [18] should include the cost of a range argument. Unless written otherwise, the communication and the CRS length are given in group elements, the prover's computational complexity is given in exponentiations, and the verifier's computational complexity is given in bilinear pairings. In each row, highlighted cells denote the best efficiency or best security (e.g., not requiring the PKE assumption) among arguments in the CRS model. Of course, a full efficiency comparison can only be made after implementing the different shuffle arguments.

This brings us to the main question of the current paper:

> *Is it possible to construct an NIZK shuffle argument in the CRS model that is comparable in efficiency with existing random oracle model NIZK shuffle arguments? Moreover, can one do it while minimizing the use of knowledge assumptions (i.e., not requiring the knowledge extractor to have access to the random coins used by all encrypters) and using a standard, non-lifted, cryptosystem?*

Our Contributions.

We give a partial answer to the main question. We propose a new pairing-based NIZK shuffle argument in the CRS model. Differently from [18], we prove the culpable soundness of the

Table 4.1: A comparison of different NIZK shuffle arguments, compared with the computationally most efficient known shuffle argument in the random oracle model [13].

	[15]	[19]	This work	[13]
$\lvert\mathrm{CRS}\rvert$	$2n+8$	$7n+6$	$8n+17$	$n+1$
Communication	$15n+120$ $(+3n)$	$6n+11$ $(+6n)$	$7n+2$ $(+2n)$	$480n$ bits
pro's comp.	$51n+246$ $(+3n)$	$22n+11$ $(+6n)$	$16n+3$ $(+2n)$	$6n$ $(+2n)$
ver's comp.	$75n+282$	$28n+18$	$18n+6$	$6n$ exp.
Lifted	No	Yes	No	No
Soundness	Culp. sound	White-box sound	Culp. sound	Sound
Arg. of knowl.	no	yes	yes	yes
PKE (knowl. assm.)	no	yes	yes	no
Random oracle	no	no	no	yes

new argument instead of white-box soundness. Compared to [15], which also achieves culpable soundness, the new argument has 3 times faster proving and more than 4 times faster verification. Compared to [15, 18], it is based on a more standard cryptosystem (Elgamal). While the new shuffle argument is still at least 2 times slower than the most efficient known random oracle based shuffle arguments, it has almost optimal *online* prover's computation. Of course, a full efficiency comparison can only be made after implementing the different shuffle arguments.

Our construction works as as follows. We first commit to the permutation ψ (by committing separately to first $n-1$ rows of the corresponding permutation matrix $\boldsymbol{\Psi}$) and to the vector $\boldsymbol{t}$ of blinding randomizers. Here, we use the *polynomial commitment scheme* (see Sect. 4.2) with $\mathsf{com}(\mathsf{ck};\boldsymbol{m};r) = (g_1, g_2^{\gamma})^{rP_0(\chi)+\sum_{i=1}^{n} m_i P_i(\chi)} \in \mathbb{G}_1 \times \mathbb{G}_2$, in pairing-based setting, where $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear pairing, g_i is a generator of $\mathbb{G}_i$ for $i \in \{1,2\}$, $(P_i(X))_{i=0}^{n}$ is a tuple of linearly independent polynomials, χ is a trapdoor, γ is a knowledge secret, and $\mathsf{ck} = ((g_1, g_2^{\gamma})^{P_i(\chi)})_{i=0}^{n}$ is the CRS. For different values of $P_i(X)$, variants of this commitment scheme have been proposed before [12, 14, 17].

We show that $\boldsymbol{\Psi}$ is a correct permutation matrix by constructing n witness-indistinguishable succinct *unit vector arguments*, each of which guarantees that a row of $\boldsymbol{\Psi}$ is a unit vector, for implicitly constructed $\boldsymbol{\Psi}_n = \boldsymbol{1_n} - \sum_{i=1}^{n-1} \boldsymbol{\Psi}_i$. We use the recent square span programs (SSP, [6]) approach to choose the polynomials $P_i(X) = y_i(X)$ so that the unit vector argument is efficient. Since unit vectors are used in many contexts, we hope this argument is of independent interest.

After that, we postulate a natural concrete verification equation for shuffles, and construct the shuffle argument from this. If privacy were not an issue (and thus $z'_i = z_{\psi(i)}$ for every i), the verification equation would just be the tautology $\prod_{i=1}^{n} \hat{e}(z'_i, g_2^{y_i(\chi)}) \overset{?}{=} \prod_{i=1}^{n} \hat{e}(z_i, g_2^{y_{\psi^{-1}(i)}(\chi)})$. Clearly, if the prover is honest, this equation holds. However, it does not yet guarantee soundness, since an adversary can use $g_1^{y_j(\chi)}$ (given in the CRS) to create $(z'_i)_{i=1}^{n}$ in a malicious way. To eliminate this possibility, by roughly following an idea from [15], we also verify that $\prod_{i=1}^{n} \hat{e}(z'_i, g_2^{\hat{y}_i(\chi)}) \overset{?}{=} \prod_{i=1}^{n} \hat{e}(z_i, g_2^{\hat{y}_{\psi^{-1}(i)}(\chi)})$ for some well-chosen polynomials $\hat{y}_i(X)$. (We note that instead of n univariate polynomials, [15] used n random variables χ_i, increasing the size of the secret key to $\Omega(n)$ bits.)

To show that the verifications are instantiated correctly, we also need a *same-message argument* that shows that commitments w.r.t. two tuples of polynomials $(y_i(X))_{i=1}^{n}$ and $(\hat{y}_i(X))_{i=1}^{n}$ commit to the same plaintext vectors. We construct an efficient same-message argument by using an approach that is (again, roughly) motivated by the QAP-based approach of [11]. This argument is an argument of knowledge, given that the polynomials $\hat{y}_i(X)$ satisfy an additional restriction.

Since we also require privacy, the actual verification equations are more complicated. In particular, $z'_i = z_{\psi(i)} \cdot \mathsf{enc}_{\mathsf{pk}}(1; t_i)$, and (say) $g_2^{y_{\psi^{-1}(i)}(\chi)}$ is replaced by the second element $g_2^{\gamma(r_i y_0(\chi) + y_{\psi^{-1}(i)}(\chi))}$ of a commitment to $\mathbf{\Psi}_i$. The resulting complication is minor (it requires one to include into the shuffle argument a single ciphertext $U \in \mathbb{G}_1^2$ that compensates for the added randomness). The full shuffle argument consists of commitments to $\mathbf{\Psi}$ and to $\boldsymbol{t}$ (both committed twice, w.r.t. the polynomials $(y_i(X))_{i=0}^n$ and $(\hat{y}_i(X))_{i=0}^n$), n unit vector arguments (one for each row of $\mathbf{\Psi}$), $n-1$ same-message arguments, and finally U.

If $\hat{y}_i(X)$ are well-chosen, then from the two verification equations and the soundness of the unit vector and same-message arguments it follows, under a new computational assumption PSP (*Power Simultaneous Product*, related to an assumption from [15]), that $z'_i = z_{\psi(i)}$ for every i.

We prove culpable soundness [15, 16] of the new argument. Since the security of the new shuffle argument does not depend on the cryptosystem either having knowledge components or being lifted, we can use Elgamal encryption [7] instead of the non-standard knowledge BBS encryption introduced in [18]. Since the cryptosystem does not have to be lifted, one can use more complex voting mechanisms with more complex ballots. The use of knowledge assumptions means that the new argument is an argument of knowledge.

The new shuffle argument can be largely precomputed by the prover and forwarded to the verifier even before the common input (i.e., ciphertexts) arrive. Similarly, the verifier can perform a large part of verification before receiving the ciphertexts. (See [24] for motivation for precomputation.) The prover's computation in the online phase is dominated by just two $(n+1)$-wide multi-exponentiations (the computation of U). The multi-exponentiations can be parallelized; this is important in practice due to the wide availability of highly parallel graphics processors.

Main Technical Challenges.

While the main objective of the current work is efficiency, we emphasize that several steps of the new shuffle argument are technically involved. Throughout the paper, we use and combine very recent techniques from the design of efficient succinct non-interactive arguments of knowledge (SNARKs, [11, 21, 6], that are constructed with the main goal of achieving efficient verifiable computation) with quite unrelated techniques from the design of efficient shuffle arguments [15, 18].

The security of the new shuffle argument relies on a new assumption, PSP. We prove that PSP holds in the generic bilinear group model, given that polynomials $\hat{y}_i(X)$ satisfy a very precise criterion. For the security of the SSP-based unit vector argument, we need $y_i(X)$ to satisfy another criterion, and for the security of the same-message argument, we need $y_i(X)$ and $\hat{y}_i(X)$ to satisfy a third criterion. The fact that polynomials $y_i(X)$ and $\hat{y}_i(X)$ that satisfy all three criteria exist is not a priori clear; $y_i(X)$ and $\hat{y}_i(X)$ (see Prop. 3) are also unlike any polynomials from the related literature on non-interactive zero knowledge.

Finally, the PSP assumption was carefully chosen so it will hold in the generic bilinear group model, and so the reduction from culpable soundness of the shuffle argument to the PSP assumption would work. While the PSP assumption is related to the SP assumption from [15], the situation in [15] was less fragile due to the use of independent random variables X_i and X_i^2 instead of polynomials $y_i(X)$ and $\hat{y}_i(X)$. In particular, the same-message argument is trivial in the case of using independent random variables.

4.2 Preliminaries

Let n be the number of ciphertexts to be shuffled. Let S_d be the symmetric group of d elements. Let $\mathbb{G}^*$ denote the group $\mathbb{G}$ without its identity element. For $a \leq b$, let $[a\,..\,b] := \{c \in \mathbb{Z} : a \leq$

$c \leq b\}$. Denote $(a, b)^c := (a^c, b^c)$. For a set of polynomials $\mathcal{F}$ that have the same domain, denote $g^{\mathcal{F}(\boldsymbol{a})} := (g^{f(\boldsymbol{a})})_{f \in \mathcal{F}}$.

A *permutation matrix* is a Boolean matrix with exactly one 1 in every row and column. If ψ is a permutation then the corresponding permutation matrix $\boldsymbol{\Psi}_\psi$ is such that $(\boldsymbol{\Psi}_\psi)_{ij} = 1$ iff $j = \psi(i)$. Thus $(\boldsymbol{\Psi}_{\psi^{-1}})_{ij} = 1$ iff $i = \psi(j)$. Clearly, $\boldsymbol{\Psi}$ is a permutation matrix iff its every row is a unit vector, and the sum of all its row vectors is equal to the all-ones vector $\mathbf{1}_n$.

Let κ be the security parameter. We denote $f(\kappa) \approx_\kappa g(\kappa)$ if $|f(\kappa) - g(\kappa)|$ is negligible in κ. We abbreviate (non-uniform) probabilistic-polynomial time by (NU)PPT. On input 1^κ, a *bilinear map generator* BP returns $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are multiplicative cyclic groups of prime order p, and $\hat{e}$ is an efficient bilinear map $\hat{e} \colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that satisfies the following two properties, where g_1 (resp., g_2) is an arbitrary generator of $\mathbb{G}_1$ (resp., $\mathbb{G}_2$): (i) $\hat{e}(g_1, g_2) \neq 1$, and (ii) $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$. Thus, $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1^c, g_2^d)$ iff $ab \equiv cd \pmod{p}$. We give BP another input, n (related to the input length), and allow p to depend on n. Finally, we assume that all algorithms that handle group elements reject if their inputs do not belong to corresponding groups.

We will now give short explanations of the main knowledge assumptions. Let $1 < d(n) < d^*(n) = \operatorname{poly}(\kappa)$ be two functions. We say that BP is

- $d(n)$*-PDL (Power Discrete Logarithm, [17]) secure* if any NUPPT adversary, given values $((g_1, g_2)^{\chi^i})_{i=0}^{d(n)}$, has negligible probability of producing χ.
- $(d(n), d^*(n))$*-PCDH (Power Computational Diffie-Hellman, [12, 14, 11]) secure* if any NUPPT adversary, given values $((g_1, g_2)^{\chi^i})_{i \in [0\,..\,d^*(n)] \setminus \{d(n)+1\}}$, has negligible probability of producing $g_1^{\chi^{d(n)+1}}$.
- $d(n)$*-TSDH (Target Strong Diffie-Hellman, [3, 21]) secure* if any NUPPT adversary, given values $((g_1, g_2)^{\chi^i})_{i=0}^{d(n)}$, has negligible probability of producing a pair of values $\left(r, \hat{e}(g_1, g_2)^{1/(\chi - r)}\right)$ where $r \neq \chi$.

For algorithms A and X_A, we write $(y; y') \leftarrow (\mathsf{A} || X_\mathsf{A})(\chi)$ if A on input χ outputs y, and X_A on the same input (including the random tape of A) outputs y' [1]. We will need knowledge assumptions w.r.t. up to 2 knowledge secrets γ_i. Let m be the number of different knowledge secrets in any concrete argument, in the current paper $m \leq 2$. Let $\mathcal{F} = (P_i)_{i=0}^n$ be a tuple of univariate polynomials, and $\mathcal{G}_1$ (resp., $\mathcal{G}_2$) be a tuple of univariate (resp., m-variate) polynomials. For $i \in [1\,..\,m]$, BP is $(\mathcal{F}, \mathcal{G}_1, \mathcal{G}_2, \gamma_i)$*-PKE (Power Knowledge of Exponent, [14]) secure* if for any NUPPT adversary A there exists a NUPPT extractor X_A, such that

$$\Pr\left[\begin{array}{l} \mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n), (g_1, g_2, \chi) \leftarrow_r \mathbb{G}_1^* \times \mathbb{G}_2^* \times \mathbb{Z}_p, \boldsymbol{\gamma} \leftarrow_r \mathbb{Z}_p^m, \\ \boldsymbol{\gamma_{-i}} = (\gamma_1, \dots, \gamma_{i-1}, \gamma_{i+1}, \dots, \gamma_m), \mathsf{aux} \leftarrow \left(g_1^{\mathcal{G}_1(\chi)}, g_2^{\mathcal{G}_2(\chi, \boldsymbol{\gamma_{-i}})}\right), \\ (h_1, h_2; (a_i)_{i=0}^n) \leftarrow (\mathsf{A} || X_\mathsf{A})(\mathsf{gk}; (g_1, g_2^{\gamma_i})^{\mathcal{F}(\chi)}, \mathsf{aux}) : \\ \hat{e}(h_1, g_2^{\gamma_i}) = \hat{e}(g_1, h_2) \wedge h_1 \neq g_1^{\sum_{i=0}^n a_i P_i(\chi)} \end{array}\right] \approx_\kappa 0 \enspace .$$

Here, aux can be seen as the common auxiliary input to A and X_A that is generated by using benign auxiliary input generation. The definition implies that aux may depend on $\boldsymbol{\gamma_{-i}}$ but not on γ_i. If $\mathcal{F} = (X^i)_{i=0}^d$ for some $d = d(n)$, then we replace the first argument in $(\mathcal{F}, \dots)$-PKE with d. If $m = 1$, then we omit the last argument γ_i in $(\mathcal{F}, \dots, \gamma_i)$-PKE.

We will use the Elgamal cryptosystem [7] $\Pi = (\mathsf{BP}, \mathsf{genpkc}, \mathsf{enc}, \mathsf{dec})$, defined as follows, in the bilinear setting.

Setup (1^κ)**:** Let $\mathsf{gk} \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathsf{BP}(1^\kappa)$.

Key Generation $\mathsf{genpkc}(\mathsf{gk})$**:** Let $g_1 \leftarrow_r \mathbb{G}_1^*$. Set the secret key $\mathsf{sk} \leftarrow_r \mathbb{Z}_p$, and the public key $\mathsf{pk} \leftarrow (g_1, h = g_1^{\mathsf{sk}})$. Output $(\mathsf{pk}, \mathsf{sk})$.

Encryption $\mathsf{enc}_{\mathsf{pk}}(m; r)$**:** To encrypt a message $m \in \mathbb{G}_1$ with randomizer $r \in \mathbb{Z}_p$, output the ciphertext $\mathsf{enc}_{\mathsf{pk}}(m; r) = \mathsf{pk}^r \cdot (1, m) = (g^r, mh^r)$.

Decryption $\mathsf{dec}_{\mathsf{sk}}(c_1, c_2)$**:** $m = c_2 / c_1^{\mathsf{sk}} = mh^r / h^r = m$.

Elgamal is clearly multiplicatively homomorphic. In particular, if $t \leftarrow_r \mathbb{Z}_p$, then for *any* m and r, $\mathsf{enc}_{\mathsf{pk}}(m;r) \cdot \mathsf{enc}_{\mathsf{pk}}(1;t) = \mathsf{enc}_{\mathsf{pk}}(m;r+t)$ is a random encryption of m. Elgamal is IND-CPA secure under the XDH assumption.

An extractable trapdoor commitment scheme consists of two efficient algorithms gencom (that outputs a CRS and a trapdoor) and com (that, given a CRS, a message and a randomizer, outputs a commitment), and must satisfy the following four security properties.

Computational binding: without access to the trapdoor, it is intractable to open a commitment to two different messages.

Trapdoor: given access to the original message, the randomizer and the trapdoor, one can open the commitment to any other message.

Perfect hiding: commitments of any two messages have the same distribution.

Extractable: given access to the CRS, the commitment, and the random coins of the committer, one can obtain the value that the committer committed to.

See, e.g., [14] for formal definitions.

We use the following extractable trapdoor *polynomial commitment scheme* that generalizes various earlier commitment schemes [12, 14, 17]. Let $n = \mathrm{poly}(\kappa)$, $n > 0$, be an integer. Let $P_i(X) \in \mathbb{Z}_p[X]$, for $i \in [0\,..\,n]$, be $n+1$ linearly independent low-degree polynomials. First, $\mathsf{gencom}(1^\kappa, n)$ generates $\mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n)$, picks $g_1 \leftarrow_r \mathbb{G}_1^*$, $g_2 \leftarrow_r \mathbb{G}_2^*$, and then outputs the CRS $\mathsf{ck} \leftarrow ((g_1^{P_i(\chi)}, g_2^{\gamma P_i(\chi)})_{i=0}^n)$ for $\chi \leftarrow_r \mathbb{Z}_p \setminus \{j : P_0(j) = 0\}$ and $\gamma \leftarrow_r \mathbb{Z}_p$. The trapdoor is equal to $\mathsf{td}_{\mathsf{com}} = \chi$.

The commitment of $\boldsymbol{a} \in \mathbb{Z}_p^n$, given a randomizer $r \leftarrow_r \mathbb{Z}_p$, is $\mathsf{com}(\mathsf{ck}; \boldsymbol{a}; r) := (g_1^{P_0(\chi)}, g_2^{\gamma P_0(\chi)})^r \cdot \prod_{i=1}^n (g_1^{P_i(\chi)}, g_2^{\gamma P_i(\chi)})^{a_i} \in \mathbb{G}_1 \times \mathbb{G}_2$. The validity of a commitment (A_1, A_2^γ) can be checked by verifying that $\hat{e}(A_1, g_2^{\gamma P_0(\chi)}) = \hat{e}(g_1^{P_0(\chi)}, A_2^\gamma)$. To open a commitment, the committer sends $(\boldsymbol{a}, r)$ to the verifier.

Theorem 1. *Denote $\mathcal{F}_{\mathsf{com}} = (P_i(X))_{i=0}^n$. The polynomial commitment scheme is perfectly hiding and trapdoor. Let $d := \max_{f \in \mathcal{F}_{\mathsf{com}}}(\deg f)$. If BP is d-PDL secure, then it is computationally binding. If BP is $(\mathcal{F}_{\mathsf{com}}, \emptyset, \emptyset)$-PKE secure, then it is extractable.*

Alternatively, we can think of com as being a commitment scheme that does not depend on the concrete polynomials at all, and the description of P_i is just given as a part of ck. We instantiate the polynomial commitment scheme with concrete polynomials later in Sect. 4.3 and Sect. 4.6.

An NIZK argument for a group-dependent language $\mathcal{L}$ consists of four algorithms, setup, gencrs, pro and ver. The setup algorithm setup takes as input 1^κ and n (the input length), and outputs the group description gk. The CRS generation algorithm gencrs takes as input gk and outputs the prover's CRS crs_p, the verifier's CRS crs_v, and a trapdoor td. (td is only required when the argument is zero-knowledge.) The distinction between crs_p and crs_v is only important for efficiency. The prover pro takes as input gk and crs_p, a statement u, and a witness w, and outputs an argument π. The verifier ver takes as input gk and crs_v, a statement u, and an argument π, and either accepts or rejects.

Some of the properties of an argument are: (i) *perfect completeness* (honest verifier always accepts honest prover's argument), (ii) *perfect witness-indistinguishability* (argument distributions corresponding to all allowable witnesses are equal), (iii) *perfect zero knowledge* (there exists an efficient simulator that can, given u, $(\mathsf{crs}_p, \mathsf{crs}_v)$ and td, output an argument that comes from the same distribution as the argument produced by the prover), (iv) *adaptive computational soundness* (if $u \notin \mathcal{L}$, then an arbitrary non-uniform probabilistic polynomial time prover has negligible success in creating a satisfying argument), and (v) *adaptive computational culpable soundness* [15, 16] (if $u \notin \mathcal{L}$, then an arbitrary NUPPT prover has negligible success in creating a satisfying argument together with a witness that $u \notin \mathcal{L}$). An argument is an *argument of knowledge*, if from an accepting argument it follows that the prover knows the witness.

4.3 Unit Vector Argument

In a unit vector argument, the prover aims to convince the verifier that he knows how to open a commitment (A_1, A_2^γ) to *some* $(\boldsymbol{e}_I, r)$, where $\boldsymbol{e}_I$ denotes the Ith unit vector for $I \in [1 \mathinner{..} n]$. We construct the unit vector argument by using square span programs (SSP-s, [6], an especially efficient variant of the quadratic arithmetic programs of [11]).

Clearly, $\boldsymbol{a} \in \mathbb{Z}_p^n$ is a unit vector iff the following $n+1$ conditions hold:

- $a_i \in \{0, 1\}$ for $i \in [1 \mathinner{..} n]$ (i.e., $\boldsymbol{a}$ is Boolean), and
- $\sum_{i=1}^n a_i = 1$.

We use the methodology of [6] to obtain an efficient NIZK argument out of these conditions. Let $\{0, 2\}^{n+1}$ denote the set of $(n+1)$-dimensional vectors where every coefficient is from $\{0, 2\}$, let $\circ$ denote the Hadamard (entry-wise) product of two vectors, let $V := \begin{pmatrix} 2 \cdot I_{n \times n} \\ \boldsymbol{1}_n^\top \end{pmatrix} \in \mathbb{Z}_p^{(n+1)\times n}$ and $\boldsymbol{b} := \begin{pmatrix} \boldsymbol{0}_n \\ 1 \end{pmatrix} \in \mathbb{Z}_p^{n+1}$. Clearly, the above $n+1$ conditions hold iff $V\boldsymbol{a} + \boldsymbol{b} \in \{0, 2\}^{n+1}$, i.e.,

$$(V\boldsymbol{a} + \boldsymbol{b} - \boldsymbol{1}_{n+1}) \circ (V\boldsymbol{a} + \boldsymbol{b} - \boldsymbol{1}_{n+1}) = \boldsymbol{1}_{n+1} \quad . \tag{4.1}$$

Let ω_i, $i \in [1 \mathinner{..} n+1]$ be $n+1$ different values. Let $Z(X) := \prod_{i=1}^{n+1}(X - \omega_i)$ be the unique degree $n+1$ monic polynomial, such that $Z(\omega_i) = 0$ for all $i \in [1 \mathinner{..} n+1]$. Let the ith Lagrange basis polynomial $\ell_i(X) := \prod_{i,j \in [1 \mathinner{..} n+1], j \neq i}((X - \omega_j)/(\omega_i - \omega_j))$ be the unique degree n polynomial, s.t. $\ell_i(\omega_i) = 1$ and $\ell_i(\omega_j) = 0$ for $j \neq i$. For a vector $\boldsymbol{x} \in \mathbb{Z}_p^{n+1}$, let $L_{\boldsymbol{x}}(X) = \sum_{i=1}^{n+1} x_i \ell_i(X)$ be a degree n polynomial that interpolates $\boldsymbol{x}$, i.e., $L_{\boldsymbol{x}}(\omega_i) = x_i$.

For $i \in [1 \mathinner{..} n]$, let $y_i(X)$ be the polynomial that interpolates the ith column of the matrix V. That is, $y_i(X) = 2\ell_i(X) + \ell_{n+1}(X)$ for $i \in [1 \mathinner{..} n]$. Let $y_0(X) = -1 + \ell_{n+1}(X)$ be the polynomial that interpolates $\boldsymbol{b} - \boldsymbol{1}_{n+1}$. We will use an instantiation of the polynomial commitment scheme with $\mathcal{F}_{\mathsf{com}} = (Z(X), (y_i(X))_{i=1}^n)$.

As in [6], we arrive at the polynomial $Q(X) = (\sum_{i=1}^n a_i y_i(X) + y_0(X))^2 - 1 = (y_I(X) + y_0(X))^2 - 1$ (here, we used the fact that $\boldsymbol{a} = \boldsymbol{e_I}$ for some $I \in [1 \mathinner{..} n]$), such that $\boldsymbol{a}$ is a unit vector iff $Z(X) \mid Q(X)$. As in [11, 6], to obtain privacy, we now add randomness to $Q(X)$, arriving at the degree $2(n+1)$ polynomial $Q_{wi}(X) = (rZ(X) + y_I(X) + y_0(X))^2 - 1$. By [11, 6], Eq. (4.1) holds iff

(i) $Q_{wi}(X) = (A(X) + y_0(X))^2 - 1$, where $A(X) = r_a Z(X) + \sum_{i=1}^n a_i y_i(X) \in \operatorname{span}(\mathcal{F}_{\mathsf{com}})$, and

(ii) $Z(X) \mid Q_{wi}(X)$.

An honest prover computes the degree $\leq n+1$ polynomial $\pi_{wi}(X) \leftarrow Q_{wi}(X)/Z(X) \in \mathbb{Z}_p[X]$, and sets the argument to be equal to $\pi_{uv}^* := g_1^{\pi_{wi}(\chi)}$ for a secret χ that instantiates X. If it exists, $\pi_{wi}(X) := Q_{wi}(X)/Z(X)$ is equal to $r^2 Z(X) + r \cdot 2(y_I(X) + y_0(X)) + \Pi_I(X)$, where for $i \in [1 \mathinner{..} n]$, $\Pi_i(X) := ((y_i(X) + y_0(X))^2 - 1)/Z(X)$ is a degree $\leq n-1$ polynomial and $Z(X) \mid ((y_i(X) + y_0(X))^2 - 1)$. Thus, computing π_{uv}^* uses two exponentiations.

We use a knowledge (PKE) assumption in a standard way to guarantee that $A(X)$ is in the span of $\{X^i\}_{i=0}^{n+1}$. As in [11, 6], we then guarantee condition (i) by using a PCDH assumption and condition (ii) by using a TSDH assumption. Here, we use the same technique as in [11] and subsequent papers by introducing an additional secret, β, and adding one group element A_1^β to the argument.

System parameters: Let com be the polynomial commitment scheme and let $\mathcal{F}_{\mathsf{com}} = (Z(X), (y_i(X))_{i=1}^n)$.
Setup $\mathsf{setup}_{uv}(1^\kappa, n)$**:** Let $\mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n)$.
CRS generation $\mathsf{gencrs}_{uv}(\mathsf{gk})$**:** Let $(g_1, g_2, \chi, \beta, \gamma) \leftarrow_r \mathbb{G}_1^* \times \mathbb{G}_2^* \times \mathbb{Z}_p^3$, s.t. $Z(\chi) \neq 0$.

$$\begin{aligned}
&\mathsf{ck} \leftarrow (g_1, g_2^\gamma)^{\mathcal{F}_{\mathsf{com}}(\chi)}, \\
&\mathsf{crs}_{uv,p} \leftarrow (\mathsf{ck}, (g_1^{2(y_i(\chi)+y_0(\chi))}, g_1^{\Pi_i(\chi)})_{i=1}^n, g_1^{\beta \cdot \mathcal{F}_{\mathsf{com}}(\chi)}), \\
&\mathsf{crs}_{uv,v} \leftarrow (g_1, g_1^{y_0(\chi)}, g_2^\gamma, g_2^{\gamma y_0(\chi)}, g_2^{\gamma Z(\chi)}, g_2^{\gamma\beta}, \hat{e}(g_1, g_2^\gamma)^{-1}).
\end{aligned}$$

Return $\mathsf{crs}_{uv} = (\mathsf{crs}_{uv,p}, \mathsf{crs}_{uv,v})$.

Common input: $(A_1, A_2^\gamma) = ((g_1, g_2^\gamma)^{Z(\chi)})^r (g_1, g_2^\gamma)^{y_I(\chi)}$ where $I \in [1\,..\,n]$.

Proving $\mathsf{pro}_{uv}(\mathsf{gk}, \mathsf{crs}_{uv,p}; A_1, A_2^\gamma; w_{uv} = (\boldsymbol{a} = \boldsymbol{e_I}, r))$**:** Set $\pi^*_{uv} \leftarrow (g_1^{Z(\chi)})^{r^2} \cdot (g_1^{2(y_I(\chi)+y_0(\chi))})^r \cdot g_1^{\Pi_I(\chi)}$. Set $A_1^\beta \leftarrow (g_1^{\beta Z(\chi)})^r g_1^{\beta y_I(\chi)}$. Output $\pi_{uv} = (\pi^*_{uv}, A_1^\beta) \in \mathbb{G}_1^2$.

Verification $\mathsf{ver}_{uv}(\mathsf{gk}, \mathsf{crs}_{uv,v}; A_1, A_2^\gamma; \pi_{uv})$**:** Parse π_{uv} as $\pi_{uv} = (\pi^*_{uv}, A_1^\beta)$. Verify that (1) $\hat{e}(\pi^*_{uv}, g_2^{\gamma Z(\chi)}) = \hat{e}(A_1 \cdot g_1^{y_0(\chi)}, A_2^\gamma \cdot g_2^{\gamma y_0(\chi)}) \cdot \hat{e}(g_1, g_2^\gamma)^{-1}$,(2) $\hat{e}(g_1, A_2^\gamma) = \hat{e}(A_1, g_2^\gamma)$, and (3) $\hat{e}(A_1, g_2^{\gamma\beta}) = \hat{e}(A_1^\beta, g_2^\gamma)$.

Set $\mathcal{F}_{uv,1} = \{1\} \cup \mathcal{F}_{\mathsf{com}} \cup X_\beta \mathcal{F}_{\mathsf{com}}$ and $\mathcal{F}_{uv,2} = Y\mathcal{F}_{\mathsf{com}} \cup \{Y, YX_\beta\}$. The formal variable X_β (resp., Y) stands for the secret key β (resp., γ). Since other elements of crs_{uv} are only needed for optimization, crs_{uv} can be computed from $\mathsf{crs}^*_{uv} = (g_1^{\mathcal{F}_{uv,1}(\chi,\beta)}, g_2^{\mathcal{F}_{uv,2}(\chi,\beta,\gamma)})$. If $n > 2$ then $1 \notin \mathrm{span}(\{Z(X)\} \cup \{y_i(X)\}_{i=1}^n)$, and thus $\{1, Z(X)\} \cup \{y_i(X)\}_{i=1}^n$ is a basis of all polynomials of degree at most $n+1$. Thus, $\mathcal{F}_{uv,1}$ can be computed iff $\{X^i\}_{i=0}^{n+1} \cup \{X_\beta \mathcal{F}_{\mathsf{com}}\}$ can be computed.

Theorem 2. *The new unit vector argument is perfectly complete and witness-indistinguishable. If* BP *is* $(n+1, 2n+3)$*-PCDH secure,* $(n+1)$*-TSDH secure, and* $(n+1, X_\beta\mathcal{F}_{\mathsf{com}}, \{YX_\beta\})$*-PKE secure, then this argument is an adaptive argument of knowledge.*

Proposition 1. *The computation of* (π^*_{uv}, A_1^β) *takes one* 2*-wide multi-exponentiation and* 1 *exponentiation in* $\mathbb{G}_1$*. In addition, it takes* 2 *exponentiations (one in* $\mathbb{G}_1$ *and one in* $\mathbb{G}_2$*) in the master argument to compute* (A_1, A_2^γ)*. The verifier computation is dominated by* 6 *pairings.*

4.4 New Same-Message Argument

In a *same-message argument*, the prover aims to convince the verifier that he knows, given two commitment keys ck and $\widehat{\mathsf{ck}}$ (that correspond to two tuples of polynomials $(P_i(X))_{i=0}^n$ and $(\hat{P}_i(X))_{i=0}^n$, respectively), how to open $(A_1, A_2^\gamma) = \mathsf{com}(\mathsf{ck}; \boldsymbol{m}; r)$ and $(\hat{A}_1, \hat{A}_2^{\hat{\gamma}}) = \mathsf{com}(\widehat{\mathsf{ck}}; \boldsymbol{m}; \hat{r})$ as commitments (w.r.t. ck and $\widehat{\mathsf{ck}}$) to the same plaintext vector $\boldsymbol{m}$ (but not necessarily to the same randomizer r).

We propose an efficient same-message argument using $\mathcal{F}_{\mathsf{com}} = (Z(X), (y_i(X))_{i=1}^n)$ as described in Sect. 4.3. In the shuffle argument, we need $(\hat{P}_i(X))_{i=0}^n$ to satisfy some specific requirements w.r.t. $\mathcal{F}_{\mathsf{com}}$, see Sect. 4.5. We are free to choose $\hat{P}_i$ otherwise. We concentrate on a choice of $\hat{P}_i$ that satisfies those requirements yet enables us to construct an efficient same-message argument.

Denote $\hat{Z}(X) = \hat{P}_0(X)$. For the same-message argument to be an argument of knowledge *and* efficient, we choose $\hat{P}_i$ such that $(\hat{P}_i(\omega_j))_{j=1}^{n+1} = (y_i(\omega_j))_{j=1}^{n+1} = 2\boldsymbol{e_i} + \boldsymbol{e_{n+1}}$ for $i \in [1\,..\,n]$. Moreover, $(\hat{Z}(\omega_j))_{j=1}^{n+1} = (Z(\omega_j))_{j=1}^{n+1} = \boldsymbol{0}_{n+1}$.

Following similar methodology as in Sect. 4.3, define

$$Q_{wi}(X) := (\hat{r}\hat{Z}(X) + \textstyle\sum_{i=1}^n \hat{m}_i \hat{P}_i(X)) - (rZ(X) + \sum_{i=1}^n m_i y_i(X)) \ .$$

Let $\hat{n}$ be the maximum degree of polynomials in $(y_i(X), \hat{P}_i(X))_{i=0}^n$, thus $\deg Q_{wi} \le \hat{n}$. Since $Q_{wi}(\omega_j) = 2(\hat{m}_j - m_j)$ for $j \in [1\,..\,n]$, $Q_{wi}(\omega_j) = 0$ iff $m_j = \hat{m}_j$. Moreover, if $\boldsymbol{m} = \boldsymbol{\hat{m}}$ then $Q_{wi}(\omega_{n+1}) = \sum_{i=1}^n \hat{m}_i - \sum_{i=1}^n m_i = 0$. Hence, $\boldsymbol{m} = \boldsymbol{\hat{m}}$ iff

(i) $Q_{wi}(X) = \hat{A}(X) - A(X)$, where $A(X) \in \mathrm{span}(\{Z(X)\} \cup \{y_i(X)\}_{i=1}^n)$, and $\hat{A}(X) \in \mathrm{span}(\{\hat{Z}(X)\} \cup \{\hat{P}_i(X)\}_{i=1}^n)$, and

(ii) there exists a degree $\le \hat{n} - (n+1)$ polynomial $\pi_{wi}(X) = Q_{wi}(X)/Z(X)$.

If the prover is honest, then $\pi_{wi}(X) = \hat{r}\hat{Z}(X)/Z(X) - r + \sum m_i \cdot ((\hat{P}_i(X) - y_i(X))/Z(X))$. Note that we do not need that $Q_{wi}(X) = 0$ as a polynomial, we just need that $Q_{wi}(\omega_i) = 0$, which is a deviation from the strategy usually used in QAP/QSP-based arguments [11].

We guarantee the conditions similarly to Sect. 4.3. The description of the argument follows. (Since it is derived as in Sect. 4.3, we omit further explanations.)

System parameters: Let $n = \mathrm{poly}(\kappa)$. Let com be the polynomial commitment scheme and let $\mathcal{F}_{\mathsf{com}} = (Z(X), (y_i)_{i=1}^{n})$ and $\hat{\mathcal{F}}_{\mathsf{com}} = (\hat{Z}(X), (\hat{P}_i)_{i=1}^{n})$, where $\hat{P}_i(X)$ is such that $y_i(\omega_j) = \hat{P}_i(\omega_j)$ for $i \in [0 \mathinner{..} n+1]$ and $j \in [1 \mathinner{..} n+1]$.
Setup $\mathsf{setup}_{sm}(1^\kappa, n)$**:** Let $\mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n)$.
CRS generation $\mathsf{gencrs}_{sm}(\mathsf{gk})$**:** Let $(g_1, g_2, \chi, \beta, \gamma, \hat{\gamma}) \leftarrow_r \mathbb{G}_1^* \times \mathbb{G}_2^* \times \mathbb{Z}_p^4$ with $Z(\chi) \neq 0$. Set $\mathsf{ck} \leftarrow (g_1, g_2^{\gamma})^{\mathcal{F}_{\mathsf{com}}(\chi)}$ and $\widehat{\mathsf{ck}} \leftarrow (g_1, g_2^{\hat{\gamma}})^{\hat{\mathcal{F}}_{\mathsf{com}}(\chi)}$. Let $\mathsf{crs}_{sm,p} \leftarrow (\mathsf{ck}, \widehat{\mathsf{ck}}, g_1^{\beta \cdot \mathcal{F}_{\mathsf{com}}(\chi)}, g_1^{\hat{Z}(\chi)/Z(\chi)}, g_1, (g_1^{(\hat{P}_i(\chi)-y_i(\chi))/$ and $\mathsf{crs}_{sm,v} \leftarrow (g_1, g_2^{\gamma}, g_2^{\hat{\gamma}}, g_2^{\gamma\beta}, g_2^{\gamma Z(\chi)})$. Return $\mathsf{crs}_{sm} = (\mathsf{crs}_{sm,p}, \mathsf{crs}_{sm,v})$.
Common input: $(A_1, A_2^{\gamma}) = \mathsf{com}(\mathsf{ck}; \boldsymbol{m}; r), (\hat{A}_1, \hat{A}_2^{\hat{\gamma}}) = \mathsf{com}(\widehat{\mathsf{ck}}; \boldsymbol{m}; \hat{r})$.
Argument generation $\mathsf{pro}_{sm}(\mathsf{gk}, \mathsf{crs}_{sm,p}; A_1, A_2^{\gamma}, \hat{A}_1, \hat{A}_2^{\hat{\gamma}}; \boldsymbol{m}, r, \hat{r})$**:** Set $\pi_{sm}^* \leftarrow g_1^{\pi_{wi}(\chi)} = (g_1^{\hat{Z}(\chi)/Z(\chi)})^{\hat{r}} \cdot g_1^{-r} \cdot \prod_{i=1}^{n} (g_1^{(\hat{P}_i(\chi)-y_i(\chi))/Z(\chi)})^{m_i}$. Set $A_1^{\beta} \leftarrow (g_1^{\beta Z(\chi)})^{r} \prod_{i=1}^{n} (g_1^{\beta y_i(\chi)})^{m_i}$. Output $\pi_{sm} = (\pi_{sm}^*, A_1^{\beta}) \in \mathbb{G}_1^2$.
Verification $\mathsf{ver}_{sm}(\mathsf{gk}, \mathsf{crs}_{sm,v}; (A_1, A_2^{\gamma}), (\hat{A}_1, \hat{A}_2^{\hat{\gamma}}); \pi_{sm})$**:**
Parse π_{sm} as $\pi_{sm} = (\pi_{sm}^*, A_1^{\beta})$. Verify that (1) $\hat{e}(g_1, A_2^{\gamma}) = \hat{e}(A_1, g_2^{\gamma})$,(2) $\hat{e}(A_1, g_2^{\gamma\beta}) = \hat{e}(A_1^{\beta}, g_2^{\gamma})$,(3) $\hat{e}(g_1, \hat{A}_2^{\hat{\gamma}}) = \hat{e}(\hat{A}_1, g_2^{\hat{\gamma}})$, and(4) $\hat{e}(\pi_{sm}^*, g_2^{\gamma Z(\chi)}) = \hat{e}(\hat{A}_1/A_1, g_2^{\gamma})$.

Let $\hat{Y}$ be the formal variable corresponding to $\hat{\gamma}$. In the following theorem, it suffices to take $\mathsf{crs}^* = (g_1^{\mathcal{F}_{sm,1}(\chi,\beta)}, g_2^{\mathcal{F}_{sm,2}(\chi,\beta,\gamma,\hat{\gamma})})$, where $\mathcal{F}_{sm,1} = \{1\} \cup \mathcal{F}_{\mathsf{com}} \cup \hat{\mathcal{F}}_{\mathsf{com}} \cup X_\beta \mathcal{F}_{\mathsf{com}} \cup \{\hat{Z}(X)/Z(X)\} \cup \{(\hat{P}_i(X) - y_i(X))/Z(X)\}_{i=1}^{n}$ and $\mathcal{F}_{sm,2} = Y \cdot (\{1, X_\beta\} \cup \mathcal{F}_{\mathsf{com}}) \cup \hat{Y} \cdot (\{1\} \cup \hat{\mathcal{F}}_{\mathsf{com}})$.

Theorem 3. *The same-message argument is perfectly complete and witness-indistinguishable. Let $\hat{n}$ be as above. If* BP *is $(\hat{n}, \hat{n}+n+2)$-PCDH secure, $\hat{n}$-TSDH secure, $(n+1, \mathcal{F}_{sm,1} \setminus (\{1\} \cup \mathcal{F}_{\mathsf{com}}), \mathcal{F}_{sm,2} \setminus Y \cdot (\{1\} \cup \mathcal{F}_{\mathsf{com}}), \gamma)$-PKE secure, and $(\hat{\mathcal{F}}_{\mathsf{com}}, \mathcal{F}_{sm,1} \setminus \hat{\mathcal{F}}_{\mathsf{com}}, \mathcal{F}_{sm,2} \setminus \hat{Y}\hat{\mathcal{F}}_{\mathsf{com}}, \hat{\gamma})$-PKE secure, then this argument is an adaptive argument of knowledge.*

The proof of the following proposition is straightforward and thus omitted.

Proposition 2. *The prover's computation is dominated by one $(W+2)$-wide and one $(W+1)$-wide multi-exponentiation in $\mathbb{G}_1$, where $0 \le W \le n$ is the number of elements in the vector $\boldsymbol{m}$ that are not in $\{0, 1\}$. The verifier's computation is dominated by 8 pairings.*

In the shuffle argument below, the prover uses $r = \hat{r}$, so prover's computation is $2W + 2$ exponentiations. For a unit vector $\boldsymbol{m}$, we additionally have $W = 0$ and computing A_1^{β} and the first two verification steps are already done in the unit vector argument anyway, so the argument only adds 1 exponentiation for the prover, and 4 pairings for the verifier.

4.5 New Assumption: PSP

We will next describe a new computational assumption (PSP) that is needed in the shuffle argument. The PSP assumption is related to but not equal to the SP assumption from [15]. Interestingly, the generic group proof of the PSP assumption relies on the Schwartz-Zippel lemma, while in most of the known interactive shuffle arguments (like [20]), the Schwartz-Zippel lemma is used in the reduction from the shuffle security to some underlying assumption.

Let let $d(n) > n$ be a function. Let $\hat{\mathcal{F}} = (\hat{P}_i(X))_{i=0}^{n}$ be a tuple of polynomials. We say $(d(n), \hat{\mathcal{F}})$ is *PSP-friendly*, if the following set is linearly independent: $\hat{\mathcal{F}}_{d(n)} := \{X^i\}_{i=0}^{2d(n)} \cup \{X^i \cdot \hat{P}_j(X)\}_{0 \le i \le d(n), 0 \le j \le n} \cup \{\hat{P}_0(X)\hat{P}_j(X)\}_{j=0}^{n}$.

Let $(d(n), \hat{\mathcal{F}})$ be PSP-friendly. Let $\mathcal{F} = (P_i(X))_{i=0}^{n}$ be a tuple of polynomials of degree $\le d(n)$. The $(\mathcal{F}, \hat{\mathcal{F}})$-*Power Simultaneous Product (PSP) assumption* states that for any $n =$

poly(κ) and any NUPPT adversary A,

$$\Pr\left[\begin{array}{l}\mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n), (g_1, g_2, \chi) \leftarrow_r \mathbb{G}_1^* \times \mathbb{G}_2^* \times \mathbb{Z}_p, \\ \mathbb{G}_1^{n+2} \ni (t, \hat{t}, (s_i)_{i=1}^n) \leftarrow \mathsf{A}(\mathsf{gk}; ((g_1, g_2)^{\chi^i})_{i=0}^{d(n)}, (g_1, g_2)^{\hat{\mathcal{F}}(\chi)}) : \\ t^{P_0(\chi)} \cdot \prod_{i=1}^n s_i^{P_i(\chi)} = \hat{t}^{\hat{P}_0(\chi)} \cdot \prod_{i=1}^n s_i^{\hat{P}_i(\chi)} = 1 \wedge (\exists i \in [1 \,..\, n] : s_i \neq 1)\end{array}\right] \approx_\kappa 0 \enspace .$$

In this section, we prove that the PSP assumption holds in the generic bilinear group model. PSP-friendliness and the PSP assumption are defined so that both the generic model proof and the reduction from the shuffle soundness to the PSP in Thm. 5 would go through. As in the case of SP, it is essential that two simultaneous products have to hold true; the simpler version of the PSP assumption with only one product (i.e., $t^{P_0(\chi)} \cdot \prod_{i=1}^n s_i^{P_i(\chi)} = 1$) does not hold in the generic bilinear group model. Differently from SP, the PSP assumption incorporates possibly distinct t and $\hat{t}$ since the same-message argument does not guarantee that the randomizers of two commitments are equal.

