Panos Louridas (GRNET)
George Tsoukalas (GRNET)
Dimitris Mitropoulos (GRNET)

Minimum Viable Product

Deliverable D5.2

February 28, 2017
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	2017-17-01	DM, GT, PL (GRNET)	Proposed table of contents
0.2	2017-27-01	DM, GT, (GR-NET)	Added “Integrating Zeus mix-net in PANORAMIX”
0.3	2017-07-02	DM (GRNET)	Added “E-voting in the PANORAMIX Context”
0.4	2017-07-23	DM (GRNET)	Updated “E-voting in the PANORAMIX Context”
0.5	2017-07-26	TZ (UEDIN)	Review
0.6	2017-07-26	PH (UoA)	Review
0.8	2017-07-28	AK (UEDIN)	Review
1.0	2016-02-28	MW,TZ (UEDIN)	Revised final version and submission to the EC

Executive Summary

This deliverable describes the first steps towards a new version of the Zeus e-voting system, which is being developed in the context of WP5.

In Section 2, the integration of Zeus' re-encryption mix-net into PANORAMIX is described. Then, in Section 3, we present how Zeus was ported to the PANORAMIX Minimum Viable Product (MVP) and provide details on how to set up a mix-net and create peers in a simple e-voting use case.

Furthermore, we have a) integrated a well known decryption mix-net into PANORAMIX and b) implemented a new, fully functional re-encryption mix-net (based on the work done in WP3) to enhance Zeus' efficiency and effectiveness. We discuss these advances in Section 4.

Contents

1. Introduction

Zeus [7] is an e-voting application developed by GRNET. It has been in production since late 2012. So far it has been used in approximately 500 elections for state and private organizations, including legally binding elections for Greek academic institutions. Zeus is the basis of the first use-case of the PANORAMIX project and it aims to support the deployment of secure election systems, which are essential processes for citizen participation in governance and in public administration.

Zeus mixes votes with a Sako-Kilian re-encryption mix-net [3], based on the ElGamal cryptosystem [6]. The mix-net shuffles the set of votes in a provably correct but practically untraceable way, thus anonymizing it. The output is the same set of unencrypted ballots as they were before submission, only in a random order that no strict subset of parties can predict. Once this is performed, any method of tallying can be applied to compute the final results.

A limiting factor in the wider adoption of Zeus and e-voting in general, is the mix-net implementation that is used by the voting platform. As mentioned above, the Zeus e-voting system currently uses a standard mix-net implementation that is simple, but not very efficient; only a few thousand voters can be handled within an acceptable amount of time (about an hour), while the needs of the system are in the order of tens or hundreds of thousands.

In WP3, we have already investigated and developed novel, much faster mix-net implementations. Through our work in WP5 we expect to be able to raise e-voting performance, in order to accommodate elections involving hundreds of thousands of voters with no extensive hardware needs. WP3 already provides the necessary cryptographic background for this task, by designing secure yet efficient mix-net protocols.

In this deliverable we describe: (1) how we integrated Zeus' mix-net into PANORAMIX (Section 2), (2) how Zeus was ported into the PANORAMIX Minimum Viable Product (MVP) (Section 3), and (3) how the development of the MVP and the implementation of the mix-net designed in WP3 facilitate the efficiency of Zeus (Section 4).

2. Integrating Zeus' Mix-net in PANORAMIX

As we mentioned in the previous chapter, Zeus uses a Sako-Kilian re-encryption mix-net. Specifically, it uses by default an ElGamal cryptosystem over the prime-order quadratic residue subgroup of a 2048-bit safe prime. Integers that are non-residues and therefore not within the ElGamal group are encoded into the group by mapping them to their opposite residue.

We have successfully integrated this mix-net into PANORAMIX. To do so, we have followed specific steps. The first basic step was to treat votes as messages. Then, we had to incorporate proofs into the PANORAMIX Application Programming Interface (API). We present the above steps in the following subsections.

2.1 Votes as Messages

PANORAMIX enables election trustees to configure and run a Zeus election process in a transparent way. In the context of PANORAMIX, each Zeus operation (e.g. mixing or decrypting votes) is assigned to a dedicated endpoint, controlled by one or more of the trustees. The ballot box, which acts as the external interface of the Zeus mix-net, is itself implemented as an endpoint.

Votes are viewed as messages that are submitted to the inbox of the ballot box endpoint. This endpoint should be created and jointly controlled by the trustees. Trustees must all reach a consensus (using a negotiation) in order to either accept or reject votes or publish the election results.

