TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Taaniel Kraavi 221968IVCM

Proving Vote Correctness in the Estonian Internet Voting System

Master's Thesis

Supervisors: Prof. Ahto Buldas
PhD

Jan Willemson
PhD

Tallinn 2024

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Taaniel Kraavi 221968IVCM

E-hääle korrektsuse tõestamine Eesti e-valimistel

Magistritöö

Juhendajad: Prof. Ahto Buldas
PhD

Jan Willemson
PhD

Tallinn 2024

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Taaniel Kraavi
12.05.2024

Abstract

Online voting has been used in Estonia for politically binding elections since 2005. The Estonian voting scheme (IVXV) uses a re-encryption mix-net for anonymising digital encrypted ballots before their decryption. While zero-knowledge proofs of vote correctness are common for online voting schemes based on homomorphic aggregation, they are less prevalent among schemes using mix-nets like IVXV. If a ballot does not contain a well-formed vote for a valid candidate, its arbitrary content might identify a voter even after ballots have been anonymised through cryptographic mixing. The zero-knowledge proofs of correct decryption issued by the decryption application cannot therefore be published for invalid votes due to the risk of leaking voter-identifying information. As a result, the public must trust an auditor with verifying that the decryption application has not arbitrarily declared ballots as invalid. This contrasts with valid votes, for which observers can request decryption proofs to verify them themselves. This work proposes a scheme for IVXV where a voter must prove that their ballot contains a vote for an eligible candidate, without revealing who the vote is for. After analysing multiple zero-knowledge proof systems, an approach based on the Bulletproofs range proofs was chosen. In addition, a prototype was developed and was used to benchmark IVXV with the proof verification added in. The results show that for well-chosen parameters, the proofs of vote correctness only have a small impact on the performance of IVXV, and the proposed approach is therefore practical. By using zero-knowledge proofs of vote correctness in IVXV, ballots for invalid candidates can be rejected during the voting phase, and therefore cannot reach the decryption stage. As a result, all zero-knowledge proofs of correct decryption can be made publicly available, e.g. published online for anyone to verify, therefore increasing the transparency and auditability of the system.

The thesis is written in English and is 73 pages long, including 9 chapters, 1 figure and 2 tables.

Annotatsioon

E-hääle korrektsuse tõestamine Eesti e-valimistel

Eestis on valimistel e-hääletamist kasutatud alates 2005. aastast. Eesti e-hääletamissüsteemis (IVXV) kasutatakse selleks, et krüptitud e-hääli enne dekrüptimist anonüümida, ümberkrüptimisel põhinevat miksnetti. Kuigi hääle korrektsuse nullteadmustõestuste kasutamine on homomorfsel agregeerimisel põhinevate e-hääletamissüsteemide puhul tavapärane, on miksnetil põhinevate skeemide puhul, nagu IVXV, nende kasutamine vähem levinud. Kui e-hääl ei sisalda nimekirjas oleva kandidaadi kohta korralikult vormistatud valikut, siis selle suvalise sisu põhjal võib hääletaja olla tuvastatav ka pärast häälte anonüümimist miksnetiga. Dekrüptimisrakenduse poolt genereeritud dekrüptimise korrektsust tõestavaid nullteadmustõestusi ei saa seega avaldada, sest sellega võib kaasneda valijaid identifitseeriva info leke. Seda, et dekrüptimisrakendus ei ole omavoliliselt sedeleid kehtetuks tunnistanud, saab kontrollida üksnes audiitor. Seevastu valiidsete sedelite puhul tohivad dekrüptimistõestusi välja nõuda ja kontrollida ka vaatlejad. Selles töös kirjeldatud IVXV täiendus näeb ette, et e-hääletaja tõestab, et on andnud e-hääle nimekirjas oleva kandidaadi poolt, seejuures lekitamata, millise kandidaadi poolt. Pärast mitme nullteadmussüsteemi analüüsimist valis autor välja *Bulletproofs*'i vahemiktõestusel põhineva lähenemise. Lisaks programmeeris autor prototüübi, mida kasutati, et koormustestida IVXVd koos tõestuste kontrollimisega. Tulemused näitavad, et hästi valitud parameetrite puhul ei ole e-hääle korrektsuse tõestustel IVXV-le märkimisväärset lisakoormust ja seega on pakutud lahendus ka reaalselt kasutatav. E-häälte korrektsust tõestavate nullteadmustõestuste kasutamine IVXVs võimaldab kehtetud sedelid juba hääletamise etapis tagasi lükata, mis väldib kehtetute sedelite jõudmist dekrüptimisfaasi. Selle tulemusel on võimalik kõik korrektse dekrüptimise nullteadmustõestused avalikustada, nt avaldada kõigile kontrollimiseks internetis, mis suurendab IVXV läbipaistvust ja auditeeritavust.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 73 leheküljel, 9 peatükki, 1 joonist, 2 tabelit.

Acknowledgements

I would like to thank my supervisors Ahto Buldas[*] and Jan Willemson[†] for supporting me throughout writing this thesis. In addition, I am thankful to Jan Willemson for proposing this research topic to me. My sincere thanks also go to Peeter Laud[†] and Sven Laur[‡] for helping me with some cryptographic technicalities in my research. I am grateful to Helger Lipmaa[‡] for helpful discussions regarding the state of the art of range proofs, and Sven Heiberg[††] for discussions surrounding IVXV. I would also like to thank Artjom Zingfeld[††] who has not only been a great colleague, but without whom benchmarking my prototype on real IVXV infrastructure would not have been possible. Lastly, I am thankful to Jelizaveta Vakarjuk[†] for reviewing a draft of this work and suggesting a number of improvements.

[*]Tallinn University of Technology
[†]Cybernetica
[‡]University of Tartu
[††]Smartmatic-Cybernetica Centre of Excellence for Internet Voting (SCCEIV)

List of abbreviations and terms

ASN.1	Abstract Syntax Notation One, a data serialisation format
CRS	Common reference string
DER	Distinguished Encoding Rules, a set of rules for ASN.1
DLOG	Discrete logarithm
IVXV	The codename for the current Estonian i-voting scheme
NEC	National Electoral Committee (EE)
NIKZP	Non-interactive zero-knowledge proof
NIST	National Institute of Standards and Technology (US)
PKI	Public key infrastructure
PPT	Probabilistic polynomial time
ROM	Random oracle model
RSA	Rivest–Shamir–Adleman, a public-key cryptosystem
SEO	State Electoral Office (EE)
SHA	Secure Hash Algorithm
SHVZK	Special honest-verifier zero-knowledge
ZK	Zero-knowledge
ZKP	Zero-knowledge proof
ZKPoK	Zero-knowledge proof of knowledge
ZKRP	Zero-knowledge range proof
ZKSM	Zero-knowledge proof of set membership
zk-SNARK	Zero-knowledge succinct non-interactive argument of knowledge

Table of contents

List of figures

List of tables

1. Introduction

Internet voting, or i-voting, was first used in Estonia for its municipal elections in 2005, with 1.9% of voters voting over the Internet [1, 2]. I-voting has been used for each election since, and the 2024 European Parliament elections in Estonia will mark the 14th i-voting event. The 2023 parliamentary elections were the first Estonian elections where i-voters were the majority, with 51.1% of participating voters casting i-votes [2]. In English, the term 'i-voting' is preferable over 'e-voting' to distinguish the Estonian system from other electronic voting technologies, such as electronic voting machines or blockchain voting.