Generic Security of the PSP Assumption.

We will briefly discuss the security of the PSP assumption in the generic bilinear group model. Similarly to [15], we start by picking a random asymmetric bilinear group $\mathsf{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathsf{BP}(1^\kappa)$. We now give a generic bilinear group model proof for the PSP assumption.

Theorem 4. *Let $\mathcal{F} = (P_i(X))_{i=0}^n$ be linearly independent with $1 \notin \operatorname{span}(\mathcal{F})$. Let $d = \max\{\deg P_i(X)\}$ and let $\hat{\mathcal{F}} = (\hat{P}_i(X))_{i=0}^n$ be such that $(d, \hat{\mathcal{F}})$ is PSP-friendly. The $(\mathcal{F}, \hat{\mathcal{F}})$-PSP assumption holds in the generic bilinear group model.*

Proof. Assume there exists a successful adversary A. In the generic bilinear group model, A acts obliviously to the actual representation of the group elements and only performs generic bilinear group operations such as multiplying elements in $\mathbb{G}_i$ for $i \in \{1, 2, T\}$, pairing elements in $\mathbb{G}_1$ and $\mathbb{G}_2$, and comparing elements to see if they are identical. hence it can only produce new elements in $\mathbb{G}_1$ by multiplying existing group elements together.

Recall that the A's input is gk and $\mathsf{crs} = (((g_1, g_2)^{\chi^i})_{i=0}^d, (g_1, g_2)^{\hat{\mathcal{F}}(\chi)})$. Hence, keeping track of the group elements we get that A outputs $t, \hat{t}, s_i \in \mathbb{G}_1$, where $\log_{g_1} t = \sum_{j=0}^d t_j \chi^j + \sum_{j=0}^n t'_j \hat{P}_j(\chi)$, $\log_{g_1} \hat{t} = \sum_{j=0}^d \hat{t}_j \chi^j + \sum_{j=0}^n \hat{t}'_j \hat{P}_j(\chi)$, and $\log_{g_1} s_i = \sum_{j=0}^d s_{ij} \chi^j + \sum_{j=0}^n s'_{ij} \hat{P}_j(\chi)$, for *known* constants t_j, t'_j, $\hat{t}_j$, $\hat{t}'_j$, s_{ij}, s'_{ij}. Taking discrete logarithms of the PSP condition $t^{P_0(\chi)} \cdot \prod_{i=1}^n s_i^{P_i(\chi)} = \hat{t}^{\hat{P}_0(\chi)} \cdot \prod_{i=1}^n s_i^{\hat{P}_i(\chi)} = 1$, we get that the two polynomials (for *known* coefficients)

$$d_1(X) := \left(\sum_{j=0}^d t_j X^j + \sum_{j=0}^n t'_j \hat{P}_j(X)\right) \cdot P_0(X) + \sum_{i=1}^n \left(\sum_{j=0}^d s_{ij} X^j + \sum_{j=0}^n s'_{ij} \hat{P}_j(X)\right) P_i(X) \enspace ,$$
$$d_2(X) := \left(\sum_{j=0}^d \hat{t}_j X^j + \sum_{j=0}^n \hat{t}'_j \hat{P}_j(X)\right) \cdot \hat{P}_0(X) + \sum_{i=1}^n \left(\sum_{j=0}^d s_{ij} X^j + \sum_{j=0}^n s'_{ij} \hat{P}_j(X)\right) \hat{P}_i(X)$$

satisfy $d_1(\chi) = d_2(\chi) = 0$. Since the adversary is oblivious to the actual representation of the group elements it will do the same group operations no matter the actual value of $X (= \chi)$; so the values t_j, ..., s'_{ij} are generated (almost[2]) independently of χ. By the Schwartz-Zippel lemma there is a negligible probability that $d_i(\chi) = 0$, for non-zero $d_i(X)$, when we choose χ randomly. Thus, with all but a negligible probability $d_1(X)$ and $d_2(X)$ are zero polynomials.

[2]A generic bilinear group adversary may learn a negligible amount of information about χ by comparing group elements; we skip this part in the proof.

Since $\mathcal{F}$ and $\{X^i\}_{i=0}^{2d} \cup \{X^i \cdot \hat{P}_j(X)\}_{i\in[0\,..\,d],j\in[0\,..\,n]}$ are both linearly independent, $\{X^i\}_{i=0}^{2d} \cup \{P_i(X)\hat{P}_j(X)\}_{i,j\in[0\,..\,n]}$ is also linearly independent. We get from $d_1(X) = 0$ that $\sum_{j=0}^{n} t'_j P_0(X)\hat{P}_j(X) + \sum_{i=1}^{n}\sum_{j=0}^{n} s'_{ij}P_i(X)\hat{P}_j(X) = 0$, which implies $s'_{ij} = 0$ for $i \in [1\,..\,n], j \in [0\,..\,n]$. Substituting these values into $d_2(X) = 0$, we get that $\left(\sum_{j=0}^{d} \hat{t}_j X^j + \sum_{j=0}^{n} \hat{t}'_j \hat{P}_j(X)\right)\hat{P}_0(X) + \sum_{i=1}^{n}\sum_{j=0}^{d} s_{ij}X^j\hat{P}_i(X) = 0$. Since $\hat{\mathcal{F}}_d$ is linearly independent, we get that all coefficients in the above equation are zero, and in particular $s_{ij} = 0$ for $i \in [1\,..\,n], j \in [0\,..\,n]$. Thus $s_i = 1$ for $i \in [1\,..\,n]$. Contradiction to the fact that the adversary is successful. □ □

4.6 New Shuffle Argument

Let Elgamal operate in $\mathbb{G}_1$ defined by gk. In a shuffle argument, the prover aims to convince the verifier that, given the description of a group, a public key, and two vectors of ciphertexts, the second vector of the ciphertexts is a permutation of rerandomized versions of the ciphertexts from the first vector. However, to achieve better efficiency, we construct a shuffle argument that is only culpably sound with respect to the next relation (i.e., $\mathcal{R}_{sh}^{\mathsf{guilt}}$-sound):

$$\mathcal{R}_{sh,n}^{\mathsf{guilt}} = \left\{ \begin{array}{l} (\mathsf{gk}, (\mathsf{pk}, (z_i)_{i=1}^n, (z'_i)_{i=1}^n), \mathsf{sk}) : \mathsf{gk} \in \mathsf{BP}(1^\kappa, n) \wedge \\ (\mathsf{pk}, \mathsf{sk}) \in \mathsf{genpkc}(\mathsf{gk}) \wedge (\forall \psi \in S_n : \exists i : \mathsf{dec}_{\mathsf{sk}}(z'_i) \neq \mathsf{dec}_{\mathsf{sk}}(z_{\psi(i)})) \end{array} \right\} .$$

The argument of [15] is proven to be $\mathcal{R}_{sh}^{\mathsf{guilt}}$-sound with respect to the same relation. See [15] or the introduction for an explanation why $\mathcal{R}_{sh}^{\mathsf{guilt}}$ is sufficient.

As noted in the introduction, we need to use same-message arguments and rely on the PSP assumption. Thus, we need polynomials $\hat{P}_j$ that satisfy two different requirements at once. First, to be able to use the same-message argument, we need that $y_j(\omega_k) = \hat{P}_j(\omega_k)$ for $k \in [1\,..\,n+1]$. Second, to be able to use the PSP assumption, we need $(d, \hat{\mathcal{F}})$ to be PSP-friendly, and for this we need $\hat{P}_j(X)$ to have a sufficiently large degree. Recall that y_j are fixed by the unit vector argument. We now show that such a choice for $\hat{P}_j$ exists.

Proposition 3. *Let $\hat{y}_j(X) := (XZ(X)+1)^{j-1}(X^2Z(X)+1)y_j(X)$ for $j \in [1\,..\,n]$, and $\hat{Z}(X) = \hat{y}_0(X) := (XZ(X)+1)^{n+1}Z(X)$. Let $\hat{\mathcal{F}}_{\mathsf{com}} = (\hat{y}_j(X))_{j=0}^n$. Then $\hat{y}_j(\omega_k) = y_j(\omega_k)$ for all j, k, and $(n+1, \hat{\mathcal{F}}_{\mathsf{com}})$ is PSP-friendly.*

Next, we will provide the full description of the new shuffle argument. Note that $(c_i)_{i=1}^n$ are commitments to the rows of the permutation matrix $\mathbf{\Psi}$, proven by the n unit vector arguments $(\pi_{uv,i})_{i=1}^n$ and by the implicit computation of c_n. We denote $\hat{E}((a,b),c) := (\hat{e}(a,c), \hat{e}(b,c))$.

System parameters: Let $(\mathsf{genpkc}, \mathsf{enc}, \mathsf{dec})$ be the Elgamal cryptosystem. Let com be the polynomial commitment scheme. Consider polynomials $\mathcal{F}_{\mathsf{com}} = \{Z(X)\} \cup (y_i(X))_{i=1}^n$ from Sect. 4.3. Let $\hat{\mathcal{F}}_{\mathsf{com}} = (\hat{y}_i(X))_{i=0}^n$ be as in Prop. 3.

Setup $\mathsf{setup}_{sh}(1^\kappa, n)$**:** Let $\mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n)$.

CRS generation $\mathsf{gencrs}_{sh}(\mathsf{gk})$**:** Let $(g_1, g_2, \chi, \beta, \gamma) \leftarrow_r \mathbb{G}_1^* \times \mathbb{G}_2^* \times \mathbb{Z}_p^3$ with $Z(\chi) \neq 0$. Let $(\mathsf{crs}_{uv,p}, \mathsf{crs}_{uv,v}) \leftarrow_r \mathsf{gencrs}_{uv}(\mathsf{gk}, n)$, $(\mathsf{crs}_{sm,p}, \mathsf{crs}_{sm,v}) \leftarrow_r \mathsf{gencrs}_{sm}(\mathsf{gk}, n)$, but by using the same $(g_1, g_2, \chi, \beta, \gamma)$ in both cases. Let $\mathsf{ck} \leftarrow (g_1, g_2^\gamma)^{\mathcal{F}_{\mathsf{com}}(\chi)}$ and $\widehat{\mathsf{ck}} \leftarrow (g_1, g_2^{\hat{\gamma}})^{\hat{\mathcal{F}}_{\mathsf{com}}(\chi)}$. Set $(D_1, D_2^\gamma) \leftarrow \mathsf{com}(\mathsf{ck}; \mathbf{1_n}; 0)$, $(\hat{D}_1, \hat{D}_2^{\hat{\gamma}}) \leftarrow \mathsf{com}(\widehat{\mathsf{ck}}; \mathbf{1_n}; 0)$. Set $\mathsf{crs}_{sh,p} \leftarrow (\mathsf{crs}_{uv,p}, \widehat{\mathsf{ck}}, g_1^{\hat{Z}(\chi)/Z(\chi)}, g_1, (g_1^{(\hat{y}_i(\chi)-y_i(}$ $\mathsf{crs}_{sh,v} \leftarrow (\mathsf{crs}_{uv,v}, g_2^{\hat{\gamma}}, \{g_2^{\gamma y_i(\chi)}, g_2^{\hat{\gamma}\hat{y}_i(\chi)}\}_{i=0}^n, D_1, D_2^\gamma, \hat{D}_1, \hat{D}_2^{\hat{\gamma}})$, and $\mathsf{td}_{sh} \leftarrow \chi$. Return $((\mathsf{crs}_{sh,p}, \mathsf{crs}_{sh,v}), \mathsf{td}_{sh})$.

Common input: $(\mathsf{pk}, (z_i, z'_i)_{i=1}^n)$, where $\mathsf{pk} = (g_1, h) \in \mathbb{G}_1^2$, $z_i \in \mathbb{G}_1^2$ and $z'_i = z_{\psi(i)} \cdot \mathsf{enc}_{\mathsf{pk}}(1; t_i) \in \mathbb{G}_1^2$.

Argument $\mathsf{pro}_{sh}(\mathsf{gk}, \mathsf{crs}_{sh,p}; \mathsf{pk}, (z_i, z'_i)_{i=1}^n; \psi, (t_i)_{i=1}^n)$**:**

(1) Let $\mathbf{\Psi} = \mathbf{\Psi}_{\psi^{-1}}$ be the $n \times n$ permutation matrix corresponding to ψ^{-1}.

(2) For $i \in [1\,..\,n-1]$:

- Set $r_i \leftarrow \mathbb{Z}_p$, $(c_{i1}, c_{i2}^\gamma) \leftarrow \mathsf{com}(\mathsf{ck}; \mathbf{\Psi}_i; r_i)$, $(\hat{c}_{i1}, \hat{c}_{i2}^{\hat{\gamma}}) \leftarrow \mathsf{com}(\widehat{\mathsf{ck}}; \mathbf{\Psi}_i; r_i)$.

```
    (3) Set $r_n \leftarrow -\sum_{i=1}^{n-1} r_i$, $(c_{n1}, c_{n2}^{\gamma}) \leftarrow (D_1, D_2^{\gamma}) / \prod_{i=1}^{n-1} (c_{i1}, c_{i2}^{\gamma})$.
    (4) Set $(\hat{c}_{n1}, \hat{c}_{n2}^{\hat{\gamma}}) \leftarrow (\hat{D}_1, \hat{D}_2^{\hat{\gamma}}) / \prod_{i=1}^{n-1} (\hat{c}_{i1}, \hat{c}_{i2}^{\hat{\gamma}})$.
    (5) For $i \in [1 \,..\, n]$: set $\pi_{uv,i} = (\pi^*_{uv,i}, c_{i1}^{\beta}) \leftarrow \mathsf{pro}_{uv}(\mathsf{gk}, \mathsf{crs}_{uv,p}; c_{i1}, c_{i2}^{\gamma}; \mathbf{\Psi}_i, r_i)$.
    (6) Set $r_t \leftarrow_r \mathbb{Z}_p$, $(d_1, d_2^{\gamma}) \leftarrow \mathsf{com}(\mathsf{ck}; \boldsymbol{t}; r_t)$, and $(\hat{d}_1, \hat{d}_2^{\hat{\gamma}}) \leftarrow \mathsf{com}(\widehat{\mathsf{ck}}; \boldsymbol{t}; r_t)$.
    (7) For $i \in [1 \,..\, n-1]$:
          • Set $(\pi^*_{sm,i}, c_{i1}^{\beta}) \leftarrow \mathsf{pro}_{sm}(\mathsf{gk}, \mathsf{crs}_{sm,p}; c_{i1}, c_{i2}^{\gamma}, \hat{c}_{i1}, \hat{c}_{i2}^{\hat{\gamma}}; \mathbf{\Psi}_i, r_i, r_i)$.
    (8) Set $\pi_{sm,d} \leftarrow \mathsf{pro}_{sm}(\mathsf{gk}, \mathsf{crs}_{sm,p}; d_1, d_2^{\gamma}, \hat{d}_1, \hat{d}_2^{\hat{\gamma}}; \boldsymbol{t}, r_t, r_t)$.
    (9) Compute $U = (U_1, U_2) \leftarrow \mathsf{pk}^{r_t} \cdot \prod_{i=1}^{n} z_i^{r_i} \in \mathbb{G}_1^2$. // The only online step
    (10) Output $\pi_{sh} \leftarrow ((c_{i1}, c_{i2}^{\gamma}, \hat{c}_{i1}, \hat{c}_{i2}^{\hat{\gamma}})_{i=1}^{n-1}, d_1, d_2^{\gamma}, \hat{d}_1, \hat{d}_2^{\hat{\gamma}}, (\pi_{uv,i})_{i=1}^{n}, (\pi^*_{sm,i})_{i=1}^{n-1}, \pi_{sm,d}, U)$
**Verification** $\mathsf{ver}_{sh}(\mathsf{gk}, \mathsf{crs}_{sh,v}; \mathsf{pk}, (z_i, z'_i)_{i=1}^{n}, \pi_{sh})$:
    (1) Let $(c_{n1}, c_{n2}^{\gamma}) \leftarrow (D_1, D_2^{\gamma}) / \prod_{i=1}^{n-1} (c_{i1}, c_{i2}^{\gamma})$.
    (2) Let $(\hat{c}_{n1}, \hat{c}_{n2}^{\hat{\gamma}}) \leftarrow (\hat{D}_1, \hat{D}_2^{\hat{\gamma}}) / \prod_{i=1}^{n-1} (\hat{c}_{i1}, \hat{c}_{i2}^{\hat{\gamma}})$.
    (3) For $i \in [1 \,..\, n]$: reject if $\mathsf{ver}_{uv}(\mathsf{gk}, \mathsf{crs}_{uv,v}; c_{i1}, c_{i2}^{\gamma}; \pi_{uv,i})$ rejects.
    (4) For $i \in [1 \,..\, n-1]$: reject if $\mathsf{ver}_{sm}(\mathsf{gk}; \mathsf{crs}_{sm,v}; c_{i1}, c_{i2}^{\gamma}, \hat{c}_{i1}, \hat{c}_{i2}^{\hat{\gamma}}; \pi_{sm,i})$ rejects.
    (5) Reject if $\mathsf{ver}_{sm}(\mathsf{gk}, \mathsf{crs}_{sm,v}; d_1, d_2^{\gamma}, \hat{d}_1, \hat{d}_2^{\hat{\gamma}}; \pi_{sm,d})$ rejects.
    (6) Check the PSP-related verification equations: // The only online step
        (a) $\prod_{i=1}^{n} \hat{E}(z'_i, g_2^{\gamma y_i(\chi)}) / \prod_{i=1}^{n} \hat{E}(z_i, c_{i2}^{\gamma}) = \hat{E}((g_1, h), d_2^{\gamma}) / \hat{E}(U, g_2^{\gamma Z(\chi)})$,
        (b) $\prod_{i=1}^{n} \hat{E}(z'_i, g_2^{\hat{\gamma} \hat{y}_i(\chi)}) / \prod_{i=1}^{n} \hat{E}(z_i, \hat{c}_{i2}^{\hat{\gamma}}) = \hat{E}((g_1, h), \hat{d}_2^{\hat{\gamma}}) / \hat{E}(U, g_2^{\hat{\gamma} \hat{Z}(\chi)})$.
```

Since $\mathsf{ck}, \widehat{\mathsf{ck}} \subset \mathsf{crs}_{sh,p}$, $(D_1, D_2^{\gamma}) = \mathsf{com}(\mathsf{ck}; \mathbf{1}_n; 0)$ and $(\hat{D}_1, \hat{D}_2^{\hat{\gamma}}) = \mathsf{com}(\widehat{\mathsf{ck}}; \mathbf{1}_n; 0)$ can be computed from the rest of the CRS. (These four elements are only needed to optimize the computation of $(c_{n1}, c_{n2}^{\gamma})$ and $(\hat{c}_{n1}, \hat{c}_{n2}^{\hat{\gamma}})$.) For security, it suffices to take $\mathsf{crs}^*_{sh} = (g_1^{\mathcal{F}_{sh,1}(\chi,\beta)}, g_2^{\mathcal{F}_{sh,2}(\chi,\beta,\gamma,\hat{\gamma})})$, where $\mathcal{F}_{sh,1} = \mathcal{F}_{uv,1} \cup \hat{\mathcal{F}}_{\mathsf{com}} \cup \{\hat{Z}(X)/Z(X)\} \cup \{(\hat{y}_i(X) - y_i(X))/Z(X)\}_{i=1}^{n}$ and $\mathcal{F}_{sh,2} = \mathcal{F}_{uv,2} \cup \hat{Y} \cdot (\{1\} \cup \hat{\mathcal{F}}_{\mathsf{com}})$.

Theorem 5. *The new shuffle argument is a non-interactive perfectly complete and perfectly zero-knowledge shuffle argument for Elgamal ciphertexts. If the $(n+1)$-TSDH, $(\hat{n}, \hat{n}+n+2)$-PCDH, $(\mathcal{F}_{\mathsf{com}}, \hat{\mathcal{F}}_{\mathsf{com}})$-PSP, $(n+1, \mathcal{F}_{sh,1} \setminus (\{1\} \cup \mathcal{F}_{\mathsf{com}}), \mathcal{F}_{sh,2} \setminus Y \cdot (\{1\} \cup \mathcal{F}_{\mathsf{com}}), \gamma)$-PKE, $(\hat{\mathcal{F}}_{\mathsf{com}}, \mathcal{F}_{sh,1} \setminus \hat{\mathcal{F}}_{\mathsf{com}}, \mathcal{F}_{sh,2} \setminus \hat{Y} \hat{\mathcal{F}}_{\mathsf{com}}, \hat{\gamma})$-PKE assumptions hold, then the shuffle argument is adaptively computationally culpably sound w.r.t. the language $\mathcal{R}^{\mathsf{guilt}}_{sh,n}$ and an argument of knowledge.*

When using a Barreto-Naehrig curve [?], exponentiations in $\mathbb{G}_1$ are three times cheaper than in $\mathbb{G}_2$. Moreover, a single $(N+1)$-wide multi-exponentiations is considerably cheaper than $N+1$ exponentiations. Hence, we compute separately the number of exponentiations and multi-exponentiations in both $\mathbb{G}_1$ and $\mathbb{G}_2$ [?, ?]. For the sake of the simplicity, Prop. 4 only summarizes those numbers.

Proposition 4. *The prover's CRS consists of $6n+7$ elements of $\mathbb{G}_1$ and $2n+4$ elements of $\mathbb{G}_2$. The verifier's CRS consists of 4 elements of $\mathbb{G}_1$, $2n+8$ elements of $\mathbb{G}_2$, and 1 element of $\mathbb{G}_T$. The total CRS is $6n+8$ elements of $\mathbb{G}_1$, $2n+8$ elements of $\mathbb{G}_2$, and 1 element of $\mathbb{G}_T$, in total $8n+17$ group elements. The communication complexity is $5n+2$ elements of $\mathbb{G}_1$ and $2n$ elements of $\mathbb{G}_2$, in total $7n+2$ group elements. The prover's and the verifier's computational complexity are as in Tbl. 4.1.*

Importantly, both the proving and verification algorithm of the new shuffle argument can be divided into offline (independent of the common input $(\mathsf{pk}, (z_i, z'_i)_{i=1}^{n})$) and online (dependent on the common input) parts. The prover can precompute all elements of π_{sh} except U (i.e., execute all steps of the proving algorithm, except step (9)), and send them to the verifier before the inputs are fixed. The verifier can verify $\pi_{sh} \setminus \{U\}$ (i.e., execute all steps of the verification algorithm, except step (6)) in the precomputation step. Thus, the online computational complexity is dominated by two $(n+1)$-wide multi-exponentiations for the prover, and $8n+4$

pairings for the verifier (note that $\hat{E}((g_1, h), d_2^{\gamma})$ and $\hat{E}((g_1, h), \hat{d}_2^{\gamma})$ can also be precomputed by the verifier).

Low online complexity is highly important in e-voting, where the online time (i.e., the time interval after the ballots are gathered and before the election results are announced) can be limited for legal reasons. In this case, the mix servers can execute all but step (9) of the proving algorithm and step (6) of the verification algorithm before the votes are even cast, assuming one is able to set a priori a reasonable upper bound on n, the number of votes. See [24] for additional motivation.

Bibliography

[1] Abe, M., Fehr, S.: Perfect NIZK with Adaptive Soundness. In: TCC 2007. LNCS, vol. 4392, pp. 118–136

[2] Blum, M., Feldman, P., Micali, S.: Non-Interactive Zero-Knowledge and Its Applications. In: STOC 1988, pp. 103–112

[3] Boneh, D., Boyen, X.: Secure Identity Based Encryption Without Random Oracles. In: CRYPTO 2004. LNCS, vol. 3152, pp. 443–459

[4] Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: CRYPTO 2004. LNCS, vol. 3152, pp. 41–55

[5] Damgård, I.: Towards Practical Public Key Systems Secure against Chosen Ciphertext Attacks. In: CRYPTO 1991. LNCS, vol. 576, pp. 445–456

[6] Danezis, G., Fournet, C., Groth, J., Kohlweiss, M.: Square Span Programs with Applications to Succinct NIZK Arguments. In: ASIACRYPT 2014 (1). LNCS, vol. 8873, pp. 532–550

[7] Elgamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Trans. on Inf. Theory **31**(4) (1985) pp. 469–472

[8] Fauzi, P., Lipmaa, H.: Efficient Culpably Sound NIZK Shuffle Argument without Random Oracles. Technical Report 2015/1112, IACR (2015) http://eprint.iacr.org/2015/1112.

[9] Furukawa, J.: Efficient and Verifiable Shuffling and Shuffle-Decryption. IEICE Transactions **88-A**(1) (2005) pp. 172–188

[10] Furukawa, J., Sako, K.: An Efficient Scheme for Proving a Shuffle. In: CRYPTO 2001. LNCS, vol. 2139, pp. 368–387

[11] Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic Span Programs and NIZKs without PCPs. In: EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645

[12] Golle, P., Jarecki, S., Mironov, I.: Cryptographic Primitives Enforcing Communication and Storage Complexity. In: FC 2002. LNCS, vol. 2357, pp. 120–135

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

[14] Groth, J.: Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In: ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340

[15] Groth, J., Lu, S.: A Non-interactive Shuffle with Pairing Based Verifiability. In: ASIACRYPT 2007. LNCS, vol. 4833, pp. 51–67

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

[17] Lipmaa, H.: Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments. In: TCC 2012. LNCS, vol. 7194, pp. 169–189

[18] Lipmaa, H., Zhang, B.: A More Efficient Computationally Sound Non-Interactive Zero-Knowledge Shuffle Argument. In: SCN 2012. LNCS, vol. 7485, pp. 477–502

[19] Lipmaa, H., Zhang, B.: A More Efficient Computationally Sound Non-Interactive Zero-Knowledge Shuffle Argument. Journal of Computer Security **21**(5) (2013) pp. 685–719

[20] Neff, C.A.: A Verifiable Secret Shuffle and Its Application to E-Voting. In: ACM CCS 2001, pp. 116–125

[21] Parno, B., Gentry, C., Howell, J., Raykova, M.: Pinocchio: Nearly Practical Verifiable Computation. In: IEEE SP 2013, pp. 238–252

[22] Sako, K., Kilian, J.: Receipt-Free Mix-Type Voting Scheme - A Practical Solution to the Implementation of a Voting Booth. In: EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403

[23] Terelius, B., Wikström, D.: Proofs of Restricted Shuffles. In: AFRICACRYPT 2010. LNCS, vol. 6055, pp. 100–113

[24] Wikström, D.: A Commitment-Consistent Proof of a Shuffle. In: ACISP 2009. LNCS, vol. 5594, pp. 4007–421

5. Initial design options for mix-nets: Prover-Efficient Commit-And-Prove Zero-Knowledge SNARKs

Zk-SNARKs (succinct non-interactive zero-knowledge arguments of knowledge) are needed in many applications, including e-voting applications. Unfortunately, all previous zk-SNARKs for interesting languages are either inefficient for the prover, or are non-adaptive and based on a commitment scheme that depends both on the prover's input and on the language, i.e., they are not commit-and-prove (CaP) SNARKs. In this chapter, we propose a proof-friendly extractable commitment scheme, and use it to construct prover-efficient adaptive CaP succinct zk-SNARKs for different languages, that can all reuse committed data. The presented scheme introduces a design option of developing a shuffle argument in mix-nets based e-voting applications for WP4 of PANORAMIX project.

5.1 Introduction

Recently, there has been a significant surge of activity in studying succinct non-interactive zero knowledge (NIZK) arguments of knowledge (also known as zk-SNARKs) [4–7, 13, 14, 19, 21, 26, 27, 31]. The prover of a zk-SNARK outputs a short (ideally, a small number of group elements) argument π that is used to convince many different verifiers in the truth of the same claim without leaking any side information. The verifiers can verify independently the correctness of π, without communicating with the prover. The argument must be efficiently verifiable. Constructing the argument can be less efficient, since it is only done once. Still, prover-efficiency is important, e.g., in a situation where a single server has to create many arguments to different clients or other servers.

Many known zk-SNARKs are non-adaptive, meaning that the common reference string, CRS, can depend on the concrete instance of the language (e.g., the circuit in the case of CIRCUIT-SAT). In an adaptive zk-SNARK, the CRS is independent on the instance and thus can be reused many times. This distinction is important, since generation and distribution of the CRS must be done securely. The most efficient known *non-adaptive* zk-SNARKs for NP-complete languages from [19] are based on either Quadratic Arithmetic Programs (QAP, for arithmetic CIRCUIT-SAT) or Quadratic Span Programs (QSP, for Boolean CIRCUIT-SAT). There, the prover computation is dominated by $\Theta(n)$ cryptographic operations (see the full version [29] for a clarification on cryptographic/non-cryptographic operations), where n is the number of the gates. QAP, QSP [19, 27] and other related approaches like SSP [14] have the same asymptotic complexity.

QSP-based CIRCUIT-SAT SNARK can be made adaptive by using universal circuits [36]. Then, the CRS depends on the construction of universal circuit and not on the concrete input circuit itself.

However, since the size of a universal circuit is $\Theta(n \log n)$, the prover computation in resulting adaptive zk-SNARKs is $\Theta(n \log^2 n)$ non-cryptographic operations and $\Theta(n \log n)$ cryptographic operations. (In the case of QAP-based arithmetic CIRCUIT-SAT SNARK, one has to use universal arithmetic circuits [33] that have an even larger size $\Theta(r^4 n)$, where r is the degree of the polynomial computed by the arithmetic circuit. Thus, we will mostly give a comparison to the QSP-based approach.)

Since Valiant's universal circuits incur a large constant $c = 19$ in the $\Theta(\cdot)$ expression, a common approach [24,34] is to use universal circuits with the overhead of $\Theta(\log^2 n)$ but with a smaller constant $c = 1/2$ in $\Theta(\cdot)$. The prover computation in the resulting adaptive zk-SNARKs is $\Theta(n \log^3 n)$ non-cryptographic operations and $\Theta(n \log^2 n)$ cryptographic operations.[1]

Another important drawback of the QSP/QAP-based SNARKs is that they use a circuit-dependent commitment scheme. To use the same input data in multiple sub-SNARKs, one needs to construct a single large circuit that implements all sub-SNARKs, making the SNARK and the resulting *new* commitment scheme more complicated.In particular, these SNARKs are not commit-and-prove (CaP [10, 23]) SNARKs. We recall that in CaP SNARKs, a commitment scheme C is fixed first, and the statement consists of commitments of the witness using C; see Sect. 5.2. Hence, a CaP commitment scheme is *instance-independent*. In addition, one would like the commitment scheme to be *language-independent*, enabling one to first commit to the data and only then to decide in what applications (e.g., verifiable computation of a later fixed function) to use it.

See Tbl. 5.1 for a brief comparison of the efficiency of proposed adaptive zk-SNARKs for NP-complete languages. SUBSET-SUM is here brought as an example of a wider family of languages; it can be replaced everywhere say with PARTITION or KNAPSACK, see the full version [29]. Here, $N = r_3^{-1}(n) = o(n2^{2\sqrt{2\log_2 n}})$, where $r_3(n)$ is the density of the largest progression-free set in $\{1, \ldots, n\}$. According to the current knowledge, $r_3^{-1}(n)$ is comparable to (or only slightly smaller than) n^2 for $n < 2^{12}$; this makes all known CaP SNARKs [16,21,26] arguably impractical unless n is really small. In all cases, the verifier's computation is dominated by either $\Theta(n)$ cryptographic or $\Theta(n \log n)$ non-cryptographic operations (with the verifier's online computation usually being $\Theta(1)$), and the communication consists of a small constant number of group elements.[2] Given all above, it is natural to ask the following question:

> The Main Question of This Paper: *Is it possible to construct* adaptive CaP *zk-SNARKs for NP-complete languages where the prover computation is dominated by a linear number of cryptographic operations?*

We answer the "main question" positively by improving on Groth's modular approach [21]. Using the modular approach allows us to modularize the security analysis, first proving the security of underlying building blocks (the product and the shift SNARKs), and then composing them to construct master SNARKs for even NP-complete languages. The security of master SNARKs follows easily from the security of the basic SNARKs. We also use batch verification to speed up verification of almost all known SNARKs.

All new SNARKs use the same commitment scheme, the interpolating commitment scheme. Hence, one can reuse their input data to construct CaP zk-SNARKs for different unrelated languages, chosen only after the commitment was done. Thus, one can first commit to some data,

[1]Recently, [13] proposed an independent methodology to improve the prover's computational complexity in QAP-based arguments. However, [13] does not spell out their achieved prover's computational complexity.

[2]We emphasize that CIRCUIT-SAT is *not* our focus; the lines corresponding to CIRCUIT-SAT are provided only for the sake of comparison. One can use proof boot-strapping [13] to decrease the length of the resulting CIRCUIT-SAT argument from $\Theta(\log n)$, as stated in [28], to $\Theta(1)$; we omit further discussion.

Table 5.1: Prover-efficiency of known *adaptive* zk-SNARKs for NP-complete languages. Here, n is the number of the gates (in the case of CIRCUIT-SAT) and the number of the integers (in the case of SUBSET-SUM). Green background denotes the best known asymptotic complexity of the *concrete* NP-complete language w.r.t. to the concrete parameter. The solutions marked with * use proof bootstrapping from [13]

Paper	Language	Prover computation		|CRS|
		non-crypt. op.	crypt. op.	
Not CaP-s				
QAP, QSP ([14,19,27])	CIRCUIT-SAT	$\Theta(n \log^2 n)$	$\Theta(n \log n)$	$\Theta(n)$
CaP-s				
Gro10 ([21])	CIRCUIT-SAT	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$
Lip12 ([26])	CIRCUIT-SAT	$\Theta(n^2)$	$\Theta(N)$	$\Theta(N)$
Lip14 + Lip12 ([26,28])*	CIRCUIT-SAT	$\Theta(N \log^2 n)$	$\Theta(N \log n)$	$\Theta(N \log n)$
Lip14 + current paper ([28])*	CIRCUIT-SAT	$\Theta(n \log^2 n)$	$\Theta(n \log n)$	$\Theta(n \log n)$
FLZ13 ([16])	SUBSET-SUM	$\Theta(N \log n)$	$\Theta(N)$	$\Theta(N)$
Current paper	SUBSET-SUM	$\Theta(n \log n)$	$\Theta(n)$	$\Theta(n)$

and only later decide in which application and to what end to use it. Importantly, by using CaP zk-SNARKs, one can guarantee that all such applications use exactly the same data.

The resulting SNARKs are not only commit-and-prove, but also very efficient, and often more efficient than any previously known SNARKs. The new CaP SNARKs have prover-computation dominated by $\Theta(n)$ cryptographic operations, with the constant in $\Theta(\cdot)$ being reasonably small. Importantly, we propose the most efficient known succinct range SNARK. Since the resulting zk-SNARKs are sufficiently different from QAP-based zk-SNARKs, we hope that our methodology by itself is of independent interest. Up to the current paper, Groth's modular approach has resulted in significantly less efficient zk-SNARKs than the QSP/QAP-based approach.

In Sect. 5.3, we construct a new natural extractable trapdoor commitment scheme (the interpolating commitment scheme). Here, commitment to $\vec{a} \in \mathbb{Z}_p^n$, where n is a power of 2, is a short garbled and randomized version $g_1^{L_{\vec{a}}(\chi)}(g_1^{\chi^n-1})^r$ of the Lagrange interpolating polynomial $L_{\vec{a}}(X)$ of $\vec{a}$, for a random secret key χ, together with a knowledge component. This commitment scheme is arguably a very natural one, and in particular its design is not influenced by the desire to tailor it to one concrete application. Nevertheless, as we will see, using it improves the efficiency of many constructions while allowing to reuse many existing results.

The new CaP zk-SNARKs are based on the interpolating commitment scheme and two CaP witness-indistinguishable SNARKs: a product SNARK (given commitments to vectors $\vec{a}$, $\vec{b}$, $\vec{c}$, it holds that $c_i = a_i b_i$; see [16, 21, 26]), and a shift SNARK (given commitments to $\vec{a}$, $\vec{b}$, it holds that $\vec{a}$ is a coordinate-wise shift of $\vec{b}$; see [16]). One can construct an adaptive CIRCUIT-SAT CaP zk-SNARK from $\Theta(\log n)$ product and shift SNARKs [21, 28], or adaptive CaP zk-SNARKs for NP-complete languages like SUBSET-SUM (and a similar CaP range SNARK) by using a constant number of product and shift SNARKs [16].

In Sect. 5.4, we propose a CaP product SNARK, that is an argument of knowledge under a computational and a knowledge (needed solely to achieve extractability of the commitment scheme) assumption. Its prover computation is dominated by $\Theta(n \log n)$ non-cryptographic and $\Theta(n)$ cryptographic operations. This can be compared to $r_3^{-1}(n)$ non-cryptographic operations in [16]. The speed-up is mainly due to the use of the interpolating commitment scheme.

In Sect. 5.5, we propose a variant of the CaP shift SNARK of [16], secure when combined with the interpolating commitment scheme. We prove that this SNARK is an adaptive argument of knowledge under a computational and a knowledge assumption. It only requires the prover to perform $\Theta(n)$ cryptographic and non-cryptographic operations.

Product and shift SNARKs are already very powerful by itself. E.g., a prover can commit to her input vector $\vec{a}$. Then, after agreeing with the verifier on a concrete application, she can commit to a different yet related input vector (that say consists of certain permuted subset of $\vec{a}$'s coefficients), and then use the basic SNARKs to prove that this was done correctly. Here, she may use the permutation SNARK [28] that consists of $O(\log n)$ product and shift SNARKs. Finally, she can use another, application-specific, SNARK (e.g., a range SNARK) to prove that the new committed input vector has been correctly formed.

In Sect. 5.6, we describe a modular adaptive CaP zk-SNARK, motivated by [16], for the NP-complete language, Subset-Sum. (Subset-Sum was chosen by us mainly due to the simplicity of the SNARK; the rest of the paper considers more applications.) This SNARK consists of three commitments, one application of the shift SNARK, and three applications of the product SNARK. It is a zk-SNARK given that the commitment scheme, the shift SNARK, and the product SNARK are secure. Its prover computation is strongly dominated by $\Theta(n)$ cryptographic operations, where n is the instance size, the number of integers. More precisely, the prover has to perform only nine ($\approx n$)-wide multi-exponentiations, which makes the SNARK efficient not only asymptotically (to compare, the size of Valiant's arithmetic circuit has constant 19, and this constant has to be multiplied by the overhead of non-adaptive QSP/QAP/SSP-based solutions). Thus, we answer positively to the stated main question of the current paper. Moreover, the prover computation is highly parallelizable, while the *online* verifier computation is dominated by 17 pairings (this number will be decreased later).

In Sect. 5.7, we propose a new CaP range zk-SNARK that the committed value belongs to a range $[L\,..\,H]$. This SNARK looks very similar to the Subset-Sum SNARK, but with the integer set $\vec{S}$ of the Subset-Sum language depending solely on the range length. Since here the prover has a committed input, the simulation of the range SNARK is slightly more complicated than of the Subset-Sum SNARK. Its prover-computation is similarly dominated by $\Theta(n)$ cryptographic operations, where this time $n := \lceil \log_2(H - L) \rceil$. Differently from the Subset-Sum SNARK, the verifier computation is dominated only by $\Theta(1)$ cryptographic operations, more precisely, by 19 pairings (also this number will be decreased later). Importantly, this SNARK is computationally more efficient than any of the existing *succinct* range SNARKs either in the standard model (i.e., random oracle-less) or in the random oracle model. E.g., the prover computation in [25] is $\Theta(n^2)$ under the Extended Riemann Hypothesis, and the prover computation in [16] is $\Theta(r^{-3}(n) \log r^{-3}(n))$. It is also significantly simpler than the range SNARKs of [12, 16], mostly since we do not have to consider different trade-offs between computation and communication.

In the full version [29], we outline how to use the new basic SNARKs to construct efficient zk-SNARKs for several other NP-complete languages like Boolean and arithmetic Circuit-SAT, Two-Processor Scheduling, Subset-Product, Partition, and Knapsack [17]. Tbl. 5.1 includes the complexity of Subset-Sum and Circuit-SAT, the complexity of most other SNARKs is similar to that of Subset-Sum zk-SNARK. It is an interesting open problem why some NP-complete languages like Subset-Sum have more efficient zk-SNARKs in the modular approach (equivalently, why their verification can be performed more efficiently in the parallel machine model that consists of Hadamard product and shift) than languages like Circuit-SAT. We note that [15] used recently some of the ideas from the current paper to construct an efficient shuffle argument. However, they did not use product or shift arguments.

In the full version [29], we show that by using batch-verification [3], one can decrease the verifier's computation of all presented SNARKs. In particular, one can decrease the verifier's computation in the new Range SNARK from 19 pairings to 8 pairings, one 4-way multi-exponentiation in $\mathbb{G}_1$, two 3-way multi-exponentiations in $\mathbb{G}_1$, one 2-way multi-exponentiation in $\mathbb{G}_1$, three exponentiations in $\mathbb{G}_1$, and one 3-way multi-exponentiation in $\mathbb{G}_2$.. Since one exponentiation is much cheaper than one pairing [9] and one m-way multi-exponentiation is much cheaper than m exponentiations [32, 35], this results in a significant win for the verifier. A similar technique can be used to also speed up other SNARKs; a good example here is the CIRCUIT-SAT argument from [28] that uses $\Theta(\log n)$ product and shift arguments. To compare, in Pinocchio [31] and Geppetto [13], the verifier has to execute 11 pairings; however, batch-verification can also be used to decrease this to 8 pairings and a small number of (multi-)exponentiations.

Finally, all resulting SNARKs work on data that has been committed to by using the interpolating commitment scheme. This means that one can repeatedly reuse committed data to compose different zk-SNARKs (e.g., to show that we know a satisfying input to a circuit, where the first coefficient belongs to a certain range). This is not possible with the known QSP/QAP-based zk-SNARKs where one would have to construct a single circuit of possibly considerable size, say n'. Moreover, in the QSP/QAP-based SNARKs, one has to commit to the vector, the length of which is equal to the total length of the input and witness (e.g., n' is the number of wires in the case of CIRCUIT-SAT). By using a modular solution, one can instead execute several zk-SNARKs with smaller values of the input and witness size; this can make the SNARK more prover-efficient since the number of non-cryptographic operations is superlinear. This emphasizes another benefit of the modular approach: one can choose the value n, the length of the vectors, accordingly to the desired tradeoff, so that larger n results in faster verifier computation, while smaller n results in faster prover computation. We are not aware of such a tradeoff in the case of the QSP/QAP-based approach.

We provide some additional discussion (about the relation between n and then input length, and about possible QSP/QAP-based solutions) in the full version [29]. Due to the lack of space, many proofs and details are only given in the full version [29]. We note that an early version of this paper, [29], was published in May 2014 and thus predates [13]. The published version differs from this early version mainly by exposition, and the use of proof bootstrapping (from [13]) and batching.

5.2 Preliminaries

By default, all vectors have dimension n. Let $\vec{a} \circ \vec{b}$ denote the Hadamard (i.e., element-wise) product of two vectors, with $(\vec{a} \circ \vec{b})_i = a_i b_i$. We say that $\vec{a}$ is a *shift-right-by-z* of $\vec{b}$, $\vec{a} = \vec{b} \gg z$, iff $(a_n, \ldots, a_1) = (0, \ldots, 0, b_n, \ldots, b_{1+z})$. For a tuple of polynomials $\mathcal{F} \subseteq \mathbb{Z}_p[X, Y_1, \ldots, Y_{m-1}]$, define $Y_m \mathcal{F} = (Y_m \cdot f(X, Y_1, \ldots, Y_{m-1}))_{f \in \mathcal{F}} \subseteq \mathbb{Z}_p[X, Y_1, \ldots, Y_m]$. For a tuple of polynomials $\mathcal{F}$ that have the same domain, denote $h^{\mathcal{F}(\vec{a})} := (h^{f(\vec{a})})_{f \in \mathcal{F}}$. For a group $\mathbb{G}$, let $\mathbb{G}^*$ be the set of its invertible elements. Since the direct product $\mathbb{G}_1 \times \cdots \times \mathbb{G}_m$ of groups is also a group, we use notation like $(g_1, g_2)^c = (g_1^c, g_2^c) \in \mathbb{G}_1 \times \mathbb{G}_2$ without prior definition. Let κ be the security parameter. We denote $f(\kappa) \approx_\kappa g(\kappa)$ if $|f(\kappa) - g(\kappa)|$ is negligible in κ.

On input 1^κ, a *bilinear map generator* BP returns $\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three multiplicative cyclic groups of prime order p (with $\log p = \Omega(\kappa)$), and $\hat{e}$ is an efficient bilinear map $\hat{e} \colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that satisfies in particular the following two properties, where g_1 (resp., g_2) is an arbitrary generator of $\mathbb{G}_1$ (resp., $\mathbb{G}_2$): (i) $\hat{e}(g_1, g_2) \neq 1$, and (ii) $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$. Thus, if $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1^c, g_2^d)$ then $ab \equiv cd \pmod{p}$. We also give BP another input,

n (intuitively, the input length), and allow p to depend on n. We assume that all algorithms that handle group elements verify by default that their inputs belong to corresponding groups and reject if they do not. In the case of many practically relevant pairings, arithmetic in (say) $\mathbb{G}_1$ is considerably cheaper than in $\mathbb{G}_2$; hence, we count separately exponentiations in both groups.