In order to interact with endpoints, both voters and trustees must first be registered to the controller using their distinct cryptographic keys. The trustees jointly create a so-called election peer, which owns the ballot box endpoint and acts as the recipient of the votes / messages. The votes can only be deciphered if all trustees agree to do so.

Before the election begins, trustees must agree on a processing schedule. Based on this schedule and after the ballot box closes, votes are forwarded to a trustee's endpoint in order to be mixed. The mixed votes are sent consecutively to the other trustees for further mixing. Off the last mixing endpoint, each trustee fetches the mixed votes to their own decryption endpoint in order to compute their partial decryption. Finally, a combining endpoint, controlled by all trustees, collects the factors and produces the decrypted election results. After all trustees agree on the results, they can be published at the outbox of the ballot box. Note that (as we described in D5.1), each trustee provides their public key for each poll. All keys are combined into a single election public key that is used to encrypt votes. At decryption time, each trustee must provide partial decryptions which are all combined to yield the final plaintext ballots. In addition, Zeus automatically creates a key and operates as a trustee for every election. As per the Helios workflow, this allows the service operator to join the appointed committee of trustees in providing anonymity guarantees.

2.2 Incorporating Proofs into the PANORAMIX API

As we discussed in D5.1, voting requires strong verification guarantees. Hence, we have added proofs to the PANORAMIX API. Specifically, the process described in the previous subsection is fully auditable, because each endpoint records the metadata of the processing it performed. In particular, a mixing endpoint records the cryptographic proof produced by the re-encryption mixing applied. An auditor can retrieve the proof and check it against the endpoint's inbox votes and outbox votes.

When a consensus of all trustees is needed for an endpoint action to be applied, the consensus is recorded along with the action. It can later be checked that all trustees have cryptographically signed the action with the exact same attributes; for instance, publishing the results requires a signed text that contains the hashes of all decrypted votes.

3. E-voting in the PANORAMIX Context

As we have mentioned earlier, Zeus had the Sako-Kilian (SK) shuffle hardcoded. Now, by using PANORAMIX, it is much more easier to switch to a more efficient shuffle. Note also that the MVP-based Zeus has already matched the functionality of old Zeus, given that we have already integrated Zeus' SK shuffle to PANORAMIX.

In this chapter we discuss how Zeus can work in the context of PANORAMIX. We provide an architecture, the negotiation process between trustees and how different mix-nets can be used by Zeus.

3.1 Zeus deployment in PANORAMIX

In this chapter we provide an overview of the Zeus architecture, based on the MVP described in D4.1. Figure 3.1 illustrates the architecture of the system.

There are three main entities involved in the architecture, namely: the *voting controller*, the *contributor computer*, and the *voter computer*. The voter controller provides several *endpoints* to the other two entities. Specifically, there is an endpoint utilized by a voter computer to send his or her ballot and an endpoint that provides specific information regarding the kind of the mix-net and the various parameters that have to be used to either encrypt or decrypt ballots. The endpoints used by the contributor computers provide all the information regarding the protocol that is being used (re-encryption mix-net specifics).

Every contributor computer contains a *wizard* component. This component can be used by the administrator to set up the mix server which in turn, will act as a mix-net peer (see D4.1 for more). The mix server contains two basic components, the *crypto module* and the PANORAMIX *client*. The latter initializes the former after interacting with the controller through the corresponding endpoints. Each user computer contains a local agent with the same components.

3.2 Mixing

In this section we provide information related to the Zeus mixing process.

3.2.1 Setting up a mix-net

Building a mix-net requires fixing a number of settings across all mix-net contributors. Apart from the cryptographic parameters, the peers must agree on the mix-net workflow.

In the case of an election mix-net, the trustees must specify the exact sequence on which the mixing and decrypting endpoints will operate.