The current Estonian i-voting scheme is codenamed 'IVXV', and was introduced in 2017 [3]. IVXV follows the double envelope postal voting scheme [3, 4], where an unmarked envelope that contains a voter's ballot is included in an outer envelope marked with the voter's information and signature. The inner envelope hides the ballot while the outer envelope enables identification. IVXV achieves this digitally with public-key cryptography. Each ballot is encrypted with an election-specific public encryption key, and each voter digitally signs their ballot using their personal private signing key[1] [3]. Encryption provides secrecy while digital signatures establish the identities of voters. Digital signatures also provide integrity for the ballots, since a third party cannot alter an encrypted and signed ballot without voiding a voter's digital signature on it. Before decryption, ballots are decoupled from their digital signatures in a process called 'mixing', which involves cryptographic re-randomisation and shuffling [3]. Re-randomisation breaks the direct digital link between a ballot and its signature, while shuffling ensures that the two cannot be correlated by auxiliary information, e.g. by the order of mix-net inputs and outputs. This ensures that choices cannot be linked back to voters, which preserves ballot secrecy.

For auditability purposes, when ballots are decrypted during the tallying phase, zero-knowledge proofs of correct decryption are created by the *key application* which performs the decryption [3]. These proofs are mathematical data that allow verifying—without needing the secret decryption key—that decrypted ballots correspond to their encrypted counterparts. Therefore, a valid proof asserts that the key application did not decrypt a ballot to a different vote than what was encrypted. Because the key application decrypts mixed and therefore anonymised ballots, these ballots cannot seemingly be linked to the original voters. This suggests that the mixed ballot box and the proofs of correct decryption can be published for anyone to verify as a transparency measure. However, this

[1]The signature can be given with the ID-card, Digi-ID, or Mobile-ID [4].

unlinkability only holds for ballots that are well-formed, i.e. where a valid candidate is encrypted according to the protocol specification. This issue is similar for paper ballots, where an incorrectly marked ballot could be associated with a voter, e.g. for a voter writing their name instead of a candidate number.

Since digital votes are encrypted on the personal devices of i-voters, it is feasible to encrypt and submit a choice that is not valid by manipulating the voting client. While these invalid votes are detected during decryption and are not counted towards the official tally, they cannot be made public since this would allow attacks on voter privacy [5–7]. For example, if invalid ballots were made public, it would become easier for a coercer to force a voter to submit an invalid vote, thereby disenfranchising the voter [5]. After forcing a voter to vote for some non-existent candidate using a modified voting client, the coercer could then verify whether this invalid vote appears in the published proofs once the election concludes. If so, the coercer would gain the assurance that the voter did not subsequently override their vote, re-voting being the current mechanism in IVXV for mitigating coercion concerns [4]. A potential solution to this problem is to implement proofs of ballot validity into the vote casting process.

1.1 Research problem

The highlighted significant shortcoming of IVXV is that currently, ballots for invalid candidates can reach the decryption stage. As a result, some decryption results and proofs may need to be withheld from the general public to prevent potential attacks [5–7], such as the aforementioned disenfranchisement attack. The current organisational measures for preventing leaking information through invalid ballots therefore result in conditional auditability, which is not ideal. This shortcoming forms the core research problem of this work.

The main focus is to find a solution to prove, in zero-knowledge, that a ballot contains the encryption of a valid candidate. That is, the voting client should prove to the vote collection server that the ballot contains a valid choice, without revealing who the vote is for. For this, both the mathematical but also the practical feasibility of different approaches must be analysed. The main research questions therefore are:

- **RQ1.** How to prove in zero-knowledge that an encrypted ballot represents a valid candidate in IVXV?
  - **RQ1.1.** Which zero-knowledge proof (ZKP) systems are suitable for this goal?
  - **RQ1.2.** Which changes are needed in IVXV to enable the use of such proofs?
  - **RQ1.3.** Is the use of such proofs practical in IVXV or can it be made practical?

- **RQ2.** How does the use of ZKPs for vote correctness impact/benefit the rest of the IVXV system?
  - **RQ2.1.** What are the performance considerations of the chosen approach?
  - **RQ2.2.** When and where should the proofs be verified?
  - **RQ2.3.** Do the proofs obsolete existing technical/organisational measures?

1.2 Contributions

The use of zero-knowledge proofs for proving ballot correctness is common for schemes with homomorphic tallying, but is less studied for general mix-net based schemes like IVXV. This work is therefore one of the first where the applicability of proofs of vote correctness for general mix-net based schemes is analysed. While the analysis in this work is specific to IVXV, it is potentially extensible to any mix-net voting scheme based on the ElGamal cryptosystem. The core contributions of this work are the following:

- Multiple works were combined into a scheme to prove in zero-knowledge, and across two separate algebraic groups, that an encrypted integer lies within a range. A group switching strategy was used to perform the proof in a computationally performant algebraic group, and link it to the computationally more expensive group that ElGamal ciphertexts are part of.
- Proofs of vote correctness were introduced to IVXV. Due to the group switching strategy, the scheme can be added to IVXV with only a small performance overhead, and with minimal modifications to the cryptosystem used in IVXV. In particular, only the digital ballot format must be changed, and a derivative of the ballot must be encrypted instead of the ballot itself.
- A proof of concept was developed in Go[2] to demonstrate that the scheme works and to benchmark the scheme's performance. The benchmarks can be used by the State's Electoral Office (SEO) to decide whether the proposed scheme can be integrated into IVXV or whether more performant alternatives must be found. While the code is of research quality rather than production quality, it can still serve as a reference for a production implementation. Notably, Go was chosen for the prototype since the IVXV server code is written in Go.

If the vote correctness proofs presented in this work are incorporated into IVXV, the problems caused by the decryption of arbitrary ballots are effectively solved. As a result, the inputs and outputs to the tallying process can be published online, therefore greatly improving the transparency and verifiability of IVXV.

[2] `https://go.dev`

Should the presented scheme not be judged performant enough by the SEO, the work presents suggestions on potential performance improvements, and highlights promising approaches for future work.

1.3 Organisation

This thesis is organised as follows. A general overview of IVXV is provided in Chapter 2. Work related to proving ballot correctness is covered in Chapter 3 and the cryptographic background is introduced in Chapter 4. The process of elimination by which the state of the art of different proof systems was reviewed and the suitable proof system was selected is detailed in Chapter 5. The full protocol for proving vote correctness is presented in Chapter 6 and its benchmarks are given in Chapter 7. Chapter 8 contains a discussion on the complexity, performance, and other details related to the proposed protocol. Finally, Chapter 9 concludes this work.

2. IVXV

IVXV was released in 2017 with the aim of improving the individual verifiability of many system components. Notably, the encryption scheme was changed from RSA[1] to ElGamal which enabled introducing both the proofs of correct decryption and the mixing process into IVXV [3]. This chapter aims to provide the reader with the background on the IVXV voting scheme necessary to understand this work. For a complete overview on IVXV, the official documentation[2] should be consulted instead.