For $\kappa = 128$, the current recommendation is to use an optimal (asymmetric) Ate pairing over Barreto-Naehrig curves [2]. In that case, at security level of $\kappa = 128$, an element of $\mathbb{G}_1/\mathbb{G}_2/\mathbb{G}_T$ can be represented in respectively 256/512/3072 bits. To speed up interpolation, we will additionally need the existence of the n-th, where n is a power of 2, primitive root of unity modulo p (under this condition, one can interpolate in time $\Theta(n \log n)$, otherwise, interpolation takes time $\Theta(n \log^2 n)$). For this, it suffices that $(n+1) \mid (p-1)$ (recall that p is the elliptic curve group order). Fortunately, given κ and a practically relevant value of n, one can easily find a Barreto-Naehrig curve such that $(n+1) \mid (p-1)$ holds; such an observation was made also in [6]. For example, if $\kappa = 128$ and $n = 2^{10}$, one can use Alg. 1 of [2] to find an elliptic curve group of prime order $N(x_0)$ over a finite field of prime order $P(-x_0)$ for $x_0 = 1753449050$, where $P(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$, $T(x) = 6x^2 + 1$, and $N(x) = P(x) + 1 - T(x)$. One can then use the curve $E : y^2 = x^3 + 6$.

In proof bootstrapping [13], one needs an additional elliptic curve group $\tilde{E}$ over a finite field of order $N(x_0)$ (see [13] for additional details). Such elliptic curve group can be found by using the Cocks-Pinch method; note that $\tilde{E}$ has somewhat less efficient arithmetic than E.

The security of the new commitment scheme and of the new SNARKs depends on the following q-type assumptions, variants of which have been used in many previous papers. The assumptions are parameterized but non-interactive in the sense that q is related to the parameters of the language (most generally, to the input length) and not to the number of the adversarial queries. All known (to us) adaptive zk-SNARKs are based on q-type assumptions about BP.

Let $d(n) \in poly(n)$ be a function. Then, BP is

- $d(n)$*-PDL (Power Discrete Logarithm) secure* if for any $n \in poly(\kappa)$ and any non-uniform probabilistic polynomial-time (NUPPT) adversary A, $\Pr[\mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n), (g_1, g_2, \chi) \leftarrow_r \mathbb{G}_1^* \times \mathbb{G}_2^* \times \mathbb{Z}_p : \mathsf{A}(\mathsf{gk}; ((g_1, g_2)^{\chi^i})_{i=0}^{d(n)}) = \chi] \approx_\kappa 0$.
- n*-TSDH (Target Strong Diffie-Hellman) secure* if for any $n \in poly(\kappa)$ and any NUPPT adversary A, $\Pr[\mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n), (g_1, g_2, \chi) \leftarrow_r \mathbb{G}_1^* \times \mathbb{G}_2^* \times \mathbb{Z}_p : \mathsf{A}(\mathsf{gk}; ((g_1, g_2)^{\chi^i})_{i=0}^{n}) = (r, \hat{e}(g_1, g_2)^{1/(\chi - r)})] \approx_\kappa 0$.

For algorithms A and X_A, we write $(y; y') \leftarrow (\mathsf{A} || X_\mathsf{A})(\chi)$ if A on input χ outputs y, and X_A on the same input (including the random tape of A) outputs y'. We will need knowledge assumptions w.r.t. several knowledge secrets γ_i. Let m be the number of different knowledge secrets in any concrete SNARK. Let $\mathcal{F} = (P_i)_{i=0}^n$ be a tuple of univariate polynomials, and $\mathcal{G}_1$ (resp., $\mathcal{G}_2$) be a tuple of univariate (resp., m-variate) polynomials. Let $i \in [1 \,..\, m]$. Then, BP is $(\mathcal{F}, \mathcal{G}_1, \mathcal{G}_2, i)$*-PKE (Power Knowledge of Exponent) secure* if for any NUPPT adversary A there exists an NUPPT extractor X_A, such that

$$\Pr\left[\begin{array}{l} \mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n), (g_1, g_2, \chi, \vec{\gamma}) \leftarrow_r \mathbb{G}_1^* \times \mathbb{G}_2^* \times \mathbb{Z}_p \times \mathbb{Z}_p^m, \\ \vec{\gamma_{-i}} \leftarrow (\gamma_1, \dots, \gamma_{i-1}, \gamma_{i+1}, \dots, \gamma_m), \mathsf{aux} \leftarrow (g_1^{\mathcal{G}_1(\chi)}, g_2^{\mathcal{G}_2(\chi, \vec{\gamma_{-i}})}), \\ (h_1, h_2; (a_i)_{i=0}^n) \leftarrow (\mathsf{A} || X_\mathsf{A})(\mathsf{gk}; (g_1, g_2^{\gamma_i})^{\mathcal{F}(\chi)}, \mathsf{aux}) : \\ \hat{e}(h_1, g_2^{\gamma_i}) = \hat{e}(g_1, h_2) \wedge h_1 \neq g_1^{\sum_{i=0}^n a_i P_i(\chi)} \end{array}\right] \approx_\kappa 0 \ .$$

Here, aux can be seen as the common auxiliary input to A and X_A that is generated by using benign auxiliary input generation. If $\mathcal{F} = (X^i)_{i=0}^d$ for some $d = d(n)$, then we replace the first argument in $(\mathcal{F}, \dots)$-PKE with d. If $m = 1$, then we omit the last argument i in $(\mathcal{F}, \dots, i)$-PKE. While knowledge assumptions are non-falsifiable, we recall that non-falsifiable assumptions are needed to design succincts SNARKs for interesting languages [20].

By generalizing [8, 21, 26], one can show that the TSDH, PDL, and PKE assumptions hold in the generic bilinear group model.

Within this paper, $m \leq 2$, and hence we denote γ_1 just by γ, and γ_2 by δ.

An extractable trapdoor commitment scheme in the CRS model consists of two efficient algorithms $\mathsf{G_{com}}$ (that outputs a CRS ck and a trapdoor) and C (that, given ck, a message m and a randomizer r, outputs a commitment $\mathsf{C_{ck}}(m; r)$), and must satisfy the following security properties.

Computational binding: without access to the trapdoor, it is intractable to open a commitment to two different messages.

Trapdoor: given access to the original message, the randomizer and the trapdoor, one can open the commitment to any other message.

Perfect hiding: commitments of any two messages have the same distribution.

Extractability: given access to the CRS, the commitment, and the random coins of the committer, one can open the commitment to the committed message.

See, e.g., [21] for formal definitions. In the context of the current paper, the message is a vector from $\mathbb{Z}_p^n$. We denote the randomizer space by $\mathfrak{R}$.

Let $\mathcal{R} = \{(u, w)\}$ be an efficiently verifiable relation with $|w| = \mathrm{poly}(|u|)$. Here, u is a statement, and w is a witness. Let $\mathcal{L} = \{u : \exists w, (u, w) \in \mathcal{R}\}$ be an **NP**-language. Let $n = |u|$ be the input length. For fixed n, we have a relation $\mathcal{R}_n$ and a language $\mathcal{L}_n$.

Following [10, 23], we will define commit-and-prove (CaP) argument systems. Intuitively, a CaP non-interactive zero knowledge argument system for $\mathcal{R}$ allows to create a common reference string (CRS) crs, commit to some values w_i (say, $u_i = \mathsf{C_{ck}}(w_i; r_i)$, where ck is a part of crs), and then prove that a subset $u := (u_{i_j}, w_{i_j}, r_{i_j})_{j=1}^{\ell_m(n)}$ (for publicly known indices i_j) satisfies that u_{i_j} is a commitment of w_{i_j} with randomizer r_{i_j}, and that $(w_{i_j}) \in \mathcal{R}$.

Differently from most of the previous work (but see also [13]), our CaP argument systems will use computationally binding trapdoor commitment schemes. This means that without their openings, commitments $u_i = \mathsf{C_{ck}}(a_i; r_i)$ themselves do not define a valid relation, since u_i can be a commitment to any a'_i, given a suitable r'_i. Rather, we define a new relation $\mathcal{R}_{\mathsf{ck}} := \{(\vec{u}, \vec{w}, \vec{r}) : (\forall i, u_i = \mathsf{C_{ck}}(w_i; r_i)) \wedge \vec{w} \in \mathcal{R}\}$, and construct argument systems for $\mathcal{R}_{\mathsf{ck}}$.

Within this subsubsection, we let vectors $\vec{u}$, $\vec{w}$, and $\vec{r}$ be of dimension $\ell_m(n)$ for some polynomial $\ell_m(n)$. However, we allow committed messages w_i themselves to be vectors of dimension n. Thus, $\ell_m(n)$ is usually very small. In some argument systems (like the Subset-Sum SNARK in Sect. 5.6), also the argument will include some commitments. In such cases, technically speaking, $\vec{w}$ and $\vec{r}$ are of higher dimension than $\vec{u}$. To simplify notation, we will ignore this issue.

A *commit-and-prove non-interactive zero-knowledge argument system* [10, 23] Π for $\mathcal{R}$ consists of an ($\mathcal{R}$-independent) trapdoor commitment scheme $\Gamma = (\mathsf{G_{com}}, \mathsf{C})$ and of a non-interactive zero-knowledge argument system $(\mathsf{G}, \mathsf{P}, \mathsf{V})$, that are combined as follows: 1. the CRS generator G (that, in particular, invokes $(\mathsf{ck}, \mathsf{td_C}) \leftarrow \mathsf{G_{com}}(1^\kappa, n)$) outputs $(\mathsf{crs} = (\mathsf{crs}_p, \mathsf{crs}_v), \mathsf{td}) \leftarrow \mathsf{G}(1^\kappa, n)$, where both crs_p and crs_v include ck, and td includes $\mathsf{td_C}$. 2. the prover P produces an argument π, $\pi \leftarrow \mathsf{P}(\mathsf{crs}_p; \vec{u}; \vec{w}, \vec{r})$, where presumably $u_i = \mathsf{C_{ck}}(w_i; r_i)$. 3. the verifier V, $\mathsf{V}(\mathsf{crs}_v; \vec{u}, \pi)$, outputs either 1 (accept) or 0 (reject). [(i)] Now, Π is *perfectly complete*, if for all $n = \mathrm{poly}(\kappa)$, $\Pr[(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{G}(1^\kappa, n), (\vec{u}, \vec{w}, \vec{r}) \leftarrow \mathcal{R}_{\mathsf{ck},n} : \mathsf{V}(\mathsf{crs}_v; \vec{u}, \mathsf{P}(\mathsf{crs}_p; \vec{u}, \vec{w}, \vec{r})) = 1] = 1$.

Since Γ is computationally binding and trapdoor (and hence u_i can be commitments to *any* messages), soundness of the CaP argument systems only makes sense together with the argument of knowledge property.

Let $b(X)$ be a non-negative polynomial. Π is *a (b-bounded-auxiliary-input) argument of knowledge* for $\mathcal{R}$, if for all $n = \mathrm{poly}(\kappa)$ and every NUPPT A, there exists an NUPPT extractor X_{A}, such that for every auxiliary input $\mathsf{aux} \in \{0,1\}^{b(\kappa)}$, $\Pr[(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{G}(1^\kappa, n), ((\vec{u}, \pi); \vec{w}, \vec{r}) \leftarrow (\mathsf{A}||X_\mathsf{A})(\mathsf{crs}; \mathsf{aux}) : (\vec{u}, \vec{w}, \vec{r}) \not\in \mathcal{R}_{\mathsf{ck},n} \wedge \mathsf{V}(\mathsf{crs}_v; \vec{u}, \pi) = 1] \approx_\kappa 0$. As in the definition of PKE, we can restrict the definition of an argument of knowledge to benign auxiliary information generators, where aux is known to come from; we omit further discussion.

Π is *perfectly witness-indistinguishable*, if for all $n = \text{poly}(\kappa)$, it holds that if $(\mathsf{crs}, \mathsf{td}) \in \mathsf{G}(1^\kappa, n)$ and $((\vec{u}; \vec{w}, \vec{r}), (\vec{u}; \vec{w}', \vec{r}')) \in \mathcal{R}^2_{\mathsf{ck},n}$ with $r_i, r'_i \leftarrow_r \mathfrak{R}$, then the distributions $\mathsf{P}(\mathsf{crs}_p; \vec{u}; \vec{w}, \vec{r})$ and $\mathsf{P}(\mathsf{crs}_p; \vec{u}; \vec{w}', \vec{r}')$ are equal. Note that a witness-indistinguishable argument system does not have to have a trapdoor.

Π is *perfectly composable zero-knowledge*, if there exists a probabilistic poly-time simulator S, s.t. for all stateful NUPPT adversaries A and $n = \text{poly}(\kappa)$, $\Pr[(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{G}(1^\kappa, n), (\vec{u}, \vec{w}, \vec{r}) \leftarrow \mathsf{A}(\mathsf{crs}), \pi \leftarrow \mathsf{P}(\mathsf{crs}_p; \vec{u}; \vec{w}, \vec{r}) : (\vec{u}, \vec{w}, \vec{r}) \in \mathcal{R}_{\mathsf{ck},n} \wedge \mathsf{A}(\pi) = 1] = \Pr[(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{G}(1^\kappa, n), (\vec{u}, \vec{w}, \vec{r}) \leftarrow \mathsf{A}(\mathsf{crs}), \pi \leftarrow \mathsf{S}(\mathsf{crs}; \vec{u}, \mathsf{td}) : (\vec{u}, \vec{w}, \vec{r}) \in \mathcal{R}_{\mathsf{ck},n} \wedge \mathsf{A}(\pi) = 1]$. Here, the prover and the simulator use the same CRS, and thus we have *same-string zero knowledge*. Same-string statistical zero knowledge allows to use the same CRS an unbounded number of times.

An argument system that satisfies above requirements is known as *adaptive*. An argument system where the CRS depends on the statement is often called *non-adaptive*. It is not surprising that non-adaptive SNARKs can be much more efficient than adaptive SNARKs.

A non-interactive argument system is *succinct* if the output length of P and the running time of V are polylogarithmic in the P's input length (and polynomial in the security parameter). A succinct non-interactive argument of knowledge is usually called *SNARK*. A zero-knowledge SNARK is abbreviated to *zk-SNARK*.

5.3 New Extractable Trapdoor Commitment Scheme

We now define a new extractable trapdoor commitment scheme. It uses the following polynomials. Assume n is a power of two, and let ω be the n-th primitive root of unity modulo p. Then,

- $Z(X) := \prod_{i=1}^n (X - \omega^{i-1}) = X^n - 1$ is the unique degree n monic polynomial, such that $Z(\omega^{i-1}) = 0$ for all $i \in [1 \,..\, n]$.
- $\ell_i(X) := \prod_{j \neq i}((X - \omega^{j-1})/(\omega^{i-1} - \omega^{j-1}))$, the *ith Lagrange basis polynomial*, is the unique degree $n-1$ polynomial, such that $\ell_i(\omega^{i-1}) = 1$ and $\ell_i(\omega^{j-1}) = 0$ for $j \neq i$.

Clearly, $L_{\vec{a}}(X) = \sum_{i=1}^n a_i \ell_i(X)$ is the interpolating polynomial of $\vec{a}$ at points ω^{i-1}, with $L_{\vec{a}}(\omega^{i-1}) = a_i$, and can thus be computed by executing an inverse Fast Fourier Transform. Moreover, $(\ell_i(\omega^{j-1}))_{j=1}^n = \vec{e}_i$ (the ith unit vector) and $(Z(\omega^{j-1}))_{j=1}^n = \vec{0}_n$. Thus, $Z(X)$ and $(\ell_i(X))_{i=1}^n$ are $n+1$ linearly independent degree $\leq n$ polynomials, and hence $\mathcal{F}_\mathsf{C} := (Z(X), (\ell_i(X))_{i=1}^n)$ is a basis of such polynomials. Clearly, $Z^{-1}(0) = \{j : Z(j) = 0\} = \{\omega^{i-1}\}_{i=1}^n$.

Definition 1 (Interpolating Commitment Scheme)**.** *Let $n = \text{poly}(\kappa)$, $n > 0$, be a power of two. First, $\mathsf{G}_\mathsf{com}(1^\kappa, n)$ sets $\mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa, n)$, picks $g_1 \leftarrow_r \mathbb{G}_1^*$, $g_2 \leftarrow_r \mathbb{G}_2^*$, and then outputs the CRS $\mathsf{ck} \leftarrow (\mathsf{gk}; (g_1^{f(\chi)}, g_2^{\gamma f(\chi)})_{f \in \mathcal{F}_\mathsf{C}})$ for $\chi \leftarrow_r \mathbb{Z}_p \setminus Z^{-1}(0)$ and $\gamma \leftarrow_r \mathbb{Z}_p^*$. The trapdoor is equal to χ.*

The commitment of $\vec{a} \in \mathbb{Z}_p^n$, given a randomizer $r \leftarrow_r \mathbb{Z}_p$, is $\mathsf{C}_\mathsf{ck}(\vec{a}; r) := (g_1^{Z(\chi)}, g_2^{\gamma Z(\chi)})^r \cdot \prod_{i=1}^n (g_1^{\ell_i(\chi)}, g_2^{\gamma \ell_i(\chi)})^{a_i} \in \mathbb{G}_1 \times \mathbb{G}_2$, i.e., $\mathsf{C}_\mathsf{ck}(\vec{a}; r) := (g_1, g_2^\gamma)^{r(\chi^n - 1) + L_{\vec{a}}(\chi)}$. The validity of a commitment (A_1, A_2^γ) is checked by verifying that $\hat{e}(A_1, g_2^{\gamma Z(\chi)}) = \hat{e}(g_1^{Z(\chi)}, A_2^\gamma)$. To open a commitment, the committer sends $(\vec{a}, r)$ to the verifier.

The condition $Z(\chi) \neq 0$ is needed in Thm. 1 to get perfect hiding and the trapdoor property. The condition $\gamma \neq 0$ is only needed in Thm. 5 to get perfect zero knowledge. Also, (a function of) γ is a part of the trapdoor in the range SNARK of Sect. 5.7.

Clearly, $\log_{g_1} A_1 = \log_{g_2^\gamma} A_2^\gamma = rZ(\chi) + \sum_{i=1}^n a_i \ell_i(\chi)$. The second element, A_2^γ, of the commitment is known as the knowledge component.

Theorem 1. *The interpolating commitment scheme is perfectly hiding and trapdoor. If* BP *is* n*-PDL secure, then it is computationally binding. If* BP *is* $(n, \emptyset, \emptyset)$*-PKE secure, then it is extractable.*

Proof. PERFECT HIDING: since $Z(\chi) \neq 0$, then $rZ(\chi)$ (and thus also $\log_{g_1} A_1$) is uniformly random in $\mathbb{Z}_p$. Hence, (A_1, A_2^γ) is a uniformly random element of the multiplicative subgroup $\langle (g_1, g_2^\gamma) \rangle \subset \mathbb{G}_1^* \times \mathbb{G}_2^*$ generated by (g_1, g_2^γ), independently of the committed value. TRAPDOOR: given χ, $\vec{a}$, r, $\vec{a^*}$, and $c = \mathsf{C}_{\mathsf{ck}}(\vec{a}; r)$, we compute r^* s.t. $(r^* - r)Z(\chi) + \sum_{i=1}^n (a_i^* - a_i)\ell_i(\chi) = 0$. This is possible since $Z(\chi) \neq 0$. Clearly, $c = \mathsf{C}_{\mathsf{ck}}(\vec{a^*}; r^*)$. EXTRACTABILITY: clear from the statement.

COMPUTATIONAL BINDING: assume that there exists an adversary A_C that outputs $(\vec{a}, r_a)$ and $(\vec{b}, r_b)$ with $(\vec{a}, r_a) \neq (\vec{b}, r_b)$, s.t. the polynomial $d(X) := (r_a Z(X) + \sum_{i=1}^n a_i \ell_i(X)) - (r_b Z(X) + \sum_{i=1}^n b_i \ell_i(X))$ has a root at χ.

Construct now the following adversary A_{pdl} that breaks the PDL assumption. Given an n-PDL challenge, since $\mathcal{F}_\mathsf{C}$ consists of degree $\leq n$ polynomials, A_{pdl} can compute a valid ck from (a distribution that is statistically close to) the correct distribution. He sends ck to A_C. If A_C is successful, then $d(X) \in \mathbb{Z}_p[X]$ is a non-trivial degree-$\leq n$ polynomial. Since the coefficients of d are known, A_{pdl} can use an efficient polynomial factorization algorithm to compute all roots r_i of $d(X)$. One of these roots has to be equal to χ. A_{pdl} can establish which one by comparing each (say) $g_1^{\ell_1(r_i)}$ to the element $g_1^{\ell_1(\chi)}$ given in the CRS. Clearly, $g_1^{\ell_1(r_i)}$ is computed from g_1 (which can be computed, given the CRS, since $1 \in \mathrm{span}(\mathcal{F}_\mathsf{C})$), the coefficients of $\ell_1(X)$, and r_i. A_{pdl} has the same success probability as A_C, while her running time is dominated by that of A_C plus the time to factor a degree-$\leq n$ polynomial. □ □

Thm. 1 also holds when instead of $Z(X)$ and $\ell_i(X)$ one uses any $n+1$ linearly independent low-degree polynomials (say) $P_0(X)$ and $P_i(X)$. Given the statement of Thm. 1, this choice of the concrete polynomials is very natural: $\ell_i(X)$ interpolate linearly independent vectors (and thus are linearly independent; in fact, they constitute a basis), and the choice to interpolate unit vectors is the conceptually clearest way of choosing $P_i(X)$. Another natural choice of independent polynomials is to set $P_i(X) = X^i$ as in [21], but that choice has resulted in much less efficient (CaP) SNARKs.

In the full version [29] we show how to use batch-verification techniques to speed up simultaneous validity verification of many commitments.

5.4 New Product SNARK

Assume the use of the interpolating commitment scheme. In a *CaP product SNARK* [21], the prover aims to convince the verifier that she knows how to open three commitments (A, A^γ), (B, B^γ), and (C, C^γ) to vectors $\vec{a}$, $\vec{b}$ and $\vec{c}$ (together with the used randomizers), such that $\vec{a} \circ \vec{b} = \vec{c}$. Thus,

$$\mathcal{R}^\times_{\mathsf{ck},n} := \left\{ \begin{array}{l} (u_\times, w_\times, r_\times) : u_\times = ((A_1, A_2^\gamma), (B_1, B_2^\gamma), (C_1, C_2^\gamma)) \wedge \\ w_\times = (\vec{a}, \vec{b}, \vec{c}) \wedge r_\times = (r_a, r_b, r_c) \wedge (A_1, A_2^\gamma) = \mathsf{C}_{\mathsf{ck}}(\vec{a}; r_a) \wedge \\ (B_1, B_2^\gamma) = \mathsf{C}_{\mathsf{ck}}(\vec{b}; r_b) \wedge (C_1, C_2^\gamma) = \mathsf{C}_{\mathsf{ck}}(\vec{c}; r_c) \wedge \vec{a} \circ \vec{b} = \vec{c} \end{array} \right\} .$$

Next, we propose an efficient CaP product SNARK. For this, we need Lem. 1.

Lemma 1. *Let* $A(X)$*,* $B(X)$ *and* $C(X)$ *be polynomials with* $A(\omega^{i-1}) = a_i$*,* $B(\omega^{i-1}) = b_i$ *and* $C(\omega^{i-1}) = c_i$*,* $\forall i \in [1\,..\,n]$*. Let* $Q(X) = A(X)B(X) - C(X)$*. Assume that (i)* $A(X), B(X), C(X) \in$

$\mathrm{span}\{\ell_i(X)\}_{i=1}^n$, *and (ii) there exists a degree* $n-2$ *polynomial* $\pi(X)$, *s.t.* $\pi(X) = Q(X)/Z(X)$. *Then* $\vec{a} \circ \vec{b} = \vec{c}$.

Proof. From (i) it follows that $A(X) = L_{\vec{a}}(X)$, $B(X) = L_{\vec{b}}(X)$, and $C(X) = L_{\vec{c}}(X)$, and thus $Q(\omega^{i-1}) = a_i b_i - c_i$ for all $i \in [1\,..\,n]$. But (ii) iff $Z(X) \mid Q(X)$, which holds iff $Q(X)$ evaluates to 0 at all n values ω^{i-1}. Thus, $\vec{a} \circ \vec{b} = \vec{c}$. Finally, if (i) holds then $\deg Q(X) = 2n-2$ and thus $\deg \pi(X) = n-2$. □ □

If privacy and succinctness are not needed, one can think of the product argument being equal to $\pi(X)$. We achieve privacy by picking $r_a, r_b, r_c \leftarrow_r \mathbb{Z}_p$, and defining $Q_{wi}(X) := (L_{\vec{a}}(X) + r_a Z(X)) \left(L_{\vec{b}}(X) + r_b Z(X)\right.$ $(L_{\vec{c}}(X) + r_c Z(X))$. Here, the new addends of type $r_a Z(X)$ guarantee hiding. On the other hand, $Q_{wi}(X)$ remains divisible by $Z(X)$ iff $\vec{c} = \vec{a} \circ \vec{b}$. Thus, $\vec{a} \circ \vec{b} = \vec{c}$ iff

(i') $Q_{wi}(X)$ can be expressed as $Q_{wi}(X) = A(X)B(X) - C(X)$ for some polynomials $A(X)$, $B(X)$ and $C(X)$ that belong to the span of $\mathcal{F}_{\mathsf{C}}$, and

(ii') there exists a polynomial $\pi_{wi}(X)$, such that

$$\pi_{wi}(X) = Q_{wi}(X)/Z(X) \ . \tag{5.1}$$

The degree of $Q_{wi}(X)$ is $2n$, thus, if $\pi_{wi}(X)$ exists, then it has degree n.

However, $|\pi_{wi}(X)|$ is not sublinear in n. To minimize communication, we let the prover transfer a "garbled" evaluation of $\pi_{wi}(X)$ at a random secret point χ. More precisely, the prover computes $\pi_\times := g_1^{\pi_{wi}(\chi)}$, using the values $g_1^{\chi^i}$ (given in the CRS) and the coefficients π_i of $\pi_{wi}(X) = \sum_{i=0}^n \pi_i X^i$, as follows:

$$\pi_\times := g_1^{\pi_{wi}(\chi)} \leftarrow \prod_{i=0}^n (g_1^{\chi^i})^{\pi_i} \ . \tag{5.2}$$

Similarly, instead of (say) $L_{\vec{a}}(X) + r_a Z(X)$, the verifier has the succinct interpolating commitment $\mathsf{C}_{\mathsf{ck}}(\vec{a}; r_a) = (g_1, g_2^\gamma)^{L_{\vec{a}}(\chi) + r_a Z(\chi)}$ of $\vec{a}$.

We now give a full description of the new product SNARK $\Pi_\times$, given the interpolating commitment scheme $(\mathsf{G_{com}}, \mathsf{C})$ and the following tuple of algorithms, $(\mathsf{G}_\times, \mathsf{P}_\times, \mathsf{V}_\times)$. Note that $\mathsf{C}_{\mathsf{ck}}(\vec{1_n}; 0) = (g_1, g_2^\gamma)$.

CRS generation $\mathsf{G}_\times(1^\kappa, n)$**:** Let $\mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa)$, $(g_1, g_2, \chi, \gamma) \leftarrow_r \mathbb{G}_1^* \times \mathbb{G}_2^* \times \mathbb{Z}_p^2$ with $Z(\chi) \neq 0$ and $\gamma \neq 0$. Let $\mathsf{crs}_p = \mathsf{ck} \leftarrow (\mathsf{gk}; (g_1, g_2^\gamma)^{\mathcal{F}_{\mathsf{C}}(\chi)})$ and $\mathsf{crs}_v \leftarrow (\mathsf{gk}; g_2^{\gamma Z(\chi)})$. Output $\mathsf{crs}_\times = (\mathsf{crs}_p, \mathsf{crs}_v)$.

Common input: $u_\times = ((A_1, A_2^\gamma), (B_1, B_2^\gamma), (C_1, C_2^\gamma))$.

Proving $\mathsf{P}_\times(\mathsf{crs}_p; u_\times; w_\times = (\vec{a}, \vec{b}, \vec{c}), r_\times = (r_a, r_b, r_c))$**:** Compute $\pi_{wi}(X) = \sum_{i=0}^n \pi_i X^i$ as in Eq. (5.1) and $\pi_\times$ as in Eq. (5.2). Output $\pi_\times$.

Verification $\mathsf{V}_\times(\mathsf{crs}_v; u_\times; \pi_\times)$**:** accept if $\hat{e}(A_1, B_2^\gamma) = \hat{e}(g_1, C_2^\gamma) \cdot \hat{e}(\pi_\times, g_2^{\gamma Z(\chi)})$.

Since one can recompute it from ck, inclusion of $g_2^{\gamma Z(\chi)}$ in the CRS is only needed to speed up the verification. Here as in the shift SNARK of Sect. 5.5, validity of the commitments will be verified in the master SNARK. This is since the master SNARKs use some of the commitments in several sub-SNARKs, while it suffices to verify the validity of every commitment only once.

To obtain an argument of knowledge, we use knowledge assumptions in all following proofs. This SNARK is not zero-knowledge since the possible simulator gets three commitments as inputs but not their openings; to create an accepting argument the simulator must at least know how to open the commitment $(A_1 B_1 / C_1, A_2^\gamma B_2^\gamma / C_2^\gamma)$ to $\vec{a} \circ \vec{b} - \vec{c}$. It is witness-indistinguishable, and this suffices for the Subset-Sum and other master SNARKs to be zero-knowledge.

Theorem 2. $\Pi_\times$ *is perfectly complete and witness-indistinguishable. If the input consists of valid commitments, and* BP *is* n*-TSDH and* $(n, \emptyset, \emptyset)$*-PKE secure, then* $\Pi_\times$ *is an* ($\Theta(n)$*-bounded-auxiliary-input) adaptive argument of knowledge.*

Proof. PERFECT COMPLETENESS: follows from the discussion in the beginning of this section. PERFECT WITNESS-INDISTINGUISHABILITY: since the argument $\pi_\times$ that satisfies the verification equations is unique, all witnesses result in the same argument, and thus this argument is witness-indistinguishable.

ARGUMENT OF KNOWLEDGE: Assume that A_{aok} is an adversary that, given $\mathsf{crs}_\times$, returns $(u_\times, \pi)$ such that $\mathsf{V}_\times(\mathsf{crs}_v; u_\times, \pi) = 1$. Assume that the PKE assumption holds, and let X_A be the extractor thatreturns openings of the commitments in $u_\times$, i.e., $(\vec{a}, r_a)$, $(\vec{b}, r_b)$, and $(\vec{c}, r_c)$. We now claim that X_A is also the extractor needed to achieve the argument of knowledge property.

Assume that this is not the case. We construct an adversary A_{tsdh} against n-TSDH. Given an n-TSDH challenge $ch = (\mathsf{gk}, ((g_1, g_2)^{\chi^i})_{i=0}^n)$, A_{tsdh} first generates $\gamma \leftarrow_r \mathbb{Z}_p^*$, and then computes (this is possible since $\mathcal{F}_\mathsf{C}$ consists of degree $\leq n$ polynomials) and sends $\mathsf{crs}_\times$ to A_{aok}. Assume $(\mathsf{A}_{aok} || X_\mathsf{A})(\mathsf{crs}_\times)$ returns $((u_\times = ((A_1, A_2^\gamma), (B_1, B_2^\gamma), (C_1, C_2^\gamma)), \pi), (w_\times = (\vec{a}, \vec{b}, \vec{c}), r_\times = (r_a, r_b, r_C)))$, s.t. $u_i = \mathsf{C}_\mathsf{ck}(w_i; r_i)$ but $(u_\times, w_\times, r_\times) \notin \mathcal{R}_{\mathsf{ck},n}^\times$. Since the openings are correct, $\vec{a} \circ \vec{b} \neq \vec{c}$ but π is accepting. According to Lem. 1, thus $Z(X) \nmid Q_{wi}(X)$.

Since $Z(X) \nmid Q_{wi}(X)$, then for some $i \in [1 \mathinner{..} n]$, $(X - \omega^{i-1}) \nmid Q_{wi}(X)$. Write $Q_{wi}(X) = q(X)(X - \omega^{i-1}) + r$ for $r \in \mathbb{Z}_p^*$. Clearly, $\deg q(X) \leq 2n-1$. Moreover, we write $q(X) = q_1(X)Z(X) + q_2(X)$ with $\deg q_i(X) \leq n - 1$. Since the verification succeeds, $\hat{e}(g_1, g_2^\gamma)^{Q_{wi}(\chi)} = \hat{e}(\pi_\times, g_2^{\gamma Z(\chi)})$, or $\hat{e}(g_1, g_2^\gamma)^{q(\chi)(\chi - \omega^{i-1}) + r} = \hat{e}(\pi_\times, g_2^{\gamma Z(\chi)})$, or $\hat{e}(g_1, g_2^\gamma)^{q(\chi) + r/(\chi - \omega^{i-1})} = \hat{e}(\pi_\times, g_2^{\gamma Z(\chi)/(\chi - \omega^{i-1})})$, or $\hat{e}(g_1, g_2^\gamma)^{1/(\chi - \omega^{i-1})} = (\hat{e}(\pi_\times, g_2^{\gamma Z(\chi)/(\chi - \omega^{i-1})}) / \hat{e}(g_1^{q(\chi)}, g_2^\gamma))^{r^{-1}}$.

Now, $\hat{e}(g_1^{q(\chi)}, g_2^\gamma) = \hat{e}(g_1^{q_1(\chi)}, g_2^{\gamma Z(\chi)}) \hat{e}(g_1^{q_2(\chi)}, g_2^\gamma)$, and thus it can be efficiently computed from $((g_1^{\chi^i})_{i=0}^{n-1}, g_2^\gamma, g_2^{\gamma Z(\chi)}) \subset \mathsf{crs}$. Moreover, $Z(X)/(X - \omega^{i-1}) = \ell_i(X) \cdot \prod_{j \neq i} (\omega^{i-1} - \omega^{j-1})$, and thus $g_2^{\gamma Z(\chi)/(\chi - \omega^{i-1})}$ can be computed from $g_2^{\gamma \ell_i(\chi)}$ by using generic group operations. Hence, $\hat{e}(g_1, g_2^\gamma)^{1/(\chi - \omega^{i-1})}$ can be computed from $((g_1^{\chi^i})_{i=0}^{n-1}, g_2^\gamma, g_2^{\gamma Z(\chi)}, (g_2^{\gamma \ell_i(\chi)})_{i=1}^n)$ (that can be computed from ch), by using generic group operations. Thus, the adversary has computed $(r = \omega^{i-1}, \hat{e}(g_1, g_2^\gamma)^{1/(\chi - r)})$, for $r \neq \chi$. Since A_{tsdh} knows $\gamma \neq 0$, he can finally compute $(r, \hat{e}(g_1, g_2)^{1/(\chi - r)})$, and thus break the n-TSDH assumption.

Hence, the argument of knowledge property follows. □ □

We remark that the product SNARK (but not the shift SNARK of Sect. 5.5) can be seen as a QAP-based SNARK [19], namely for the relation $\vec{a} \circ \vec{b} - \vec{c}$. (Constructing a QAP-based shift SNARK is possible, but results in using different polynomials and thus in a different commitment scheme.)

The prover computation is dominated by the following: (i) one $(n+1)$-wide multi-exponentiation in $\mathbb{G}_1$. By using the Pippenger's multi-exponentiation algorithm for *large* n this means approximately $n + 1$ bilinear-group multiplications, see [32]. For small values of n, one can use the algorithm by Straus [35]; then one has to execute $\Theta(n / \log n)$ bilinear-group exponentiations. (ii) three polynomial interpolations, one polynomial multiplication, and one polynomial division to compute the coefficients of the polynomial $\pi_{wi}(X)$. Since polynomial division can be implemented as 2 polynomial multiplications (by using pre-computation and storing some extra information in the CRS, [27]), this part is dominated by two inverse FFT-s and three polynomial multiplications.

The verifier computation is dominated by 3 pairings. (We will count the cost of validity verifications separately in the master SNARKs.) In the special case $C_1 = A_1$ (e.g., in the *Boolean SNARK*, where we need to prove that $\vec{a} \circ \vec{a} = \vec{a}$, or in the *restriction SNARK* [21], where we need to prove that $\vec{a} \circ \vec{b} = \vec{a}$ for a *public* Boolean vector $\vec{b}$), the verification equation can be simplified to $\hat{e}(A_1, B_2^\gamma / g_2^\gamma) = \hat{e}(\pi_\times, g_2^{\gamma Z(\chi)})$, which saves one more pairing. In the full version [29], we will describe a batch-verification technique that allows to speed up simultaneous verification of several product SNARKs.

Excluding gk, the prover CRS together with ck consists of $2(n+1)$ group elements, while the verifier CRS consists of 1 group element. The CRS can be computed in time $\Theta(n)$, by using an algorithm from [4].

5.5 New Shift SNARK

In a *shift-right-by-z* SNARK [16] (shift SNARK, for short), the prover aims to convince the verifier that for 2 commitments (A, A^γ) and (B, B^γ), he knows how to open them as $(A, A^\gamma) = \mathsf{C}_{\mathsf{ck}}(\vec{a}; r_a)$ and $(B, B^\gamma) = \mathsf{C}_{\mathsf{ck}}(\vec{b}; r_b)$, s.t. $\vec{a} = \vec{b} \gg z$. I.e., $a_i = b_{i+z}$ for $i \in [1 \mathinner{..} n-z]$ and $a_i = 0$ for $i \in [n-z+1 \mathinner{..} n]$. Thus,

$$\mathcal{R}^{\mathsf{rsft}}_{\mathsf{ck},n} := \left\{ \begin{array}{l} (u_\times, w_\times, r_\times) : u_\times = ((A_1, A_2^\gamma), (B_1, B_2^\gamma)) \wedge w_\times = (\vec{a}, \vec{b}) \wedge \\ r_\times = (r_a, r_b) \wedge (A_1, A_2^\gamma) = \mathsf{C}_{\mathsf{ck}}(\vec{a}; r_a) \wedge \\ (B_1, B_2^\gamma) = \mathsf{C}_{\mathsf{ck}}(\vec{b}; r_b) \wedge (\vec{a} = \vec{b} \gg z) \end{array} \right\} .$$

An efficient shift SNARK was described in [16]. We now reconstruct this SNARK so that it can be used together with the interpolating commitment scheme. We can do it since the shift SNARK of [16] is *almost* independent of the commitment scheme. We also slightly optimize the resulting SNARK; in particular, the verifier has to execute one less pairing compared to [16].

Our strategy of constructing a shift SNARK follows the strategy of [21, 26]. We start with a concrete verification equation that also contains the argument, that we denote by π_1. We write the discrete logarithm of π_1 (that follows from this equation) as $F_\pi(\chi) + F_{con}(\chi)$, where χ is a secret key, and $F_\pi(X)$ and $F_{con}(X)$ are two polynomials. The first polynomial, $F_\pi(X)$, is identically zero iff the prover is honest. Since the spans of certain two polynomial sets do not intersect, this results in an efficient adaptive shift SNARK that is an argument of knowledge under (two) PKE assumptions.

Now, for a non-zero polynomial $Z^*(X)$ to be defined later, consider the verification equation $\hat{e}(A_1, g_2^{\gamma Z^*(\chi)})/\hat{e}(B_1 \pi_1, g_2^\gamma) = 1$ (due to the properties of pairing, this is equivalent to verifying that $\pi_1 = A_1^{Z^*(\chi)}/B_1$), with (A_1, A_2^γ) and (B_1, B_2^γ) being interpolating commitments to $\vec{a}$ and $\vec{b}$, and $\pi_1 = g_1^{\pi(\chi)}$ for some polynomial $\pi(X)$. Denote $r(X) := (r_a Z^*(X) - r_b) Z(X)$. Taking a discrete logarithm of the verification equation, we get that $\pi(X) = (r_a Z(X) + \sum_{i=1}^n a_i \ell_i(X)) Z^*(X) - (r_b Z(X) + \sum_{i=1}^n b_i \ell_i(X)) = Z^*(X) \sum_{i=1}^n a_i \ell_i(X) - \sum_{i=1}^n b_i \ell_i(X) + r(X) = \left(\sum_{i=1}^{n-z} a_i \ell_i(X) + \sum_{i=n-z+1}^n a_i \ell_i(X)\right) Z^*($ $r(X) - \sum_{i=1}^{n-z} b_{i+z} \ell_{i+z}(X) - \sum_{i=1}^z b_i \ell_i(X)$. Hence, $\pi(X) = F_\pi(X) + F_{con}(X)$, where

$$\begin{aligned} F_\pi(X) &= \left(\sum_{i=1}^{n-z} (a_i - b_{i+z}) \ell_i(X) + \sum_{i=n-z+1}^n a_i \ell_i(X)\right) \cdot Z^*(X) \ , \\ F_{con}(X) &= \left(\sum_{i=z+1}^n b_i (\ell_{i-z}(X) Z^*(X) - \ell_i(X)) - \sum_{i=1}^z b_i \ell_i(X)\right) + r(X) \ . \end{aligned}$$

Clearly, the prover is honest iff $F_\pi(X) = 0$, which holds iff $\pi(X) = F_{con}(X)$, i.e., $\pi(X)$ belongs to the span of $\mathcal{F}_{z-\mathsf{rsft}} := (\ell_{i-z}(X) Z^*(X) - \ell_i(X))_{i=z+1}^n, (\ell_i(X))_{i=1}^z, Z(X) Z^*(X), Z(X))$. For the shift SNARK to be an argument of knowledge, we need that

(i) $(\ell_i(X) Z^*(X))_{i=1}^n$ is linearly independent, and

(ii) $F_\pi(X) \cap \operatorname{span}(\mathcal{F}_{z-\mathsf{rsft}}) = \emptyset$.

Together, (i) and (ii) guarantee that from $\pi(X) \in \operatorname{span}(\mathcal{F}_{z-\mathsf{rsft}})$ it follows that $\vec{a}$ is a shift of $\vec{b}$.

We guarantee that $\pi(X) \in \operatorname{span}(\mathcal{F}_{z-\mathsf{rsft}})$ by a knowledge assumption (w.r.t. another knowledge secret δ); for this we will also show that $\mathcal{F}_{z-\mathsf{rsft}}$ is linearly independent. As in the case of the product

SNARK, we also need that (A_1, A_2^γ) and (B_1, B_2^γ) are actually commitments of n-dimensional vectors (w.r.t. γ), i.e., we rely on two PKE assumptions.

Denote $\mathcal{F}_\pi := \{\ell_i(X)Z^*(X)\}_{i=1}^n$. For a certain choice of $Z^*(X)$, both (i) and (ii) follow from the next lemma.

Lemma 2. *Let $Z^*(X) = Z(X)^2$. Then $\mathcal{F}_\pi \cup \mathcal{F}_{z-\mathsf{rsft}}$ is linearly independent.*

Proof. Assume that there exist $\vec{a} \in \mathbb{Z}_p^n$, $\vec{b} \in \mathbb{Z}_p^n$, $c \in \mathbb{Z}_p$, and $d \in \mathbb{Z}_p$, s.t. $f(X) := \sum_{i=1}^n a_i\ell_i(X)Z^*(X) + \sum_{i=z+1}^n b_i (\ell_{i-z}(X)Z^*(X) - \ell_i(X)) - \sum_{i=1}^z b_i\ell_i(X) + cZ(X)Z^*(X) + dZ(X) = 0$. But then also $f(\omega^{j-1}) = 0$, for $j \in [1\,..\,n]$. Thus, due to the definition of $\ell_i(X)$ and $Z(X)$, $\sum_{i=1}^n b_i\vec{e}_i = \vec{0}_n$ which is only possible if $b_i = 0$ for all $i \in [1\,..\,n]$. Thus also $f'(X) := f(X)/Z(X) = \sum_{i=1}^n a_i\ell_i(X)Z^*(X)/Z(X) + cZ^*(X) + d = 0$. But then also $f'(\omega^{j-1}) = 0$ for $j \in [1\,..\,n]$. Hence, $cZ^*(\omega^{j-1}) + d = d = 0$. Finally, $f''(X) := f(X)/Z^*(X) = \sum_{i=1}^n a_i\ell_i(X) + cZ(X) = 0$, and from $f''(\omega^{j-1}) = 0$ for $j \in [1\,..\,n]$, we get $\vec{a} = \vec{0}_n$. Thus also $c = 0$. This finishes the proof. □ □

Since the argument of knowledge property of the new shift SNARK relies on $\pi(X)$ belonging to a certain span, similarly to [16], we will use an additional knowledge assumption. That is, it is necessary that there exists an extractor that outputs a witness that $\pi(X) = F_{con}(X)$ belongs to the span of $\mathcal{F}_{z-\mathsf{rsft}}$.

Similarly to the product SNARK, the shift SNARK does not contain $\pi(X) = F_{con}(X)$, but the value $\pi_{\mathsf{rsft}} = (g_1, g_2^\delta)^{\pi(\chi)}$ for random χ and δ (necessary due to the use of the second PKE assumption), computed as

$$\begin{aligned}\pi_{\mathsf{rsft}} \leftarrow &(\pi_1, \pi_2^\delta) = (g_1, g_2^\delta)^{\pi(\chi)}\\ = &\textstyle\prod_{i=z+1}^n ((g_1, g_2^\delta)^{\ell_{i-z}(\chi)Z^*(\chi) - \ell_i(\chi)})^{b_i} \cdot \prod_{i=1}^z ((g_1, g_2^\delta)^{\ell_i(\chi)})^{-b_i} \cdot\\ &((g_1, g_2^\delta)^{Z(\chi)Z^*(\chi)})^{r_a} \cdot ((g_1, g_2^\delta)^{Z(\chi)})^{-r_b} \enspace .\end{aligned} \tag{5.3}$$

We are now ready to state the new shift-right-by-z SNARK Π_{rsft}. It consists of the interpolating commitment scheme and of the following three algorithms:

CRS generation $\mathsf{G}_{\mathsf{rsft}}(1^\kappa, n)$**:** Let $Z^*(X) = Z(X)^2$. Let $\mathsf{gk} \leftarrow \mathsf{BP}(1^\kappa)$, $(g_1, g_2, \chi, \gamma, \delta) \leftarrow \mathbb{G}_1^* \times \mathbb{G}_2^* \times \mathbb{Z}_p^3$, s.t. $Z(\chi) \neq 0$, $\gamma \neq 0$. Set $\mathsf{ck} \leftarrow (\mathsf{gk}; (g_1, g_2^\gamma)^{\mathcal{F}_{\mathsf{C}}(\chi)})$, $\mathsf{crs}_p \leftarrow (\mathsf{gk}; (g_1, g_2^\delta)^{\mathcal{F}_{z-\mathsf{rsft}}(\chi)})$, $\mathsf{crs}_v \leftarrow (\mathsf{gk}; (g_1, g_2^\delta)^{Z(\chi)}, g_2^{\delta Z(\chi)Z^*(\chi)})$. Return $\mathsf{crs}_{\mathsf{rsft}} = (\mathsf{ck}, \mathsf{crs}_p, \mathsf{crs}_v)$.