PANORAMIX enables the contributors to agree on these settings in a cryptographically secure way, before launching the mix-net. Through a negotiation they can formally agree on the creation of the involved endpoints with the described behavior:

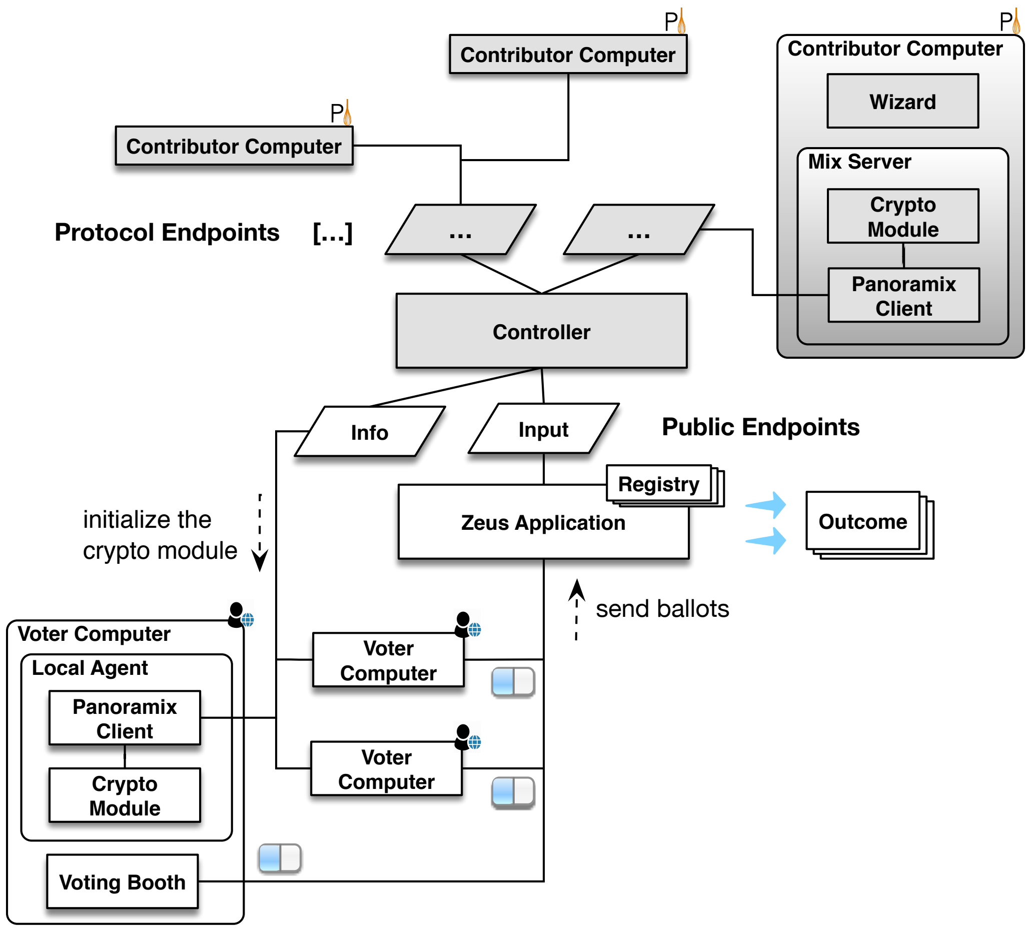

Figure 3.1: The Architecture of the Zeus e-voting System based on the PANORAMIX MVP.

- The "ballot box" endpoint is owned jointly by all trustees; it delegates processing to other mix-net endpoints and expects to pull the results from the "combine" endpoint (see below).
- The first trustee's mixing endpoint retrieves the votes cast at the ballot box, mixes them and uploads the mixed votes to its outbox.
- The mixing endpoint of each other trustee, consecutively, fetches the mixed votes from the outbox of the previous trustee's endpoint for further mixing.
- The decryption endpoint of each trustee, in parallel, retrieves the mixed votes from the last mixing endpoint, in order to compute their decryption factors.
- The "combine" endpoint, controlled by all trustees, fetches the factors from the outbox of all decryption endpoints and computes the election results.

To support such a setup, PANORAMIX allows an endpoint description to specify the endpoints it expects to retrieve its input from. This facilitates a static workflow which can be easily streamlined, with each endpoint polling other endpoints for their output. It also allows a clean separation between the mix-net's public and private interfaces: the ballot box's inbox and outbox constitutes the public interface; while actual processing is delegated to the other endpoints, which can be private.

3.2.2 Peer creation

First we need to create a joint peer. This fixes the cryptographic settings across the mix-net, but does not involve any application settings. A simple negotiation is enough.