Many verifiability definitions have been proposed for i-voting protocols [8], but the fundamental idea is that it should be possible to verify that the election results match with the intent of voters. Individual verifiability should allow voters to verify that their intended vote was indeed stored in the ballot box (cast-as-intended). Universal verifiability should allow anyone to verify that the election results match with the ballot box contents. Verifiability should not infringe on ballot secrecy however, which means that no-one should be able to determine how a voter voted. Ballot secrecy is a constitutional requirement in Estonia [9, §60, §156].

Because i-voting is done on the personal devices of voters, and not on trusted hardware in voting booths, there is a risk of coercion, vote manipulation, and vote selling. Therefore, an important functionality of IVXV is the capability of voters to override their i-vote, which helps mitigate these issues. Votes can be overridden in two ways [4]: by casting a new i-vote (re-voting), or by casting a paper vote. During the i-voting phase, voters can vote multiple times, with their most recent vote overriding their previous votes since only one vote must be counted per voter. However, the last i-vote is not necessarily the last valid i-vote [10], which is an important distinction under the assumption that the voting client cannot be trusted. If a voter also casts a paper vote, their i-votes are not considered at all. In short, only the last i-vote cast by a voter is counted, unless the voter has also voted on paper, in which case no i-votes are counted for this voter.

The four main parties involved in the scheme are the voter, the collector, the processor, and the tallying party [4]. Since the election organiser performs the tallying, the term *organiser* is used hereinafter to refer to the tallying party. I-voting itself is split into four main phases: setup, voting, processing, and tallying [4]. The phases may not overlap, and each phase

[1] More specifically, RSA with the OAEP padding scheme was used.

[2] `https://www.valimised.ee/et/e-haaletamine/dokumendid` Note: some documents are only available in Estonian.

begins when the previous stage ends. The overlap with the general election phases and physical voting is not described here.

In Estonia, anyone can register themselves as an observer for an election [11, §19[4]]. In turn, observers can observe part of the setup, processing and tallying phases. The following summaries of election phases are based on the author's working knowledge of the system[3], on his experience as an observer, on the IVXV source code [12], and on [4, 13, 14].

2.1 Election setup

During the setup phase, the data necessary for the election is prepared, the digital systems are configured, and the official i-voting client is published. Moreover, the election-specific key-pair used to encrypt and decrypt the votes cast in the election is generated with the *key application* on an air-gapped computer with no persistent storage. The key generation itself is a public ceremony, meaning that the use of the key application can be observed by the registered observers. The observers gain no knowledge of the secret key or other secret information. Due to the described setting, the key generation process can be considered as being a *transparent setup* (or trusted setup) process.

The election key is generated following a (t, n)-threshold scheme, where the secret key is split into n shares and distributed among n key-holders. To perform operations with the secret key (e.g. decryption), at least t out of n key-holders must collaborate. These key-holders are trusted members of the State's Electoral Office and of the National Electoral Committee (NEC), and the values $t = 5, n = 9$ have been used since 2017 [15–19]. During a normal election process, the key is not reassembled before the tallying phase, and no party is therefore able to decrypt arbitrary ballots during the voting and processing phases. While the secret decryption key is split into shares, the public key is not, and this public key is published[4] online after the key generation process. For simplicity, and unless specified otherwise, the decryption key is considered to be unavailable and the public key is considered to always be available in the rest of this work.

2.2 Voting

To cast an i-vote, a voter should download the official *voting client* published by the SEO. The voting client enables the voter to select their preferred candidate, encrypt the ballot, digitally sign it, and send it to the collector. The collector then returns to the voting

[3] At the time of writing, the author was a protocol analyst at SCCEIV, the technical developer of IVXV.
[4] While the public key is not published on the website of the elections, it is available e.g. from the open endpoint providing the configuration file for the verification application.

client the vote qualifying elements, which the client verifies to attest whether the collector performed all required operations. While the voting client verifies whether the collector behaved properly, there needs to be a separate mechanism for verifying whether the voting client behaved properly. For this, the voter can use the *verification application* on a second device to verify that their vote was cast and processed correctly. However, the verification application does not deter a voter who intends to cast an invalid ballot.

The *collector* is a combination of servers which perform multiple tasks. It provides to the voting client the data necessary for casting a vote, such as the candidate list available to the voter. It also receives ballots from the voting client, stores them in the digital ballot box, and serves requests for the verification application. Upon receiving a ballot, the collector performs feasible validity checks, such as verifying that the signature on the ballot is valid and that the voter is eligible to vote. The collector also acquires a timestamp for the ballot from a trusted timestamping service, since the time on voter devices cannot be trusted. After having stored the ballot in the digital ballot box, the collector returns the vote qualifying elements to the voting client.

2.3 Processing and tallying

After the voting phase concludes, the processor obtains the digital ballot box and other integrity information such as checksums and logs from the collector. In practice, the election organiser also holds the role of the processor. Using the *processing application*, the organiser verifies the integrity of the data supplied by the collector, including the integrity of the ballot box and of the signatures on the ballots. The organiser also discards votes overridden by re-voting and paper voting to keep only eligible i-votes, and strips the digital signatures from these votes. Finally, the organiser uses the mixing application (mix-net) to cryptographically anonymise the ballots, since otherwise they could still be correlated with the stripped digital signatures. While the processing phase is observable, the data generated by the processing application cannot be made publicly available since it contains sensitive information. For example, it contains the ID codes of voters and ballot timestamps which could be used to determine whether a voter re-voted.

The anonymised encrypted ballots are then transferred to an air-gapped computer for decryption with the key application. Notably, while the key application has the capability to verify whether the structure of encrypted ballots is correct, this option is disabled in practice for performance reasons. The assumption is that this check was performed during the processing stage, and the output data of the processing stage is considered trusted. Before the votes can be decrypted, the trusted parties holding the secret key shares must present their key-shares to the key application. Once the quorum has been reached,

decryption can begin. In addition to decrypting the ballots, the key application also verifies whether the decrypted results are valid, tallies the valid results, and generates the proofs of correct decryption. While the decryption phase is observable, not all outputs can be made public in this phase either.

Decryption is the step where the core problem considered in this work arises. Intuitively, in a good cryptosystem, the encrypted form of some data should not leak any information about this data. As such, given only an encrypted ballot and no auxiliary information, it should not be possible to determine whether a valid choice is indeed encrypted. This is why the validity of decrypted ballots must currently be checked by the key application. The key application is given the list of candidates and the secret key, and so it can decrypt votes and compare the outputs with what is specified by the candidate list. Since there is no knowing what sort of information leak can be caused by arbitrary ballot contents, the current practice in IVXV is to not publish the invalid votes. These invalid votes are not made available to the observers either. As a result, the observers must trust an auditor with verifying that the key application did not arbitrarily declare votes as invalid.

As a result, the decryption process currently produces conditional outputs: if no invalid votes appear, everything can be made public. Otherwise, only the valid votes and their proofs can be made public. In the latter case, the tally is no longer verifiable by third parties, which hinders the universal verifiability of IVXV. A mechanism is therefore needed to prevent ballots from reaching the final anonymised ballot box, so that the decrypted ballot box could always be matched against the tally results and decryption proofs.