Common input: $u_{\mathsf{rsft}} = ((A_1, A_2^\gamma), (B_1, B_2^\gamma))$.

Proving $\mathsf{P}_{\mathsf{rsft}}(\mathsf{crs}_p; u_{\mathsf{rsft}}; w_{\mathsf{rsft}} = (\vec{a}, \vec{b}), r_{\mathsf{rsft}} = (r_a, r_b))$**:** return $\pi_{\mathsf{rsft}} \leftarrow (\pi_1, \pi_2^\delta)$ from Eq. (5.3).

Verification $\mathsf{V}_{\mathsf{rsft}}(\mathsf{crs}_v; u_{\mathsf{rsft}}; \pi_{\mathsf{rsft}} = (\pi_1, \pi_2^\delta))$**:** accept if $\hat{e}(\pi_1, g_2^{\delta Z(\chi)}) = \hat{e}(g_1^{Z(\chi)}, \pi_2^\delta)$ and $\hat{e}(B_1\pi_1, g_2^{\delta Z(\chi)}) = \hat{e}(A_1, g_2^{\delta Z(\chi)Z^*(\chi)})$.

Since crs_v can be recomputed from $\mathsf{ck} \cup \mathsf{crs}_p$, then clearly it suffices to take CRS to be $\mathsf{crs}_{\mathsf{rsft}} = (\mathsf{gk}; g_1^{\mathcal{F}_{\mathsf{C}}(\chi) \cup \mathcal{F}_{z-\mathsf{rsft}}(\chi)}, g_2^{\gamma\mathcal{F}_{\mathsf{C}}(\chi) \cup \delta\mathcal{F}_{z-\mathsf{rsft}}(\chi)})$.

Theorem 3. *Let $Z^*(X) = Z(X)^2$, $y = \deg(Z(X)Z^*(X)) = 3n$. Π_{rsft} is perfectly complete and witness-indistinguishable. If the input consists of valid commitments, and BP is y-PDL, $(n, \mathcal{F}_{z-\mathsf{rsft}}, Y_2\mathcal{F}_{z-\mathsf{rsft}}, 1)$-PKE, and $(\mathcal{F}_{z-\mathsf{rsft}}, \mathcal{F}_{\mathsf{C}}, Y_1\mathcal{F}_{\mathsf{C}}, 2)$-PKE secure, then Π_{rsft} is an ($\Theta(n)$-bounded-auxiliary-input) adaptive argument of knowledge.*

The prover computation is dominated by two $(n + 2)$-wide multi-exponentiations (one in $\mathbb{G}_1$ and one in $\mathbb{G}_2$); there is no need for polynomial interpolation, multiplication or division. The communication is 2 group elements. The verifier computation is dominated by 4 pairings. In the full version [29], we describe a batch-verification technique that allows to speed up simultaneous

```
Let $\vec{b} \in \{0,1\}^n$ be such that $\sum_{i=1}^n S_i b_i = s$.
Let $(B_1, B_2^\gamma)$ be a commitment to $\vec{b}$.
Construct a product argument $\pi_1$ to show that $\vec{b} \circ \vec{b} = \vec{b}$.
Let $(C_1, C_2^\gamma)$ be a commitment to $\vec{c} \leftarrow \vec{S} \circ \vec{b}$.
Construct a product argument $\pi_2$ to show that $\vec{c} = \vec{S} \circ \vec{b}$.
Let $(D_1, D_2^\gamma)$ be a commitment to $\vec{d}$, where $d_i = \sum_{j \geq i} c_j$.
Construct a shift-right-by-1 argument $(\pi_{31}, \pi_{32}^\delta)$ to show that $\vec{d} = (\vec{d} - \vec{c}) \gg 1$.
Construct a product argument $\pi_4$ to show that $\vec{e}_1 \circ (\vec{d} - s\vec{e}_1) = \vec{0}_n$.
Output $\pi_{\mathsf{ssum}} = (B_1, B_2^\gamma, C_1, C_2^\gamma, D_1, D_2^\gamma, \pi_1, \pi_2, \pi_{31}, \pi_{32}^\delta, \pi_4)$.
```

Figure 5.1: The new SUBSET-SUM SNARK Π_{ssum} (prover's operations)

verification of several shift SNARKs. Apart from gk, the prover CRS and ck together contain $4n+6$ group elements, and the verifier CRS contains 3 group elements.

A shift-left-by-z (necessary in [28] to construct a permutation SNARK) SNARK can be constructed similarly. A rotation-left/right-by-z SNARK (one committed vector is a *rotation* of another committed vector) requires only small modifications, see [16].

5.6 New Subset-Sum SNARK

For fixed n and $p = n^{\omega(1)}$, the NP-complete language SUBSET-SUM over $\mathbb{Z}_p$ is defined as the language $\mathcal{L}_n^{\text{SUBSET-SUM}}$ of tuples $(\vec{S} = (S_1, \ldots, S_n), s)$, with $S_i, s \in \mathbb{Z}_p$, such that there exists a vector $\vec{b} \in \{0,1\}^n$ with $\sum_{i=1}^n S_i b_i = s$ in $\mathbb{Z}_p$. SUBSET-SUM can be solved in pseudo-polynomial time $O(pn)$ by using dynamic programming. In the current paper, since $n = \kappa^{o(1)}$ and $p = 2^{O(\kappa)}$, pn is not polynomial in the input size $n \log_2 p$.

In a SUBSET-SUM SNARK, the prover aims to convince the verifier that he knows how to open commitment (B_1, B_2^γ) to a vector $\vec{b} \in \{0,1\}^n$, such that $\sum_{i=1}^n S_i b_i = s$. We show that by using the new product and shift SNARKs, one can design a prover-efficient adaptive SUBSET-SUM zk-SNARK Π_{ssum}. We emphasize that SUBSET-SUM is just one of the languages for which we can construct an efficient zk-SNARK; Sect. 5.7 and the full version [29] have more examples.

First, we use the interpolating commitment scheme. The CRS generation $\mathsf{G}_{\mathsf{ssum}}$ invokes CRS generations of the commitment scheme, the product SNARK and the shift SNARK, sharing the same gk, g_1, g_2, γ, and trapdoor $\mathsf{td} = \chi$ between the different invocations. (Since here the argument must be zero knowledge, it needs a trapdoor.) Thus, $\mathsf{crs}_{\mathsf{ssum}} = \mathsf{crs}_{\mathsf{rsft}}$ for $z = 1$.

Let $\vec{e}_i$ be the ith unit vector. The prover's actions are depicted by Fig. 5.1 (a precise explanation of this SNARK will be given in the completeness proof in Thm. 4). This SNARK, even without taking into account the differences in the product and shift SNARKs, is both simpler and moth efficient than the SUBSET-SUM SNARK presented in [16] where one needed an additional step of proving that $\vec{b} \neq \vec{0}_n$.

We remark that the vector $\vec{d}$, with $d_i = \sum_{j \geq i} c_j$, is called either a *vector scan*, an *all-prefix-sums*, or a *prefix-sum* of $\vec{c}$, and $(\pi_{31}, \pi_{32}^\delta)$ can be thought of as a *scan SNARK* [16] that $\vec{d}$ is a correct scan of $\vec{c}$.

After receiving π_{ssum}, the verifier computes $S_1' \leftarrow \prod_i (g_1^{\ell_i(\chi)})^{S_i}$ as the first half of a commitment to $\vec{S}$, and then performs the following verifications: (i) Three commitment validations: $\hat{e}(B_1, g_2^\gamma) = \hat{e}(g_1, B_2^\gamma)$, $\hat{e}(C_1, g_2^\gamma) = \hat{e}(g_1, C_2^\gamma)$, $\hat{e}(D_1, g_2^\gamma) = \hat{e}(g_1, D_2^\gamma)$. (ii) Three product argument verifications: $\hat{e}(B_1/g_1, B_2^{\gamma}) = \hat{e}(\pi_1, g_2^{\gamma Z(\chi)})$, $\hat{e}(S_1', B_2^{\gamma}) = \hat{e}(g_1, C_2^{\gamma}) \cdot \hat{e}(\pi_2, g_2^{\gamma Z(\chi)})$, $\hat{e}(g_1^{\ell_1(\chi)}, D_2^{\gamma}/(g_2^{\gamma \ell_1(\chi)})^s) = \hat{e}(\pi_4, g_2^{\gamma Z(\chi)})$. (iii) One shift argument verification, consisting of two equality tests: $\hat{e}(\pi_{31}, g_2^{\delta Z(\chi)}) = \hat{e}(g_1^{Z(\chi)}, \pi_{32}^{\delta})$, $\hat{e}(D_1/C_1\pi_{31}, g_2^{\delta Z(\chi)}) = \hat{e}(D_1, g_2^{\delta Z(\chi) Z^*(\chi)})$.

Theorem 4. Π_{ssum} *is perfectly complete and perfectly composable zero-knowledge. It is an ($\Theta(n)$-bounded-auxiliary-input) adaptive argument of knowledge if* BP *satisfies n-TSDH and the same assumptions as in Thm. 3 (for $z = 1$).*

The prover computation is dominated by three commitments and the application of 3 product SNARKs and 1 shift SNARK, i.e., by $\Theta(n \log n)$ non-cryptographic operations and $\Theta(n)$ cryptographic operations. The latter is dominated by nine ($\approx n$)-wide multi-exponentiations (2 in commitments to $\vec{c}$ and $\vec{d}$ and in the shift argument, and 1 in each product argument), 7 in $\mathbb{G}_1$ and 4 in $\mathbb{G}_2$. The argument size is constant (11 group elements), and the verifier computation is dominated by *offline* computation of two $(n+1)$-wide multi-exponentiations (needed to once commit to $\vec{S}$) and *online* computation of 17 pairings (3 pairings to verify π_2, 2 pairings to verify each of the other product arguments, 4 pairings to verify the shift argument, and 6 pairings to verify the validity of 3 commitments). In the full version [29], we will describe a batch-verification technique that allows to speed up on-line part of the verification of the SUBSET-SUM SNARK.

As always, multi-exponentiation can be sped up by using algorithms from [32,35]; it can also be highly parallelized, potentially resulting in very fast parallel implementations of the zk-SNARK.

5.7 New Range SNARK

In a *range SNARK*, given public range $[L \mathinner{..} H]$, the prover aims to convince the verifier that he knows how to open commitment (A_1, A_2^{γ}) to a value $a \in [L \mathinner{..} H]$. That is, that the common input (A_1, A_2^{γ}) is a commitment to vector $\vec{a}$ with $a_1 = a$ and $a_i = 0$ for $i > 1$.

We first remark that instead of the range $[L \mathinner{..} H]$, one can consider the range $[0 \mathinner{..} H - L]$, and then use the homomorphic properties of the commitment scheme to add L to the committed value. Hence, we will just assume that the range is equal to $[0 \mathinner{..} H]$ for some $H \geq 1$. Moreover, the efficiency of the following SNARK depends on the range length.

The new range SNARK Π_{rng} is very similar to Π_{ssum}, except that one has to additionally commit to a value $a \in [0 \mathinner{..} H]$, use a specific sparse $\vec{S}$ with $S_i = \lfloor (H + 2^{i-1})/2^i \rfloor$ [11,30], and prove that $a = \sum_{i=1}^{n} S_i b_i$ for the committed a. Since $\vec{S} = (S_i)_{i=1}^{n}$ does not depend on the instance (i.e., on a), the verifier computation is $\Theta(1)$. On the other hand, since the commitment to a is given as an input to the prover (and not created by prover as part of the argument), Π_{rng} has a more complex simulation strategy, with one more element in the trapdoor.

Let $n = \lfloor \log_2 H \rfloor + 1$. Define $S_i = \lfloor (H + 2^{i-1})/2^i \rfloor$ for $i \in [1 \mathinner{..} n]$ and $\vec{S} = (S_i)$. We again use the interpolating commitment scheme. To prove that $a \in [0 \mathinner{..} H]$, we do the following.

The CRS generation $\mathsf{G}_{\mathsf{rng}}$ invokes the CRS generations of the commitment scheme, the product SNARK and the shift SNARK, sharing the same gk and trapdoor $\mathsf{td} = (\chi, \delta/\gamma)$ between the different invocations. In this case, the trapdoor has to include δ/γ (which is well defined, since $\gamma \neq 0$) since the simulator does not know how to open (A_1, A_2^{γ}); see the proof of Thm. 5 for more details. We note that the trapdoor only has to contain δ/γ, and not γ and δ separately. The CRS also contains the first half of a commitment $S_1' \leftarrow \prod (g_1^{\ell_i(\chi)})^{S_i}$ to $\vec{S}$, needed for a later efficient verification of the argument π_2. Clearly, the CRS can be computed efficiently from $\mathsf{crs}_{\mathsf{rsft}}$ (for $z = 1$).

The prover's actions on input (A_1, A_2^{γ}) are depicted by Fig. 5.2 (further explanations are given in the concise completeness proof in Thm.5). The only differences, compared to the prover computation of Π_{ssum}, are the computation of b_i on step 1, and of π_4 on step 2. After receiving

```
Let $a = \sum_{i=1}^{n} S_i b_i$ for $b_i \in \{0,1\}$.
Let $(B_1, B_2^\gamma)$ be a commitment to $\vec{b}$.
Construct a product argument $\pi_1$ to show that $\vec{b} = \vec{b} \circ \vec{b}$.
Let $(C_1, C_2^\gamma)$ be a commitment to $\vec{c} \leftarrow \vec{S} \circ \vec{b}$.
Construct a product argument $\pi_2$ to show that $\vec{c} = \vec{S} \circ \vec{b}$.
Let $(D_1, D_2^\gamma)$ be a commitment to $\vec{d}$, where $d_i = \sum_{j \geq i} c_i$.
Construct a shift argument $(\pi_{31}, \pi_{32}^\delta)$ to show that $\vec{d} = (\vec{d} - \vec{c}) \gg 1$.
Construct a product argument $\pi_4$ to show that $\vec{e}_1 \circ (\vec{d} - \vec{a}) = \vec{0}_n$.
Output $\pi_{\mathsf{rng}} = (B_1, B_2^\gamma, C_1, C_2^\gamma, D_1, D_2^\gamma, \pi_1, \pi_2, \pi_{31}, \pi_{32}^\delta, \pi_4)$.
```

Figure 5.2: The new range argument Π_{rng}

π_{rng}, the verifier performs the following checks: (i) Four commitment validations: $\hat{e}(A_1, g_2^\gamma) = \hat{e}(g_1, A_2^\gamma)$, $\hat{e}(B_1, g_2^\gamma) = \hat{e}(g_1, B_2^\gamma)$, $\hat{e}(C_1, g_2^\gamma) = \hat{e}(g_1, C_2^\gamma)$, $\hat{e}(D_1, g_2^\gamma) = \hat{e}(g_1, D_2^\gamma)$. (ii) Three product argument verifications: $\hat{e}(B_1/g_1, B_2^\gamma) = \hat{e}(\pi_1, g_2^{\gamma Z(\chi)})$, $\hat{e}(S_1', B_2^\gamma) = \hat{e}(g_1, C_2^\gamma) \cdot \hat{e}(\pi_2, g_2^{\gamma Z(\chi)})$, $\hat{e}(g_1^{\ell_1(\chi)}, D_2^\gamma / A_2^\gamma) = \hat{e}(\pi_4, g_2^{\gamma Z(\chi)})$. (iii) One shift argument verification, consisting of two equality tests: $\hat{e}(\pi_{31}, g_2^{\delta Z(\chi)}) = \hat{e}(g_1^{Z(\chi)}, \pi_{32}^\delta)$, $\hat{e}(D_1/C_1\pi_{31}, g_2^{\delta Z(\chi)}) = \hat{e}(D_1, g_2^{\delta Z(\chi) Z^*(\chi)})$.

Theorem 5. Π_{rng} *is perfectly complete and composable zero-knowledge. If* BP *satisfies n-TSDH and the assumptions of Thm. 3, then* Π_{rng} *is an adaptive ($\Theta(n)$-bounded-auxiliary-input) argument of knowledge.*

The prover computation is dominated by three commitments and the application of three product arguments and one shift argument, that is, by $\Theta(n \log n)$ non-cryptographic operations and $\Theta(n)$ cryptographic operations. The latter is dominated by nine ($\approx n$)-wide multi-exponentiations (2 in commitments to $\vec{c}$ and $\vec{d}$ and in the shift argument, and 1 in each product argument), seven in $\mathbb{G}_1$ and four in $\mathbb{G}_2$. The argument size is constant (11 group elements), and the verifier computation is dominated by 19 pairings (3 pairings to verify π_2, 2 pairings to verify each of the other product arguments, 4 pairings to verify the shift argument, and 8 pairings to verify the validity of 4 commitments). In this case, since the verifier does not have to commit to $\vec{S}$, the verifier computation is dominated by $\Theta(1)$ cryptographic operations.

The new range SNARK is significantly more computation-efficient for the prover than the previous range SNARKs [12, 16] that have prover computation $\Theta(r_3^{-1}(n) \log n)$. Π_{rng} has better communication (11 versus 31 group elements in [16]), and verification complexity (19 versus 65 pairings in [16]). Moreover, Π_{rng} is also simpler: since the prover computation is quasi-linear, we do not have to consider various trade-offs (though they are still available) between computation and communication as in [12, 16]. In the full version [29], we will use batch verification to further speed up the verification of the Range SNARK.

Bibliography

[1] Aranha, D.F., Barreto, S. L. M., Longa, P., Ricardini, J.E.: The Realm of the Pairings, In: SAC 2013. LNCS, vol. 8282, pp. 3–25

[2] Barreto, P.S.L.M., Naehrig, M.: Pairing-Friendly Elliptic Curves of Prime Order. In: SAC 2005. LNCS, vol. 3897, pp. 319–331

[3] Bellare, M., Garay, J.A., Rabin, T.: Batch Verification with Applications to Cryptography and Checking. In: LATIN 1998. LNCS, vol. 1380, pp. 170–191

[4] Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge. In: CRYPTO (2) 2013. LNCS, vol. 8043, pp. 90–108

[5] Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Scalable Zero Knowledge via Cycles of Elliptic Curves. In: CRYPTO (2) 2014. LNCS, vol. 8617, pp. 276–294

[6] Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. In: USENIX 2014, pp. 781–796

[7] Bitansky, N., Chiesa, A., Ishai, Y., Ostrovsky, R., Paneth, O.: Succinct Non-interactive Arguments via Linear Interactive Proofs. In: TCC 2013. LNCS, vol. 7785, pp. 315–333

[8] Boneh, D., Boyen, X.: Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. J. Cryptology **21**(2) (2008) pp. 149–177

[9] Bos, J.W., Costello, C., Naehrig, M.: Exponentiating in Pairing Groups. In: SAC 2013. LNCS, vol. 8282, pp. 438–455

[10] Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally Composable Two-Party and Multi-Party Secure Computation. In: STOC 2002, pp. 494–503

[11] Chaabouni, R., Lipmaa, H., shelat, a.: Additive Combinatorics and Discrete Logarithm Based Range Protocols. In: ACISP 2010. LNCS, vol. 6168, pp. 336–351

[12] Chaabouni, R., Lipmaa, H., Zhang, B.: A Non-Interactive Range Proof with Constant Communication. In: FC 2012. LNCS, vol. 7397, pp. 179–199

[13] Costello, C., Fournet, C., Howell, J., Kohlweiss, M., Kreuter, B., Naehrig, M., Parno, B., Zahur, S.: Geppetto: Versatile Verifiable Computation. In: IEEE SP 2015, pp. 253–270

[14] Danezis, G., Fournet, C., Groth, J., Kohlweiss, M.: Square Span Programs with Applications to Succinct NIZK Arguments. In: ASIACRYPT 2014 (1). LNCS, vol. 8873, pp. 532–550

[15] Fauzi, P., Lipmaa, H.: Efficient Culpably Sound NIZK Shuffle Argument without Random Oracles. In: CT-RSA 2016. LNCS, vol. 9610

[16] Fauzi, P., Lipmaa, H., Zhang, B.: Efficient Modular NIZK Arguments from Shift and Product. In: CANS 2013. LNCS, vol. 8257, pp. 92–121

[17] Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Series of Books in the Mathematical Sciences. W. H. Freeman (1979)

[18] Gathen, J., Gerhard, J.: Modern Computer Algebra. 2 edn. Cambridge University Press (2003)

[19] Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic Span Programs and NIZKs without PCPs. In: EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645

[20] Gentry, C., Wichs, D.: Separating Succinct Non-Interactive Arguments from All Falsifiable Assumptions. In: STOC 2011, pp. 99–108

[21] Groth, J.: Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In: ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340

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

[23] Kilian, J.: Uses of Randomness in Algorithms and Protocols. PhD thesis, Massachusetts Institute of Technology, USA (1989)

[24] Kolesnikov, V., Schneider, T.: A Practical Universal Circuit Construction and Secure Evaluation of Private Functions. In: FC 2008. LNCS, vol. 5143, pp. 83–97

[25] Lipmaa, H.: On Diophantine Complexity and Statistical Zero-Knowledge Arguments. In: ASIACRYPT 2003. LNCS, vol. 2894, pp. 398–415

[26] Lipmaa, H.: Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments. In: TCC 2012. LNCS, vol. 7194, pp. 169–189

[27] Lipmaa, H.: Succinct Non-Interactive Zero Knowledge Arguments from Span Programs and Linear Error-Correcting Codes. In: ASIACRYPT 2013 (1). LNCS, vol. 8269, pp. 41–60

[28] Lipmaa, H.: Efficient NIZK Arguments via Parallel Verification of Benes Networks. In: SCN 2014. LNCS, vol. 8642, pp. 416–434

[29] Lipmaa, H.: Prover-Efficient Commit-And-Prove Zero-Knowledge SNARKs. TR 2014/396, IACR (2014) Available at http://eprint.iacr.org/2014/396

[30] Lipmaa, H., Asokan, N., Niemi, V.: Secure Vickrey Auctions without Threshold Trust. In: FC 2002. LNCS, vol. 2357, pp. 87–101

[31] Parno, B., Gentry, C., Howell, J., Raykova, M.: Pinocchio: Nearly Practical Verifiable Computation. In: IEEE SP 2013, pp. 238–252

[32] Pippenger, N.: On the Evaluation of Powers and Monomials. SIAM J. Comput. **9**(2) (1980) pp. 230–250

[33] Raz, R.: Elusive Functions and Lower Bounds for Arithmetic Circuits. Theory of Computing **6**(1) (2010) pp. 135–177

[34] Sadeghi, A.R., Schneider, T.: Generalized Universal Circuits for Secure Evaluation of Private Functions with Application to Data Classification. In: ICISC 2008. LNCS, vol. 5461, pp. 336–353

[35] Straus, E.G.: Addition Chains of Vectors. Amer. Math. Monthly **70** (1964) pp. 806–808

[36] Valiant, L.G.: Universal Circuits (Preliminary Report). In: STOC 1976, pp. 196–203

6. Initial design options for mix-nets: Perfectly Anonymous Messaging via Secure Multiparty Computation

In this chapter, we present below ‘XYZ’, a design of anonymous messaging system that provides *perfect anonymity* and can scale in the order of hundreds of thousands of users. The main approach of the presented solution is to isolate two suitable *ideal functionalities*, called dialing and conversation, that when used in succession realize anonymous messaging. With this as a starting point, the proposed solution applies a secure multiparty computation (SMC) to instantiate them with *information theoretic* security in the semi-honest model. The use of a parallelization technique enables to scale to a large number of users, without sacryfying privacy. The presented solution can also provide a degree of forward security on the client side and can be instantiated in a variety of different ways with different SMC implementations overall, illustrating how SMC is a competitive alternative to traditional mix-nets and DC-nets for anonymous communication serving a new design option for WP4 and WP7 of PANORAMIX project.

6.1 Introduction

In an era in which privacy in communications is becoming increasingly important, it is often the case that two parties want to communicate anonymously, that is they want to exchange messages while hiding the very fact that they are in conversation. A major problem in this setting is hiding the communication metadata: while existing cryptographic techniques (e.g., secure point-to-point channels implemented with TLS) are sufficiently well developed to hide the communication content, they are not intended for hiding the metadata of the communication such as its length, its directionality, and the identities of the communicating end points. Metadata are particularly important, arguably some times as important to protect as the communication content. The importance of metadata is reflected in General Michael Hayden’s quote “We kill people based on metadata”[1] and in the persistence of security agencies with programs like PRISM (by the NSA) and TEMPORA (by the GCHQ) in collecting metadata for storage and mining.

Anonymous communication has been pioneered in the work of Chaum, with mix-nets [8] and DC-nets [7] providing the first solutions to the problem of sender-anonymous communication. In particular, a mix-net enables the delivery of a set of messages from n senders to a recipient so that the recipient is incapable of mapping messages to their respective senders. A DC-net on the other hand, allows n parties to implement an anonymous broadcast channel so that any one of them can use it to broadcast a message to the set of parties without any participant being able to distinguish the source. While initially posed as theoretical constructs, these works have evolved to actual systems that have been implemented and tested, for instance in the

[1]Complete quote: “We kill people based on metadata. But that’s not what we do with this metadata.” General M. Hayden. The Johns Hopkins Foreign Affairs Symposium. 1/4/2014.

case of Mixminion [13], that applies the mix-net concept to e-mail, in the case of Vuvuzela [28] that applies the mix-nets concept to messaging and in the case of Dissent [29] that implements DC-nets in a client-server model.

It is important to emphasize that the adversarial setting that we wish to protect against is a model where the adversary has a *global view* of the network, akin say to what a global eavesdropper would have if they were passively observing the Internet backbone, rather than a localized view that a specific server or sub-network may have. Furthermore, the adversary may manipulate messages as they are transmitted and received from users as well as block users adaptively. Note that in a more "localized" adversary setting one may apply concepts like Onion routing [27], e.g., as implemented in the Tor system [15], or Freenet [10] to obtain a reasonable level of anonymity with very low latency. Unfortunately such systems are susceptible to traffic analysis, see e.g., [20], and thus they cannot withstand a global adversary.

Given the complexity of the anonymous communication problem in general, we focus our application objective to the important special case of *anonymous messaging*, i.e., bidirectional communication with moderately low latency that has small payloads. The question we ask is whether it is possible to achieve it with *perfect privacy* while scaling to *hundreds of thousands* of users. In particular we consider two types of entities in our problem specification, clients and servers, and we ask how is it possible that the servers assist the clients that are online to communicate privately without leaking *any* type of metadata to a global adversary, apart that they are using the system. Furthermore, we seek a decentralized solution, specifically one that no single entity in the system can break the privacy of the clients even if it is compromised. We allow the adversary to completely control the network as well as a subset of the servers and adaptively drop clients' messages or manipulate them as it wishes.

Our Results We present "XYZ", an anonymous private messaging service that supports perfect privacy, under a well specified set of assumptions, and can scale to hundreds of thousands of users. In our solution we adopt a different strategy compared to previous approaches to anonymous communication. Specifically, we provide a way to cast the problem of anonymous messaging natively in the setting of secure multiparty computation (SMC). SMC, since its initial proposal, [17], is known to be able to distribute and compute securely any function, nevertheless, it is typically considered to be not particularly efficient for a large number of parties and thus inconsistent with problems like anonymous messaging. Nevertheless, the commodity-based approach [3] (client-server model), and more recent implementation efforts such as Fairplay [4], VIFF [12], Sharemind [6], PICCO [31], ObliVM [22] increasingly suggest otherwise.

We first propose two ideal functionalities that correspond to the dialing operation and the conversation operation. The XYZ system operation proceeds in distinct rounds, where in each round an invocation of either the dialing or the conversation ideal functionality is performed. The dialing functionality enables clients to either choose to dial another client or check whether anyone is trying to dial them (in practice in most rounds the overwhelming majority of clients will be in dial-checking mode). If a matching pair is determined by the ideal functionality, the caller will be notified that the other client has accepted their call and the callee will be notified about the caller. Moreover, the ideal functionality will deliver to both clients a random tag (that can be thought of the equivalent of a "dead drop" or "rendezvous" point). Subsequently the clients can access the conversation functionality using the established random tag. When two clients use the same random tag in the conversation functionality, their messages are swapped and thus they can send messages to each other (even concurrently).

The two ideal functionalities provide a useful abstraction of the anonymous messaging problem. We proceed now to describe how they can be implemented by an SMC system. It is easy to see that a straightforward implementation of the functionality programs will result in a circuit of size $\Theta(n^2)$ where n is the number of online users accessing the functionalities. Such a solution would be clearly not scalable. We provide more efficient implementations that achieve

$O(n \log n)$ circuit complexity in both cases with very efficient constants using state of the art oblivious sorting algorithms.

Given our high level functionality realizations we proceed to an explicit implementation in the Sharemind system, [6]. We provide code in the Qt platform of Sharemind and explicit benchmarks for the Dialing and Conversation solutions. The Sharemind platform provides a 3-server implementation of information theoretically secure SMC. Our results benchmark for thousands of users in a reasonable latency (little over a minute) that is consistent with messaging.

In order to increase our performance and scale to the order of hundreds of thousands of users we provide a parallelized implementation of the conversation functionality that maintains perfect privacy. Parallelization is a non-trivial problem in our setting since we would like to maintain perfect privacy across the whole user set; thus, a simplistic approach that breaks users into chunks solving dialing and conversation independently will isolate them to smaller "communication islands"; if two users have to be on the same island in order to communicate, this will lead to privacy loss that we would like to avoid. Our parallelized solution manages to make completely oblivious the interaction between islands essentially providing the same level of security as the single SMC instance solution. In this way, by utilizing a large number of servers we are able to scale the system hundreds of thousands of users, cf. Figure **??**. Beyond the enhanced level of privacy that our approach provides (perfect privacy assuming an honest majority among the servers realizing the two functionalities) our system has the unique characteristic that it is highly extensible to incorporate policies for *spam and malware prevention* that are expressed as regular expressions. This is another feature that stems from our SMC approach that distinguishes our solution from previous solutions based on DC-nets or mix-nets (where it is hard to process the transmitted information through a regular expression filter). Finally, our system also provides *forward secrecy*, in the sense that if any client or server is compromised it will be impossible to decrypt previous communication contents or metadata.

Related Work in Anonymous Messaging Our work is most closely related to the Vuvuzela system [28] that uses mixnets and addition of fake messages as noise to achieve a differentially private (cf. [16]) solution to anonymous messaging. Expectedly, differential privacy provides a guarantee that is weaker than perfect privacy. In the context of anonymous messaging differential privacy provides a bound that any observation strategy of the attacker is subject to, when trying to distinguish between two possible user actions (e.g., dialing or dial-checking) while every other entity is stable in its operation. The Vuvuzela system uses mix-nets to facilitate the dialing and conversation operations something that results in leakage. As it is demonstrated, this leakage can be controlled with the addition of fake messages by each server in order to obscure the real number of messages exchanged. Further comparison to Vuvuzela, especially its dialing protocol is provided in Table 6.1. Another related system is Riposte, [11] which uses DC-nets and SMC to implement a distributed database that users can anonymously write and read from. Specifically, they implement the write stage on the database as a "reverse" private information retrieval (PIR, [9]) where the client spreads suitable information for writing in the database. Subsequently, when used for messaging, users can read using PIR from the position in the database that the sender wrote the message (which can be a random position calculated from key information available to the users). In the end, Riposte can scale to millions of users but it requires many hours to perform a complete operation; a significant bottleneck is the write-operation that requires $O(\sqrt{L})$ client communication for an L-long database (where L should be proportional to the number of users in order to handle collisions between write requests in the setting of messaging). In contrast, in our system, client bandwidth is independent from the number of users. Other related schemes like Herbivore, [23] Dissent [29] and Pynchon Gate [24] use much smaller anonymity sets than Vuvuzela and Riposte because they scale essentially quadratically in the number of active users.

Organization After shortly presenting some preliminary topics in section 6.2, we present the two ideal functionalities, Dialing and Conversation, that together solve the anonymous messaging problem (section 6.3). In sections 6.4 and 6.5 we will propose a way to implement the aforementioned functionalities in a secure and privacy-preserving way, using secure multiparty computation. In section 6.6, we introduce a novel way to parallelize our protocols in order to achieve even better performance. Finally, in section 6.7, we combine the results of the previous sections and describe the architecture of a system that enables users to communicate anonymously.

6.2 Preliminaries

Secure Multiparty computation Secure Multiparty Computation (SMC or MPC), is an area of cryptography concerned with methods and protocols that enable a set of users $u_1, \ldots, u_n$ with private data $d_1, \ldots, d_n$ to compute the result of a public function $F(d_1, \ldots, d_n)$, without revealing their private inputs. Secure computation was formally introduced as secure two-party computation (2PC) in 1982 by Andrew Yao, [30] and was soon expanded to the multi-party setting. There exist several generic SMC constructions that receive as input a description of an algorithm (in some form) and the distribution of the inputs among the data owners and produce as output the description of a secure protocol that implements the algorithm in a privacy-preserving manner. In most cases, some form of secret sharing of the inputs, such as additive or Shamir secret sharing [25] is used and the protocol proceeds to produce a sharing of the output. Most known SMC frameworks, such as Fairplay [4], VIFF [12], Sharemind [6] etc. need the function as a circuit, either made up of boolean gates or as an arithmetic circuit over a sufficiently large field $GF(p)$. This is a highly non-trivial matter as most useful functions use loops or recursion. Generally, each implementation follows either the Yao paradigm, the GMW [17] or some combination of those. The main difference of these two approaches is that Yao's approach requires the generation of a garbled circuit and the evaluation of the function on it, whereas the GMW approach requires communication during the evaluation of any multiplication (or binary AND) gate.

Our work will be presented in a manner that makes it easy to implement using any of the aforementioned protocols and therefore we will not elaborate further on them. As a general idea, clients will break their input into shares and forward each share to a server. Then, the servers will interactively compute the desired output shares, which in turn will be returned to the respective clients. More specifically, each client will be allocated a virtual wire with a specific wire id that will be the same in all servers and her input and output will be transferred by this wire.

Oblivious sorting Sorting is used as a vital part of many algorithms. In the context of secure multiparty computation, sorting an array of values without revealing their final position, is called oblivious sorting.

The first approach to sorting obliviously is using a data-independent algorithm and performing each compare and exchange execution obliviously. This approach uses sorting networks to perform oblivious sorting. Sorting networks are circuits that solve the sorting problem on any set with an order relation. Sorting networks are devices built up only of wires carrying values and comparator modules that connect pairs of wires and that swap these values if they are not in the desired order (according to a given order relation). What sets sorting networks apart from general comparison sorts is that their sequence of comparisons is set in advance, regardless of the outcome of previous comparisons. Various algorithms exist to construct simple and efficient networks of depth $\mathcal{O}(log^2 n)$ and size $\mathcal{O}(nlog^2 n)$. The three more used ones are Batcher's odd-even mergesort and bitonic sort [2] and Shellsort [26]. All three of these networks are simple in principle and efficient. Sorting networks that achieve the theoretically optimal $\mathcal{O}(logn)$ and

$\mathcal{O}(nlogn)$ complexity in depth and total number of comparisons, such as the AKS-network [1] exist, but the constants involved are so large that make them impractical for use. Note that even for 1 billion values, i.e., $n = 10^9$, it holds that $\log n < 30$ so, in practice, the extra log factor is preferable to the large constants. A major drawback of all sorting network approaches is that sorting a matrix by one of its columns would require oblivious exchange operations of complete matrix rows, which would be very expensive.

In recent years techniques have been proposed from Hamada et. al [19] to use well known data-dependent algorithms such as quicksort or radix sort in an oblivious manner to achieve very efficient implementations, especially when considering a small number of SMC servers, which is very often the case. This approach uses the "shuffling before sorting" idea, which means that if a vector has already been randomly permuted, information leaked about the outcome of comparisons does not leak information about the initial and final position of any element of the vector. More specifically, the variant of quicksort proposed in [19], needs on average $\mathcal{O}(\log n)$ rounds and a total of $\mathcal{O}(n \log n)$ oblivious comparisons. Complete privacy is guaranteed when the input vector contains no equal sorting keys, and in the case of equal keys, their number may leak. Furthermore, performance of the algorithm is data-dependent and generally depends on the number of equal elements, with the optimal case being that no equal pairs exist. Practical results have shown [5] that this quicksort variant is the most efficient oblivious sorting algorithm available, when the input keys are constructed in a way that makes them unique.

Another algorithm following the ideas above, is Hamada's oblivious radix sort [18]. This variant of radix sort is not based on comparisons and it is very efficient, when considering a rather small fixed number of servers and a reasonable fixed size of the sorting keys (e.g. 32 or 64 bits). In this setting, the algorithm has a round complexity of $\mathcal{O}(1)$ and a total communication complexity of $\mathcal{O}(n \log n)$. Furthermore, its running time is data-independent and it can also handle vectors with equal values without leakage. Practical results have shown that this algorithm is the optimal solution when dealing with inputs which may have equal elements.

To sum up, we have briefly introduced three approaches to oblivious sorting. Sorting networks are inherently data oblivious but their performance is not practical when dealing with a large number of inputs and/or with large amount of data to be sorted according to a key. The other two approaches use shuffling techniques before sorting and are able to produce practically interesting results. More specifically, in our algorithms we will use oblivious radix sort when we care about leaking the number of equal elements, and quicksort when elements are guaranteed to be distinct, or when leaking the number of equals can be tolerated.

Sharemind Sharemind [6] is a secure multiparty computation framework that offers a higher level representation of the circuit being computed in the form of a program written in a C-like language, namely the SecreC language. It uses three-server protocols that offer security in the presence of an honest server majority. That is, we assume that no two servers will collude in order to brake the systems privacy. In the presentation of our solution we will use Sharemind as a prototype building platform but that does not mean that our solution cannot be easily customized to be compatible with any multiparty computation framework. Sharemind offers many built-in functions that make programming of privacy-preserving software easier. Of special interest are the oblivious sorting and oblivious choice methods implemented. Considering that our approach relies heavily on those two operations, Sharemind enabled us to produce working code easily in order to emulate and test our proposals.

6.3 Ideal anonymous messaging

We will present a solution in the form of two ideal functionalities that together solve the problem of anonymous communication. An ideal functionality is a protocol run by a trusted third party that computes the desired result. Our solution will make use of the idea of rendezvous points, inspired by Vuvuzela and encompass two distinct functionalities. The Dialing functionality, which consists of the computation of a rendezvous point for a given pair of users who want to communicate, and the Conversation functionality which represents the actual exchange of messages. It is important to note that we have made the assumption that a user who wants to dial another user knows the said user's public key. This assumption is non trivial, but the problem of an anonymous public key infrastructure is out of the scope of this paper. The two ideal functionalities are presented in figures **??** and **??**, respectively.

Dialing Functionality $\mathcal{F}_{\mathsf{DIAL}}$

Running with a set of users $D = \{u_1, \ldots, u_n\}$ and an ideal adversary $\mathcal{A}$, proceeds as follows:

– Upon receiving:

  – (DIAL, u_i, u_j) requests from k users, each originating from user u_i,

  – $(\text{DIALCHECK}, u_j)$ requests from $n-k$ users, each originating from user u_j,

  compute a random value t_{u_i,u_j} if two requests of the form $(\text{DIALCHECK}, u_j)$ and (DIAL, u_i, u_j) have been received and forward it to both users u_i, u_j. If more DIAL requests have been received that match the same DIALCHECK request, any of them may be chosen by $\mathcal{A}$. If no DIAL requests have been received for a DIALCHECK, return void.

Figure 6.1: The ideal functionality $\mathcal{F}_{\mathsf{DIAL}}$.

Conversation Functionality $\mathcal{F}_{\mathsf{CONV}}$

Running with a set of users $D = \{u_1, \ldots, u_n\}$ and an ideal adversary $\mathcal{A}$, proceeds as follows:

– Upon receiving:

  (a) (CONV, t_i, y_i) from all parties $u_i \in D$ (some may be controlled by $\mathcal{A}$),

  (b) a list $L = \{u_k, \cdots, u_l\}$ of blocked users from $\mathcal{A}$,

  compute the permutation π (as defined in section 6.3) over the unblocked users and send message y_{π_i} to user i.

Figure 6.2: The ideal functionality $\mathcal{F}_{\mathsf{CONV}}$.

Explaining the symbols:

- y_i : the message that some user u_i sends to a user u_j, encrypted with a key known to u_j.

- t_i : A string that stands for the rendezvous point. It contains information about the receiver of the message and is a value that will be shared by the two communicating parties.

- $\mathcal{A}$: The adversary has full control of the network and the ability to corrupt all but the two users communicating and some of the servers.

- π : The permutation π used in figure **??** is defined based on tuples of the form (t_i, y_i) as shown below :

– if $t_i = t_j \Rightarrow \begin{cases} \pi(i) = j \\ \pi(j) = i \end{cases}$

– else if $\forall j \neq i : t_i \neq t_j \Rightarrow \pi(i) = i$

If more than two tuples have the same t value, then let the ideal adversary $\mathcal{A}$ choose how they are going to be paired. Intuitively π represents the exchange of messages.

The Dialing functionality aims at computing a shared random value between two users that want to communicate. When a user u_i wants to start a conversation session with another user u_j, she sends a dial request and the functionality generates a random shared value. This shared value is then used by the Conversation functionality to match users who want to exchange messages and to facilitate this exchange. Specifically, two message requests that have the same t value are matched together and their contents are swapped. The use of a random rendezvous point in the establishment of a communication channel between two users averts any denial of service attacks targeting specific users by other users at the conversation phase. This ideal functionality serves as a general idea of what we are trying to achieve with more specific details coming along with the respective implementations in the next sections.

The remainder of this paper focuses on achieving the functionalities described above in a distributed and secure way. As a general design, we are going to implement two protocols, the Dialing and the Conversation protocol, using secure multiparty computation techniques to securely evaluate the corresponding functions with the presence of a number of servers (3 in our implementation), assuming an honest server majority (server number count can vary depending on the implementation framework). Clients will divide their input into shares and forward each one to a server using a secure channel. The servers will proceed to produce the desired output shares and then return these to the respective clients in order for them to be reproduced.

6.4 Implementing the Dialing functionality

6.4.1 Dialing Protocol

The Dialing or Dial-Dialcheck protocol will enable clients to notify others that they want to start a conversation, assuming they know the other party's public key, much like how the telephone protocol works. The protocol will work in rounds to deter possible timing attacks and in each round each online client will either send a Dial request or a Dialcheck request, which will be indiscriminate from each other. The protocol given will implement the ideal functionality $\mathcal{F}_{\mathsf{DIAL}}$. First we will present an intermediate representation describing the functionality in a mathematical manner and then we will proceed with an efficient algorithm implementing it. Below we will use the character "C" as a label to denote a DIALCHEK.

Dialing Functionality round r intermediate representation

Input: a sequence of n tuples $\langle a_1, \ldots, a_n \rangle =$
$\langle (i_1, j_1), (i_2, j_2), \ldots, (i_n, j_n) \rangle$
and a sequence of public keys $\langle k_1, \ldots, k_n \rangle$

Output: a sequence of size n, $\langle b_1, \ldots, b_n \rangle$ returning dialers' pk's to DIALCHECK requests (or zero)

```
For each $i \leftarrow 1, \ldots, n$
if $a_i.1 \neq k_i$ AND $a_i.2 \neq k_i$ then
    $a_i.1 = 0$
    $a_i.2 = 0$
```

```
end if
For each $i = 1, \dots, n$
if $a_i.1 = C$ AND $\exists j \in \{1, \dots, n\} : (a_j.2 = a_i.2)$ then
    $b_i \leftarrow a_j.1$
else
    $b_i \leftarrow 0$
end if
return  sequence $b$
Comment: each user computes the shared value $t_{u_i,u_j}$ as shown later in this section
```

Dialers input a tuple of the form (i, j) where i and j are the public keys of the dialer and the dialee respectively. Dial checkers input a tuple of the form (C, j), where C is a special value designated to show a dial check and different from any possible id/public key value, and j is the checker's own pk. Additionally, a list of the user's public keys is provided as input by an untrusted third party (more on this on later sections) with public key k_i belonging to the submitter of tuple a_i. As a first step, the protocol checks if any of the first two members of each tuple (namely $a_i.1$ and $a_i.2$) is equal to the submitter's public key. This check serves two purposes. First, it averts impersonation attacks, where a user might pose as another user to get access to dialing requests destined for the latter one. In a tuple of the form (C, j_i), which signifies a dial check, if the second member of the tuple is not the submitter's public key, then the dial check is discarded. The second use of this check is that any denial of service attack that floods a specific user with dials, in order to avert him from collecting the genuine ones, cannot be made anonymously. In the case of a dial to a user other than one's self, the first member of the dial tuple is guaranteed to be the submitter's own public key and thus the source of a dos attack cannot be hidden. How the list of the submitters' public keys is generated and guaranteed to be correct will be discussed in section 6.7 where we talk about the general architecture of the system.