Endpoint creation. In the case of the Zeus mix-net we start off a negotiation based on the following JSON (JavaScript Object Notation) canonical form:

```
[
  {
      "peer_id": joint_peer,
      "endpoint_id": "ballotbox_election",
      "endpoint_type": "ZEUS_BALLOT_BOX",
      "links": [{"from_endpoint_id": "combine",
                 "from_box": "OUTBOX",
                 "to_box": "PROCESSBOX"}]
  },
  {
      "peer_id": peer1,
      "endpoint_id": "mix_peer1",
      "endpoint_type": "ZEUS_SK_MIX",
      "links": [{"from_endpoint_id": "ballotbox_election",
                 "from_box": "ACCEPTED",
                 "to_box": "INBOX"}]
  },
  {
      "peer_id": peer2,
      "endpoint_id": "mix_peer2",
      "endpoint_type": "ZEUS_SK_MIX",
      "links": [{"from_endpoint_id": "mix_peer1",
                 "from_box": "OUTBOX",
                 "to_box": "INBOX"}]
  },
  {
      "peer_id": peer1,
      "endpoint_id": "decr_peer1",
      "endpoint_type": "ZEUS_SK_PARTIAL_DECRYPT",
      "links": [{"from_endpoint_id": "mix_peer2",
                 "from_box": "OUTBOX",
                 "to_box": "INBOX"}]
  },
  {
      "peer_id": peer2,
      "endpoint_id": "decr_peer2",
      "endpoint_type": "ZEUS_SK_PARTIAL_DECRYPT",
      "links": [{"from_endpoint_id": "mix_peer2",
                 "from_box": "OUTBOX",
                 "to_box": "INBOX"}]
  },
  {
      "peer_id": joint_peer,
      "endpoint_id": "combine",
      "endpoint_type": "ZEUS_SK_COMBINE",
```

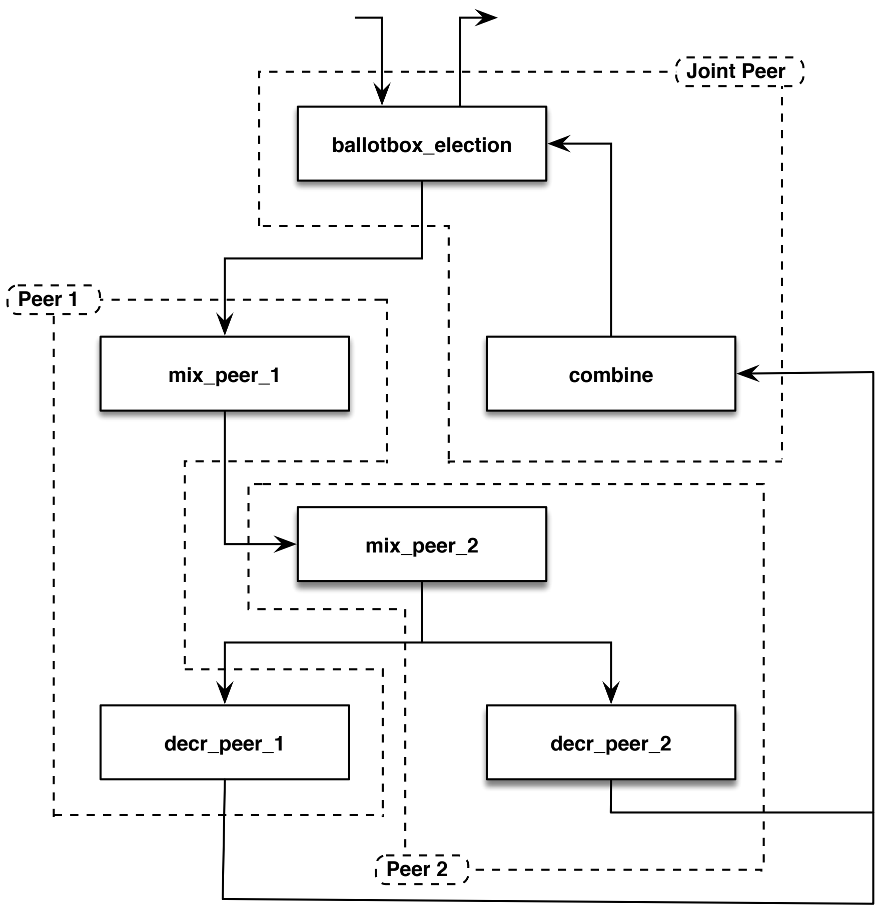

Figure 3.2: A Graph containing different peers, endpoints and their relation. A **mix_peer_*** endpoint is related to mixing and a **decr_peer_*** endpoint is related to decryption.

```
        "links": [{"from_endpoint_id": "decr_peer1",
                    "from_box": "OUTBOX",
                    "to_box": "INBOX"},
                   {"from_endpoint_id": "decr_peer2",
                    "from_box": "OUTBOX",
                    "to_box": "INBOX"}]
    }
]
```

This describes a graph of endpoints: each list element is a prescription to create an endpoint. The attribute “links” describes where each endpoint takes its input from (be it the input of the INBOX or the PROCESSBOX). We observe that there are different types of endpoints e.g. an endpoint used only for mixing, another for partial decryption etc. Figure 3.2 illustrates the graph that corresponds to the above negotiation.