2.4 Ballot structure

In IVXV, the plaintext ballot, i.e. the unencrypted ballot, is a text file that contains the candidate's choice according to a specific format. The exact format is legally fixed [20] and its specification in Augmented Backus-Naur Form[5] can be found in Appendix 2. More generally, the plaintext ballot contains the candidate's district, their candidate number, the name of their party, and their name as written on the candidate list. Candidate numbers are unique for parliamentary and European Parliament elections, in which case the four-digit district code is fixed to '0000' [20]. For local elections, the numbers are unique only within the electoral district, which is the municipality, and the district code is then the municipality's EHAK code[6]. Candidate numbers start at 101 within the district and are incrementally assigned to candidates by drawing lots. While no number should be skipped between 101 and the highest candidate number, a gap may appear if a candidate's

[5]As defined in RFC5234 (`https://www.rfc-editor.org/rfc/rfc5234`)

registration is cancelled before the candidate lists are locked, e.g. if a candidate dies.

As an example, let candidate ‘Taher ElGamal’ be a candidate with the number ‘128’ in a district with the code ‘1955’. Additionally, let Taher ElGamal be part of the ‘Famous cryptographers’ party. Then, the unpadded plaintext for Taher ElGamal would be

```
1955.128%x1FFamous cryptographers%x1FTaher ElGamal
```

in the local government elections. The first four digits represent the EHAK district code and the following digits represent the candidate number. The subsequent text strings represent the party name and the candidate’s name, respectively. `%x1F` represents the Unicode Unit Separator character.

To convert the plaintext ballot into an integer for encryption, it is first padded to a fixed size, and then the byte representation of the file is interpreted as a big-endian integer [21]. Clearly, two candidates with consecutive candidate numbers are unlikely to have their ballots be represented by consecutive integers since the low order bytes represent names and not numbers. More generally, the current digital ballot format is highly structured, a property which is not preserved by mathematical operations on the data.

[6]EHAK codes are four-digit identifiers used to classify Estonian administrative and settlement units.

3. Related work

There are two areas in the IVXV i-voting scheme where zero-knowledge proofs are used: verification of mixing integrity and verification of correct ballot box decryption. The different attacks against the IVXV voting scheme expose the properties needed for implementing countermeasures, properties which partly differ for both uses of ZKPs. The process for discovering research materials thus started with attack discovery, followed by snowballing through references up to the root primitives of the subject. Once the root primitives were identified, reverse snowballing and additional search served as the basis for finding state of the art research to consider for the literature review.

The main database used to find unfamiliar, relevant research was Scopus[1] by Elsevier, especially for material with respect to ZKPs themselves. Due to the complexity, esoteric nature, and recency of research surrounding the problems at hand, citation counts were not a reliable metric for selection. Rather, the primary selection argument was the field-reputation of the authors and journals where the works have been published. In addition, the tables of contents of the Advances in Cryptology conference series — comprising CRYPTO, EUROCRYPT, and ASIACRYPT — were searched through for relevant materials. Less systematically, additional references were sourced by browsing the Cryptology ePrint Archive[2], from discussions with researchers at the E-Vote-ID[3] conference of 2023, and the proceedings of previous conference events.

The Scopus filter used was

```
KEY(zero-knowledge OR SNARK) AND KEY(membership OR
    range-proof) AND NOT KEY(lattice-based)
```

with the exclusion of lattice-based schemes due to their incompatibility with IVXV. From there onwards, titles and abstracts were used to discard initial superfluous material. Then, conclusions and literature reviews of the remaining papers were evaluated to pick the candidates for a full review.

[1] https://www.scopus.com/
[2] https://eprint.iacr.org
[3] https://e-vote-id.org/

3.1 Attacks based on arbitrary ballot contents

In 2017, Wikström et al. proposed ways for attackers to leverage elections to force a dilemma upon an election's organisers [6]. The idea behind the attacks is that sensitive information may be embedded in a ballot, which would then have to be revealed during the counting or auditing process. Intuitively, to mitigate such attacks without impacting transparency, arbitrary information must be prevented from reaching the ballot box. The attacks proposed by Wikström et al. are not restricted to electronic voting, and do not impact a voting system itself.

In 2022, Müller described a theoretical attack against IVXV, in which an attacker can learn the votes of several voters by crafting a cleverly encoded ballot [7]. The attack relies on multiple assumptions, one of which is the ability of the attacker to submit an invalid ballot, but also recover its decryption. While this attack is not practically feasible under the current organisational safeguards used in IVXV, it does highlight a weakness in the protocol. Currently, the IVXV decryption application outputs invalid votes into a separate file [22] which is not shared with the general public, but is accessible to auditors [3]. Still, Müller showed that the resilience of IVXV is misrepresented against the intended threat model [3, 7].

Additionally, the key application did not previously output ZKPs of correct decryption for invalid ballots at all [22]. This, in turn, prevented a complete audit from being conducted: there was no cryptographic proof that a decrypted vote labelled as invalid indeed was invalid. For the 2024 elections, the author fixed this problem by adding the capability to the key application to generate ZKPs of correct decryption also for invalid ballots[4]. While the auditor can now verify that ballots deemed invalid were indeed correctly decrypted, observers must still trust the auditor and are not allowed to verify the proofs themselves.

3.2 Proofs of vote validity for homomorphic tallying

To eliminate the possibility of the aforementioned attacks, proofs of vote validity can be used. That is, a voter must issue a proof alongside their encrypted ballot that a legal option was encrypted. The challenge is doing so without revealing what was encrypted, i.e. in zero-knowledge.

Mathematical proofs of vote correctness were, to the best of the author's knowledge, first introduced by Cohen and Fischer in 1985 [23]. The main idea of the scheme is that voters

[4]At the time of writing, the source code for the 2024 elections was not yet published.

prepare unmarked ballots which have the encryptions of a yes and a no value. Voters then prove that their ballot has indeed only those values without revealing which is which. To cast a vote, voters select the desired value and submit it to the election organiser. Finally, the organiser combines the votes and publishes the tally and a proof that the latter is correct. Cohen and Fischer also describe how their scheme could be extended to more than two choices and propose intractable problems suitable as a security basis of such schemes.

A significant downside to the scheme of Cohen and Fischer is, however, that the organiser can decrypt the individual votes [23]. Hence, the proofs of correctness are necessary for voters to verify the consistency of the tally, and not for the organiser, who can simply decrypt votes. Benaloh and Yung later contributed an improvement to the scheme which addressed this very problem by splitting the organiser into parties which must collaborate for computing the tally [24]. Their work also established the terminology of *marking* to designate voters selecting their choice. Benaloh then elaborated on those ideas further and introduced the use of secret sharing for privacy preservation, and of homomorphic encryption and tallying for the proof process [25].

All three schemes require interactive proofs of vote correctness [23–25] and serve to prove the correctness of the tally. The schemes therefore do not consider that a party may need to verify vote correctness before any tallying. In 1997, Cramer, Gennaro and Schoenmakers first used zero-knowledge vote correctness proofs in the context of ElGamal ciphertexts [26]. Notably, the proofs are non-interactive (NIZKP), and verifiable without access to any secret value, but are still only usable with homomorphically tallied votes. This approach remains the most widely used proof system in homomorphic tally voting systems [27].

Ideally, similar NIZKPs should be generated for votes with more complex or abstract information to that of voting systems with homomorphic tally. However, unlike for voting systems with homomorphic tally [28–32], such proofs of correctness are not strictly required for the validity of traditional ‘decrypt then tally’ schemes. There is therefore significantly less research available which targets the validation of arbitrary digital ballots.