At the end, the protocol produces meaningful output only for dial checkers who have a dial request by another user. This output is the pk of the user who dialed them. All other outputs are meaningless and could be zero or have another special value. It has to be noted that a dial checker can have multiple incoming dial requests. In this protocol it is not specified which request will actually come through, but that one of them will.

After having received (or having sent) a dial request from (to) a user u_j, user u_i can calculate the shared rendezvous point for each (conversation) round r as follows:

$$t_i = H(s_{u_i,u_j}, r)$$
$$\text{where } s_{u_i,u_j} = DH(pk_{u_i}, pk_{u_j})$$

where H is a standard cryptographic hash function, r is the round number, pk_{u_i} is the public key of user u_i, and DH marks a (non interactive) Diffie-Hellman key exchange operation. We emphasize the fact that this t value is at least 64 bits long. If user u_i doesn't want to communicate, but wants to protect her privacy she computes a rendezvous point as above with $pk_{u_j} =$ rand and sends a zero or random message. In this case, the message returned is the message she sent. Here it has to be noted that the random rendezvous point calculation can be considered to be somewhere in the middle between the Dialing and Conversation functionalities as the seed of the pseudorandom hash function is generated from the first functionality and the specific rendezvous point for each round from the second. The algorithm realizing the Dialing protocol is presented next.

Algorithm 1 describes an implementation of functionality $\mathcal{F}_{\mathsf{DIAL}}$in a manner suitable for secure multiparty computation. More precisely, inputs are considered wires bearing a wire id (wid). First checking is performed for each input tuple, in a possibly parallel manner, to exclude rogue requests by setting them to zero. Sorting is then performed using the oblivious

Algorithm 1 Dialing round r

Input: a sequence of n tuples $\langle a_1, \dots, a_n \rangle = \langle (i_1, j_1, wid_1), (i_2, j_2, wid_2), \dots, (i_n, j_n, wid_n) \rangle$ along with a sequence of n public keys $\langle k_1, \dots, k_n \rangle$

Output: a sequence of size n, $\langle b_1, \dots, b_n \rangle$ returning dialers' pk's to check requests (or zero)

```
1. For each $i \leftarrow 1, \dots, n$
if $a_i.1 \neq k_i$ AND $a_i.2 \neq k_i$ then
    $a_i.1 \leftarrow 0$
    $a_i.2 \leftarrow 0$
end if
2. Sort tuples $\langle a_1, \dots, a_n \rangle$ according to second coordinate using oblivious radix sort.
3. For each $i \leftarrow 1, \dots, n$
if $a_i.1 = C$ AND $a_i.2 = a_{i-1}.2$ then
    $b'_i \leftarrow (a_{i-1}.1, a_i.3)$
else if $a_i.1 = C$ AND $a_i.2 = a_{i+1}.2$ then
    $b'_i \leftarrow (a_{i+1}.1, a_i.3)$
else
    $b'_i \leftarrow (0, a_i.3)$
end if
4. Sort tuples $\langle b'_1, \dots, b'_n \rangle$ according to second coordinate using quicksort, then ignore second coordinate and produce sequence $\langle b_1, \dots, b_n \rangle = \langle b'_1.1, \dots, b'_n.1 \rangle$
return sequence $b$
Post-processing: Each client calculates rendezvous point $t_i$ for each round $r, r+1, \dots$ she needs.
```

radix sort algorithm of [18] to sort the tuples according to their second coordinate, which in the case of a Dial is the recipient's pk and in the case of a Dialcheck is a checker's own pk. In reality, due to the fact that radix sort scales linearly to the length of the sorting key, we need to use an alternative to the public key specified in the algorithm. As this value will not be used for encryption, but only for identification purposes, it can be a short username acquired be a client when entering the system and agreed upon by all the involved servers. Another more straightforward option would be to use a public key fingerprint (e.g. 128 bit MD5). Then, requests are processed individually by looking at both of their neighbours to determine if there is a Dial for any given Dialcheck request. Of course, requests at the first and last place of the sorted vector need only look at one neighbour. Sorting enables us to claim that any Dialcheck request of a user will have a suitable Dial request as its neighbour or not at all. After checking and producing an intermediate result b', the algorithm needs to sort the requests according to their wire id's in order for the correct requests to be forwarded to each user. The latter sort, performed according to the wire id's of the requests can be implemented by the quicksort algorithm of [19], as the key values, that is the wire id's, are guaranteed to be distinct.

At the end, each user who submitted a valid dial check request gets one if any of the public keys of possible users that dialed him in that given round. On the other hand, dialers get a dummy output in order to be indiscriminable from dial checkers.

From everything presented in this section we can conclude that:

Theorem 1 *Algorithm 1 implements the ideal functionality $\mathcal{F}_{\mathsf{DIAL}}$ of figure* **??**.

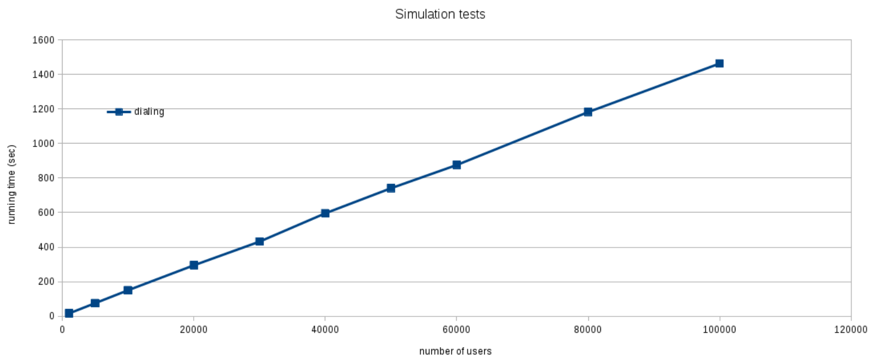


Figure 6.3: Dialing simulation results

6.4.2 Performance prediction of Dialing protocol

We have implemented the Dialing protocol by running a SecreC program on a local 1 Gbps LAN cluster with 12-core 3 GHz Hyper-Threading CPU and 48 GB of RAM operated by the Sharemind team. We used the offered implementations for the radix sort and the quicksort algorithms. Our simulation results are presented in figure **??**. As we expected, running times scale nearly linearly, according to the $\mathcal{O}(nlogn)$ cost of our sorting algorithms. Our protocol can serve 20.000 users with latency around 5 minutes and 40.000 users with latency around 10 minutes. These figures may appear large, but dialing need not be performed in very short time intervals.

6.4.3 Comparison with previous solutions

An interesting comparison would be that of our Dialing protocol with the one presented in Vuvuzela [28]. Vuvuzela uses an approach where all users submit dial requests, some of them dummy, which consist of the sender's public key encrypted by the recipient's public key. Then, all requests are mixed by a decryption mixnet consisting of three servers and real ones are partitioned in big batches according to the subset of users they are intended for. Then each user has to download the entire batch, which could be in the order of a few megabytes large, and check if she can decrypt any of the said requests. This of course results in increased bandwidth needs for both the server and the clients and additionally quite substantial computational burden on the client's side.

As our two protocols are independent, our Dialing protocol could be used in conjunction with another system that uses shared values to exchange messages, such as Vuvuzela. As it is now it could accommodate up to 50.000 clients with respectable total latency and thus substitute Vuvuzela's Dialing protocol, in this range of client populations. A comparison of our Dialing protocol to that of Vuvuzela can be found in table 6.1. Further scaling is possible using a parallelization technique that is presented in section 6.6.

Our Dialing protocol is very efficient in terms of both bandwidth needs for the server and the client, and computational need on the client's side. This is because it follows a point-to-point approach that returns only one message to each client. Furthermore, security guarantees of our protocol are these of the secure multiparty framework we use and in the case of Sharemind it is cryptographic security with an honest server majority, compared with the differential privacy offered by Vuvuzela. Concerning forward secrecy, our protocol can be made to offer a certain degree by using ideas presented in section 6.7.2. Finally, when comparing according to total latency and scalability, that is including the time allocated by the Vuvuzela client protocol for the clients to download the requests, our protocol is slightly inferior to that of Vuvuzela. As a final note on scalability, our Dialing protocol could be parallelized to run on multiple SMC

Table 6.1: Comparison of Dialing protocols (n: number of users)

* latency can be decreased by parallelizing the Dialing protocol (see section 6.6)

	this paper	Vuvuzela
client computation	1 op	$\mathcal{O}(n)$ ops
client bandwidth	1 request	$\mathcal{O}(n)$ requests
server bandwidth	n requests	$\mathcal{O}(n^2)$ requests
privacy	cryptographic	differential
forward secrecy	possible	no
latency	medium*	low
honest servers	majority	1

systems in order to achieve much better performance, as presented in section 6.6.5.

6.5 Implementing the Conversation functionality

6.5.1 Conversation protocol

We will begin describing our Conversation protocol, which facilitates message exchange, by presenting a mathematical intermediate step towards our algorithm. At this point, we have to highlight our assumption that a valid message at the input has its least significant bit equal to 0. This flag which could also be a discrete fourth member of our tuple, is useful at the parallelization of our protocol presented in section 6.6.

Conversation functionality round r intermediate representation

Preliminary: each user computes a rendezvous point t value for round r
Input: a sequence of n tuples $\langle a_1, \dots, a_n \rangle =$
$\langle (t_1, m_1), (t_2, m_2), \dots, (t_n, m_n) \rangle$
Output: a sequence of size n, $\langle b_1, \dots, b_n \rangle$ carrying messages to their intended recipients

```
For each $i \leftarrow 1, \dots, n$
if $\exists j \in \{1, \dots, n\} : a_j.1 = a_i.1$ then
  $b_i \leftarrow a_j.2 + 1$
else
  $b_i \leftarrow a_i.2$
end if
return sequence $b$
```

The Conversation protocol, as described in functionality $\mathcal{F}_{\textsf{CONV}}$, works in rounds and facilitates the exchange of messages having the same t value. This value represents the rendezvous point computed by the two communicating parties (users u_i and u_j) at the final part of the Dialing protocol. It is expected that no more than two messages will have the same t value due to its large bit-size. We point out that when a message is exchanged its LSB is set to 1. Now let's proceed to the algorithmic implementation of our protocol.

As can be seen from algorithm 2, the Conversation protocol is implemented similarly to the Dialing protocol. The input tuples are sorted by their rendezvous points (t) and then messages are exchanged between neighbouring elements, when their t values match. Sorting guarantees that requests with the same t value will reside in neighbouring indexes of the sorted

Algorithm 2 Conversation round r

Input: a sequence of n tuples $\langle a_1,\dots,a_n\rangle = \langle (t_1, m_1, wid_1), (t_2, m_2, wid_2), \dots, (t_n, m_n, wid_n)\rangle$

Output: a sequence of size n, $\langle b_1,\dots,b_n\rangle$ carrying messages to their intended recipients

```
1. Sort tuples $\langle a_1,\dots,a_n\rangle$ according to first coordinate $(a_i.1)$ using oblivious radix sort.
2. For each $i \leftarrow 1,\dots,n-1$
if $a_i.1 = a_{i+1}.1$ then
    $b'_i \leftarrow (a_{i+1}.1, a_i.3)$
    $b'_{i+1} \leftarrow (a_i.1, a_{i+1}.3)$
end if
3. Sort tuples $\langle b'_1,\dots,b'_n\rangle$ according to second coordinate using quicksort, then ignore second
coordinate and produce sequence $\langle b_1,\dots,b_n\rangle = \langle b'_1.1,\dots,b'_n.1\rangle$
return sequence $b$
```

vector. Furthermore, the random nature of the rendezvous points, along with their relatively high bit length make it highly improbable that there will be a conflict in the t value. In this case, a conversation may not take place as intended and it is up to the client to handle this highly improbable case. We note here that if messages are end-to-end encrypted between the conversing parties, then a collision will only result in a dropped message that will be detected and re-transmitted by the client.

From everything presented in this section, we can conclude that:

Theorem 2 *Algorithm 2 implements the ideal functionality $\mathcal{F}_{\mathsf{CONV}}$ of figure* **??**.

6.5.2 Spam / Malware detection

It is easy to see that our approach is directly extensible to incorporate any plaintext processing operation that can be described as a boolean circuit on the input message. Specifically, given, say a regular expression filter that checks for spam or malware based on e.g., signatures, we can extract a short circuit description for fixed input length as long as the message size and incorporate it as part of the SMC implementation. More specifically, the construct of Laud et al. in [21] enables the oblivious evaluation of any DFA with the cost of only one multiplication per input character when the DFA is publicly known.

6.6 Parallelizing our protocols

6.6.1 Introduction

As discussed in the previous sections, our protocols while satisfying very strong privacy guarantees, are by themselves not as scalable as desired to serve hundreds of thousands of users in a real-time manner. Therefore, we would like to also propose a way to combine a number of such protocols in a way that will lead to a more scalable system. Against the trend that sacrifices privacy in order to gain scalability, we want to maintain the strict privacy guarantees of our system. Therefore, in our novel parallelized approach we will relax our quality of service guarantees. That is, in each round, an adjustably small number of requests that would have been served when using algorithms 1 and 2 for the Dialing and the Conversation protocols respectively will fail to do so. The probability that some client request will not be served can be made arbitrarily small in the expense of performance, as described later in this section. As evident by the mathematical and algorithmic representation of our two protocols, in both cases, the integral part of our two functions is detecting duplicate key entries and performing an action on these pairs. In this section, we will introduce a parallelized oblivious duplicate detection approach for uniformly distributed key entries. Our approach will allow a margin for error which will be measured by the quality of service metric, introduced below:

Definition 1 $qos = \frac{duplicates_{found}}{duplicates_{total}}$

The above definition can straitforwardly be interpreted as the ratio of successful dials or message exchanges in the parallelized version of our protocols over their count in our initial non parallel approach. We leverage the fact that the pairing keys are uniformly distributed and partition requests among the servers based on the fact that equal keys are likely to be located at the same range of different arrays after sorting.

Our approach will be demonstrated by two algorithms, one for the Conversation and one for the Dialing protocols. In our examples, we will use two SMC server islands (e.g. Sharemind 3-server platforms) to explain how parallelization is achieved but the method can easily be applied to any number of such islands.

6.6.2 Parallelizing the Conversation protocol

In figure 6.2, we can see how we can combine two SMC islands, each one potentially consisting of 3 or more servers, to achieve better performance and more decentralization.

In our figure, we assume two SMC islands, and assign half of the incoming requests ($\frac{n}{2}$) of the form (t_i, m_i, wid_i) to each of them. For example, the first island gets requests from clients with wire id's $\{1, \dots, \frac{n}{2}\}$ and the second with wire id's $\{\frac{n}{2}+1, \dots, n\}$. Then, each island independently sorts its requests according to their t coordinate. As a next step, intuitively we want to send all low values of t to the first island and all high values of t to the second one. Thus, the first island keeps the lower half (plus $\delta\frac{n}{4}$) of its sorted requests and receives the lower half (plus $\delta\frac{n}{4}$) of the sorted requests of the second. The second island does the opposite, keeping the upper half of its requests and receiving the upper half of the first island's requests. Due to the fact that these t values are randomly generated, they are uniformly distributed and we expect identical values to generally fall in the same half. Practically, this transmission of data can be made on a peer to peer level between each of the servers of the two islands. That is, the first server of the first island will communicate through a secure channel with the first server of the second island etc. The additional requests ($d = \delta\frac{n}{2}$), apart from the halves assigned to eacj island, serve the purpose of calibrating the quality of service parameter of the mechanism. The bigger the value of d, the less likely is that two requests with the same t values will end up in different islands and communication will not take place.

As a next step, each island merges the two sorted lists according to their t values and performs the necessary exchange of messages as described in the Conversation algorithm. Then, after dropping the now useless t values, each island sorts its requests independently according to their wid coordinate. After this step, it is guaranteed that the first $\frac{n}{4} + \delta\frac{n}{4}$ requests of the first island originated from the first island and the rest $\frac{n}{4} + \delta\frac{n}{4}$ from the second. The same is true for the second island. At this point, the message exchange has been performed and the messages must reach their intended recipients, based on their wire id. Thus, each island sends each message request back to the island it originated from. As a next step, each island merges the requests designated for it and ends up with $\frac{n}{2} + d$ requests, some of which have duplicate wire id's. The duplicate requests must then be combined in a meaningful way before proceeding with the algorithm.

For a pair of messages m_1 and m_2 that must be combined, one of the following must be true:

- $m_1 = m_2$, either both of them are carrying a message from another user or none of them

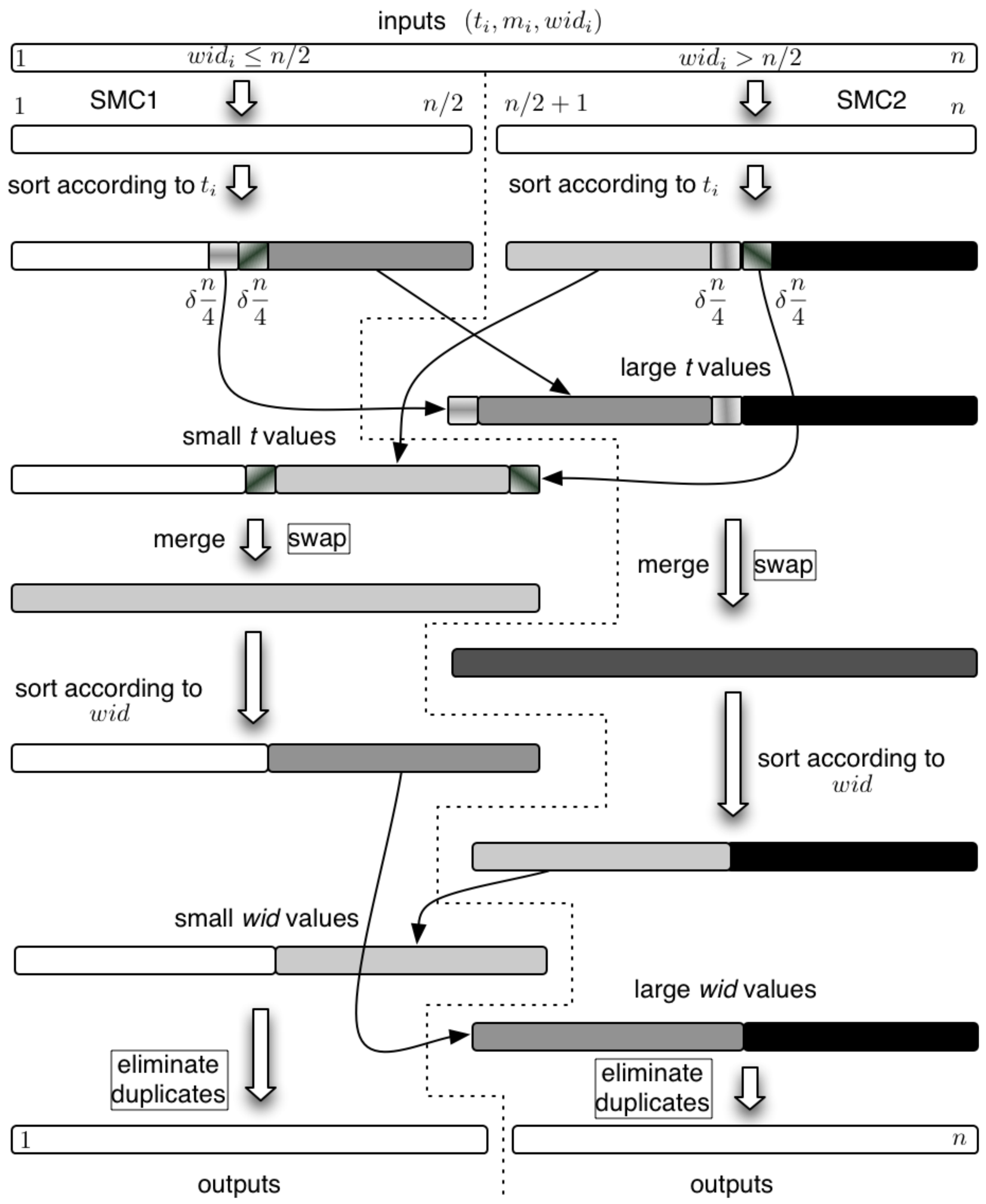


Figure 6.4: Parallel operation of two SMC islands performing the Conversation Protocol.

- $m_1 \neq m_2$, that is one of the two requests carries a message from another user and the other carries the original message sent.

So to combine the two messages, we apply algorithm 3. This algorithm eliminates duplicates by combining two requests of the form (m_1, wid), (m_2, wid) with same wire id's so that if one of them carries a message that was the result of an exchange operation (LSB=1), then this is the message that survives. At the end of this combination we have the valid message form a tuple of the form (m, wid) and the invalid one a tuple $(m', 0)$, which will be discarded at the end.

Algorithm 3 Eliminate duplicates

Input: a sequence of $n + d$ tuples $\{a_1, \dots, a_{n+d}\} = \{(m_1, wid_1), (m_2, wid_2), \dots, (m_n, wid_n)\}$, which is the sorted output (according to the wire id's) of the merge step of the algorithm and contains d duplicates.

Output: a sequence b of size n, that is the output of the protocol.

```
For each $i \leftarrow 1, \dots, n + d - 1$
if $a_i.2 = a_{i+1}.2$ then
  if $a_i.1 \mod 2 = 0$ then
    $a_i.1 = a_{i+1}.1$
  end if
  $a_{i+1}.1 = 0$
end if
sort sequence $a$ according to $a_i.2$ using quicksort. Sorted sequence is $a'$.
return sequence $b = (a'_{d+1}, \dots, a'_n)$
```

Finally, each island sorts its requests according to their wire id's and discards the leftmost d requests. Then the valid messages are forwarded to their recipients.

The combination of two islands, as explained above, can be generalized to the combination of S islands with each one handling $\frac{n}{S}$ of the requests. Quality of service may decline with increased island numbers but it can be controlled and is predicted to remain at a very high level.For example 10 SMC islands can serve 10.000 clients each, for a total of 100.000 clients. The work done by each island is only a little more than what a standalone system serving 20.000 clients would perform. More details on the projected performance of our protocol can be found in section 6.6.4. Everything that concerns the way communication is performed between an SMC island and its client pool follows what has been said about a standalone SMC system in previous sections.

6.6.3 Quality of Service Analysis

Using the parallelized algorithm of section 6.6.2, there is a probability that a client will not be served, that is a valid message exchange may not take place. In this case, the client can just retransmit her message. An analysis of the probability of such an event taking place in the two island case, follows:

Let n be an even number, representing the number of users, and $\mathcal{C}$ an arbitrary set of disjoint subsets from $\binom{[n]}{2}$, representing the pairs of users currently in conversation. Consider the following probabilistic procedure of assigning rendezvous points to the users in each round: for each $i \in [n]$, pick a random value $v_i \leftarrow \{0,1\}^\lambda$ and if it happens that $i \in P = \{i, j\} \in \mathcal{C}$ such that t_j is defined already, set $t_i = t_j$; Else, set $t_i = v_i$. Consider now the vector $\langle t_1, \dots, t_n \rangle$ and sort it, resulting to the vector $\langle t'_1, \dots, t'_n \rangle$. Define the event BAD_δ to be the event that $\mathrm{MSB}(t'_i) = 1$ for some $i \leq (1 - \delta)n/2$ where $\delta \in (0, 1)$ is a parameter, or that $\mathrm{MSB}(t'_i) = 0$ for some $i \geq (1 + \delta)n/2$.

If $\mathcal{C} = \emptyset$, it holds that the most significant bit is uniformly distributed over $\{0, 1\}$ in each draw t_i and it follows that the mean of number of times it will be selected to be 1 is $n/2$. We recall the two-sided Chernoff bound $\Pr[|X - \mu| \leq \delta\mu] \leq 2\exp(-\delta^2\mu/3)$ where X is the Binomial distribution with mean μ and $\delta \in (0, 1)$. Consider X_i to be equal to $\mathrm{MSB}(t_i) = 1$ and we let $X = \sum_{i=1}^{n} X_i$. Observe that indeed X is following the Binomial distribution with mean $\mu = n/2$ and that the event BAD_δ can only happen if the number of t_i's with $\mathrm{MSB}(t_i) = 1$ deviate by a factor $(1 - \delta)$ below or $(1 + \delta)$ above the mean. It follows immediately that $\Pr[\mathsf{BAD}_\delta] \leq 2\exp(-\delta^2 n/6)$.

In the general case, for arbitrary $\mathcal{C}$, observe that the most significant bit for each draw t_i is uniformly selected unless it happens that $\{i, j\} \in \mathcal{C}$. It follows we have $n' = n - |\mathcal{C}|$ draws, where $0 \leq |\mathcal{C}| \leq n/2$ since the elements of $\mathcal{C}$ are subsets of size 2 that are disjoint. The i-th draw selects element $t_{f(i)}$ where f is a mapping from $[n']$ to a subset of equal size in $[n]$ that drops the largest element from each conversation pair. We denote by P the set of all elements of $[n']$ so that $f(i)$ participates in a conversation in $\mathcal{C}$. We now define the random variable Y_i as the most significant bit of the i-th draw among the n' ones we perform and we let $Y = \sum_{i=1}^{n'} c_i \cdot Y_i$ where $c_i = 2$ if $i \in P$ and $c_i = 1$ otherwise. Observe that if the event BAD_δ happens it should be that $Y \notin [(1 - \delta)n/2, (1 + \delta)n/2]$. Note that by linearity of expectation it holds that $E[Y] = n/2$ hence the mean, compared to the previous case, has not changed. It is easy to extend the Chernoff tail bound to a tail bound and obtain a tail bound for Y that will be exponentially decreasing with n' (alternatively one may use the Hoeffding bound). Specifically we can prove the following.

Proposition 1 *For any $n \in \mathbb{N}$, any $\delta \in (0, 1)$ and any set $\mathcal{C}$ of disjoint subsets of cardinality 2 from $[n]$, it holds that $\Pr[\mathsf{BAD}_\delta] \leq 2\exp(-\delta^2 n/12)$.*

So for a total of ten thousand users and two servers and with $\delta = 0.08$, the probability that someone will not be served is less than one percent.

However, this is an upper bound and in practise the quality of service is much better. To better assess it we ran experiments on the Sage platform [14] for $n = 100,000$ users and $S = 10, 20$ servers, with each experiment run 200 times. These parameters were chosen as they express a possible usage scenario. An assumption made when running the experiments was that half of the clients were communicating with someone and half were idle. This parameter, however didn't seem to influence the results in a substantial way. In figure **??**, we can see the relation of the qos (quality of service) and the server overhead (how many more requests a server had to process compared to the original number) variables.

As we can see from figure **??**, The quality of service for a 10 island system with each island processing 10000 requests is very high (96%) even when no extra requests are taken. Whith an overhead of 10% (11000 requests per server), the possibility of even a single failure is very close to zero. The 20 island scenario naturally lags behind in quality of service when no extra requests are taken, begining at 88%. Nearly perfect service is achieved with an overhead of 40%, that is 7000 requests per server compared to the original 5000. Generally, we see that the quality of service even when partitioning the requests among 20 islands is very high and this form of parallelization can be very rewarding.

6.6.4 Performance of the Conversation protocol

We have implemented the Conversation protocol by running our Sharemind, "SecreC" programs on a local 1 Gbps LAN cluster with 12-core 3 GHz Hyper-Threading CPU and 48 GB of RAM. Concerning message length, experiments asserted our expectation that message length ranging from 64 to 640 bits does not significantly influence the performance of our protocol, due to the nature of the sorting algorithms used. Furthermore, cryptographic operations by the servers,

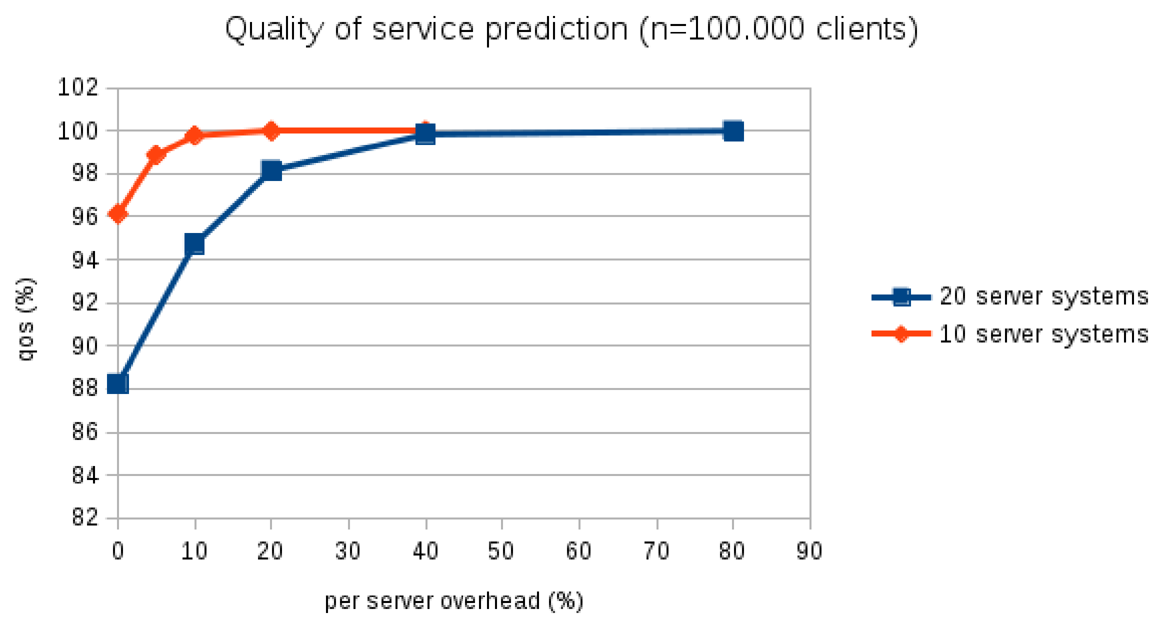


Figure 6.5: Quality of Service simulation results

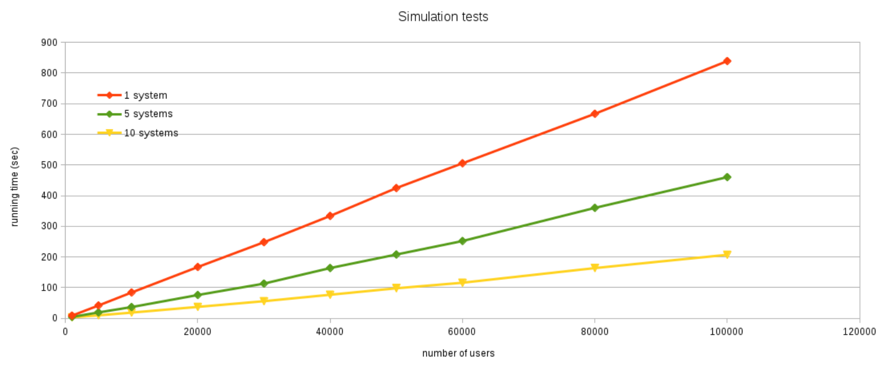


Figure 6.6: Conversation simulation results

such as decrypting and encrypting the shares have not be taken into account in the testing, but their cost is similar to single TLS connections. This overhead can be neutralized by having the servers process the requests in a pipelined fashion, that is decrypting the requests for round $r + 1$ while processing the requests of round r and in any case it is a very small overhead given that we use symmetric cryptography for encrypting the shares. The results are presented in figure **??**. As we can see, our single system protocol can serve 10.000 users with a latency a little over a minute.

As for the parallel case, our parallel algorithm requires 1 sort and 1 merge on the t_i's and 2 sorts and 1 merge on the wid's. The total cost at each island is then about double of the original cost of running the protocol on only one 3-server system. Consequently, the running time of our system for S SMC islands running in parallel and for n total clients will be the equivalent of running a single SMC system with $n' = \frac{2n}{S}$ clients. Simulations match these expectations and a running time of a little under two minutes can be achieved for 60.000 users by running 10 SMC islands in parallel. In our simulations we have not used code optimizations but we have not included the communication overhead of sending and receiving the requests. However, that is expected to be small because the total number of requests transmitted between the server systems is only double the amount of total requests (plus a small number of the extra requests taken). On a final note, we can see that our parallelized system can support 100.000 users with latency around 200 seconds.

6.6.5 Parallelizing the Dialing protocol

The technique used to parallelize the Dialing protocol is very similar to the one used for the conversation protocol, described in section 6.6.2. There are two key differences that make the Dialing protocol somewhat harder to parallelize, compared to the Conversation protocol. Firstly, Dial and Dialcheck requests are paired according to the public keys of the clients and not according to shared random values. This does not guarantee the near-uniform distribution of values leveraged in the parallelization of the Conversation protocol. Secondly, in the Dialing protocol there is a possibility that one client will be the recipient of many Dial requests. We should consider this when assessing the system's privacy. To overcome these issues and to effectively make the parallelization of the Dialing protocol identical in nature to that of the Conversation protocol, we delve into the inherent capabilities of SMC systems in two steps. First, the SMC islands participating in the protocol agree on a shared random value. Then each SMC island applies a Psedorandom Function (PRF), which is basically a MAC, in an oblivious way,to each of the public keys consisting the second part of a Dial or Dialcheck tuple. That is the part according to which requests are paired and exchanged. The key used for this procedure is naturally the shared random key generated in the first part of the protocol. The PRF, which behaves like a random oracle, can be implemented using a keyed hash function or by encrypting the values e.g. with AES. The two-step procedure described above guarantees the uniformity of the distributions of the former public keys, while preserving the equality relation where it existed. That is, the Dialing protocol can still be carried out exactly the same way. We will now present a technique to parallelize the Dialing protocol using 2 SMC islands. The technique described can easily expand to any number of such islands. We assume that requests are originally divided among two islands and the wire id's of the requests attributed to the first one are strictly smaller than those of the second. To parallelize the protocol, after applying the PRF, we sort the tuples of the form (i_1, j_1, wid_1) according to their second (now randomized) coordinate. As a next step, we partition each island's set of values in half and have one island handle the two lower halves and the other the two upper halves. The idea behind this is that a uniform distribution of the j values means that the lower halves of both servers will contain values in the same range, with the same naturally applying to the upper halves as well. As a result, identical values that originally resided in different servers are likely to end up to the same server after this step. To further enhance this likelihood, each server also takes an extra chunk of requests of size δ times the chunk it would get from each server (see figure). After this partition step, the tuples need to be merged according to their j values. Then, the protocol works as in the non parallel case, that is Dialcheck requests are matched to their neighbouring Dials and the necessary values are passed to the Dialckeck requests as in algorithm 1 (check "C" and substitute in figure **??**). At that point the basic functionality has been achieved and we need a way to get the requests back to their original positions at their server of origin. To achieve this, we sort the tuples according to their wire id's (third coordinate). As mentioned above, requests are originally partitioned according to those wid's and thus at this point, after the sort, it is guaranteed that the lower half of the requests at each server originated from the first server and the upper half from the second. As a next step, requests are returned to their server of origin according to the logic just described, and then merged according to their wid's. This would conclude the procedure if $\delta = 0$, but the extra requests taken from each server result in double requests at this stage. These double requests correspond to the same client (same wid) but their first coordinate may or may not be identical. Specifically, when talking about Dialcheck requests, which are the ones of interest, one of the two may contain the public key of a Dialer and the other may be empty. That would be a result of a scenario

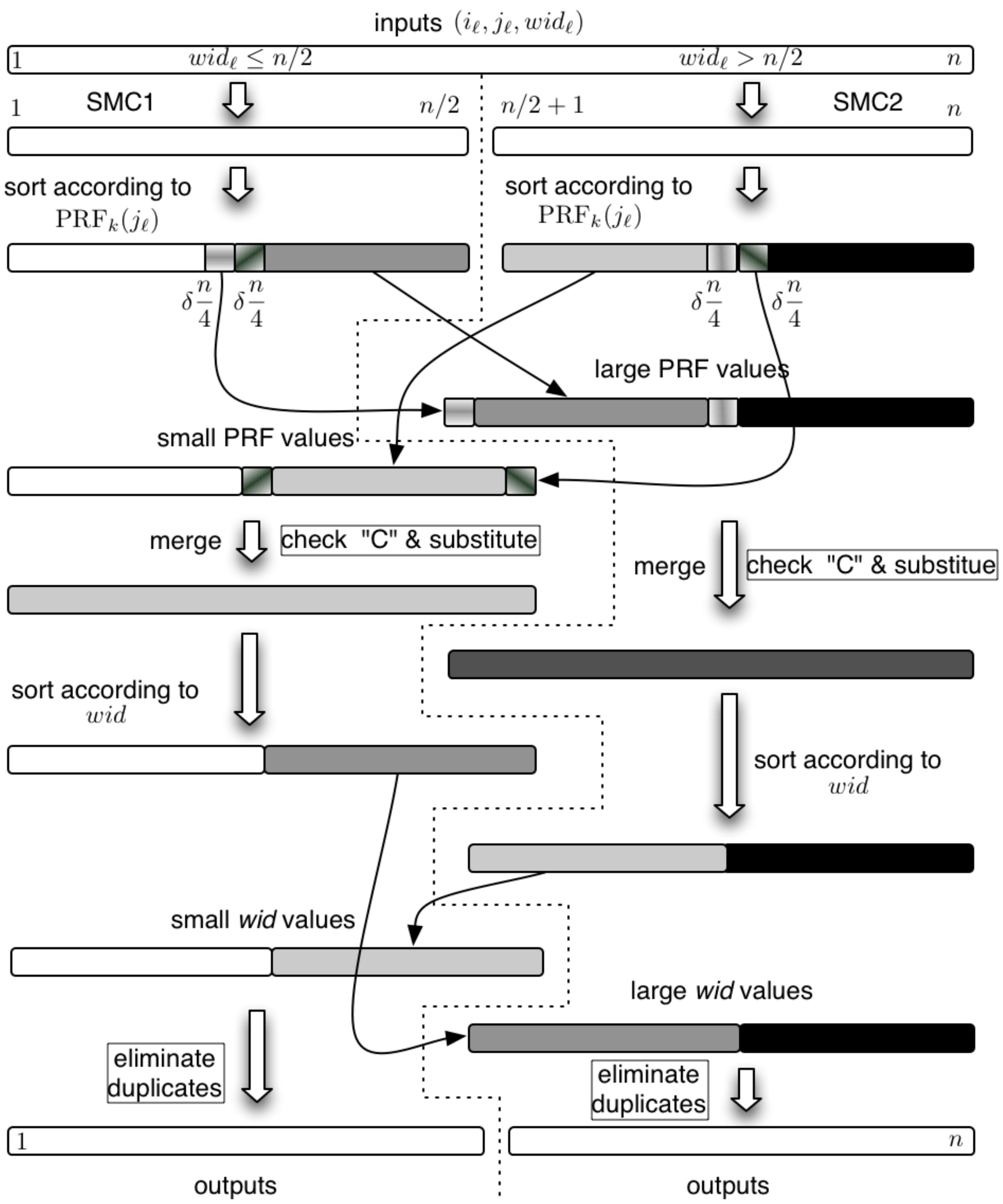


Figure 6.7: Parallel operation of two SMC islands performing the Dialing Protocol.

in which one of those requests ends up at a server that has a matching Dial request and the other at a server that doesn't. In this case, the request that is empty (first coordinate equals to 0) should be deleted. In any other case in which the two requests are both empty or both contain different values, any of the two can be deleted e.g. always the first one encountered. This process (eliminate duplicates in figure **??**) can be carried out by a simple check similar to the one used for the parallelized Conversation protocol (algorithm 3). After this final step, the sequence of the output requests is identical to the one of the corresponding input requests and thus they are ready to be returned to their recipients.

6.7 Building an anonymous communication service

6.7.1 System architecture

The complete architecture of the system is shown in figure 6.3 and will include apart from the secure computation servers, an entry and an output server used to handle client requests. The entry and output servers may be located on the same or on different physical machines.

The execution of the Dialing protocol is outlined below:

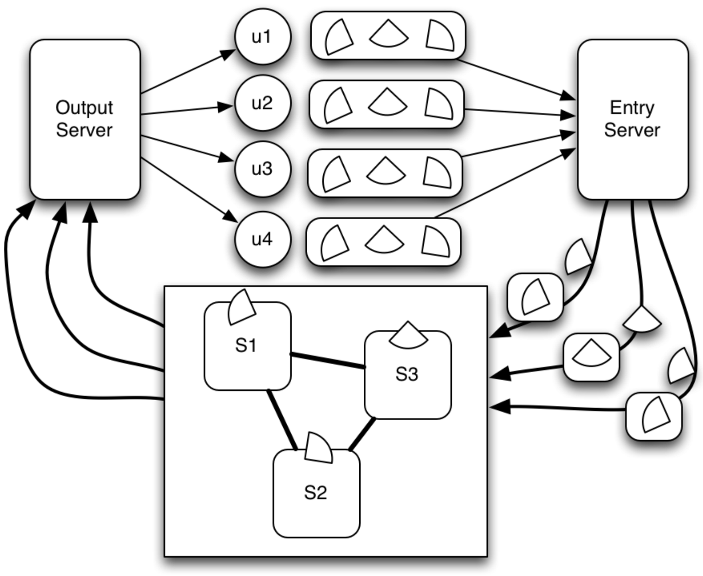


Figure 6.8: XYZ general architecture

Dialing round r

1. Each **client** generates a tuple $a = (i, j)$, which can either be a Dial or Dialcheck request, constructed as shown in functionality $\mathcal{F}_{\mathsf{DIAL}}$.

2. Each **client** produces a secret sharing of a and encrypts each share with the public key of the intended SMC server. For example $j \rightarrow a_1, a_2, a_3$ and $s_k = Enc(pk_{server_k}, a_k)$, $k \leftarrow \{1, 2, 3\}$, assuming three SMC servers.

3. Each **client** forwards the request of the form (s_1, s_2, s_3) signed with her secret key along with her public key to the entry server.

4. **Entry server** checks signature and if valid assigns a wire id to the respective user.

5. **Entry server** forwards request shares of the form $(s_k|wireid_{ik}|pk_{ik})$ to SMC server k. Note that $wireid_{ik}$ are shares of the wire id and pk_{ik} are shares of client's u_i public key, both generated by the entry server.

6. **SMC servers** decrypt request shares and execute multiparty dialing algorithm (algorithm 1).

7. **SMC servers** encrypt result shares with the recipient's key and send them to the output server.

8. **Output server** collects the shares and forwards them to the respective clients.

9. **Client** decrypts her shares and reconstructs her output of the protocol.

In the Dialing protocol, each client generates requests as described above, signs them and forwards them to the entry server along with her own public key for reasons of signature verification. It is important to note that communication between the clients and the entry server is encrypted with the keys of the SMC servers. This is why the entry server alone cannot compromise client privacy. The entry server performs two other tasks apart of the signature verification and assigning to each user a wire id that is then shared in 3 parts and forwarded along with the request to the SMC servers. This wire id is given in plaintext and is added to the decrypted user request shares as the third member of the tuple that forms algorithm's 1 input. Finally, the entry server also forwards a sharing of the public keys that correspond to each request submitter, as requested by the Dialing algorithm. Both the wire id's and the public keys are shared and forwarded in plaintext to the SMC servers. Sorting in general has the ability to produce any permutation on a given input vector and thus an oblivious sorting procedure will produce a sequence that will look completely random, despite the fact that at the start some of the information of the input was publicly known.

For the output to be generated, the SMC servers first decrypt the incoming shares. Then, they calculate the result of the dialing algorithm in a privacy-preserving manner and re-encrypt the result shares with the respective clients' encryption keys. Finally, they send these encrypted shares to the output server, which in turn forwards them to the clients, according to the client-wire id relationship established at the entry server. For the Conversation protocol, we have:

Conversation round r

1. Each **client** i generates a tuple $a = (t_i, m_i)$, as presented in functionality $\mathcal{F}_{\mathsf{CONV}}$.

2. Each **client** produces a secret sharing of a and encrypts each share with the public key of the intended SMC server. For example $j \rightarrow a_1, a_2, a_3$ and
$s_k = Enc(pk_{server_k}, a_k)$, $k \leftarrow \{1, 2, 3, \}$, assuming three SMC servers.

3. Each **client** forwards the request of the form (s_1, s_2, s_3) to the entry server.

4. **Entry server** assigns a wire id to the respective user.

5. **Entry server** forwards request shares of the form $(s_k | wireid_{ik})$ to SMC server k. Note that $wireid_{ik}$ are shares of the wire id generated by the entry server.

6. **SMC servers** decrypt request shares and execute multiparty conversation algorithm (algorithm 2).

7. **SMC servers** encrypt result shares with the recipient's public key and sends them to output server.

8. **Output server** collects the shares and forwards them to the respective clients.

9. **Client** decrypts her shares and reconstructs her output of the protocol.

The Conversation protocol works much like the Dialing one. One difference is that now the entry server doesn't have to make any checks concerning the users' requests and identities. This is a direct result of the use of a pseudorandom rendezvous point that is known only to the 2 users that have establishes a connection through the Dialing protocol. The entry server still shares and forwards the shares of the wire id's assigned to each user. The rest of the protocol is identical to the Dialing one described earlier, apart of course of the fact that the SMC servers execute the Conversation algorithm as described in algorithm 2.

6.7.2 Key management

In all previous sections, it was assumed that a client knew the public key of another client whom she wanted to dial and that all SMC server-client communication was performed using public key algorithms. To provide better solutions to these issues we introduce the concept of client registration.

Each client u_i can apply for registration at the entry server using her public key and optionally a username. The entry server will forward the registration request to each SMC server along with the client's public key. Each SMC server will construct a master key for user u_i using a keyed pseudorandom function with the user's public key as input, that is $s^{u_i}_{serveri} = PRF_k(pk_{u_i})$. This key will then be encrypted with the client's public key and sent back to her through the entry server. From this point onward, the client can encrypt her requests with a key generated by applying a pseudorandom function with the master key and the round number, $s_i = PRF_{s^{u_i}_{serveri}}(r)$ and use symmetric key encryption to communicate with the server. To additionally achieve forward secrecy, a client can forward at the first stage of registration, along with her public key and username, a fresh public key for communication with each server, encrypted with the server's public key. This key will in turn be kept by the server and used to produce the master key. When the client wishes, say every 24 hours, she can send a renew request with a new public key and thus achieve some form of forward secrecy on the client side.