The mix-net contributors (peers) inspect the definition, negotiate in order to change e.g. the endpoint_ids (or any other attributes not shown in the above definition), and finally create the endpoints according to the definition.

Working with the endpoints. We provide some basic actions related to the endpoints of the e-voting protocol.

1. End-users post messages to INBOX of the main endpoint.
2. Owners close the endpoint. Messages then will move to ACCEPTED.

3. Transmission: The endpoint that expects to get its input from the main endpoint, polls its status until it becomes CLOSED and then retrieves the accepted messages.
4. The running endpoint is closed, messages are processed and uploaded, and the endpoint moves to state PROCESSED.
5. Similarly, steps 3 and 4 are executed by remaining peers according to their specified links.
6. The main endpoint waits for the last endpoint to finish processing (as specified in links), then pushes the processed messages to its own processbox. After acknowledging the processed messages though a negotiation, the output of the mix-net is available at the main endpoint's outbox.

4. Deploying Different mix-nets

PANORAMIX enables interested third parties to start integrating the functionality it provides through an easy to use and fully documented API (see D4.1). By using the API different mix-nets can be integrated to the platform in an easy manner. Based on the above we managed to integrate the Sphinx decryption mix-net [4]. In addition, we have developed a prototype mix-net based on the re-encryption mix-net designed by Fauzi et al. [5]. This mix-net can be employed by Zeus through PANORAMIX. We discuss these advances in the following sections.

4.1 Sphinx

After deploying Sphinx to PANORAMIX, we used it to develop a prototype private chat room (which could also work as a private poll room). Notably, we have presented it as a demo at the 10th International Conference on Computers, Privacy & Data Protection (CPDP 2017). This prototype demonstrates how messages can be broadcast anonymously, without allowing an eavesdropper or even legitimate users to figure out which user sent which message (for more information refer to D4.2).

To start a negotiation in the case of Sphinx mix-net (static routing) we show a potential definition:

```
[
  {
      "peer_id": joint_peer,
      "endpoint_id": "our_sphinx_mix-net",
      "endpoint_type": "SPHINXMIX_STATIC_GW",
      "size_min": 3,
      "links": [{"from_endpoint_id": "peer2_mix",
                 "from_box": "OUTBOX",
                 "to_box": "PROCESSBOX"}]
  },
  {
      "peer_id": peer1,
      "endpoint_id": "peer1_mix",
      "endpoint_type": "SPHINXMIX_STATIC",
      "size_min": 3,
      "links": [{"from_endpoint_id": "our_sphinx_mix-net",
                 "from_box": "ACCEPTED",
                 "to_box": "INBOX"}]
  },
  {
      "peer_id": peer2,
      "endpoint_id": "peer2_mix",
      "endpoint_type": "SPHINXMIX_STATIC",
      "size_min": 3,
      "links": [{"from_endpoint_id": "peer1_mix",
                 "from_box": "OUTBOX",
                 "to_box": "INBOX"}]
  }
]
```

4.2 PANORAMIX Re-encryption mix-net

In the context of WP5, we developed the re-encryption mix-net proposed by Fauzi et al. [5]. Specifically, this is a new non-interactive zero-knowledge proof (NIZK) [2] shuffle argument. In a standard zero-knowledge proof one party (the prover) can prove to another party (the verifier) that a given statement is true, without conveying any information apart from the fact that the statement is indeed true. NIZK proofs are a variant of zero-knowledge proofs in which no interaction is necessary between prover and verifier. This new shuffle argument has a simple structure, where the first verification equation ascertains that the prover has committed to a permutation matrix, the second verification equation ascertains that the same permutation was used to permute the ciphertexts, and the third verification equation ascertains that input ciphertexts were "correctly" formed.