Using homomorphic tallying in Estonia has been considered and studied before, but implementing such a change would require a significant architectural redesign of the current IVXV scheme [33]. Moreover, when considering local elections where different regions have different candidates, votes must contain information linking them to a specific region [3] which was not considered in [33]. While this could be achieved by unique candidate numbers across the country or by collecting ballots into separate ballots boxes, both approaches would require further analysis for assessing their practicality.

3.3 Proofs of vote validity in practice

In the Civitas voting scheme [28], the re-encryption proofs of Hirt and Sako [34] are used to prove that a vote is the re-encryption of an existing ciphertext. That is, encryptions for all valid candidates are published by the election authority, and the voter proves that their ballot is a re-encryption of one of these votes. In essence, this is a zero-knowledge proof of set-membership since the proof must not reveal which ballot was re-encrypted. This paradigm was revisited and extended by Joaquim [35] who proposed the addition of an additional proof of structure. In the scheme, the ballot is obtained from the re-encryptions of multiple options and an additional ZKP is used to prove the structure of the combination. The structure proof is based on the proof of knowledge of representation by Brands [36], but is a novel addition for ElGamal by Joaquim.

In the Kryvos voting system [37], zero-knowledge succinct arguments of knowledge (zk-SNARK) are used to prove ballot validity. More specifically, the proof system due to Groth [38] is used to prove that the committed vote shares correspond to the valid choice space. VoteAgain [39] uses the zero-knowledge set-membership proofs by Bayer and Groth [40] to prove that a ballot represents a valid candidate. In the code-based SwissPost voting scheme [41] used in Switzerland, voters use codes delivered to them by post to vote for specific candidates. A mechanism specific to their code-based implementation is also used to verify that a voter has cast a vote for a candidate they are allowed to vote for. However, it is unclear whether the verification method is novel or if it is an adaptation of an existing scheme.

4. Preliminaries

4.1 Notation

Let $\lambda \in \mathbb{N}$ denote a security parameter and let 1^λ be its unary representation. A probabilistic polynomial time (PPT) adversary $\mathcal{A}$ is a probabilistic interactive Turing Machine that runs in polynomial time in the security parameter λ. Let $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ be the integers modulo q with representatives $\mathbb{Z}_q = [0, q-1]$, and let $[a, b]$ be an interval in $\mathbb{Z}$.

Given a finite set S, $s \xleftarrow{\$} S$ denotes the sampling of s uniformly at random from S. For a randomised algorithm A with input x, let $y \leftarrow A(x; r)$ denote its execution with explicit randomness r. When the randomness needs not be explicit, $y \leftarrow A(x)$ is used instead, with the assumption that r is then sampled accordingly.

Definition 1 (DLOG)**.** *Let $\langle g \rangle = \mathbb{G}$ be a q-element cyclic group generated by g, with q dependent on a security parameter λ. The discrete logarithm (DLOG) is hard in $\mathbb{G}$ if for all non-uniform PPT adversaries $\mathcal{A}$, there exists a negligible function μ such that*

$$\Pr[x \xleftarrow{\$} \mathbb{Z}_q, y \leftarrow g^x : x = \mathcal{A}(g, y)] \leq \mu(\lambda) \, .$$

A DLOG-group is then a group where the DLOG is hard. From hereinafter, $\mathbb{G}$ is used to denote a DLOG-group of prime order, i.e. with a prime number of elements. Multiplicative notation for groups is used throughout.

4.2 Commitment schemes

Commitment schemes provide a way to commit—i.e. bind oneself—to a value without (initially) revealing it. Depending on the use case, the committer may (but needs not) subsequently reveal the committed value. A commitment scheme must be hiding so as not to reveal the committed value, and binding so that commitments can only be opened to one value. In this work, commitments must not be opened to protect vote secrecy, and thus the definition without an opening algorithm is considered.

Definition 2 (Commitment scheme)**.** *A non-interactive commitment scheme consists of a pair of PPT algorithms* $(\mathsf{Setup}, \mathsf{Com})$ *such that*

- *The setup algorithm* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *generates the public parameters* pp *for a security parameter* λ*. The public parameters specify the message space* $\mathcal{M}_{\mathsf{pp}}$*, randomness space* $\mathcal{R}_{\mathsf{pp}}$*, and commitment space* $\mathcal{C}_{\mathsf{pp}}$*.*
- *The commitment algorithm and* pp *specify the function* $\mathsf{Com}_{\mathsf{pp}} : \mathcal{M}_{\mathsf{pp}} \times \mathcal{R}_{\mathsf{pp}} \to \mathcal{C}_{\mathsf{pp}}$*. For a message* $m \in \mathcal{M}_{\mathsf{pp}}$*, the algorithm selects* $r \xleftarrow{\$} \mathcal{R}_{\mathsf{pp}}$ *uniformly at random, and computes the commitment* $c \leftarrow \mathsf{Com}_{\mathsf{pp}}(m; r)$*.*

Definition 3 (Hiding)**.** *A commitment scheme is* hiding *if for any non-uniform PPT adversary* $\mathcal{A}$*, there exists a negligible function* μ *such that*

$$\left| \Pr \left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pp}), \\ b \xleftarrow{\$} \{0,1\}, c \leftarrow \mathsf{Com}_{\mathsf{pp}}(m_b) : b = \mathcal{A}(c) \end{array} \right] - \frac{1}{2} \right| \leq \mu(\lambda),$$

where $\mathcal{A}$ *outputs* $m_0, m_1 \in \mathcal{M}_{\mathsf{pp}}$*. If* $\mu(\lambda) = 0$*, the scheme is said to be* perfectly *hiding.*

Definition 4 (Binding)**.** *A commitment scheme is* binding *if for any non-uniform PPT adversary* $\mathcal{A}$*, there exists a negligible function* μ *such that*

$$\Pr \left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); (m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(\mathsf{pp}) : \\ \mathsf{Com}_{\mathsf{pp}}(m_0; r_0) = \mathsf{Com}_{\mathsf{pp}}(m_1; r_1) \wedge x_0 \neq x_1 \end{array} \right] \leq \mu(\lambda),$$

where $\mathcal{A}$ *outputs* $m_0, m_1 \in \mathcal{M}_{\mathsf{pp}}$ *and* $r_0, r_1 \in \mathcal{R}_{\mathsf{pp}}$*. If* $\mu(\lambda) = 0$*, the scheme is said to be* perfectly *binding.*

Definition 5 (Pedersen commitment [42])**.** *Let* $\mathbb{G}$ *be a cyclic DLOG-group of prime order* q*. The setup algorithm then outputs random generators* $g, h \xleftarrow{\$} \mathbb{G}$ *such that* $\log_g(h)$ *is not known to anyone. To commit to* $m \in \mathbb{Z}_q$*, the committer samples* $r \xleftarrow{\$} \mathbb{Z}_q$ *uniformly at random and computes*

$$\mathsf{Com}_{\mathsf{pp}}(m; r) = g^m h^r.$$

Definition 6 (ElGamal commitment)**.** *Let the parameter generation be as for the Pedersen commitment. To commit to* $m \in \mathbb{Z}_q$*, the committer samples* $r \xleftarrow{\$} \mathbb{Z}_q$ *uniformly at random and computes*

$$\mathsf{Com}_{\mathsf{pp}}(m; r) = (g^r, g^m h^r).$$

Under the DLOG, Pedersen commitments are perfectly hiding and computationally binding, while ElGamal commitments are perfectly binding and computationally hiding. These results are well known and the proofs are therefore not reproduced here, but can be found in e.g. [43].