To address the problem of the PKI, we can have each client download the whole database of public keys and usernames kept by the entry server or use a private information retrieval (PIR) protocol [9] to reduce communication complexity.

Bibliography

[1] M. Ajtai, J. Komlós, and E. Szemerédi. An 0 (n log n) sorting network. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 1–9. ACM, 1983.

[2] K. E. Batcher. Sorting networks and their applications. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, pages 307–314. ACM, 1968.

[3] D. Beaver. Commodity-based cryptography. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 446–455. ACM, 1997.

[4] A. Ben-David, N. Nisan, and B. Pinkas. Fairplaymp: a system for secure multi-party computation. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 257–266. ACM, 2008.

[5] D. Bogdanov, S. Laur, and R. Talviste. A practical analysis of oblivious sorting algorithms for secure multi-party computation. In *Secure IT Systems*, pages 59–74. Springer, 2014.

[6] D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A framework for fast privacy-preserving computations. In *Computer Security-ESORICS 2008*, pages 192–206. Springer, 2008.

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

[8] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[9] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *Journal of the ACM (JACM)*, 45(6):965–981, 1998.

[10] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies*, pages 46–66. Springer, 2001.

[11] H. Corrigan-Gibbs, D. Boneh, and D. Mazières. Riposte: An anonymous messaging system handling millions of users. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 321–338. IEEE, 2015.

[12] I. Damgård, M. Geisler, M. Krøigaard, and J. B. Nielsen. Asynchronous multiparty computation: Theory and implementation. In *Public Key Cryptography–PKC 2009*, pages 160–179. Springer, 2009.

[13] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type iii anonymous remailer protocol. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 2–15. IEEE, 2003.

[14] T. S. Developers. *SageMath, the Sage Mathematics Software System (Version x.y.z)*, YYYY. `http://www.sagemath.org`.

[15] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[16] C. Dwork. Differential privacy. In *Automata, languages and programming*, pages 1–12. Springer, 2006.

[17] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229. ACM, 1987.

[18] K. Hamada, D. Ikarashi, K. Chida, and K. Takahashi. Oblivious radix sort: An efficient sorting algorithm for practical secure multi-party computation. *IACR Cryptology ePrint Archive*, 2014:121, 2014.

[19] K. Hamada, R. Kikuchi, D. Ikarashi, K. Chida, and K. Takahashi. Practically efficient multi-party sorting protocols from comparison sort algorithms. In *Information Security and Cryptology–ICISC 2012*, pages 202–216. Springer, 2012.

[20] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson. Users get routed: Traffic correlation on tor by realistic adversaries. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 337–348. ACM, 2013.

[21] P. Laud and J. Willemson. Universally composable privacy preserving finite automata execution with low online and offline complexity. *IACR Cryptology ePrint Archive*, 2013:678, 2013.

[22] C. Liu, X. S. Wang, K. Nayak, Y. Huang, and E. Shi. Oblivm: A programming framework for secure computation. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 359–376. IEEE, 2015.

[23] M. Robson, M. Polte, S. Goel, and E. Sirer. Herbivore: A scalable and efficient protocol for anonymous communication. Technical report, Cornell University, 2003.

[24] L. Sassaman, B. Cohen, and N. Mathewson. The pynchon gate: A secure method of pseudonymous mail retrieval. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 1–9. ACM, 2005.

[25] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[26] D. L. Shell. A high-speed sorting procedure. *Communications of the ACM*, 2(7):30–32, 1959.

[27] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on*, pages 44–54. IEEE, 1997.

[28] J. Van Den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152. ACM, 2015.

[29] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson. Dissent in numbers: Making strong anonymity scale. In *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 179–182, 2012.

[30] A. C. Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 160–164. IEEE, 1982.

[31] Y. Zhang, A. Steele, and M. Blanton. Picco: a general-purpose compiler for private distributed computation. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 813–826. ACM, 2013.

Part III

Definitions of privacy

7. Anonymization With guaranteed privacy

Big data, i.e. the online collection of people's behviours, is becoming a major driver of the digital economy. Data and its analysis form the resources of tomorrow's economy. Several businesses that have collected such massive amounts of data are actively looking for ways of monetizing it. However, while there are great economic opportunities there are also societal risks. Big data collection and analysis may allow sensitive inferences about people's life. For example, genomic or health data which are major drivers of big data may allow inferences about disposition to certain illnesses or personality traits. Future employers could leverage that information to deny access to certain career paths. Even shopping data may reveal such sensitive health-care related information as the case of Target's advertising shows [6]. The PANORAMIX WP6 use-case, relating to gathering in privacy-preserving ways this type of statistics and aggregates aims to reduce such privacy risks.

Hence, it is a societal challenge to balance these objectives of spurring economic growth and preserving personal privacy. Solutions include a variety of approaches from self-controlling, privacy-respecting behavior, legal regulation to technical protection means. No single solution can work by itself and any technical approach needs to integrate into the legal framework. In particular, the upcoming EU data protection regulation includes the categories of personal, pseudonymized and anonymized data. Personal (and pseudomyized) data may only be used for the purpose it has been collected for and if such data is used for other purposes – which may relate to monetization – the data needs to be anonymized. Anonymization means the removal of all personal identifiers, such that no de-identification is possible without the original data set. This proves to be a challenging technical task and seems to be very error-prone. The goal of the SAP Product Security Research project AWARE "Anonymization With guARantEed privacy" is to provide a framework for the data protection officer to apply anonymization with measurable and reliable guarantees. As such, it features a privacy parameter for each method that can be appropriately set balancing privacy versus utility. Hence, the AWARE project aims to enable the monetization of personal big data sources, i.e. preserving sufficient utility, while also preserving the privacy of the data.

7.1 Why Anonymization so far does not work

A fundamental challenge in anonymization is that the simple removal of an identifier does not protect against de-identification. The data itself may provide sufficient possibility for inference with external data sources that de-identification is possible. Consider this small example: The list of all companies with their name, field of operation, revenue, and country. This list would contain an entry SAP, IT, 17.56 billion EUR[1], Germany. Even, if we were to remove the name SAP simply from its size the other information would allow a de-identification. We list a number of specific challenges in preventing this type of de-identification.

[1]2014 figures

- Linking to external sources. A famous case of de-identification of medical records was put forward by Latanya Sweeney [12]. She linked an anonymized table of medical records – including age, gender and zipcode – with the voter registration list and de-identified the governor of Massachusetts. The conclusion is that anonymized data may never be seen by itself, but only in conjunction with other data sources. This attack led to the development of k-anonymity.

- The curse of high dimensionality. Some data is inherently hard to anonymize. High-dimensional data is such a case, since anonymization must then also cover a high number of dimensions [1]. An example of such an attack that also used external data sources was the Netflix de-identification [10]. The anonymized set of Netflix recommendations was de-identified using the public IMDB recommendations. Another example of such an attack is the recent de-anonymization of credit card data [3]. Given a few sample purchases credit card holders could be identified.

- Patterns. Data may have inherent patterns that remain over time. Hence a small de-anonymized sample may suffice to de-anonymize entire data sets. An example attack of this kind is the de-anonymization of smart meter data [7].

The consequence of these challenges is that almost no data can be left unmodified for proper anonymization. In most cases the idea of sensitive, unmodified data is challenged by the findings listed above. Therefore a different model of anonymization was necessary. This model is differential privacy [4] which provides a metric of data perturbation, such that any inference becomes hard. In the next section we will introduce the foundations of this metric and the mechanisms that can achieve it.

7.2 Guaranteed Privacy

We now introduce the notion of differential privacy and illustrate how it can be integrated into the data analysis process. As we will highlight in Section 7.1.1, differential privacy is a mathematical provable guarantee on data leakage for an individual who is participating in a database. This is achieved by the application of mechanisms for the controlled generation of random noise to mask the original of a query result or to directly sanitize the original data. A detailed discussion of this differentiation is given in Section 7.1.2. The underlying statistical foundations are illustrated in Section 7.1.3.
The chapter will show that differential privacy is achieving a compromise between privacy and utility to prevent an adversary with auxiliary information about the database participants of isolating an individual. The content is based on [5].

7.2.1 Metric

To ensure a common understanding for the definitions that are introduced in this section, we will first briefly outline some common notations for differential privacy and give an overview example.

- In the following a database is interpreted as a vector of n entries from a value domain D.

- The domain D is representing the set of all possible attribute values by which database records are represented.

- An entry within the vector is representing a specific individual's (tuple) attribute value, for example a salary or an coordinate.

- A query is represented as a function $f\colon D^n \mapsto \mathbb{R}^d$, partitioning the database vector into a subset of d bins (i.e a histogram).

An example that we will formulate in this notation is a counting query for a specific attribute value. In this example $D = \{0, 1\}$ for each tuple in respect to possessing the attribute for counting. The count query result over the tuples in database A is thus expressed as $f : \sum_{i=1}^{n} A_i$.
Before we will formally define differential privacy, we will extend our example to a differentially private count for illustration. The notion of differential privacy expresses the likelihood that a query on two databases A and B, which only differ in the presence of one participant, answers this counting query with a similar result on both databases. So even while the formulation of achieving the outcome for a query independent of someone's presence or absence may sound counterintuitive at the first time, it literally expresses the desire to protect an individual from isolation based on a query outcome. Thus, the formulated maximum distance of one participant between A and B expresses the desire for protection against isolation of a single individual record from a result of a database record.
The protection is achieved by wrapping the original counting query into a so called noise generating mechanism M, which adds an amount of controlled noise to the result of the original count. In differential privacy, the level of noise can be dynamically defined to enable higher utility or privacy as we will show later within this chapter.
In general, the offered utility/privacy guarantee is expressed as e^{ϵ} for an individual within the database, where ϵ represents the privacy loss that a participant faces due to being in the database. This is formally expressed as

$$Pr[M(A) \in S] \leq exp(\epsilon) \times Pr[M(B) \in S].$$

While for small values of ϵ the bound is close to $1 \pm \epsilon$, it has to be underlined that the bound widens exponentially and already small increases in ϵ significantly lower the privacy guarantee. Thus, by increasing (decreasing) ϵ and thereby decreasing (increasing) limiting the utility that can be gathered from a function $f(A)$ on the database A. The specific enforcement of differential privacy has to be performed by a perturbation mechanism which introduces noise to cover an original result. The amount of noise depends on two parameters:

1. The level of desired utility/privacy expressed through ϵ.

2. The amount to which the participation of an individual changes the result of a database query. This notion is referred to as *sensitivity* and expressed for a query function as Δf.

Thus, for the scaling of noise it essential to estimate the sensitivity of a query operation $f(A)$. For illustration, we will refer to our example of a count query again. For a count operation, the maximum change that the presence of an individual can cause to the query result is determined by her number of tuples that fulfill the count evaluation. If we assume that a participant possesses at maximum one tuple in the database the sensitivity would be $\Delta f = 1$. In contrast, for the evaluation of Sums and Averages the sensitivity of a function has to be specifically bounded. If we take for example the derivation of an average over several salaries, the sensitivity would be represented by the maximum obtainable salary. Commonly, several differentially private mechanisms can be executed sequentially. If, for example, questions on specific databases are repeated, or uncorrelated databases are involved in multiple questions, the privacy leakage of ϵ will add up. Thus, overall accuracy should deteriorate with the amount of questions asked under consideration of database correlation. This is expressed by the sequential composition. If two ϵ-differentially private mechanisms with independent noise distributions, e.g. M_1 and M_2, on databases $A \approx B$ are composed in a new mechanism M_3, then $M_3(M_1(B), M_2(B))$ will provide privacy according to 2ϵ. Thus, it is critical to introduce and monitor a privacy budget, limiting an analyst to eventually learn a true value by consuming more ϵ. The privacy budget definition and allocation is thus clearly a non-trivial activity. Generalized to a series of r queries and to ϵ-differential privacy, sequential composition is expressed as $\sum_i(\epsilon_i)$-differential privacy.

$$\prod_i Pr[M_i^r(A) = r_i] \leq \prod_i Pr[M_i^r(B) = r_i] \times \prod_i exp(\epsilon_i).$$

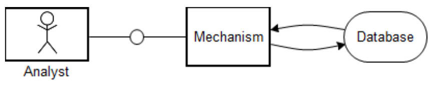
To conclude, the introduced metrics are summarized by the following formal notation for a differentially private mechanism M on database A as

$$M(A) = f(A) + Noise(\epsilon, \Delta f).$$

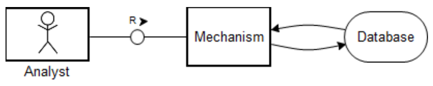
7.2.2 Privacy enforcement

We will now elaborate on how differential privacy can be integrated and enforced in the data analytics process. Naturally, differential privacy can be on individual data as well as statistical aggregates. Thus, we will in the following evaluate these dimensions.

First, depending on the the fact whether an expected set of differentially private queries on data is known in advance, mechanisms can be run in the *non-interactive model* or *interactive model*. Within the non-interactive model, a differentially private version of a database A is generated by a database owner and released to data analyst. One example for a non-interactive model is a sanitized database which consists only of a set of differentially private statistics on the original data and selected perturbed attribute values. Another example might be the restriction of interaction between the analyst and the original database by a set of static differentially private queries, thus limiting interaction as depicted in Figure 7.1b. While the non-interactive model has the drawback that the scope of possible analyses is limited, it also provides the benefit that the monitoring effort on the privacy budget is lower (especially in the case of a sanitized database). This is due to the fact that original data is no longer available in the fixed differentially private version of database. In contrast, the interactive model allows an analyst, by using mechanisms as building blocks, to formulate differentially private queries against a database and ask new questions about the original data. Thus, the data can be analyzed under different sensitivities and ϵ values in bidirectional communication model as depicted by Figure 7.1a. This model clearly provides the benefit of higher expression. However, it comes at the cost of having to monitor a privacy budget (i.e. the amount of consumed ϵ/asked questions) for the original data. While interactive mechanisms are generally utilizable in a non-interactive way, some mechanisms cannot be fully utilized in a the interactive model (e.g. the Exponential mechanism in Section 7.1.3 is non-interactive due to the predefined Range r).

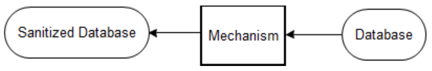


(a) The interactive model

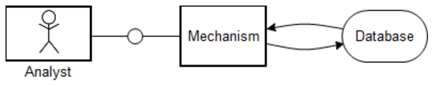


(b) The non-interactive model

Second, perturbation according to differential privacy can be enforced on *outputs* (e.g. statistical aggregates) or by *randomizing inputs* and thus ensuring a differentially private calculation. This model is also referred to as local (input) and central (output) randomization. A main approach for enforcement in local differential privacy is the randomized response mechanism, which will be introduced in Section 7.1.3. This architecture enables individuals to keep their data element, which can be interpreted as a database containing only a single tuple, differentially private from an untrusted third party (i.e. a database administrator). An example for input perturbation would be the release of a database of salaries, in which each salary is perturbed by a differentially private mechanism before the release. The released and differentially private version of an original database is referred to as the sanitized version, as depicted by Figure 7.2a. In contrast, an example for output perturbation is represented by the scenario in



(a) Input perturbation



(b) Output perturbation.

which an average is calculated on original salary data and then perturbed by a differentially private mechanism before being released to the analyst. This case is illustrated by Figure 7.2b.

In general, the biggest difference between the mentioned approaches is situated within their need for defining a privacy budget beforehand and the capability of releasing individual datasets or statistical aggregates.

7.2.3 Distortion mechanisms

In the following we will introduce the most common mechanisms for adding differentially private noise in data analysis. The use of a respective mechanism is mostly motivated by the insensitivity and stability of input data in regards to noise and by the enforcement model. We will feature the Laplace mechanism for numeric perturbation in Section 7.1.3, the Exponential mechanism for the perturbation of numeric and categorical values in Section 7.1.3, and the randomized response mechanism for local perturbation in Section 7.1.3

Laplace mechanism

The Laplace mechanism of [5] is suited for the enforcement of differential privacy on numerical valued queries which provide the analyst with a real valued answer. An example for such an operation would be a count query (or any other statistical aggregate). As the name already indicates, the Laplace mechanism samples noise from an underlying Laplace distribution. The Laplace distribution is a symmetric exponential distribution centered around mean μ with scaling factor λ. It is adapted to a differentially private version by adding sufficient noise to cover the presence of an individual database record using $\lambda = \Delta f/\epsilon$. As can be directly inferred from the specified value of λ, the level of required noise is growing (1) as the sensitivity increases and (2) as the privacy guarantee ϵ decreases. With the Laplace mechanism, several *different* input values can possibly be mapped to the *same* output value, where the probability distributions are centered on the individual input values. Therefore, reconstruction of the original value based on the mechanism result is hard. This is also referred to as the sliding property of the Laplace distribution as illustrated in Figure 7.3.

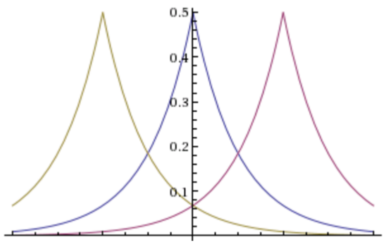


Figure 7.3: Illustration of the sliding property for the Laplace function.

Concluding, we will follow the definition of the Laplace mechanism with a query function f on database A with privacy guarantee ϵ as

$$M(A, f(\cdot), \epsilon) = f(A) + Lap\left(\frac{\Delta f}{\epsilon}\right).$$

The approach can be readily integrated into queries, as due to the well formulated Laplace distribution there is no need to formulate custom cumulative density functions for sampling.

For completeness, it is particularly interesting to evaluate the choice of the Laplace distribution over the well-known normal distribution (i.e. Gaussian distribution). While both are suited for the perturbation for numerical queries, they differ in the distribution of their probability mass. Hereby, it has to be noted that the Normal distribution has, when fixing the variance, less probability mass assigned around the mean μ and smaller tails than the Laplace distribution. This is illustrated in Figure 7.4 where the Normal distribution is illustrated in red and the Laplace distribution in blue. The handling of deviations around the mean (squared) is motivating the use of l^2-sensitivity in the Gaussian mechanism.

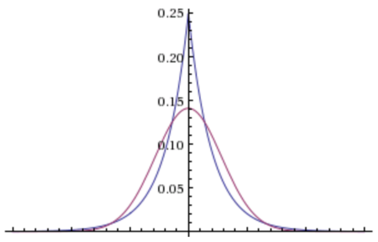


Figure 7.4: Comparison of the Laplace- and Normal distribution with $\mu = 0$, and $\beta = 2$ resp. $\sigma = \sqrt{8}$.

Exponential mechanism

The Exponential mechanism of [9] is suited for differentially private perturbation of arbitrary non-numeric and numeric functions. In contrast to the previously mentioned Laplace mechanism, the Exponential mechanism is thus designed for structural information domains which are (1) not robust and (2) sensitive to additive noise (i.e. where already a little amount of noise makes a high difference in the output result).

This adaptability is achieved by the definition of a query depending quality function q which calculates a numerical utility score for every possible query outcome. We will refer to the set of possible query outcomes as R. It can best be imagined as the enumeration of all feasible and logical correct answers that a query can receive. The quality function is then calculated for every value in R. When R is very large, the algorithm runtime becomes a challenge as the probability for every quality function has to be determined. A quality function for $r \in \mathrm{R}$ on database A is denoted $q(A, r)$. The sensitivity Δq for the Exponential mechanism is defined as the largest difference in the output of the quality function for two databases that differ in a single participant and for all r.

The Exponential mechanism is designed to assign exponentially more weight to high utility scores and pick r with probability

$$Pr \propto exp\left(\frac{\epsilon q(A,r)}{2\Delta q}\right).$$

Depending on the granularity of R, a penalty for discretization might occur. This means that there might be an element $r' \notin R$ that would achieve a higher quality score than all $r \in R$. This is called a discretization penalty, and highly depending on the formulation of R. An example for a perturbation possible by the Exponential mechanism would be a query that figures out the most common eye color of participants within the database. In this example R would encompass a set of possible eye colors, e.g. Blue, Green, Yellow, Grey, and an utility value is calculated by the quality function for each eye color. Sampling is then achieved by normalizing the obtained utility values for each $r \in \mathrm{R}$ on the interval $[0, 1]$, generating a random number in the interval $[0, 1]$, and then selecting the value r of the interval in which the random number is located. It is interesting to note that the error guarantee thus mainly depends on R (i.e. the discretization of the possible result range) and less on the amount of records in the database. We could, for example, produce an error compared to the true result by not incorporating it into the set R of possible answers.

Randomized response

A mechanism for the individual perturbation of discrete values is represented by the randomized response approach. The approach presented in [13] was originally designed to introduce plausible deniability for survey participants that answer on delicate questions, concerning illegal behavior for example. The concept is that a replier hides his true answer to a question by throwing a random coin, and acts according to the result of the random coin. Usually, at least two coin flips are executed before reporting. Privacy is ensured by the deniability of any reported answer, and increases with by the amount of coin tosses. However, by knowing the noise generation procedure it is still possible to draw conclusions from the obtained answers without being able to isolate a single individual. Thus, randomized response is resembling a Bernoulli experiment with two possible outcomes. Telling yes when the truth is yes, and telling yes when the truth is no. Of course, the experiment can also be formulated vice versa for the contrary case. For illustration that randomized response is a differentially private mechanism, we will pick up the example of [5]. In this example, survey participants are asked a question whether they participated in an illegal activity and have to answer according to the following protocol.

1. Flip a coin.
2. If tails, then respond truthfully.
3. If heads, then flip a second coin and respond "Yes" if heads and "No" if tails.

The amount of true yes responses can be approximated by reforming the the probability estimation:

$$Pr[\text{Yes}] = Pr[\text{Yes}|\text{Tails}] + Pr[\text{Yes}|\text{Heads}] = \frac{1}{2} \times Pr[\text{Yes}|\text{Truth} = \text{Yes}] + \frac{1}{4},$$

$$\hat{Pr}[\text{Yes}|\text{Truth} = \text{Yes}] = 2\left(\frac{\#\ \text{Yes}}{\#\ \text{Replies}} - \frac{1}{4}\right).$$

A meaningful implication of the randomized response mechanism is that ϵ directly depends on the response design. The above example will provide a fixed ϵ of $\ln(3)$ due to:

$$\epsilon = ln\left(\frac{3/4}{1/4}\right) = ln\left(\frac{Pr[\text{Response} = \text{Yes}|\text{Truth} = \text{Yes}]}{Pr[\text{Response} = \text{Yes}|\text{Truth} = \text{No}]}\right).$$

Randomized response has thus the potential to still approximate the true distribution, resp. answer, over multiple individually perturbed replies. This can be utilized for database scenarios in which distributed systems report their original data under the use of randomized response to a central data analysis platform.

7.3 Utility vs. Privacy

Obviously, adding noise to individual data elements decreases the accuracy of their information and hence also decreases their utility at an increase of their privacy. Both extrema are not practical for data outsourcing, that is completely randomized data does not allow any kind of data analysis, while plain data does not provide any level of privacy. However, the anonymization mechanisms are designed to support simple data analysis like arithmetic mean or more complex analysis like machine learning algorithms even after (some level) data sanitization. In this section we evaluate empirically what privacy parameters for differential privacy can be used in practice, while still allowing useful data analysis. Additionally, we examined different analytic functions with relation to their utility combined with differential privacy. Summarized, there are three parameters that influence the way the data can be sanitized, namely:

1. What level of privacy must be guaranteed.
2. What level of utility and what kind of analytic function is needed.
3. What kind of data and what amount of data is analysed.

In the current version, we focused on evaluating non-interactive mechanisms (more information on this topic is given in Section 7.1.2).

7.3.1 Experiments

The first experiments examine the effects of differential privacy on numerical data, in more detail the Laplace mechanism (see Section 7.1.3) has been applied on salary information. Here, we tested different analytic functions on sanitized information, for example arithmetic mean, median, and maximum determination.

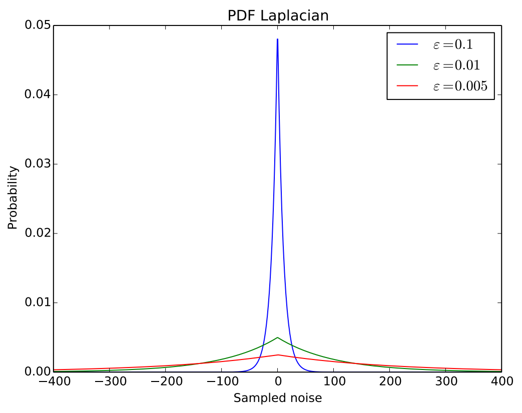
The second experiment demonstrates the usage of differential privacy for location-based systems, where the sensitive data has two dimensions.

Laplace mechanism

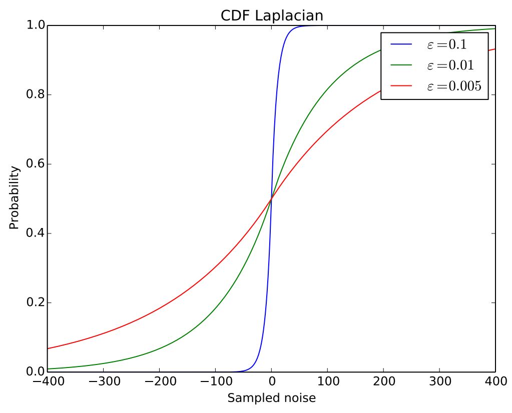
In our first experiment we prototyped the standard Laplace mechanism as described in Section 7.1.3. More particular, we normalized $\Delta = 1$ resulting in noise with probability density function (PDF)

$$Lap\left(x|\frac{1}{\epsilon},0\right) = \frac{\epsilon}{2}exp(-|x|\epsilon).$$

For a first visualization we plotted the PDF for different values of ϵ, namely $\epsilon_1 = 0.1, \epsilon_2 = 0.05, \epsilon_3 = 0.005$, in Figure 7.5a. As we can see, the diversity of the sampled noise value increases with decreasing ϵ. That is, one single value is probably distorted with greater noise value, while



(a) Laplace distribution for different values of ϵ.



(b) Laplace distribution for different values of ϵ.

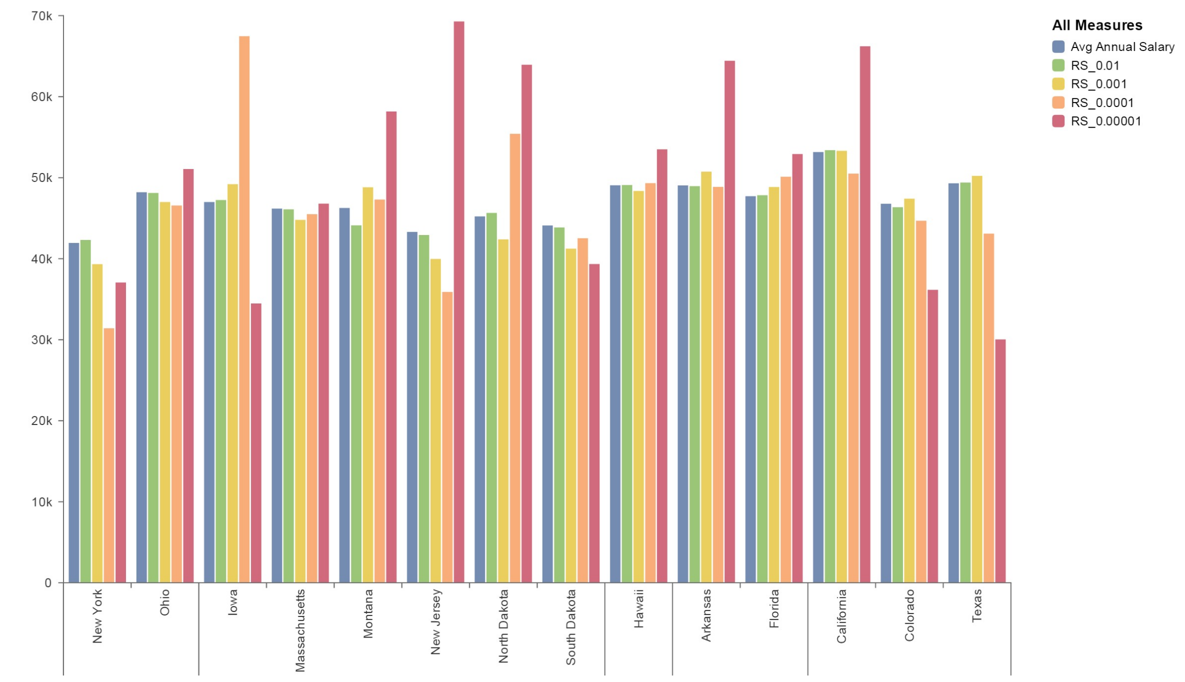
the expected values stays 0. As a result, increasing the amount of data entries, i.e. the number of noisy values that are summed up, the level of noise in the aggregation does not increase but decreases.

For a better understanding of the Laplace mechanism and how different choices of ϵ influence the noise distribution, we also plotted the cumulative distribution function (CDF)

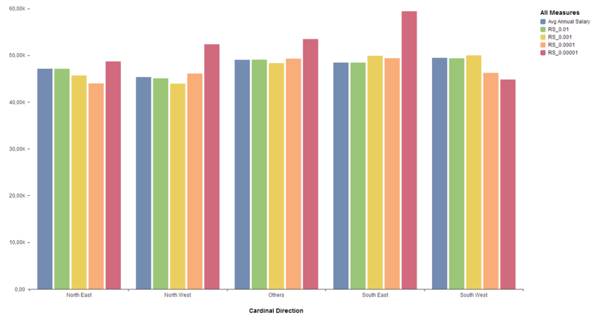
$$F_\epsilon(x) = \int_{-\infty}^{x} Lap\left(u|\frac{1}{\epsilon},0\right) du = \frac{1}{2} + \frac{1}{2}sign(x) - (1 - exp\left(-|x|\epsilon\right)).$$

The resulting Figure 7.5b can be interpreted as follows: for (relatively big) $\epsilon_1 = 0.1$ (nearly) all noise values fall approximately between -50 and 50. On the other hand, for $\epsilon_3 = 0.005$ about 10% of all sampled noise values are smaller than -400 and because of symmetry properties the same holds for noise values greater than 400. This is, roughly 20% of all sampled noise values change the numerical data by a value greater than 400.

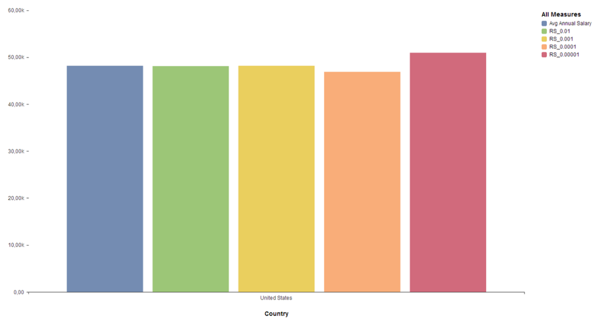
Arithmetic Mean



(a) Average salary with different values of ϵ grouped by state.



(b) Average salary with different values of ϵ grouped by cardinal region.



(c) Average salary with different values of ϵ.

Figure 7.6: Arithmetic mean for varying granularity and different privacy parameters.

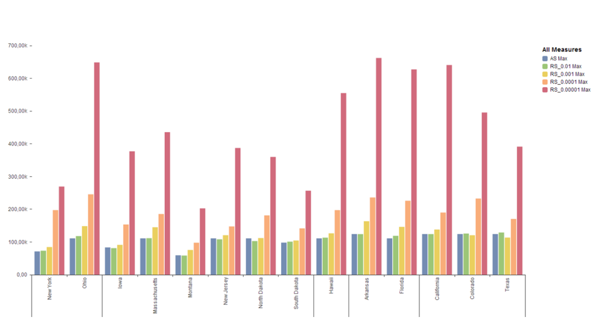
After this preliminary analysis of the Laplace mechanism we apply it to a first use case: Given a database consisting of salary data of approximately 1500 employees located in 14 different US states, the database should be sanitized utilizing the Laplace mechanism. However, we wish to learn the average salary of employees grouped by either single US-states, cardinal directions or United States global, i.e. by modifying the granularity we vary the amount of data that is aggregated.

In order to provide a visualization of the effects of differential privacy, we sanitized all salary values with different values of $\epsilon_i \in \{0.01, 0.001, 0.0001, 0.00001\}$. Original data is represented by blue bars, and different increasing privacy levels are shown as green, yellow, orange, and red bars in Figures 7.6a, 7.6b, and 7.6c.

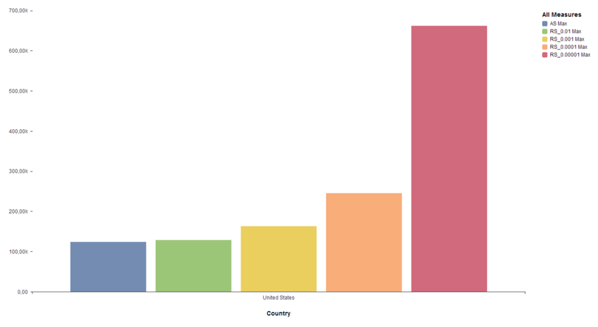
As one can see, greater amount of data per group increases utility for a fixed ϵ, e.g. the difference between original data (blue bar) and the smallest value of ϵ (greatest privacy) decreases with increasing group size, e.g. fine granularity (grouped by US-states) vs. coarse granularity (grouped by coarse direction). Furthermore, the effect of individual data (that should stay private) on the aggregated value – e.g. the average – decreases with the amount of aggregated data, hence privacy of individual data increases with the amount of data. Taken together, these two facts show that for data aggregation, both privacy and utility increase with increasing data size.

Rank-based Statistics

In contrast to the previous section, in this section we examine effects of the Laplace mechanism on rank-based statistics where individual data is more weighted on the analytic result, e.g. maximum respectively minimum operation or median. Regarding the maximum value, a greater amount of noisy data increases the probability that an extreme noise value is sampled. The maximum of a subgroup (e.g. New York, Ohio, Iowa. . .) is at most as noisy as the maximum of bigger group consisting of all the subgroups (e.g. the whole United States). This effect can be observed in Figure 7.7a in comparison with Figure 7.7b.

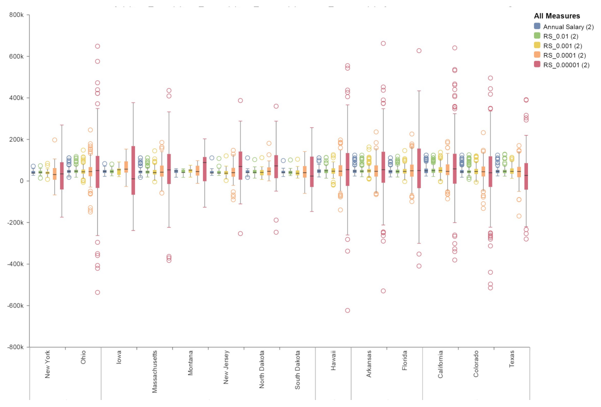


(a) Maximum salary with different values of ϵ grouped by cardinal region.

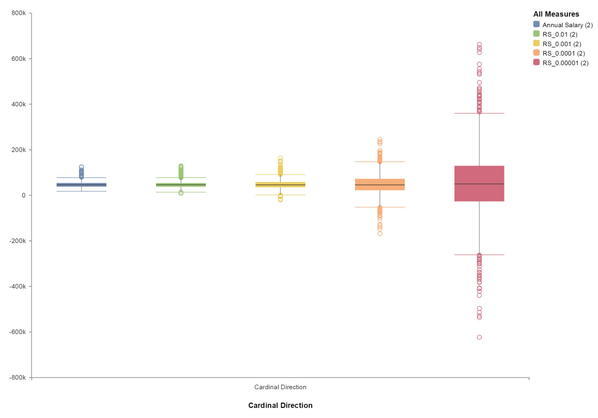


(b) Maximum salary with different values of ϵ in one global group.

Due to symmetry of the Laplace distribution, the same argument holds for the minimum operation; without countermeasures, the noisy salary value could even become negative (actually, this happens in our experiments as depicted in Figure 7.6a and 7.6b).



(a) Median salary with different values of ϵ grouped by cardinal region.



(b) Median salary with different values of ϵ in one global group.

One direct consequence of an increasing privacy level is an increasing variance of the noise values (compare with Section 7.2.1 and Figures 7.5a, 7.5b). Since this noise is added to the sensitive data, the sanitized data has also a greater variance than original data – this effect can be observed in Figures 7.8a and 7.8b. While this increasing variance increases the domain size and hence has high impact on the first and the third quartiles, the median (i.e. the second quartile) changes not nearly as extreme as the other quartiles. In more detail, for fine granularity (grouped by states), the most extreme error rate occurs in the group "Iowa", where the 1st quartile changes from $39k$ to $-66k$, the median changes from $46k$ to $10k$ and the 3rd quartile changes from $52k$ to $167k$. In contrast, for coarse granularity (only one group containing all data), the 1st quartile changes from $37k$ to $-26k$, the median changes from $46k$ to $49k$ and the 3rd quartile changes from $54k$ to $129k$.

Independent Geo-Location

In this experiment, we examined a location-based privacy mechanism. Consider, for example, a mobile network operator who can track the location of each customer. This data could be interesting for different data analysts, however, while outsourcing is prohibited due to data protection regulation, anonymized data can be outsourced. In this section we evaluate sanitization methods presented in [2] regarding security and practicability.

There are multiple ways to describe a position on the plane[2], for instance using the cartesian coordinate system in two dimensions or polar coordinate system. In the following, we assume our coordinates are given geographic coordinates consisting of latitude and longitude. This is, we assume a location is described as a point in $\mathbb{R}^2$. From a high-level perspective the location-based privacy mechanism works like the sanitization mechanisms described before: given a plain sensitive location $x \in \mathbb{R}^2$, instead of reporting x, a point $z \in \mathbb{R}^2$ is generated randomly according to a noise function.

In [2] a level of privacy is defined within a given radius, resulting in a possible informal definition of geo-indistinguishability as follows:

A mechanism satisfies ϵ-geo-indistinguishability iff for any radius $r > 0$, the user enjoys ϵr-privacy within r.

More particular, the level of privacy l is proportional of the radius r and ϵ, i.e. $l = \epsilon r$, so the smaller l the higher the privacy. Given this definition it is obvious, that ϵ depends on the unit of r. For example, assume that distances are measured in kilometers and $\epsilon = 1$. Changing the unit of distanced to meters results in a transformation of $\epsilon = 0.001$ to guarantee the same privacy level.

A modification of the general definition of differential privacy (compare Section 7.1.1) concludes to the following noise mechanism. Without going into formal details and without giving proofs, the modifications in [2] work as follows: The PDF of the used the noise mechanism for a given parameter $\epsilon = \frac{l}{r}$, the actual location $x \in \mathbb{R}^2$ and any other point $z \in \mathbb{R}^2$, is set to:

$$D_\epsilon(x)(z) = \frac{\epsilon^2}{2\pi} e^{(-\epsilon d(x,z))}$$

where $d(\cdot,\cdot)$ is the distance function between two points. Defining a mechanism for location-based privacy that samples according to this PDF satisfies ϵ-geo-indistuishability.

By standard transformation from cartesian coordinates $(x, y) \in \mathbb{R}^2$ to a system of polar coordinates $(r, \theta) \in \mathbb{R} \times [0, 2\pi]$ the PDF can be transformed to:

$$D_\epsilon(r, \theta) = \frac{\epsilon^2}{2\pi} r e^{-\epsilon r}.$$

This function has one big advantage for practical application, namely, the two random variables r and θ that represent the radius respectively the angle are independent and can be drawn

[2]This is only an approximation of the earth surface, but is accurate as long as the area of interest does not become too large.

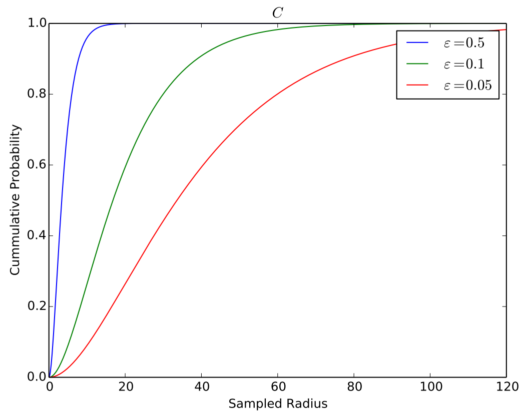
independently. This is, θ can be sampled as a uniformly distributed number in the interval of $[0, 2\pi]$ (or $[0°, 360°]$) while r is sampled from a distribution according the following probability density function

$$D_{\epsilon,R}(r) = \epsilon^2 r e^{-\epsilon r}.$$

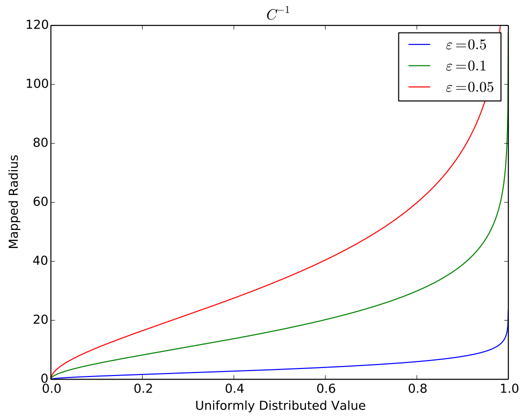
Integrating this formula results in a cumulative distribution function

$$C_\epsilon(r) = 1 - (1 + \epsilon r)e^{-\epsilon r}$$

depicted in Figure 7.9a for three different ϵ values.



(a) $C_\epsilon(r)$ for different values of ϵ.



(b) C^{-1} for different values of ϵ.

This function can be interpreted in the following way: For $\epsilon = 0.1$ (green line in Figure 7.9a), the noisy location is at most 20m away from the original location with probability roughly 65%.

Inverting the cumulative distribution function results in

$$C_\epsilon^{-1}(p) = -\frac{1}{\epsilon}(W_{-1}(\frac{p-1}{e}) + 1).$$

where W_{-1} is the Lambert W function (the -1 branch) and can be used for sampling r efficiently as noted in [2]. For different privacy parameters $\epsilon_i \in \{0.5, 0.1, 0.05\}$ this function is plotted in Figure 7.9b.

In conclusion, we sample the random noise values r and θ as follows:

r : A value p is sampled uniformly in the interval $[0, 1]$ and $C_\epsilon^{-1}(p)$ is output.

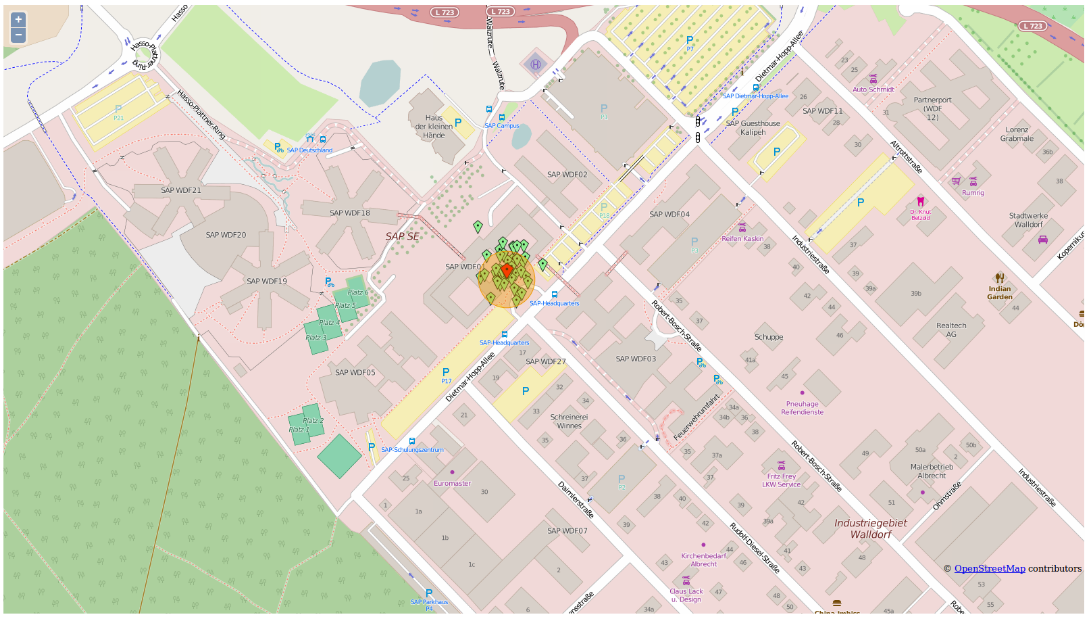
θ : A value θ is sampled uniformly in the interval $[0, 2\pi]$ and returned.

Given an original location $(x, y) \in \mathbb{R}^2$ and sampled noise, the original location is moved by direction θ and distance r resulting in a sanitized version of the location.