Initial tests indicate that this mix-net is faster than the Sako-Kilian mix-net that is currently employed by Zeus. We intend to further test it and use it in an election very soon because our goal is to provide robust and verifiable private elections hat scale up to 100K - 1M ballots. The implementation of the mix-net can be found in the following URL: https://github.com/grnet/ac16

4.3 Security Considerations

As we discussed in D5.1, there are many security and performance requirements that must be taken into account in the e-voting context. In the following, we discuss how we have considered them while 1) integrating the Sphinx mix-net, and 2) implementing the re-encryption mix-net.

Messages must be verifiable, which means that senders must be convinced that their message will be delivered. In this sense, both mix-nets provide means for verification of the correctness of the messages in the output batch. Anonymity is also very important because an adversary should not be able to link senders with the message they submitted. Messages should also be private, meaning that only the recipient can read and publish the messages. Finally, the integrity of the message is very important which means that a message comes out of the mix-net without any modifications. As we discussed previously, the MVP already can use the new and faster re-encryption mixnet presented in Section 4.2. Distribution of Trust is a major issue in e-voting. Typically, the e-voting committee should include an independent institution, the computer system administration, and all major competing interests in the election. In the context of our current MVP, a mix-net like the one mentioned earlier can be easily composed by contributed servers of various parties or organizations that do not share infrastructure. These parties should be also able to verify the security of the mix-net.

4.4 Usability Considerations

As we have discussed extensively in D5.1, an e-voting application should be easy to learn how to use and easy to remember how to use. Hence, in the PANORAMIX compatible Zeus version we have considered numerous usability issues. Note that, in the context of PANORAMIX usability involves: a) voters and b) non-experts who wish to deploy a mix server and contribute to the process. Our considerations involved:

- **Efficiency:** Goals are easy to accomplish quickly and with few or no user errors.
- **Intuitiveness:** Interfaces are easy to learn and navigate; buttons, headings, and help / error messages are simple to understand
- **Low perceived workload:** Interfaces appear easy to use, rather than intimidating, demanding and frustrating.

Trade-offs between usability and security have been resolved towards usability. For example, the audit vote feature was made inconspicuous for the voter whereas originally in the Benaloh challenge [1] the voter has to make an explicit choice every time.

Another example is that in the voter workflow adopted in the initial version of Zeus, the e-voting system relies on the user to record cryptographic identifiers themselves. However, requiring that the identifiers are copied and pasted into a form would force users to acknowledge and perhaps keep the identifier around enabling them to later verify the procedure. We plan to update the workflow in the new version of Zeus in order to bring balance to security and usability. We will further discuss this in the upcoming deliverables (e.g. D5.3).

The election control panel and the control panel for trustees is not intuitive enough for non-technical trustees to operate on their own. As a result trustees often rely too much on the technical operator of the user interface for controlling the voting process and even delegating trustee operations to them (e.g., put their key in portable storage).

Enhanced usability will make participants more independent promoting overall security and trustworthiness.

We will attempt to address the aforementioned usability issues in the upcoming months and in the context of WP5. We will explain how we fulfilled thosed requirements (together with other ones like performance and availability) in D5.3 and D5.4.

5. Conclusions

In this deliverable we discussed how the PANORAMIX MVP elevates the functionality of Zeus. Researchers and practitioners can easily integrate their mix-nets in PANORAMIX. In the context of WP5 we have highlighted this feature by integrating Zeus current re-encryption mix-net and the Sphinx mix-net in the PANORAMIX platform. Furthermore we have implemented a new, functional re-encryption mix-net based on the work done in WP3. Given that Zeus is already running on the PANORAMIX-based infrastructure, it can take advantage of this mix-net in order to become more efficient (which is actually the case based on our initial results). We are planning to further test the integration of the newly implemented re-encryption mix-net (discussed in Section 4.2) and eventually use it in the next version of Zeus.

Bibliography

[1] Josh Benaloh. Simple verifiable elections, 2006.

[2] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 103–112, New York, NY, USA, 1988. ACM.

[3] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981.

[4] George Danezis and Ian Goldberg. Sphinx: A compact and provably secure mix format. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, SP '09, pages 269–282, Washington, DC, USA, 2009. IEEE Computer Society.

[5] Prastudy Fauzi, Helger Lipmaa, and Michal Zajac. A shuffle argument secure in the generic model. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, pages 841–872, 2016.

[6] Bruce Schneier. *Applied Cryptography (2Nd Ed.): Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[7] Georgios Tsoukalas, Kostas Papadimitriou, Panos Louridas, and Panayiotis Tsanakas. From Helios to Zeus. In *2013 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '13, Washington, D.C., USA, August 12-13*, 2013.