Both Pedersen and ElGamal commitments are additively homomorphic, which is shown by the following:

$$\begin{aligned}\mathsf{Com}_{\mathsf{pp}}(m_1; r_1) \cdot \mathsf{Com}_{\mathsf{pp}}(m_2; r_2) &= g^{m_1} h^{r_1} \cdot g^{m_2} h^{r_2} \\ &= g^{m_1} g^{m_2} \cdot h^{r_1} h^{r_2} \\ &= g^{m1+m2} \cdot h^{r_1+r_2} \\ &= \mathsf{Com}_{\mathsf{pp}}(m_1 + m_2; r_1 + r_2).\end{aligned}$$

Many cryptographic protocols rely on this property, and the homomorphism of Pedersen and ElGamal commitments is fundamental also to this work.

4.3 Encryption schemes

Public-key encryption schemes provide a way to hide data using an encryption key such that only a party having the corresponding decryption key can recover the data from the encryption. Public-key encryption schemes are also called public-key cryptosystems.

Definition 7 (Public-key cryptosystem)**.** *A public-key cryptosystem consists of a triple of PPT algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *such that*

- *The key generation algorithm* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *generates the keypair for a security parameter* λ*, where* pk *is the public key and* sk *is the private (secret) key. The public key* pk *specifies the message space* $\mathcal{M}_{\mathsf{pk}}$*, randomness space* $\mathcal{R}_{\mathsf{pk}}$*, and ciphertext space* $\mathcal{C}_{\mathsf{pk}}$*.*
- *For a message (plaintext)* $m \in \mathcal{M}_{\mathsf{pk}}$*, the encryption algorithm selects* $r \xleftarrow{\$} \mathcal{R}_{\mathsf{pk}}$ *uniformly at random, and computes the ciphertext* $c \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(m; r)$*.*
- *For a ciphertext* $c \in \mathcal{C}_{\mathsf{pk}}$*, the deterministic decryption algorithm recovers the plaintext* $m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c)$*, or outputs* $\perp$ *if the decryption failed.*

Definition 8 (ElGamal cryptosystem [44])**.** *Let* $\langle g \rangle = \mathbb{G}$ *be a cyclic DLOG-group of prime order* q *generated by* g*. The secret key* $x \xleftarrow{\$} \mathbb{G}$ *is randomly sampled, and the public key is computed as* $\mathsf{pk} \leftarrow g^x$*. To encrypt a message* $m \in \mathbb{G}$*, randomness* $r \xleftarrow{\$} \mathbb{Z}_q$ *is randomly sampled, and the ciphertext is computed as*

$$\mathsf{Enc}_{\mathsf{pk}}(m; r) = (g^r, m \cdot \mathsf{pk}^r) = (u, v).$$

The ciphertext (u, v) *is decrypted with*

$$\mathsf{Dec}_x(u, v) = v \cdot u^{-x} = m \cdot \left(g^x\right)^r \cdot \left(g^r\right)^{-x} = m.$$

Encrypting g^m instead of m results in the *lifted* ElGamal cryptosystem. This also means that the decrypted message becomes g^m instead of m, and to recover m, the discrete logarithm of g^m must be computed. This is not a problem when m is small, e.g. $m < 2^{16}$, since the values $g^0, g^1, \ldots, g^{2^{16}-1}$ can feasibly be precomputed. In turn, the performance impact of recovering m through a lookup table of the precomputed values is negligible.

The lifted ElGamal cryptosystem is functionally equivalent to the ElGamal commitment (Definition 6) for any party who does not know x, i.e. the discrete logarithm of h to the base g. If a party knows x, they can recover m by decrypting the ciphertext/commitment, and therefore the hiding property is lost. On the other hand, lifted ElGamal ciphertexts are interoperable with Pedersen commitments, which makes them compatible with many zero-knowledge proof systems based on Pedersen commitments.

4.4 Zero-knowledge proofs of knowledge

Zero-knowledge proofs of knowledge (ZKPoK) enable proving the truth of statements about secret information while preserving the confidentiality of the secret information [45]. Zero-knowledge arguments are ZKPs that hold only for computationally bounded provers. In this work, only computationally bounded entities are considered, and the terms 'proof' and 'argument' are used interchangeably for simplicity.

Let Setup be a probabilistic polynomial time setup algorithm that on input 1^λ generates a common reference string (CRS) σ available to all parties. For a polynomial time decidable ternary relation $\mathcal{R}$, the CRS-dependent language

$$\mathcal{L}_\sigma = \{u \mid \exists w : (\sigma, u, w) \in \mathcal{R}\}$$

is the set of statements u that have a witness w in the relation $\mathcal{R}$.

Let a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ be two probabilistic polynomial time stateful interactive algorithms. A Σ-protocol for $\mathcal{R}$ is an interactive three-move proof system which allows the prover to convince a verifier in zero-knowledge that $\mathcal{R}$ holds for u. Assuming that Setup has already been ran, the interaction between the prover and verifier is the following:

1. Given $(\sigma, u, w) \in \mathcal{R}$, the prover generates an initial message a and sends it to the verifier. This message is sometimes called the commitment.
2. The verifier samples a challenge $c \xleftarrow{\$} \{0,1\}^\lambda$ and returns it to the prover.
3. The prover computes a response z to the challenge c and sends it to the verifier.

The tuple of exchanged messages (a, c, z) is called a *transcript*. The verifier then runs $\mathcal{V}(\sigma, u, (a, c, z))$ which returns 1 if it accepts the proof and 0 otherwise. A transcript is called *accepting* if $\mathcal{V}(\sigma, u, (a, c, z)) = 1$.

$(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ is a Σ-protocol for $\mathcal{R}$ if it is a special honest-verifier zero-knowledge argument of knowledge as defined below. The following definitions are adapted from [46] which are themselves based on [47], and from [48].

Definition 9 (Argument of knowledge)**.** *The triple* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *is called an* argument of knowledge *for relation* $\mathcal{R}$ *if it is perfectly complete and computationally special sound.*

Definition 10 (Perfect completeness)**.** $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *has* perfect completeness *if*

$$\Pr\left[\begin{array}{ll} \sigma \leftarrow \mathsf{Setup}(1^\lambda), a \leftarrow \mathcal{P}(\sigma, u, w), & : \quad (\sigma, u, w) \notin \mathcal{R} \vee \\ c \xleftarrow{\$} \{0,1\}^\lambda, z \leftarrow \mathcal{P}(c) & \quad \mathcal{V}(\sigma, u, (a, c, z)) = 1 \end{array}\right] = 1.$$

Definition 11 (Computational special soundness)**.** $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *is computationally* 2-special sound *if there exists an efficient extraction algorithm* Ext *that can compute the witness given two accepting transcripts with the same initial message. Formally, for any PPT adversary* $\mathcal{A}$*, there must exist a negligible function* μ *such that*

$$\Pr\left[\begin{array}{ll} \sigma \leftarrow \mathsf{Setup}(1^\lambda), (u, a, c_1, z_1, c_2, z_2) \leftarrow \mathcal{A}(\sigma), & : (\sigma, u, w) \in \mathcal{R} \\ w \leftarrow \mathsf{Ext}(\sigma, u, a, c_1, z_1, c_2, z_2) & \end{array}\right] \geq 1 - \mu(\lambda),$$

where $\mathcal{A}$ *outputs distinct* $c_1, c_2 \in \{0,1\}^\lambda$ *and for all* $i \in \{1, 2\}$ *the transcript is accepting, i.e.* $\mathcal{V}(\sigma, u, (a, c_i, z_i)) = 1$*.* $\kappa = \mu(\lambda)$ *is the* knowledge error *of the protocol.*