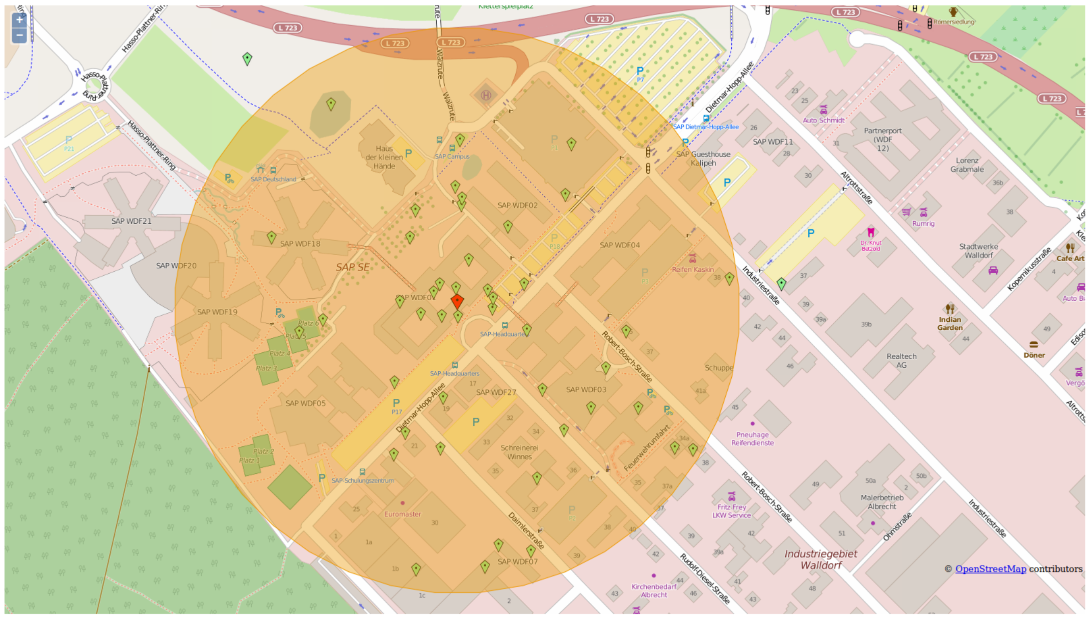
For empirical studies we have implemented a first prototype in Python 3, using NumPy[3] for efficient sampling according to the previously mentioned mechanism (particularly the Lambert W function is included in NumPy), and GeoPy[4] for location transformation and moving the location according the sampled noise. Finally, we visualize the effects of differential privacy for location-based systems utilizing OpenStreetMap[5]. We used OpenStreetMap due to its fast prototyping possibilities, but certainly other software could be used for visualization as well.

We emphasize that in our experiment, we sanitized exactly one location but not a series of locations. The original position (depicted as red marker in Figures7.10a, 7.10b) is assumed to

[3]https://pypi.python.org/pypi/numpy
[4]https://pypi.python.org/pypi/geopy
[5]https://www.openstreetmap.org



(a) $C_\epsilon(r)$ for different values of ϵ.



(b) C^{-1} for different values of ϵ.

be at “49.293551, 8.641904” (the SAP SE Headquarter) and we tested different $\epsilon_1 = 0.1$ and $\epsilon_2 = 0.01$ with fixed $l = 5$ value, so $r_1 = 50$ and $r_2 = 500$. For both sets of parameters we sanitized the original location 50 times and plotted all sanitized position at one map.

While ϵ_1 does leak the specific building our (fictional) person is located but not the exact room number, ϵ_2 does hide this information and only leaks the approximate location (e.g. the person is located on the SAP campus).

7.4 Research Questions

In recent years many research results on differential privacy have been produced. Yet, many important questions remain unsolved. Particularly, as an applied research group we are interested in applications of differential privacy to different use cases. As such, we are in search of relevant technical problems that have so far not been sufficiently solved. One example we were able to identify is text.

7.4.1 Text

Anonymization of text is a difficult task. Text contains rich information that often is not directly accessible to machines. For example, sentiment analysis can identify the attitude of the author towards its subject or stylometric analyses may identify authors. Nevertheless, text may also reveal sensitive information about its subject. Identifying all personally identifiable information is one task, but may not be sufficient, since above analyses show that significant information may be read between the lines. Hence, we are faced with a similar challenge that lead to development of differential privacy: there is no information that can be left unmodified. Consequently, methods based on differential privacy seem to be a viable approach. On the one hand, applying differential privacy to text is not straigth-forward. It is not clear how perturb a word: before or after stemming? It should be a similar word, since random character sequences probably do not make any sense, but what is similar? Furthermore, data is very high-dimensional (many words) and certainly has patterns (as already outlined above). On the other hand, being able to reliable anonymize (any) text is highly valuable and relevant. Anonymizing medical records is not the only application, but also blog posts or employee performance ratings.

7.5 Summary

We have observed that anonymization can be a valuable tool in a big data economy, but proper anonymization is hard due to a number of inherent challenges. Differential privacy is one mechanism that can provide a suitable mechanism and guarantee to balance privacy and utility. We have shown in a number of experimental studies that such mechanisms can provide sufficient privacy and utility at the same time, e.g. in computing averages or revealing geo-locations. In summary, we see differential privacy as a potential target for future product integration as well as an area for future applied research.

Bibliography

[1] Charu Aggarwal. On k-anonymity and the curse of dimensionality. In Proceedings of the 31st Very Large DataBase (VLDB) Conference, 2005.

[2] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pages 901–914. ACM, 2013.

[3] Yves-Alexandre de Montjoye, Laura Radaelli, Vivek Kumar Singh, and Alex "Sandy" Pentland. Unique in the shopping mall: On the reidentifiability of credit card metadata. Science, 347, 2015.

[4] Cynthia Dwork. Differential privacy. In Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II (ICALP), 2006.

[5] Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. Foundations and Trends in Theoretical Computer Science, 9(3-4):211–407, August 2013.

[6] Kashmir Hill. How Target Figured Out A Teen Girl Was Pregnant Before Her Father Did, 2012. http://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did/.

[7] Marek Jawurek, Martin Johns, and Konrad Rieck. Smart metering de-pseudonymization. In Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC), 2011.

[8] Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data (SIGMOD), 2009.

[9] Frank McSherry and Kunal Talwar. Mechanism Design via Differential Privacy. In 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2007.

[10] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In Proceedings of the 2008 IEEE Symposium on Security and Privacy (S&P), 2008.

[11] Aaron Roth. New Algorithms for Preserving Differential Privacy. PhD thesis, 2010.

[12] Latanya Sweeney. Weaving technology and policy together to maintain confidentiality. Journal of Law, Medicine and Ethics, 25, 1997.

[13] Stanley L. Warner. Randomized response: a survey technique for eliminating evasive answer bias. Journal of the American Statistical Association, 60(309):63–66, 1965.

7.6 Appendix - Formulary

7.6.1 Notation

Input Domain Notation:

- Value Domain: D.
- Database: $\mathbb{N}^{|D|}$.
- Histogram $i \in D$ for the vector of database A: A_i.
- Query: $f : D^n \to \mathbb{R}^d$.

Database distance: Database distance according to l^1-norm (Manhattan distance)

$$\|A\|_1 = \sum_{i=1}^{|A|} |A_i|$$

7.6.2 Composition theorems

The following is based on [5] and [8].
Sequential composition:

$$\prod_i Pr[M_i^r(A) = r_i] \leq \prod_i Pr[M_i^r(B) = r_i] \times \prod_i exp(\epsilon_i).$$

The above formulation can be optimized by lowering the bound in case that the constellation of input data sets for a mechanism is disjoint (e.g. uncorrelated). When mechanisms are combined and each mechanism is processing an arbitrary part of the disjoint input (i.e. output fields are uncorrelated), as expressed by M_i (X $\cap$ D_i) where D_i are disjoint subsets of the input domain D, ϵ is no longer derived by the sum of all individual operations but by $\max_i(\epsilon_i)$.
Parallel composition:

$$\prod_i Pr[M_i^r(A) = r_i] \leq \prod_i Pr[M_i^r(B) = r_i] \times exp\left(\max_{i=1,\ldots,r}(\epsilon_i)\right).$$

Please view the above formulations as guarantees for the privacy leakage. The analyst who is for example invoking either of the two constellations is going to have exactly the same ϵ amount deducted from his privacy budget.

7.6.3 Laplace mechanism

Probability density function: The probability density function for a random variable x is defined by

$$Lap(x|\lambda, \mu) = \frac{1}{2\lambda} exp\left(-\frac{|x-\mu|}{\lambda}\right),$$

thus for $\mu = 0$ and ϵ-differential privacy

$$Lap\left(x|(\frac{\Delta f}{\epsilon}), 0\right) = \frac{\epsilon}{2\Delta f} exp\left(-\frac{\epsilon|x|}{\Delta f}\right).$$

Sampling: A uniform distributed random variable U in the interval [1/2, 1/2] is utilized to sample from the inverse CDF.

$$X = \mu - bsign(U) \times ln(1 - 2|U|)$$

Sensitivity:

$$\Delta_1(f) = \max_{\substack{A,B \in N^{|D|} \\ \|A-B\|_1=1}} \|f(A) - f(B)\|_1$$

7.6.4 Exponential mechanism

Probability density function: By introduction of a normalizing constant in the denominator the probability density function is formulated and scaled to an interval [0,1]. The algorithm uses half ϵ to protect the event when the presence of a database record causes one utility function to increase and another to decrease. Thus leading to the following notation.

$$\mathcal{E}(\epsilon, A, q(\cdot), \epsilon) = \frac{exp\left(\frac{\epsilon q(A,r)}{2\Delta q}\right)}{\sum exp\left(\frac{\epsilon q(A,r)}{2\Delta q}\right)}$$

Sampling: Sampling can then be achieved by generating a uniformly distributed random variable in the interval [0,1], resp. $[0,\sum \exp\left(\frac{\epsilon q(A,r)}{2\Delta q}\right)]$ for the unnormalized interval, and selecting the cost function of the interval in which the random number is located.

Sensitivity:

$$\Delta q = \max_{\|A-B\|_1=1} |q(A,r) - q(B,r)|.$$

Accuracy: While the Exponential mechanism does not guarantee to pick the maximum quality function on database A (OPT_q (A)), it will return a high scoring quality function result (q(r*) in comparison to OPT_q (A) within a bound fixed by the following:

$$Pr\left[q(r*) \le OPT_q(A) - \frac{2\Delta}{\epsilon}(ln(|R|) + t)\right] \le exp(-t).$$

Accuracy can be well illustrated by the example of the author of [11] where the accuracy for picking the most common eye color of an audience via the exponential mechanism. If we define R =Blue, Red, Green, Brown, Purple than the result will provide an eye color that is shared by OPT_q(A)$-\frac{2}{\epsilon}(ln(5) + 3)$ people. The error guarantee is thus mainly depending on R (i.e. the discretization of the possible result range). Which implies that in the above example, the accuracy guarantee is independent of number of people and thus the comparative error is small if database is large.

7.6.5 Randomized response

Privacy: À posteriori ϵ inference based on study design. For example for two coin flips:

$$\epsilon = ln\left(\frac{Pr[\text{Response} = \text{Desired outcome}|\text{Truth} = \text{Desired outcome}]}{Pr[\text{Response} = \text{Desired outcome}|\text{Truth} \neq \text{desired outcome}]}\right).$$

8. Definitions of privacy: Empirical Evaluation of Privacy via Website Fingerprinting

Anonymous networks, such as those to be built in PANORAMIX WP4, face several threats from traffic analysis, and a method to evaluate their security is needed. One key threat against low-latency anonymity systems are fingerprinting attacks, which enables an attacker to infer the source of a web page or other accessed resource or communication. In the litrature those are termed 'website fingerprinting attacks' due to their applicability in determining the website borwsed anonymously in through exisitng systems such as Tor. In this chapter, we present a new website fingerprinting attack based on fingerprints extracted from random decision forests and its evaluation. The proposed attack performs better than current state-of-the-art attacks even against website fingerprinting defenses. Investigation of possible attacks against anonymous protocols informs the development of the PANORAMIX infrastracture to protect against this type of attacks, particularly to support low-latency mixing in PANORAMIX WP7. In this chapter we show that none of the existing defences are entirely safe, requiring PANORAMIX to develop novel approaches.

8.1 Introduction

Traditional encryption obscures only the content of communications and does not hide metadata such as the time, size and direction of traffic. Anonymous communication systems obscure both content and metadata, preventing a passive attacker from being able to infer the source or destination of communication.

Anonymous communications tools, such as Tor [10], route traffic through relays to hide its ultimate destination. Tor is designed to be a low-latency system to support interactive activities such as instant messaging and web browsing, and does not significantly alter the shape of network traffic. This allows an attacker to exploit information leaked via the order, timing and volume of resources requested from a website. As a result, many works have shown that website fingerprinting attacks are possible even when a client is doing encrypted browsing or using an anonymity tool such as Tor [27, 17, 14, 19, 24, 7, 35, 15, 34, 32].

Website fingerprinting is commonly formulated as a classification problem. An attacker wishes to know whether a client browses one of n web pages. The attacker first collects many examples of traffic traces from each of the n web pages by performing web-requests through the protection mechanism under attack; features are extracted and a machine learning algorithm is trained to classify the website using those features. When a client browses a web page, the attacker passively collects the traffic, passes it in to their classifier and checks if the client visited one of the n web pages. In the literature this is referred to as the closed-world scenario – a client is restricted to browse a limited number of web pages, monitored by the attacker. However, the closed-world model has been criticised for being unrealistic [15, 25] since a client is unlikely to only browse a limited set of web pages. The open-world scenario attempts to model a more realistic set-up where the attacker monitors a small number of web pages, but allows a client to additionally browse to a large world size of unmonitored web pages.

Despite some preliminary work by Panchenko et al. [24], there is a notable absence of feature analysis in the website fingerprinting literature. Instead features are picked based on heuristic arguments. Once features and a classifier have been chosen the pipeline is simple: an attacker trains on a corpus of previously collected traffic instances, and waits to collect test traces from which they infer what web page a client is browsing. Techniques such as Naive-Bayes [14], k-Nearest Neighbour [34], decision tree [15], SVM [24] and N-grams [11] have all been used to perform website fingerprinting attacks.

Our attack uses random decision forests [5], an ensemble method using multiple decision trees. We use random forests because they have been shown to perform well in classification tasks [13], [28], [16] and allow for analysis of feature importance [12]. Furthermore, they allow us to extract fingerprints to perform identification in an open-world.

The key contributions of this work are as follows:

- In section 8.3.3 we present a new attack, k-fingerprinting, based on extracting a fingerprint for a web page via random forests. We show k-fingerprinting is more accurate and faster than other state-of-the-art website fingerprinting attacks [34], [7].

- In section 8.5 we perform analysis of the features used in this and prior work to determine which yield the most information about an encrypted or anonymized web page. We show that simple features such as counting the number of packets in a sequence leaks more information about the identity of a web page than complex features such as packet ordering or packet inter-arrival time features.

- We consider a larger open-world setting than has been considered in prior works. Previously the largest open-world study considered 5,000 unmonitored web pages [34]. In section 8.7 we experiment with an open-world size of 100,000 collected via Tor while in 8.8 and 8.8.3 we experiment with open-world sizes of 7,000 and 17,000 collected via a standard web browser, reflecting a more realistic website fingerprinting attack over multiple browsing sessions. Section 8.7 contains an open-world size that is an order of magnitude larger than the current largest open-world website fingerprinting work [34] [1].

- In section 8.7 we show that an attacker need only train on a small fraction of the total data to achieve a low false positive rate, greatly reducing the start-up cost an attacker would need to perform the attack.

- In section 8.9 we observe that the error rate is uneven and so it may be advantageous to throw away some training information that could confuse a classifier. An attacker can learn the error rate of their attack from the training set, and use this information to select which web pages they wish to monitor in order to minimize their error rates.

- In section 8.10 we evaluate k-fingerprinting against many popular website fingerprinting defenses and show it outperforms the state-of-the-art attack k-NN [34].

- In section 8.11 we show training k-fingerprinting is an order of magnitude faster than the state-of-the-art attack k-NN [34].

[1] [15] considers an open world size of ∼35K but only tried to separate monitored pages from unmonitored pages instead of further classifying the monitored pages to the correct website. The authors assume the adversary monitors four pages: google.com, facebook.com, wikipedia.org and twitter.com. They trained a classifier using 36 traces for each of the Alexa Top 100 web pages, including the web pages of the monitored pages. The four traces for each of the monitored sites plus one trace for each of the unmonitored sites up to ∼35K are used for testing.

- We confirm that browsing over Tor does not provide any additional protection against fingerprinting attacks over browsing using a standard web browser. Furthermore we show that k-fingerprinting is highly accurate on Tor hidden services as well as standard web pages, and that Tor hidden services can be distinguished from standard web pages.

8.2 Related Work

Website fingerprinting has been studied extensively. Early work by Wagner and Schneier [30], Cheng and Avnur [9] exposed the possibility that encrypted HTTP GET requests may leak information about the URL, conducting preliminary experiments on a small number of websites. They asked clients in a lab setting to browse a website for 5-10 minutes, pausing two seconds between page loading. With caching disabled they were able to correctly identify 88 pages out of 92 using simple packet features. Early website fingerprinting defenses were usually designed to safeguard against highly specific attacks. In 2009, Wright et al. [36] designed 'traffic morphing' that allowed a client to shape their traffic to look as if it was generated from a different website. They were able to show that this defense does well at defeating early website fingerprinting attacks that heavily relied on exploiting unique packet length features [27, 17].

In a similar fashion, Tor pads all packets to a fixed-size cells of 512 bytes. Tor also implemented randomized ordering of HTTP pipelines [26] in response to the attack by Panchenko et al. [24] who used packet ordering features to train an SVM classifier. This attack on Tor achieved an accuracy of 55%, compared to a previous attack that did not use such fine grained features achieving 3% accuracy on the same data set using a Naive-Bayes classifier [14]. Other defenses such as the decoy defense [24] loads a camouflage website in parallel to a legitimate website, adding a layer of background noise. They were able to show using this defense attack accuracy of the SVM again dropped down to 3% despite using intelligent features such as packet orderings.

Luo et al. [20] designed the HTTPOS fingerprinting defense at the application layer. HTTPOS acts as a proxy accepting HTTP requests and obfuscating them before allowing them to be sent. It modifies network features on the TCP and HTTP layer such as packet size, packet time and payload size, along with using HTTP pipelining to obfuscate the number of outgoing packets. They showed that HTTPOS was successful in defending against a number of classifiers [4, 8, 17] and [27].

More recently Dyer et al. [11] created a defense, BuFLO, that combines many previous countermeasures, such as fixed packet sizes and constant rate traffic. Dyer et al. showed this defense improved upon other defenses at the expense of a high bandwidth overhead. Cai et al. [6] made modifications to the BuFLO defense based on rate adaptation again at the expense of a high bandwidth overhead. Following this Nithyanand et al. [22] proposed Glove, that groups website traffic into clusters that cannot be distinguished from any other website in the set. This provides information theoretic privacy guarantees and reduces the bandwidth overhead by intelligently grouping web traffic in to similar sets.

Cai et al. [7] modified the kernel in Panchenko et al.'s SVM to improve an attack on Tor, and was further improved in an open-world setting by Wang and Goldberg in 2013 [35], achieving a True Positive rate of over 0.95 and a False Positive rate of 0.002 when monitoring one web page. Wang et al. [34] conducted attacks on Tor using large open-world sets. Using a k-nearest neighbour classifier they achieved a True Positive rate of 0.85 and False Positive rate of 0.006 when monitoring 100 web pages out of 5100 web pages. More recently Wang and Goldberg [33] suggested a defense using a browser in half-duplex mode – meaning a client cannot send multiple requests to servers in parallel. In addition to this simple modification they add random padding and show they can even foil an attacker with perfect classification accuracy with a comparatively (to other defenses) small bandwidth overhead. Finally Wang and Goldberg [32] took website fingerprinting attacks out of the lab. By maintaining an up-to-date training set and splitting a full packet sequence in to components comprising of different web page load traces they show that practical website fingerprinting attacks are possible. By considering a time gap of 1.5 seconds between web page loads, their splitting algorithm can successfully parse a single packet sequence in to multiple packet sequences with no loss in website fingerprinting accuracy.

Website fingerprinting defenses attempt to make all packet sequences look as similar as possible to foil classifiers, at the expense of bandwidth and latency. Website fingerprinting defenses can be separated into two categories, simulatable and non-simulatable [34]. Simulatable defenses operate on an input packet sequence and output another packet sequence, based upon packet features such as direction, size and time. Their advantage is that they do not have to be applied from applications that have access to sensitive client data, such as an extension in the browser. They do not require any more information than would be available to an attacker. Examples of simulatable defenses include BuFLO [11], CS-BuFLO [6], background noise [24], Tor packet padding, traffic morphing [36]. Non-simulatable defenses applied at the application layer include HTTPOS [20] and Tor's randomization of packet orderings. Both types of defense come at the expense of bandwidth or time overhead and may not be tolerable to the average client wishing to browse online with little latency. For example BuFLO pads all packets to a fixed size, leading to a bandwidth overhead of 190%.

8.3 Attack Design

We consider an attacker that can passively collect a client's encrypted or anonymized web traffic, and aims to infer which web resource is being requested. Dealing with an open-world, makes approaches based purely on classifying previously seen websites inapplicable. Therefore k-fingerprinting aims to define a distance-based classifier, similar to the k-NN [34] approach. It manages unbalanced sized classes and assigns meaningful distances between packet sequences, where close-by 'fingerprints' denote requests likely to be for the same resources.

8.3.1 Threat model

We make the following usage assumptions following Juarez et al. [15]: The client browses to one web page at a time, and does not perform multi-tab browsing. The attacker is able to perfectly infer the start and end of the page load (for our data sets we chose a cut off point of 20 seconds after which an attacker would stop recording traffic). The client browses the web but does not perform any other actions that create network traffic such as downloading via BitTorrent or using VoIP.

The only information that the attacker may extract from the observed web-browsing activity is the timing and volume of incoming and outgoing traffic, as transformed by the protection mechanism chosen. For example, an attacker observing Tor will be observing padded cells, while an attacker observing web-browsing under traffic morphing [36] may be observing payloads that are padded so that they conform to a specified target set of web pages.

The attacker is able to use the protection mechanism under study to retrieve a number of pages under observation, as well as a number of other random pages, to use as training data. Furthermore, the network conditions under which these training traces are requested are indistinguishable from, or can be made arbitrarily similar to, the network conditions under which target clients will be performing requests.

8.3.2 Extracting k-fingerprints from random forests

Random forests are a classification technique consisting of an ensemble of decision trees, taking a consensus vote of how to classify a new object. They have been shown to perform well in classification, regression [16], [5] and anomaly detection [18]. Each tree in the forest is trained using labeled objects represented as feature vectors of a fixed size. Training includes some randomness to prevent over-fitting: the training set for each tree is sampled from the available training set with replacement. Due to the bootstrap sampling process there is no need for k-fold cross validation to measure k-fingerprinting performance, it is estimated via the unused training samples on each tree [5]. This is referred to as the *out-of-bag* score.

In this work we use random forests to extract a fingerprint for each traffic instance, instead of using directly the classification output of the forest. We define a distance metric between two traces based on the output of the forest: given a feature vector each tree in the forest associates a leaf identifier with it, forming a vector of leaf identifiers for the item, which we refer to as the *fingerprint*.

Once fingerprint vectors are extracted for two traces, we use the Hamming[2] distance to calculate the distance between these fingerprints[3].

We classify a test instance as the label of the closest k training instances via the Hamming distance of fingerprints – assuming all labels agree. We evaluate the effect of varying k, the number of fingerprints used for comparison, in sections 8.6 and 8.8.

This leafs vector output from a trained random forest classifier represents a robust fingerprint: we expect similar traffic sequences are more likely to fall on the same leaves than dissimilar traffic sequences. This is the case since the forest has been trained on a classification task, thus selecting decision branches that keep traces from the same websites in the same leafs, and those from different ones apart.

We can vary the number of training instances k a fingerprint should match, to allow an attacker to trade the True Positives for False Positives. This is not possible using directly the classification of the random forest. By using a k closest fingerprint technique for classification, the attacker can choose how they wish to decide upon final classification[4]. For the closed-world scenario we do not need the additional fingerprint layer for classification, we can simply use the classification output of the random forest since all classes are balanced and our attack does not have to differentiate between False Positives and False Negatives. For the closed-world scenario we measure the mean accuracy of the random forest on the given test data and labels.

8.3.3 The k-fingerprinting attack

The k-fingerprinting attack proceeds as follows: The attacker chooses which web pages they wish to monitor and captures network traffic generated via loading the monitored web pages and some unmonitored web pages. These target traces for monitored websites, along with many traces for unmonitored websites, are used to train a random forest for classification. Given a packet sequence representing each training instance of a monitored web page, it is converted to a fixed length fingerprint as described in Section 8.3.2 and stored.

The attacker now passively collects instances of web page loads from a client's browsing session. A fingerprint is extracted from the newly collected packet sequence, as described in section 8.3.2. The attacker then computes the Hamming distance of this new fingerprint against the corpus of fingerprints collected during training. In the open-world scenario we follow the Wang et al. [34] method for final classification. For each test instance with a given leaf vector fingerprint, we select the k training instances with minimum Hamming distances to this leaf vector. A test instance is classified as a monitored page if and only if all k fingerprints agree on classification, otherwise the test instance is classified as an unmonitored page.

We define performance measures for the attack as follows:

[2]We experimented with using the Hamming, Euclidean, Mahalanobis and Manhattan distance functions and found Hamming to provide the best results.

[3]For example, given the Hamming distance function $d : V \times V \to \mathbb{R}$, where V is the space of leaf symbols, we expect given two packet sequences generated from loading *google.com*, with fingerprints vectors f_1, f_2 and a packet sequence generated from loading *facebook.com* with fingerprint f_3, that $d(f_1, f_2) < d(f_1, f_3)$ and $d(f_1, f_2) < d(f_2, f_3)$.

[4]We chose to classify a traffic instance as a monitored page if all k fingerprints agree on the label, but an attacker could choose some other metric such as majority label out of the k fingerprints.

- **True Positive Rate.** The probability that a monitored page is classified as the correct monitored page.
- **True Negative Rate.** The probability that an unmonitored page is correctly classified as an unmonitored page.
- **False Positive Rate.** The probability that an unmonitored page is incorrectly classified as a monitored page.
- **False Negative Rate.** The probability that a monitored page is incorrectly classified as a different monitored page or an unmonitored page.

8.4 Data gathering

We chose to collect two data sets, one collected via Tor, DS_{Tor}, and one collected via a standard web broswer, DS_{Norm}. DS_{Norm} consists of 30 instances from each of 55 monitored web pages, along with 17,000 unmonitored web pages chosen from Alexas top 20,000 web sites [1]. We collected DS_{Norm} using a number of `Amazon EC2 instances`[5], `Selenium`[6] and the headless browser `PhantomJS`[7]. We used `tcpdump`[8] to collect network traces for 20 seconds with 2 seconds between each web page load. Monitored pages were collected in batches of 30 and unmonitored web pages were collected successively. Page loading was performed with no caches and time gaps between multiple loads of the same web page, as recommended by Wang and Goldberg [35]. We chose to monitor web pages from Alexa's top 100 web sites [1] to provide a comparison with the real world censored web pages used in the Wang et al. [34] data set. DS_{Tor} was collected in a similar manner to DS_{Norm} but was collected via the Tor browser. DS_{Tor} consists of two subsets of monitored web pages: (i) 100 instances from each of the 55 top Alexa monitored web pages and (ii) 80 instances from each of 30 popular Tor hidden services[9]. The unmonitored set is comprised of the top 100,000 Alexa web pages, excluding the top 55.

For comparison to previous work, we also use the Wang et al. data set [34], which collected 90 instances from each of 100 monitored sites, along with 5000 unmonitored web pages. The Wang et al. monitored web pages are various real-world censored websites from UK, Saudi Arabia and China providing a realistic set of web pages an attacker[10] may wish to monitor. The unmonitored web pages are chosen at random from Alexa's top 10,000 websites – with no intersection between monitored and unmonitored web pages.

This allows us to validate k-fingerprinting on two different data sets while allowing for direct comparison against the state-of-the-art k-Nearest Neighbour attack [34]. We can also infer how well the attack works on censored web pages which may not have small landing pages or be set up for caching like websites in the top Alexa list. Testing k-fingerprinting on both real-world censored websites and top alexa websites indicates how the attack performs across a wide range of websites.

We vary the number of stored fingerprints k between 1 and 10 and vary the number of unmonitored pages we train on: for the attack on 7000 unmonitored web pages we train between 1 and 6500 unmonitored pages, for the attack on 17,000 unmonitored web pages we train between 1000 and 15,000 unmonitored pages, for the attack on 100,000 unmonitored web pages we train between 2000 and 16,000 unmonitored pages and for the Wang et al. [34] data set we train

[5]https://aws.amazon.com/ec2/
[6]http://www.seleniumhq.org/
[7]http://phantomjs.org/
[8]http://www.tcpdump.org/
[9]A Tor hidden service is a website that is hosted on a Tor relay and so both server and client remain anonymous to one another and any external observers. We chose hidden services to fingerprint based on popularity as listed by the .onion search engine http://www.ahmia.fi/
[10]For example an ISP or nation state.

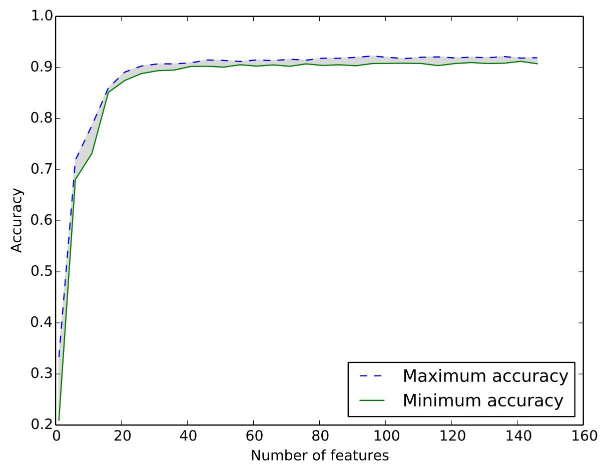


Figure 8.1: Accuracy of k-fingerprinting in a closed-world setting as the number of features is varied.

between 1 and 4500 unmonitored pages. The variations in unmonitored training instances simulates different scenarios under which an attacker can train on different world sizes. We show that an attacker need only train on a small fraction of the unmonitored web pages to achieve a low false positive rate.

For the sake of comparison, according to a study by research firm Nielsen [3] the number of unique websites visited per month by an average client in 2010 was 89. Another study [23, 15] collected web site statistics from 80 volunteers in a virtual office environment. Traffic was collected from each volunteer for a total of 40 hours. The mean unique number of websites visited per volunteer was 484, this is substantially smaller than the world sizes we consider in our experiments. However, we note that the data was collected in a lab setting that may not realistically reflect a clients browsing habits.

8.5 Feature selection

Our first contribution is a systematic analysis of feature selection. All experiments in this section were performed with the Wang et al. data set [34] so as to allow direct comparison with their attack results.

We train a random forest classifier in the closed-world setting using a feature vector comprised of features in the literature, and labels corresponding to the monitored sites. We use the gini coefficient as the purity criterion for splitting branches and estimate feature importance using the standard methodology described by Breiman [2], [5], [12]. Each time a decision tree branches on a feature the weighted sum of the gini impurity index for the two descendent nodes is higher than the purity of the parent node. We add up the gini decrease for each individual feature over the entire forest to get a consistent measure of feature importance.

We explain each feature used and following this perform feature analysis. Some of the features in the feature set have different lengths due to the different lengths of packet sequences, in this case we pad these features with 0's, and extract a feature vector of length 150 from every packet sequence.

Figure 8.1 illustrates the effect of using a subset of features for random forrest classification. A number of experiments were performed by training a random forest classifier to establish feature importance; and then training a new random forest with only a subset of the most informative features. More specifically, we train a random forest using subsets of the most informative features in batches of five. As we increase the number of features used we observe a monotonic increase in accuracy; however there are diminishing returns as we can achieve nearly the same accuracy using the 30 most important features, as when using more. Though we could have achieved near same accuracy with an order of magnitude fewer features, we chose to use 150 features because the difference in training time when using less features was negligible.

Figure 8.2 identifies the top-20 ranked features and illustrates their variability across 100 repeated experiments. As seen in figure 8.1 there is a reduction in gradient when combining the top 15 features compared to using the top 10 features. Figure 8.2 shows that the top 13 features are comparatively much more important than the rest of the top 20 features, hence there is only a slight increase in accuracy when using the top 15 features compared to using the top 10. After the drop between the rank 13 and rank 14 features, feature importance falls steadily until feature rank 40, after which the reduction in feature importance is less prominent[11]. Note that there is some interchangeability in rank between features, we assign ranks based on the average rank of a feature over the 100 experiments.

Feature set list

Feature importance was computed for each feature over 100 experiments, we order them by the mean feature importance score. From each packet sequence we extract the following features:

- **Number of packets statistics.** We extract the total number of packets, along with the number of incoming and outgoing packets for the total transmission. These features are used in [34, 24, 11]. The number of incoming packets during transmission is the most important feature, and together with the number of outgoing packets during transmission are always two of the five most important features. The total number of packets in transmission has rank 10.

- **Incoming & outgoing packets as fraction of total packets.** The number of incoming and outgoing packets as a fraction of the total number of packets. A variation of this feature is used in [24]. These are always two of the five most important features.

- **Packet ordering statistics.** For each successive incoming and outgoing packet we include a feature that indicates the total number of packets seen before it in the sequence. Variations of these features are used in [34, 24] and [7]. The standard deviation of the ougoing packet ordering list is the most important of these features with rank 4, the average of the ougoing packet ordering list has rank 7. The standard deviation of the incoming packet ordering list has rank 12 and the average of the incoming packet ordering list has rank 13.

- **Concentration of outgoing packets.** We split the packet sequence into non-overlapping chunks of 20 packets. We then count the number of outgoing packets in each of these chunks. We extract along with the entire chunk sequence, the standard deviation, mean, median and max of the sequence of chunks. This provides a snapshot of where outgoing packets are concentrated. A variant of this feature is used in [34]. The features that make up the concentration list are between the 15th and 30th most important features, but also make up the bulk of the 75 least important features. The concentration list mean has rank 11, the standard deviation has rank 16, the maximum has rank 30 and the median has rank 65.

- **Concentration of incoming & outgoing packets in first & last 30 packets.** We count the number of incoming and outgoing packets in the first and last 30 packets. A variation of this feature is used in [34]. The number of incoming and outgoing packets in the first thirty packets has rank 19 and 20, respectively. The number of incoming and outgoing packets in the last thirty packets has rank 50 and 55, respectively.

[11]The total feature importance table is shown in Appendix 8.14.1.

- **Number of packets per second.** We count the number of packets per second, along with the mean, standard deviation, min, max, median. The standard deviation feature has rank 38, maximum has rank 42, mean has rank 44, median has rank 50 and minimum has rank 117.

- **Alternative concentration features.** This subset of features is based on the concentration of outgoing packets feature list. We split the outgoing packets feature list in to 20 evenly sized subsets and sum each subset. This creates a new list of features. Similarly to the concentration feature list, the alternative concentration feature list are regularly in the top 20 most important features and bottom 50 features. Note though concentration features are never seen in the top 15 most important features whereas alternative concentration features are - at rank 14 and 15 - so information is gained by summing the concentration subsets.

- **Packet inter-arrival time statistics.** For the total, incoming and outgoing packet streams we extract the lists of inter-arrival times between packets. For each list we extract the max, mean, standard deviation, and third quartile. A variation of this feature is used in [4]. These features have rank between 40 and 70.

- **Transmission time statistics.** For the total, incoming and outgoing packet sequences we extract the first, second, third quartile and total transmission time. This feature is used in [34]. These features have rank between 30 and 50. The total transmission time for incoming and outgoing packet streams are the most important out of this subset of features.

- **Alternative number of packets per second features.** For the number of packets per second feature list we create 20 even sized subsets and sum each subset. The sum of all subsets is the 9th most important feature. The features produced by each subset are in the bottom 50 features - with rank 101 and below. The important features in this subset are the first few features with rank between 66 and 78, that are calculated from the first few seconds of a packet sequence.

Our analysis concludes that the total number of incoming packets is the most informative feature. This is expected as different web pages have different resource sizes, that are poorly hidden by encryption or anonymization. The number of incoming and outgoing packets as a fraction of the total number of packets are also informative for the same reason. After the inclusion of the 40 most important features, using additional features gives only incremental increases in accuracy.

The least important features are from the padded concentration of outgoing packets list, since the original concentration of outgoing packets lists were of non-uniform size and so have been padded with zeros to give uniform length. Clearly, if most packet sequences have been padded with the same value this will provide a poor criterion for splitting, hence being a feature of low importance. Packet concentration statistics, while making up the bulk of “useless features” also regularly make up a few of the top 30 most important features, they are the first few items that are unlikely to be zero. In other words, the first few values in the packet concentration list do split the data well.

Packet ordering features have rank 4, 7, 12 and 13, indicating these features are a good criterion for splitting. Packet ordering features exploit the information leaked via the way in which browsers request resources and the end server orders the resources to be sent. This supports conclusions in [34], [7] about the importance of packet ordering features.

We also found that the number of incoming and outgoing packets in the first thirty packets, with rank 19 and 20, were a more important feature than the number of incoming and outgoing packets in the last thirty packets, with rank 50 and 55. In the alternative number packets per

Table 8.1: k-fingerprinting results for k=3 while varying the number of unmonitored training pages.

Training pages	**True Positive rate**	**False Positive rate**
0	0.90 ± 0.02	0.750 ± 0.010
1500	0.88 ± 0.02	0.013 ± 0.007
2500	0.88 ± 0.01	0.007 ± 0.001
3500	0.88 ± 0.01	0.005 ± 0.001
4500	0.87 ± 0.02	0.009 ± 0.001

second feature list the earlier features were a better criterion for splitting than the later features in the list. This supports claims by Wang et al. [34] that the beginning of a packet sequence leaks more information than the end of a packet sequence. In contrast to Bissias et al. [4] we found packet inter-arrival time statistics, with rank between 40 and 70, only slightly increase the attack accuracy, despite being a key feature in their work.

8.6 k-fingerprinting the Wang et al. data set

We first evaluate k-fingerprinting on the Wang et al. data set [34]. This data set was collected over Tor, and thus implements padding of packets to fixed-size cells (512-bytes) and randomization of request orders [26]. Thus the only information available to k-fingerprinting is full cell timing and volume features. As described in section 8.4 there are 100 monitored web pages and 5000 unmonitored web pages in the Wang et al. data set. We train on 60 out of the 90 instances for each monitored page; we vary the number of unmonitored pages on which we train. For the attack evaluation we use fingerprints of length 200 and 150 features. Final classification is as described in section 8.3.3, if all k fingerprints agree on classification a test instance is classified as a monitored web page, otherwise it is classified as an unmonitored web page.

The k-NN classifier [34] is similar to k-fingerprinting. The classifier is trained upon a set of labelled packet sequences $D_{train} = \{P_1, ..., P_n\}$, then given a new packet sequence Q_1 the classifier computes the distance with all training points $d(Q_1, P_i)$ for $i \in \{1, .., n\}$. Q_1 is then classified as the label of the k closest training instances. Wang et al. use a weighted distance function that learns weights that discriminate against features that do not provide much information. We show that k-fingerprinting performs better than the state-of-the-art k-NN classifier [34]. k-fingerprinting also requires fewer features than the k-NN attack – although most of the features used in k-NN are redundant when attacking Tor. The k-NN attack uses their weighting scheme to generate features that allows packet size features to be ignored.

8.6.1 Attack on Tor

The scenario for the attack is as follows: an attacker, within the threat model described in section 8.3.1, monitors 100 web pages; they wish to know whether a client is visiting one of those pages, and establish which one. The client can browse to any of these web pages or to 5000 unmonitored web pages, which the attacker one classifies in bulk as an unmonitored page.

Using the k-fingerprinting method for classifying a web page we measure a True Positive rate of 0.88 ± 0.01 and a False Positive rate of 0.005 ± 0.001 when training on 3500 unmonitored web pages and k, the number of training instances used for classification, set at k=3. k-fingerprinting achieves better accuracy than the state-of-the-art k-NN attack that has a True Positive rate of 0.85 ± 0.04 and a False Positive rate of 0.006 ± 0.004. Given a monitored web page k-fingerprinting will misclassify this page 12% of the time, while k-NN will misclassify with 15% probability.

Best results are achieved when training on 3500 unmonitored web pages. Table 8.1 reports True and False Positive rates when using different numbers of unmonitored web pages for training with $k = 3$. As we train more unmonitored web pages we decrease our False Positive rate with almost no reduction in True Positive rate. After training 3500 unmonitored pages there is no decrease in False Positives and so no benefit in training more unmonitored web pages. This scheme allows an attacker to decrease False Positives at a cost of decreasing True Positives. This allows an attacker to tune the classifier to either low False Positives or high True Positives depending on the desired application of the attack.

Figure 8.3 illustrates how classification accuracy changes as, k, the number of fingerprints used for classification changes. For a low k the attack achieves a high True Positive rate with high False Positives, as we increase the value of k we reduce the number of misclassifications since it is less likely that all k fingerprints will belong to the same label, but we also reduce the number of True Positives. We find that altering the number of fingerprints used for classification, k, affects the True Positive and False Positive rate more than the number of unmonitored training pages. This suggests that while it is advantageous to have a large world size of unmonitored pages, increasing the number of unmonitored training pages does not increase accuracy of the classifier dramatically. This supports Wang et al.'s [34] claims to the same effect.

Closed-World. In the closed-world scenario in which the client can only browse within the 100 monitored web pages we achieve 0.91 ± 0.01 accuracy. This is comparable to the k-NN accuracy of 0.91 ± 0.03. If we were to use the random forest for final classification in the open-world scenario we would falsely inflate our attack accuracy, since the unmonitored class is much larger than any of the monitored classes. For the closed-world scenario we do not need the additional fingerprint layer for classification, and simply use the classification output of the random forest.

Fingerprint length. Changing the length of the fingerprint vector will affect k-fingerprinting accuracy. For a small fingerprint length there may not be enough diversity to provide an accurate measure of distance over all packet sequences. Figure 8.4 shows the resulting True Positive rate and False Positive rate as we change the length of fingerprints in the Wang et al. [34] data set. The attack and set up is the same as in section 8.6.1, we train on 60 out the 90 instances for each monitored web page. We set k=1 and train on 4000 unmonitored web pages. Using only a fingerprint of length one results in a True Positive rate of 0.51 and high False Positive rate of 0.904. Clearly using a fingerprint of length one results in a high False Positive rate since there is a small universe of leaf symbols from which to create the fingerprint. A fingerprint of length 20 results in a True Positive rate of 0.87 and low False Positive rate of 0.013. After this there are diminishing returns for increasing the length of the fingerprint vector.

8.7 Attack evaluation on DS_{Tor}

We now evaluate k-fingerprinting on DS_{Tor}. First we evaluate the attack given a monitored set of the top 55 Alexa web pages, with 100 instances for each web page. Then we evaluate the attack given a monitored set of 30 Tor hidden services, with 80 instances for each hidden service. The unmonitored set remains the same for both evaluations, the top 100,000 Alexa web pages with one instance for each web page.

8.7.1 Alexa web pages monitored set

Table 8.2 shows the accuracy of k-fingerprinting as the number of unmonitored training pages is varied. For the monitored web pages, 70 instances per web page were trained upon and testing was done on the remaining 30 instances of each web page. As expected, the false positive rate decreases as the number of unmonitored training samples grows. Similarly to section 8.6.1 there is only marginal loss in terms of true positives while we see a large reduction in the false positive rate as the number of training samples grows. Meaning an attacker will not have to compromise on true positives to decrease the false positive rate; when scaling the number of unmonitored training samples from 2% to 16% of the entire set the true positive rate decreases from 93% to 91% while the false positive rate decreases from 3.2% to 0.3%.

Table 8.2: Attack results on top Alexa sites for k=2 while varying the number of unmonitored training pages.

Training pages	True Positive rate	False Positive rate
2000	0.93 ± 0.03	0.032 ± 0.010
4000	0.93 ± 0.01	0.018 ± 0.007
8000	0.92 ± 0.01	0.008 ± 0.002
16000	0.91 ± 0.02	0.003 ± 0.001

Clearly the attack will improve as the number of training samples grows, but in reality an attacker may have limited resources and training on a significant fraction of 100,000 web pages may be unfeasible. Figure 8.5 shows the true positive and false positive rate of k-fingerprinting as the number of unmonitored web pages used for testing grows while the number of unmonitored web pages used for training is kept at 2000, for different values of k. We may think of this as the evaluation of success of k-fingerprinting as a client browses to more and more web pages over multiple browsing sessions. Again 70 out of 100 instances were used for training for each monitored web page. Clearly for a small k, both true positives and false positives will be comparitively high. Given that, with k=5 only 2.5% of unmonitored web pages are falsely identified as monitored web pages, out of 100,000 unmonitored web pages. Both the true positive rates and false positive rates remain steady regardless on the number of unmonitored web pages; an attacker can arbitrarily reduce the false positive rate by increasing the number of neighbours used for comparison, albeit at the expense of the true positive rate.

8.7.2 Hidden services monitored set

Table 8.3 shows the accuracy of k-fingerprinting as the number of unmonitored training pages is varied. For the monitored set, 60 instances per hidden service were trained upon and testing was done on the remaining 20 instances of each hidden service. Again we see a marginal loss in terms of true positives while we see a large reduction in the false positive rate as the number of training samples grows. When scaling the number of unmonitored training samples from 2% to 16% of the entire set the true positive rate decreases from 82% to 81% while the false positive rate decreases by an order of magnitude from 0.2% to 0.02%. Meaning when training on 16% of the unmonitored set only 16 unmonitored web pages out of 84,000 were misclassified as a Tor hidden service. In comparison to the Alexa web pages monitored set the true positives is around 10% lower, while the false positive rate is also vastly reduced. This is clear evidence that Tor hidden services are easy to distinguish from standard web pages loaded over Tor.

Table 8.3: Attack results on Tor hidden services for k=2 while varying the number of unmonitored training pages.