Theorem 1. *Let* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *be a* Σ*-protocol for relation* $\mathcal{R}$ *with challenge length* $|c|$*. Then* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *has knowledge error* $2^{-|c|}$*.*

This is a well known result in cryptography and so the proof is omitted here. A detailed proof is presented in [49] and in [50]. Informally, the knowledge error designates the maximal probability of producing an accepting proof for a statement u without knowing the corresponding witness w. The intuition behind a proof of knowledge is then that a (potentially malicious) prover $\mathcal{P}^*$ that can convince a verifier with probability $\varepsilon > \kappa$ must indeed known a valid witness. This is known as knowledge-soundness and is precisely what is implied by special soundness.

Definition 12 (Public-coin)**.** *An argument* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *is called* public-coin *if all messages sent by the verifier are chosen uniformly at random and independently of the messages*

sent by the prover, i.e. the challenges correspond to the verifier's randomness.

Definition 13 (Special honest-verifier zero-knowledge (SHVZK))**.** *A public-coin argument* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *is called a* special honest verifier zero-knowledge *argument for* $\mathcal{R}$ *if there exists a probabilistic polynomial time simulator* $\mathcal{S}$ *such that for any interactive non-uniform PPT adversary* $\mathcal{A}$*, there exists a negligible function* μ *such that*

$$\left| \Pr\left[\begin{array}{l} \sigma \leftarrow \mathsf{Setup}(1^\lambda), (u, w, c) \leftarrow \mathcal{A}(\sigma), \\ a \leftarrow \mathcal{P}(\sigma, u, w), z \leftarrow \mathcal{P}(c) : \mathcal{A}(a, z) = 1 \end{array} \right] - \Pr\left[\begin{array}{l} \sigma \leftarrow \mathsf{Setup}(1^\lambda), (u, w, c) \leftarrow \mathcal{A}(\sigma), \\ (a, z) \leftarrow \mathcal{S}(\sigma, u, c) : \mathcal{A}(a, z) = 1 \end{array} \right] \right| \leq \mu(\lambda)$$

where $\mathcal{A}$ *outputs* (u, w, c) *such that* $(\sigma, u, w) \in \mathcal{R}$ *and* $c \in \{0,1\}^\lambda$*.*

4.4.1 Non-interactive zero-knowledge

Interactive zero-knowledge proofs are generally non-transferable [51, §2.6.2], which means that only the verifier involved in the protocol is convinced by the proof [51, §1.6.6]. Intuitively, an observer overseeing the interaction between the prover and verifier cannot know that the prover and verifier have not colluded beforehand to create an unsound proof. For example, the observer cannot be convinced that the verifier truly generated the challenge randomly, and that the challenge was not known to the prover beforehand. Conversely, a transferable zero-knowledge proof is publicly verifiable [51, §1.6.6], i.e. any potential verifier, and not just a verifier involved in the protocol, must be able to verify it. In turn, it is possible to forego the interaction with a verifier completely [51, §2.6.3]. For example, the ZKPs of correct decryption used in IVXV are non-interactive and transferable: the key application generates them on its own [22], and they are verifiable by anyone [5].

The Fiat-Shamir [52] transform can be used to turn an interactive public-coin zero-knowledge protocol into a non-interactive protocol [53]. However, the resulting protocol is only provably secure in the random oracle model (ROM). In the ROM, all parties have access to a perfectly random function $\mathsf{RO} : \{0,1\}^* \to \{0,1\}^{2\lambda}$. The output space must be of size $2^{2\lambda}$ to maintain a security level of λ against collision attacks. In practice, the ROM is instantiated with a strong cryptographic hash function (e.g. from the SHA-3 family), and the security guarantees are heuristic rather than provable. The proofs of vote correctness described in (Chapter 6) are made non-interactive via the Fiat-Shamir transform.

4.5 Lifted ElGamal in IVXV

For compatibility with many of the ZKP schemes presented in the next chapter, the classic ElGamal cryptosystem used in IVXV must be replaced with its lifted counterpart. While this is a change to the existing cryptosystem of IVXV, the security guarantees do not change and implementation should not be problematic. Indeed, the only required changes are the encryption of g^m instead of m, and the use of a precomputed table for the decryption to map g^m back to m.

Additionally, for the proofs of vote correctness to work (Chapter 6), the ballot must directly represent the candidate number as an integer. This is a change to the current ballot format presented in Section 2.4. While this change makes the plaintext ballot illegible in a text editor, sacrificing human-readable structure for mathematical structure is necessary for the scheme to work. Since the voting client handles the creation and encryption of the ballot itself, the voter is never exposed to the plaintext ballot directly, and so the change does not impact voters. Moreover, simply using the candidate number also aligns with the paper ballots used in Estonia, where voters must only write the candidate number on the ballot [20]. Still, the legislation governing the format of the ballots will have to be changed as a result.

5. Selecting a proof system

The desired goal is to prove that an encrypted ballot contains a valid candidate identifier, without leaking which candidate the ballot is for. To simplify the problem, the encrypted ballot can be viewed as a binding and hiding commitment to a candidate identifier. This enables the use of 'commit-and-prove' techniques [54], where a statement is proven in zero-knowledge relative to a committed value. The stated problem can thus be reduced to the general task of proving set membership in zero-knowledge, i.e. that a secret value belongs to a set.

This chapter contains a brief literature review on commit-and-prove techniques for ZK proofs of set membership (ZKSM) and ZK range proofs (ZKRP). Since the aim of this chapter is to illustrate the discovery and thought process behind selecting a practical approach for IVXV, this review is not meant to be comprehensive. Moreover, the differences between the approaches are not necessarily clear-cut, and thus the section segmentation provides only a loose outline. To limit the scope of this work, options were discarded as soon as they compared unfavourably to another (seemingly) viable approach. This chapter should therefore be viewed as the process of elimination by which the final proof system was chosen. While the reasons for discarding schemes are explained, not all such choices were necessarily objective. The field is vast and it is clear that alternative approaches should be explored in the future.

5.1 Selection criteria

For minimal changes to IVXV, the chosen scheme should be compatible with finite field arithmetic, the lifted ElGamal cryptosystem, and rely only on the discrete logarithm problem. Relying on the DLOG avoids introducing new security assumptions into IVXV. It follows that schemes based on Pedersen/ElGamal commitments are preferable over other approaches.

Since the mathematical complexity of the approach impacts the transparency and auditability of IVXV for the general public, it is also a contributing factor in selecting an appropriate scheme. Part of the subjectivity in the choice thus stems from the perceived complexity of the available approaches. In other words, approaches requiring the least amount of specialised mathematical background are preferable, as long as the resulting scheme remains practical. For example, if a scheme based on elliptic curves has similar

performance as a scheme based on bilinear pairings over elliptic curves, the former is preferable due to having one less 'layer'. For the same reason, approaches making use of general purpose ZK proof systems for arithmetic circuits (e.g. [38, 55, 56]) were not considered in this work.