Training pages	True Positive rate	False Positive rate
2000	0.82 ± 0.03	0.0020 ± 0.0015
4000	0.82 ± 0.04	0.0007 ± 0.0006
8000	0.82 ± 0.02	0.0002 ± 0.0001
16000	0.81 ± 0.02	0.0002 ± 0.0002

Similarly to figure 8.5, figure 8.6 shows the true positive and false positive rate of k-fingerprinting as the number of unmonitored web pages used for testing grows while the number of unmonitored web pages used for training is kept at 2000, for different values of k. Monitored training was done on 60 out of the 80 instances, with the remaining 20 left for testing. Both the true positive rate and false positive rate is lower than in figure 8.5. For example, given

100,000 unmonitored pages, using k=5, the false positive rate is 0.2% which equates to only 200 unmonitored pages being falsely classified as monitored pages.

It is clear that an attacker need only train on a small fraction of data to launch a powerful fingerprinting attack. It is also clear that Tor hidden services are easily distinguished from standard web pages, rendering them vulnerable to website fingerprinting attacks. We attribute the lower false positive rate of Tor hidden services when compared to a monitored training set of standard web page traffic to this distinguishability. A standard web page is more likely to be confused with another standard web page than a Tor hidden service.

8.8 Attack evaluation on DS_{Norm}

Besides testing on DS_{Tor} and the Wang et al. [34] data set we tested the efficacy of k-fingerprinting on DS_{Norm}. This allows us to establish how accurate k-fingerprinting is over a standard web browsing session.

8.8.1 Attack on encrypted browsing sessions

An encrypted browsing session does not pad packets to a fixed size and the attacker may extract the following features in addition to time features:

- **Size transmitted.** For each packet sequence we extract the total size of packets transmitted, in addition, we extract the total size of incoming packets and the total size of outgoing packets.
- **Size transmitted statistics.** For each packet sequence we extract the average, variance, standard deviation and maximum packet size of the total sequence, the incoming sequence and the outgoing sequence.

We evaluate the efficacy of k-fingerprinting when a client is browsing the internet without Tor but with encryption. The attacker will have access to packet size information as well as packet timings from which they can infer information about the web page the client is browsing. Apart from this modification in available features, the attack scenario is similar: An attacker monitors a client browsing online and attempts to infer which web pages they are visiting. The only difference is that browsing with the Transport Layer Security (TLS) protocol, or Secure Sockets Layer (SSL) protocol, versions 2.0 and 3.0, exposes the destination IP address and port. The attack is now trying to infer which web page the client is visiting from the known website[12].

For this attack the attacker monitors 55 web pages, they wish to know if the client has visited one of these pages. The client can browse to any of these web pages or to 7000 other web pages, which the attacker does not care to classify other than as unmonitored pages. We train on 20 out of the 30 instances for each monitored page and vary the number of unmonitored pages we train.

Table 8.4: Attack results for k=2 while varying the number of unmonitored training pages.

Training pages	True Positive rate	False Positive rate
0	0.95 ± 0.01	0.850 ± 0.010
1000	0.92 ± 0.01	0.020 ± 0.001
2000	0.90 ± 0.01	0.010 ± 0.004
3000	0.89 ± 0.02	0.010 ± 0.001
4000	0.87 ± 0.02	0.004 ± 0.001
5000	0.86 ± 0.01	0.004 ± 0.001
6000	0.86 ± 0.01	0.005 ± 0.002

[12]Note that the data sets are composed of traffic instances from some websites without SSL and TLS, as well as websites using the protocols.

Despite more packet sequence information to exploit, the larger cardinality of world size gives rise to more opportunities for incorrect classifications. The attack achieves a True Positive rate of 0.87 ± 0.02 and a False Positive rate of 0.004 ± 0.001. We achieved best results when training on 4000 unmonitored web pages. Table 8.4 report on results for training on different number of unmonitored web pages, with $k = 2$. Figure 8.7 shows our results when modifying the number of fingerprints used (k) and training on 2000 unmonitored pages. We find that altering the number of unmonitored training pages decreases the False Positive rate while only slightly decreasing the True Positive rate. This mirrors our experimental findings from the Wang et al. data set.

Closed-World. In the closed-world scenario in which the client can only browse within the 55 monitored web pages we achieve 0.96 ± 0.02 accuracy. In this setting we do not need the additional fingerprint layer for classification, we can simply use the classification output of the random forest.

Number of monitored training pages in closed-world. Figure 8.8 shows the *out-of-bag* score[13] as we change the number of *monitored* pages we train. We found that training on any more than a third of the data gives roughly the same accuracy.

8.8.2 Attack without packet size features

DS_{Norm} was not collected via Tor and so also contains packet size information. We remove this to allow for comparison with DS_{Tor} and the Wang et al. data set which was collected over Tor. This also gives us a baseline for how much more powerful k-fingerprinting is when we have additional packet size features available.

Table 8.5: Attack results for k=2 while varying the number of unmonitored training pages.

Training pages	True Positive rate	False Positive rate
0	0.90 ± 0.01	0.790 ± 0.020
1000	0.85 ± 0.01	0.019 ± 0.001
2000	0.83 ± 0.01	0.009 ± 0.001
3000	0.83 ± 0.02	0.009 ± 0.001
4000	0.81 ± 0.02	0.006 ± 0.001
5000	0.81 ± 0.01	0.005 ± 0.002
6000	0.80 ± 0.02	0.005 ± 0.001

We achieved a True Positive rate of 0.81 ± 0.01 and False Positive rate of 0.005 ± 0.002 when training on 5000 unmonitored web pages. Table 8.5 shows our results at other sizes of training samples, with $k = 2$. Removing packet size features reduces the True Positive rate by over 0.05 percentile points and increases the False Positive rate by 0.001 percentile points. Clearly packet size features improve our classifier in terms of correct identifications but do not decrease the number of unmonitored test instances that were incorrectly classified as a monitored page.

Closed-World. In the closed-world scenario in which the client can only browse within the 55 monitored web pages k-fingerprinting is 0.91 ± 0.02 accurate. Showing that in the closed-world scenario attack accuracy improves by 5% when we include packet size features.

8.8.3 Attack on larger world size

We run k-fingerprinting with the same number of monitored sites but increase the numbered of unmonitored sites to 17,000. We evaluate when we have both time and size features available.

[13]Defined in section 8.3.2.

Figure 8.9 shows the results of k-fingerprinting while varying the number of fingerprints (k) used for classification, from between 1 and 10, for various experiments trained with different numbers of unmonitored pages. We see that the attack results are comparable to the attack on 7000 unmonitored pages, meaning there is no degradation in attack accuracy when we increase the world size by 10,000 web pages. Training on approximately 30% of the 7000 unmonitored web pages k-fingerprinting gives a True Positive rate of over 0.90 and a False Positive rate of 0.01 for k=1. Training on approximately 30% of the 17,000 unmonitored web pages k-fingerprinting gives a True Positive rate of 0.90 and a False Positive rate of 0.006 for k=1.

The fraction of unmonitored pages that were incorrectly classified as a monitored page decreased as we increased our world size. In other words, out of 12,000 unmonitored pages only 72 were classified as a monitored page, with this figure dropping to 24 if we use k=10 for classification. This provides a strong indication that k-fingerprinting can scale to a real-world attack in which a client is free to browse the entire internet, with no decrease in attack accuracy.

8.9 Fine grained false positives

Closed World

We observe that the classification error is not uniform across all web pages[14]. Some pages are misclassified many times, and confused with many others, while others are never misclassified. An attacker can leverage this information to estimate the misclassification rate of each web page instead of using the global average misclassification rate.

An attacker can use their training set of web pages to estimate the misclassification rate of each web page, by splitting the training set in to a smaller training set and validation set. Since both sets are from the original training set the attacker has access to the true labels. The attacker then computes the misclassification rate of each web page, which they can use as an estimation for the misclassification rate when training on the entire training set and testing on new traffic instances.

Figures 8.10 and 8.11 show the global misclassification rate for a varying number of monitored pages. Monitored pages are first ordered in terms of the misclassification rate they have, ordered from smallest to largest. From figure 8.10, using the Wang et al. data set, we see that if the attacker considers only the top 50% on web pages in terms of per page misclassification rate, the true global misclassification rate and global misclassification rate estimated by the attacker drop by over 70%. Similarly from figure 8.11, using DS_{Norm}, if the attacker considers only the top 50% on web pages in terms of per page misclassification rate, the true global misclassification rate and global misclassification rate estimated by the attacker drop by over 80%. This allows an attacker to train on monitored pages and then cull the pages that have too high an error rate, allowing for more confidence in the classification of the rest of the monitored pages.

The gap between the attacker's estimate and the misclassification rate of the test set is largely due to the size of the data set. Figure 8.10 has a smaller error of estimate than figure 8.11 because the Wang et al. data set has 60 instances per monitored page, in comparison DS_{Norm} has 20 instances per monitored page. In practice, an attacker cannot expect perfect alignment; they are generated from two different sets of data, the training and training + test set. Nevertheless the attacker can expect this difference to decrease with the collection of more training instances.

[14]See additional evidence in Appendix 8.14.2.

Open World on Alexa monitored set of DS_{Tor}

In addition to computing the misclassification rates in a closed-world scenario, an attacker can compute the true positive rate and false positive rates for monitored and unmonitored pages. A naive approach to this problem would be to first find which fingerprints contribute to the many misclassifications and remove them. Our analysis shows that the naive approach of removing "bad" fingerprints that contribute to many misclassifications is floored[15].

We again observed that the classification error is not uniform across all web pages. Similar to the closed-world scenario, an attacker can use their training set of web pages to estimate the true positive and false positives rates of each web page, by splitting the training set in to a smaller training set and validation set. Since both sets are from the original training set the attacker has access to the true labels. The attacker then computes the true positive and false positive rates of each web page, which they can use as an estimation for the rates when training on the entire training set and testing on new traffic instances. More specifically we split, for the monitored training set of 70 instance for each of the Alexa top 55 web pages, into smaller training sets of 40 instances and validation sets of 30 instances. This is used as a misclassification estimator for the full training set of 70 instances against the true test set of 30 instances, that is an estimator of how often each monitored web page will be misclassified. Similarly we split the unmonitored training in half, one set as a smaller training set and the other as a validation set.

Figures 8.12, 8.13, 8.14, 8.15 show the true positive and false positive rate under this scenario for a varying number of unmonitored pages. Monitored pages are first ordered in terms of the misclassification rate they have, ordered from best to worst in terms of their true positive rate. As the size of the unmonitored training set increases so too does the accuracy both the attackers estimate of the false positive rate, and the correct false positive rate. Nevertheless even with a small unmonitored training set of 2000 web pages, which is then split in to a training set of 1000 web pages and a validation set of 1000 web pages, an attacker can accurately estimate the false positive rate of the attack if some of the monitored web pages were removed. For example, using only the best 20 monitored web pages (in terms of true positive rate), an attacker would estimate that using those 20 web pages as a monitored set, the false positive rate would 0.012. Using the entire data set we see that the real false positive rate of these 20 web pages is 0.010; the attacker has nearly precisely estimated the utility of removing a large fraction of the original monitored set. There is a small difference between estimated and the actual false positive rate in all of figures 8.12, 8.13, 8.14 and 8.15. Furthermore there is little benefit in training more unmonitored data if the attacker wants to accurately estimate the false positive rate; figure 8.12 has a similar gap between the estimate false positive rate and real false positive rate when compared to figure 8.15.

From 8.12, 8.13, 8.14, 8.15 it is evident even with small original training set, an attacker can identify web pages that are likely to be misclassified and then accurately calculate the utility of removing these web pages from their monitored set.

8.10 Attack on hardened defenses

For direct comparison we tested our random forest classifier in a closed-world scenario on various defenses against the k-NN attack using the Wang et al. data set [34]. Note that most of these defenses require large bandwidth overheads that may render them unusable for the average client. We test against the following defenses:

- **BuFLO [11].** This defense sends packets at a constant size during fixed time intervals. This potentially extends the length of transmission and requires dummy packets to fill in gaps.

[15]See additional evidence in Appendix 8.14.3.

- **Decoy pages [24].** This defense loads a decoy page when- ever another page is loaded. This provides background noise that degrades the accuracy of an attack.
- **Traffic morphing [36].** Traffic morphing morphs a clients traffic to look like another set of web pages. A client chooses the source web pages that they would like to defend, as well as a set of target web pages that they would like to make the source processes look like.
- **Tamaraw [31].** Tamaraw operates similarly to BuFLO but fixes packet sizes depending on their direction. Outgo- ing traffic is fixed at a higher packet interval, this reduces overhead as outgoing traffic is less frequent.

Table 8.6 shows the performance of k-fingerprinting against k-NN under various website fingerprinting defenses in a closed-world setting on 100 different web pages - meaning an attacker monitors these web pages and a client can only browse to these web pages. Under every defense k-fingerprinting is comparable or achieves better results than the k-NN attack. Note that k-fingerprinting does equally well when traffic morphing is applied compared to no defense. As Lu et al. [19] note, traffic morphing is only effective when the attacker restricts attention to the same features targeted by the morphing process. Our results confirm that attacks can succeed even when traffic morphing is employed.

Table 8.6: Attack comparison under various website fingerprinting defenses.

Defenses	**This work**	***k*-NN [34]**	**Overhead (%)**
No defense	0.91 ± 0.01	0.91 ± 0.03	0
Morphing [36]	0.90 ± 0.03	0.82 ± 0.06	50 ± 10
Tamaraw [31]	0.10 ± 0.01	0.09 ± 0.02	96 ± 9
Decoy pages [24]	0.37 ± 0.01	0.30 ± 0.06	130 ± 20
BuFLO [11]	0.21 ± 0.02	0.10 ± 0.03	190 ± 20

8.11 Attack Summary

Past and current works on website fingerprinting either use the artificial closed-world model or an open-world model that is limited in size. The current largest studies using an open-world scenario by Wang et al. [34], and Panchenko et al. [24], both consider 5000 unmonitored sites. Our study considers 55 monitored web pages and unmonitored world sizes of 7,000, 17,000 and 100,000 web pages. By reducing the number of monitored web pages and number of examples we train upon, and increasing the number of unmonitored web pages we greatly increase the chance of False Positives – since we have more unmonitored sites that could be classified as a monitored site. This reflects realistic conditions where an attacker would like to monitor a small number of web pages out of a large universe of web pages they do not care about.

Best attack results on data sets were achieved when we train on approximately two thirds of the unmonitored web pages. Despite this results from DS_{Tor} show that an attacker can achieve a very small false positive rate while only training on 2% of the unmonitored data. Training on 2% of 100,000 unmonitored web pages greatly reduces the attack set up costs while only marginally reducing the accuracy, providing a realistic scenario under which an attack could be launched. Figure 8.8 illustrates that compared to training on a small number of monitored instances increasing the size of the monitored training set only incrementally increases accuracy. Results on all data sets also suggest that altering k, the number of fingerprints used for classification, has a greater influence on accuracy than the number of training samples. By varying the number of k training instances considered when classifying a test instance, an attacker may trade the True Positive rate for the False Positive rate.

Figure 8.1 illustrates that the attack achieves approximately the same accuracy using the best 30 features, as when using more of them. Using packet size features in addition to timing features increases the True Positive rate by 5% but does not dramatically decrease the False Positive rate. Similarly from figure 8.4 we see that k-fingerprinting has nearly the same True Positive and False Positive rates using fingerprints of length 20 as it does for fingerprints of length 200.

In terms of type of web page, k-fingerprinting achieves the same accuracy regardless of the target monitored set. The monitored set in the Wang et al. data set consists of some websites not found in Alexa 10,000 list [1], and the $DS_{Tor/Norm}$ monitored sets were taken from the top 100 Alexa websites. Although we do see a reduction in the false positive rate when the target monitored set are Tor hidden services due to the distinguishibility between the hidden services and the unmonitored web pages.

We also highlight the non-uniformity of classification performance: when a monitored web page is misclassified, it is usually misclassified on multiple tests. We show that an attacker can use their training set to estimate the error rate of k-fingerprinting per web page, and select targets with low misclassification rates.

k-fingerprinting is more accurate and uses fewer features than state-of-the-art attacks. Furthermore k-fingerprinting is faster than current state-of-the-art website fingerprinting attacks. On the Wang et al. data set training time for 6,000 monitored and 2,500 unmonitored training pages is 30.738 CPU seconds on an 1.4 GHz Intel Core i5z. The k-NN attack [34] has training time per round of 0.064 CPU seconds for 2500 unmonitored training pages. For 6,000 rounds training time is 384.0 CPU seconds on an AMD Opteron 2.2 GHz cores. This can be compared to around 500 CPU hours using the attack described by Cai et al. [7]. Testing time per instance for k-fingerprinting is around 0.1 CPU seconds, compared to 0.1 CPU seconds to classify one instance for k-NN and 450 CPU seconds for the attack described by Cai et al. [7].

8.12 Discussion of Practicalities

Website fingerprinting research has been criticised for not being applicable to real-world scenarios [15], [25]. We have shown that a website fingerprinting attack can scale to the number of traffic instance an attacker may sample over long period of time with hardly any false positives. We have also shown how a realistic attack may wish to throw away some training information which could confuse the classifier. However, here we present limitations of our and other website fingerprint attacks:

Multitab browsing. Website fingerprinting attacks have so far only considered a client that browses the internet using a single tab. The ability to separate traffic into relevant packet streams when a client browses online has so far not been researched – and our work shines no light on this topic. As Juarez et al. note that real-world browsing session tend to be performed with multiple tabs [21], [29].

Short-lived websites. Website content rapidly changes which will negatively affect the accuracy of a website fingerprinting attack [15]. As the content of a website changes so will the generated packet sequences, if an attacker cannot train on this new data then an attack will suffer. However we note that an attack will suffer from the ephemeral nature of websites at different rates depending on the type of website being monitored. For example, an attack monitoring a news or social media site can expect a faster degradation in performance compared to an attack monitoring a landing page of a top 10 Alexa site [1]. Also note Tor does not cache by default, so if in the realistic scenario where an attacker wanted to monitor *www.facebook.com* a client would be forced to navigate to the facebook landing page, which hosts content that is long lived.

Network conditions and noise. In reality an attacker will not be able to perfectly replicate the network conditions of a client's browsing session. This means the training set the attacker collected before the attack will not be a perfect representation of the traffic they wish to monitor.

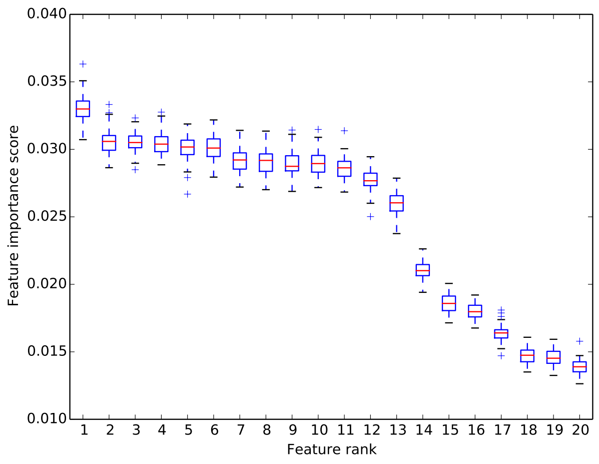
It is also highly unlikely a client will browse the internet with no other background traffic present. Both of these things will limit the practicality of a real-world website fingerprinting attack.
Feature importance. One limitation of our feature importance analysis is that our implementation of random forests uses axis-aligned splits and so cannot capture the non-linear relationships that features have with one another. Packet features may have dependency relationships between one another that cannot be captured by the attack.

8.13 Conclusion

Website fingerprinting attacks are a serious threat to a client's online privacy. Clients of both Tor and standard web browsers are at risk from website fingerprinting attacks regardless of whether they browse to hidden services or standard websites. k-fingerprinting improves on state-of-the-art attacks in terms of both speed and accuracy. We have shown that current website fingerprinting defenses either do not defend against k-fingerprinting or incur such a high bandwidth cost that it renders the defense unfeasible. Using random forests to extract robust fingerprints of web pages we can perform an attack that increases True Positives and decreases False Positives when compared to state-of-the-art website fingerprinting attacks. Additionally we showed that misclassification rates of web pages is highly non-uniform; patterns of misclassification can be exploited to perform a more accurate attack.

We also conducted feature analysis of features used in the attack, these features are often used in other website fingerprinting works. We found that simple features such as counting the number of incoming and outgoing packets were more important than complex features such as packet inter-arrival times or packet ordering features.

Our world size is the biggest used in any website fingerprinting study so far. k-fingerprinting achieves good results even when an attacker trains on a small fraction of the total data. Untrustworthy data within that small fraction can then be filtered and removed before the attack is launched to later yield better results, showing that long term website fingerprinting attacks on a targeted client is a realistic possibility.
Reproducibility. All code is available through code repositories under a liberal open source license and and data will be deposited in open data repositories.



№	Feature Description
1.	Number of incoming packets.
2.	Number of outgoing packets as a fraction of the total number of packets.
3.	Number of incoming packets as a fraction of the total number of packets.
4.	Standard deviation of the outgoing packet ordering list.
5.	Number of outgoing packets.
6.	Sum of all items in the alternative concentration feature list.
7.	Average of the outgoing packet ordering list.
8.	Sum of incoming, outgoing and total number of packets.
9.	Sum of alternative number packets per second.
10.	Total number of packets.
11.	Average of concentration of outgoing packets in chunks of 20 packets feature list.
12.	Standard deviation of the incoming packet ordering list.
13.	Average of the incoming packet ordering list.
14.	Alternative packet concentration feature list - 1st item.
15.	Alternative packet concentration feature list - 2nd item.
16.	Standard deviation of concentration of outgoing packets in chunks of 20 packets feature list.
17.	Packet concentration feature list - 2nd item.
18.	Packet concentration feature list - 3rd item.
19.	The total number of incoming packets stats in first 30 packets.
20.	The total number of outgoing packets stats in first 30 packets.

Figure 8.2: The 20 most important features.

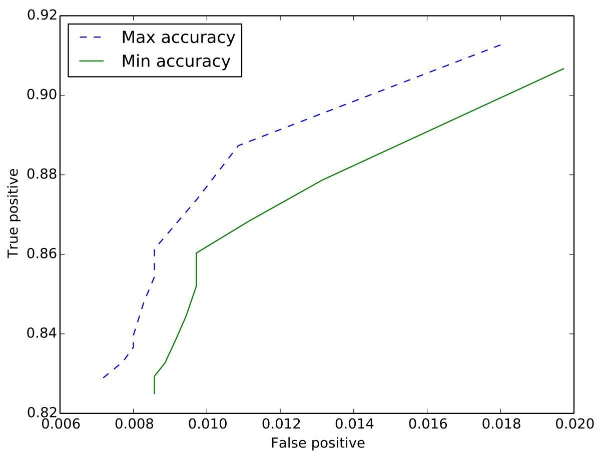


Figure 8.3: Attack results for 1500 unmonitored training pages while varying the number of fingerprints used for comparison, k, over 10 experiments.

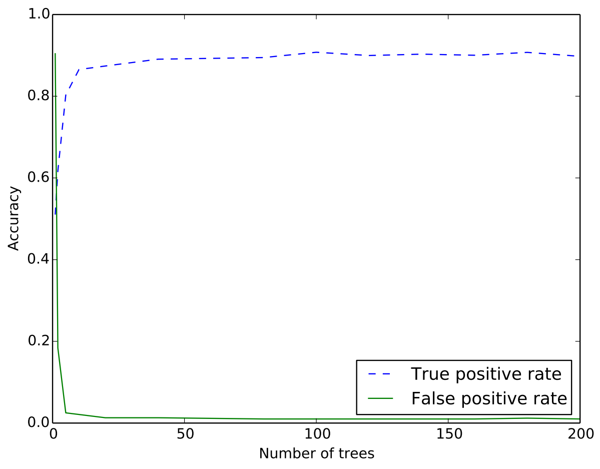


Figure 8.4: Accuracy of k-fingerprinting as we vary the number of trees in the forest. We train on 4000 unmonitored training pages and set k=1.

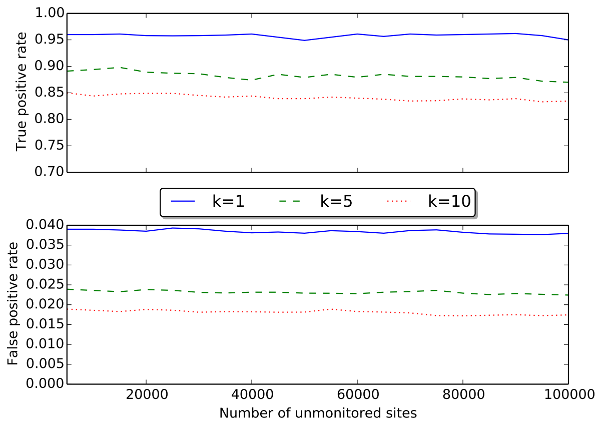


Figure 8.5: Attack accuracy on DS_{Tor} with Alexa monitored set.

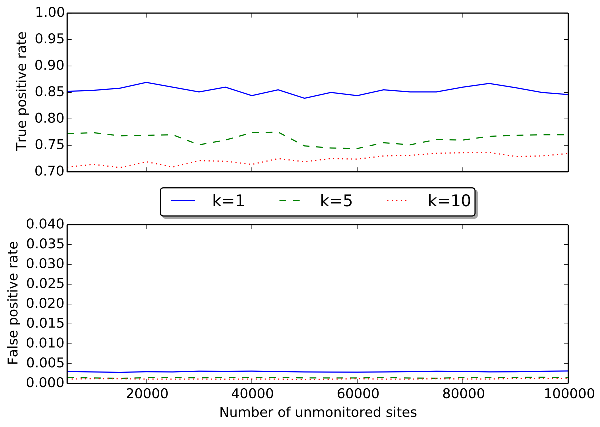


Figure 8.6: Attack accuracy on DS_{Tor} with Tor hidden services monitored set.

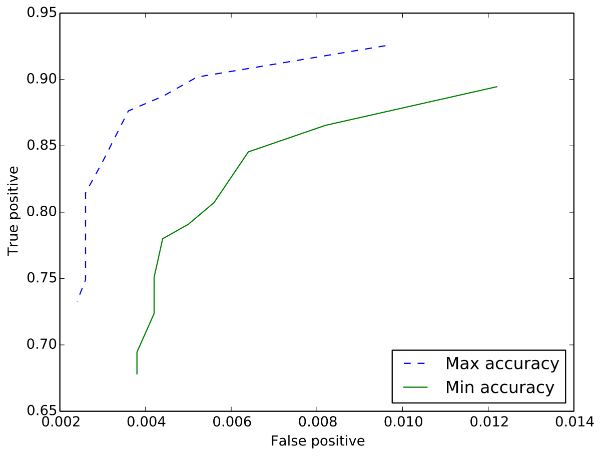


Figure 8.7: Attack results for 2000 unmonitored training pages while varying the number of fingerprints used for comparison, k, over 10 experiments.

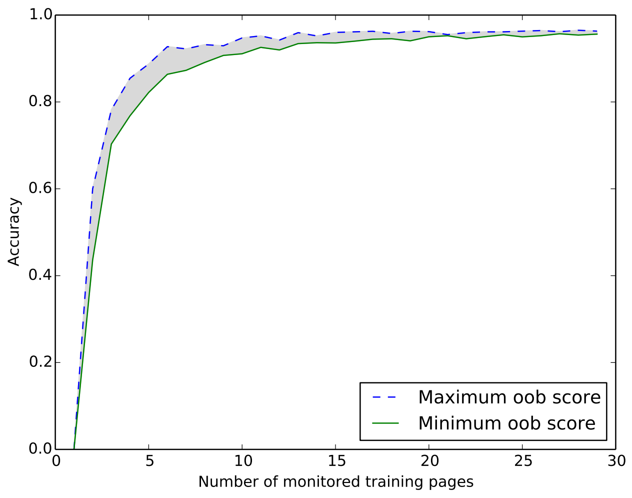


Figure 8.8: Attack out-of-bag score while varying the number of monitored training pages.

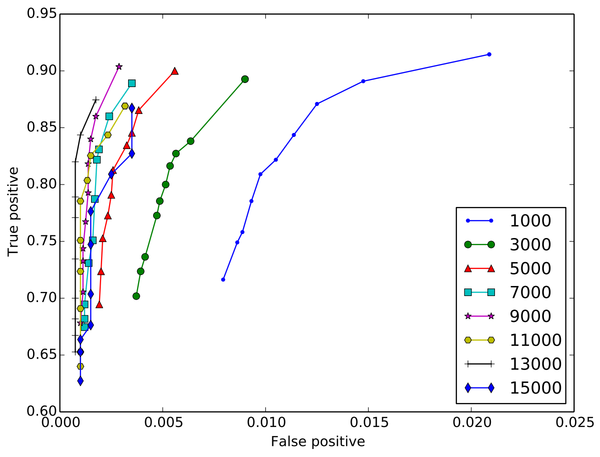


Figure 8.9: Attack accuracy for 17,000 unmonitored web pages. Each line represents a different number of unmonitored web pages that were trained, while varying k, the number of fingerprints used for classification.

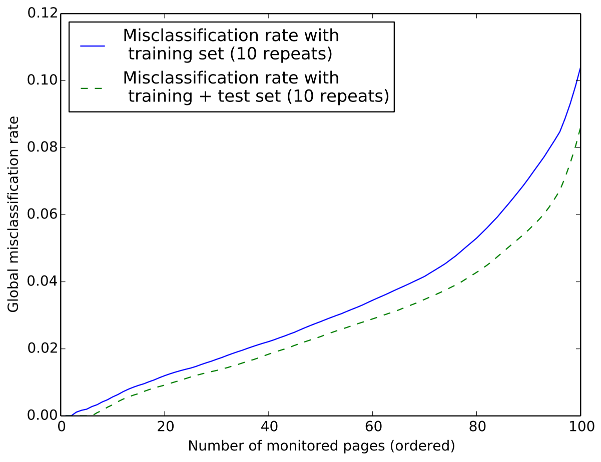


Figure 8.10: The global misclassification rate when considering different numbers of monitored pages from the Wang et al. data set. The monitored pages are ordered in terms of smallest misclassification rate to largest.

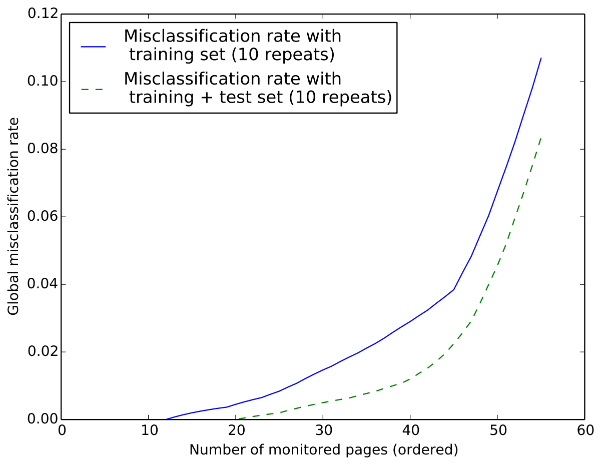


Figure 8.11: The global misclassification rate when considering different numbers of monitored pages from DS_{Norm}. The monitored pages are ordered in terms of smallest misclassification rate to largest.

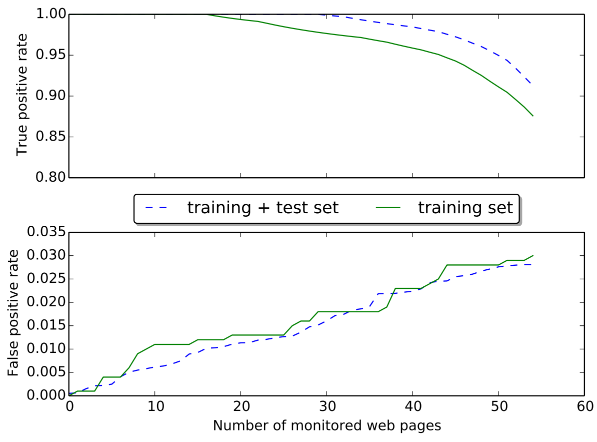


Figure 8.12: Rates for training on 1000 unmonitored pages, testing on 1000, and comparison when training on the full 2000 unmonitored pages and testing on the remaining 98000 unmonitored pages in DS_{Tor}, k=3.

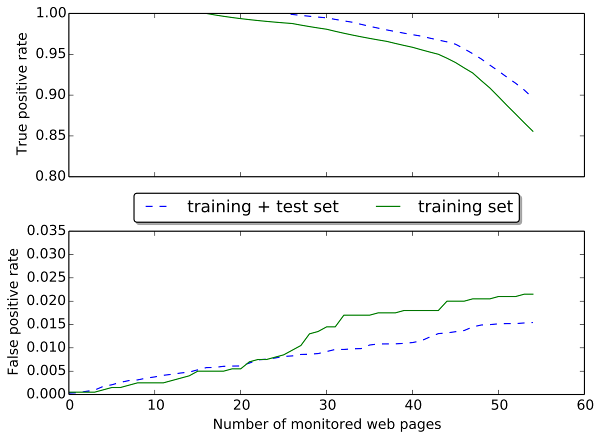


Figure 8.13: Rates for training on 2000 unmonitored pages, testing on 2000, and comparison when training on the 4000 unmonitored pages and testing on the remaining 96000 unmonitored pages in DS_{Tor}, k=3.

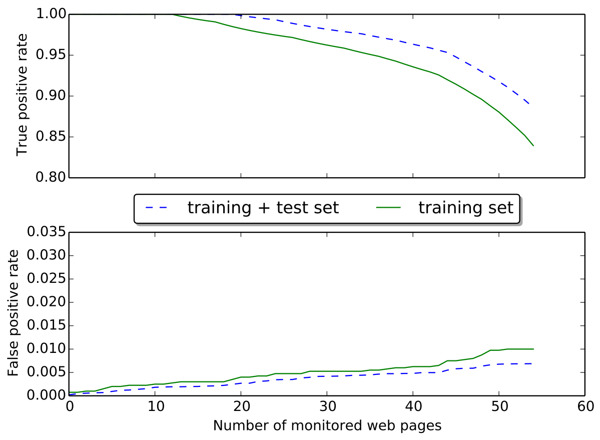


Figure 8.14: Rates for training on 4000 unmonitored pages, testing on 4000, and comparison when training on the full 8000 unmonitored pages and testing on the remaining 92000 unmonitored pages in DS_{Tor}, k=3.

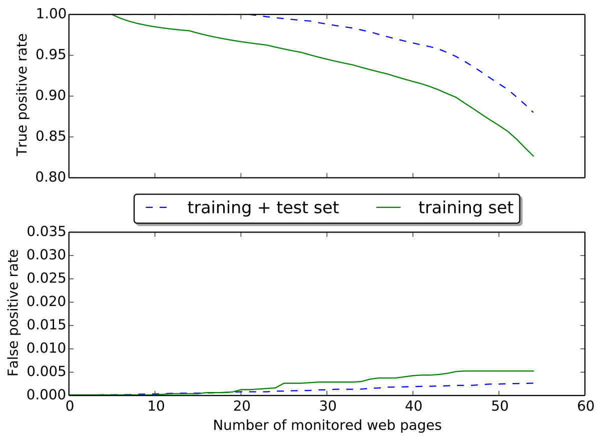


Figure 8.15: Rates for training on 8000 unmonitored pages, testing on 8000, and comparison when training on the full 16000 unmonitored pages and testing on the remaining 84000 unmonitored pages in DS_{Tor}, k=3.

Bibliography

[1] Alexa The Web Information Company, [Accessed August 2015].

[2] Leo Breiman. Random Forests, [Accessed July 2015].

[3] The Nielsen Company, [Accessed July 2015].

[4] George Dean Bissias, Marc Liberatore, David Jensen, and Brian Neil Levine. "Privacy Vulnerabilities in Encrypted HTTP Streams". In *Proceedings of the 5th International Conference on Privacy Enhancing Technologies*, pages 1–11, 2006.

[5] Leo Breiman. "Random Forests". *Mach. Learn.*, 45(1):5–32, 2001.

[6] Xiang Cai, Rishab Nithyanand, and Rob Johnson. "CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense". In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 121–130, 2014.

[7] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. "Touching from a distance: website fingerprinting attacks and defenses". In *ACM Conference on Computer and Communications Security*, pages 605–616, 2012.

[8] Shuo Chen, Rui Wang, XiaoFeng Wang, and Kehuan Zhang. "Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow". In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, pages 191–206, 2010.

[9] Heyning Cheng, , Heyning Cheng, and Ron Avnur. "Traffic Analysis of SSL Encrypted Web Browsing", 1998.

[10] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. "Tor: The Second-Generation Onion Router". In *Proceedings of the 13th USENIX Security Symposium*, pages 303–320, 2004.

[11] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. "Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail". In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, pages 332–346, 2012.

[12] Jerome H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine". *Annals of Statistics*, 29:1189–1232, 2000.

[13] Pall Oskar Gislason, Jon Atli Benediktsson, and Johannes R. Sveinsson. "Random Forests for Land Cover Classification". *Pattern Recogn. Lett.*, 27(4):294–300, March 2006.

[14] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. "Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naive-bayes Classifier". In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, pages 31–42, 2009.

[15] Marc Juárez, Sadia Afroz, Gunes Acar, Claudia Díaz, and Rachel Greenstadt. "A Critical Evaluation of Website Fingerprinting Attacks". In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 263–274, 2014.

[16] A. Liaw and M. Wiener. "Classification and Regression by randomForest". *R News: The Newsletter of the R Project*, 2(3):18–22, 2002.

[17] Marc Liberatore and Brian Neil Levine. "Inferring the source of encrypted HTTP connections". In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 255–263, 2006.

[18] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation-Based Anomaly Detection". *ACM Trans. Knowl. Discov. Data*, 6(1):3:1–3:39, March 2012.

[19] Liming Lu, Ee-Chien Chang, and Mun Choon Chan. "Website Fingerprinting and Identification Using Ordered Feature Sequences". In *Proceedings of the 15th European Conference on Research in Computer Security*, pages 199–214, 2010.

[20] Xiapu Luo, Peng Zhou, Edmond W. W. Chan, Wenke Lee, Rocky K. C. Chang, and Roberto Perdisci. "HTTPOS: Sealing information leaks with browser-side obfuscation of encrypted flows". In *In Proc. Network and Distributed Systems Symposium (NDSS)*, 2011.

[21] Mozilla Labs. Test Pilot: Tab Open/Close Study: Results. https://testpilot.mozillalabs.com/testcases/tab-open-close/results.html. Accessed July 2015.

[22] Rishab Nithyanand, Xiang Cai, and Rob Johnson. "Glove: A Bespoke Website Fingerprinting Defense". In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 131–134, 2014.

[23] A. Stolerman M. V. Ryan P. Brennan P. Juola, J. I. Noecker Jr and R. Greenstadt. "A Dataset for Active Linguistic Authentication". In *IFIP WG 11.9 International Conference on Digital Forensics*, 2013.

[24] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. "Website fingerprinting in onion routing based anonymization networks". In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society, WPES*, pages 103–114, 2011.

[25] Mike Perry. "A Critique of Website Traffic Fingerprinting Attacks". *https://blog.torproject.org/blog/critique-website-traffic-fingerprinting-attacks*, Accessed June 2015.

[26] Mike Perry. "Experimental defense website traffic fingerprinting". *https://blog.torproject.org/blog/experimental-defense-website-traffic-fingerprinting*, Accessed June 2015.

[27] Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu. "statistical identification of encrypted web browsing traffic". In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 19–, 2002.

[28] Vladimir Svetnik, Andy Liaw, Christopher Tong, J. Christopher Culberson, Robert P. Sheridan, and Bradley P. Feuston. "Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling". *Journal of Chemical Information and Computer Sciences*, 43(6):1947–1958, 2003.

[29] C. von der Weth and M. Hauswirth. DOBBS: Towards a Comprehensive Dataset to Study the Browsing Behavior of Online Users. CoRR, abs/1307.1542, 2015.

[30] David Wagner and Bruce Schneier. "Analysis of the SSL 3.0 Protocol". In *Proceedings of the 2nd Conference on Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2*, pages 4–4, 1996.

[31] T. Wang and I. Goldberg. "Comparing website fingerprinting attacks and defenses". Technical Report, 2013.

[32] T. Wang and I. Goldberg. "On Realistically Attacking Tor with Website Fingerprinting". Technical Report, 2015.

[33] T. Wang and I. Goldberg. "Walkie-Talkie: An Effective and Efficient Defense against Website Fingerprinting". Technical Report, 2015.

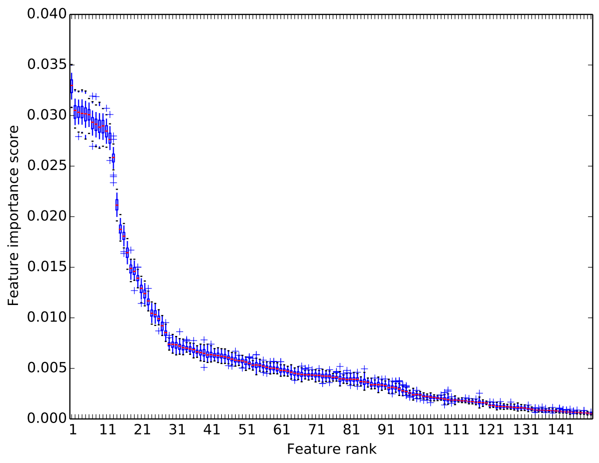
[34] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. "Effective Attacks and Provable Defenses for Website Fingerprinting". In *Proceedings of the 23rd USENIX Security Symposium*, pages 143–157, 2014.

[35] Tao Wang and Ian Goldberg. "Improved Website Fingerprinting on Tor". In *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, pages 201–212, 2013.

[36] Charles V. Wright, Scott E. Coull, and Fabian Monrose. "Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis". In *In Proceedings of the 16th Network and Distributed Security Symposium*, pages 237–250, 2009.

8.14 Appendix

8.14.1 Total feature importance.



№	Feature Description
131.	Packet concentration feature list - 34th item.
132.	Packet concentration feature list - 39th item.
133.	Alternative packet concentration feature list - 20th item.
134.	Packet concentration feature list - 40th item.
135.	Packet concentration feature list - 24th item.
136.	Packet concentration feature list - 23th item.
137.	Packet concentration feature list - 48th item.
138.	Packet concentration feature list - 46th item.
139.	Packet concentration feature list - 45th item.
140.	Packet concentration feature list - 22th item.
141.	Packet concentration feature list - 55th item.
142.	Packet concentration feature list - 42th item.
143.	Packet concentration feature list - 47th item.
144.	Packet concentration feature list - 51th item.
145.	Packet concentration feature list - 36th item.
146.	Packet concentration feature list - 44th item.
147.	Packet concentration feature list - 41th item.
148.	Packet concentration feature list - 54th item.
149.	Packet concentration feature list - 52th item.
150.	Packet concentration feature list - 53th item.

Figure 8.16: The figure shows the feature importance score for all 150 features in order. The table gives the description for the 20 least important features.

8.14.2 Confusion matrix for closed-world simulated attack on Tor.

Figure 8.17 shows the confusion matrix in our closed-world scenario, that is, it shows the 49 misclassifications (out of 550). We see that some persistent misclassification patterns of web pages appear, for example web page 54 is classified correctly four times but is misclassified as web page 0 six times. The misclassification rate in figure 8.17 is 0.09 but this is the average error rate across all web pages.

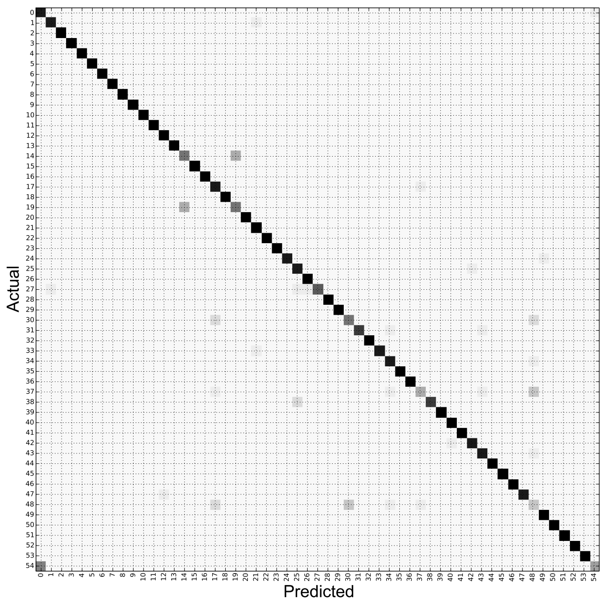


Figure 8.17: Confusion matrix for closed-world attack on Tor using DS_{Norm}. F1 score = 0.913, Accuracy: 0.915, 550 items.

8.14.3 Good vs. bad fingerprints

Figure 8.18 shows the 50 fingerprints that cause the most misclassifications, and also shows for those same fingerprints the number of correct classifications they make. As we can see nearly all "bad" fingerprints actually contribute to many correct classifications.

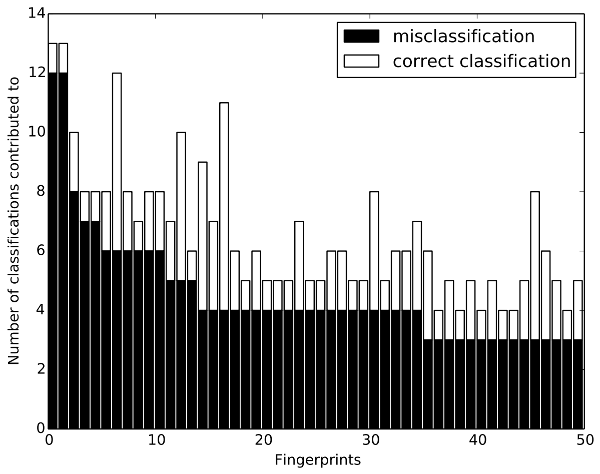


Figure 8.18: The fingerprints that lead to the most misclassifications and the "good" classifications they contribute to. Training on 2000 unmonitored pages and testing on 10000 unmonitored pages with k=3.