What exactly is practical is also loosely defined. Since i-voters are already used to fast vote casting times, proving time should not impact it by much, e.g. by < 1s on the majority of devices. Moreover, the computational burden should lie on the prover, and verification should be fast enough for the vote collector to withstand projected peak voting loads. Due to the lack of a specific target for verification times, the most performant solutions were preferred, provided that the perceived complexity did not appear prohibitive. Performance was only preferred over complexity when the time-difference was more than one order of magnitude, i.e. at least tenfold.

5.2 Set membership proofs

It is unclear when exactly zero-knowledge proofs of set membership first appeared in literature, and there seems to be no dedicated survey in literature summarising and comparing different ZKSM approaches. However, set membership proofs are partly linked with group signatures [57], where a signature proves its issuer's membership in a group without publicly revealing their identity. Chen and Pedersen [58] presented a group signature scheme based on a protocol for proving knowledge of one out of many witnesses without revealing the specific witness. It can thus be argued that their approach provided the first tool for proving set membership in a loose sense of zero-knowledge. Although it does not cover the advancements of the past decade, one of the most comprehensive summaries on (non-)membership proofs was given by Bayer and Groth in 2013 [40].

It is not uncommon for ZKSM constructions to appear as a building block or by-product, rather than as a primary result. For example, Groth and Kohlweiss [48] proposed a ZK proof system for Bitcoin privacy, which can additionally be used to prove that a value belongs to a set. The idea behind the scheme is to give a proof that a certain Pedersen commitment is a commitment to 0. If this is indeed the case for a value in a list, then a value from this list was necessarily committed. An earlier approach by Bayer and Groth [40] also uses Pedersen commitments to yield proofs of membership or non-membership of a secret in a list. However, contrary to the 0-commitment approach, this approach is based on polynomial evaluation and requires two commitments per secret. The main improvement of the scheme over earlier membership proof schemes was the absence of a trusted third party [40].

Regarding explicit ZKSM research, Camenisch, Chaabouni and Shelat [59] presented an approach where a prover proves that they know a signature on one of the elements of a set. The authors also derived a ZKRP from the signature approach, and additionally proposed an alternative ZKSM based on cryptographic accumulators. Approaches based on cryptographic accumulators seem to currently be the most popular constructions for efficient ZKSMs [60].

5.2.1 Accumulator-based set membership

Cryptographic accumulators were first introduced by Benaloh and de Mare [61] and enable compressing a set of elements into a short digest: the *accumulator*. While the original approach was based on iterative exponentiation in an RSA group, accumulators have since been extended to other constructions and are a core tool for many ZKSMs [60, 62].

Commit-and-prove zero-knowledge proofs of set (non-)membership were formalised by Benarroch et al. in 2019 [60]. In the original preprint, the authors only considered approaches based on cryptographic accumulators to achieve succinct ZKSMs, and split them into three categories: Merkle tree and lattice-based, RSA-based, and pairing-based. However, a revised, 'full' version of their work was published in 2023 [63], which contains a section on subsequent developments in the field. The authors also presented efficient ZKSM constructions based on RSA accumulators ($\mathsf{MemCP}_{\mathsf{RSA}}$, $\mathsf{MemCP}_{\mathsf{RSAPrm}}$) and bilinear pairings ($\mathsf{MemCP}_{\mathsf{VC}}$). The difference between $\mathsf{MemCP}_{\mathsf{RSA}}$ and $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ is that the former works with sets of arbitrary elements, while $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ requires a set of primes of equal bit-length. As of May 2024, their RSA-based constructions seem to remain the state of the art regarding set membership proofs based on RSA accumulators.

More recently, Campanelli, Hall-Andersen and Kamp presented a novel accumulator-based ZKSM construction called 'Curve Trees' which is both efficient, and has transparent setup [62]. The approach is based on shallow Merkle trees and 2-cycles of elliptic curves and requires an existing and efficient commit-and-prove system such as Bulletproofs [46] to work. The brief overview on related work in their article complements, but does not encompass the overview of ZKSM approaches by Benarroch et al.

The main argument of Campanelli et al. against the schemes presented in [60] is that they do not offer transparent setup. However, in the 2023 version, Benarroch et al. showed that their RSA accumulator scheme can also be instantiated over hidden order groups—e.g. class groups—for transparent setup [63, Appendix 4]. As a drawback, Benarroch et al. estimated that verification would take approximately 2.3s in a class group with a 2048-bit discriminant. This is a significant markup compared to the 20–30ms verification times for

their $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ construction with a 2048-bit modulus [63]. Furthermore, cryptanalytic results show that a 2048-bit discriminant does not offer 128-bit security level in a class group [64], further worsening performance for a 128-bit security level. Trusted setup is however not a concern in IVXV, where election keys are generated on an air-gapped machine during a public ceremony. That is, the RSA group could very well be generated on the same machine using the pre-audited tool.

5.2.2 Selecting an accumulator

While Benarroch et al. did not include benchmarks for their pairing-based approach, they did provide benchmarks for the RSA-approach instantiated with Bulletproofs. Campanelli, Hall-Andersen and Kamp also provided benchmarks for their scheme instantiated with Bulletproofs. Unfortunately, the benchmarks are not directly comparable since the set sizes are not comparable, and the difference in approaches makes extrapolation difficult.

Notably, both $\mathsf{MemCP}_{\mathsf{RSAPrm}}$ and $\mathsf{MemCP}_{\mathsf{VC}}$ require the use of a commit-and-prove range proof. For $\mathsf{MemCP}_{\mathsf{RSA}}$, the range proof is replaced by a proof that the committed element hashes to the correct prime. This stems from the 'hash-to-prime' mapping used to map the arbitrary set elements to a set of primes in a collision resistant manner. While the authors did not provide a detailed benchmark for $\mathsf{MemCP}_{\mathsf{RSA}}$ approach, they indicated that the approach is potentially slower than the approach for sets of primes [63].

The conclusion is thus that both state of the art approaches require an underlying commit-and-prove system for either range proofs, or proving arbitrary statements using constraint systems. As such, the use of a ZKRP instead of proving ZKSM could potentially reduce verification times and proof complexity even further, at the cost of losing the flexibility of arbitrary sets. A potential solution to restore this flexibility is to map set elements to consecutive integers, similarly to how arbitrary set elements are mapped to primes for $\mathsf{MemCP}_{\mathsf{RSA}}$.

While tree-based approaches can yield constant-size proofs with fast verification times, they generally require the use of general-purpose zk-SNARKS [63]. As such, they do not satisfy the requirements set forth in Section 5.1. Pairing-based ZKSMs were also not pursued further due to the additional hardness assumptions of pairings and since the ZKRP approach seemed promising.

5.3 Range proofs

A range proof is a specific type of set membership proof, where the set is an integer range (an interval). Compared to proofs of set membership, the literature on ZKRPs seems to be more comprehensive and interlinked. Christ et al. [65] summarised the state of the art regarding ZKRPs in a fresh survey (2024), and claimed that it provides a complete description of existing ZKRP techniques. This claim is not entirely accurate, as the authors did not mention approaches based on lookup arguments [66]. The authors also did not consider signature-based approaches [59, 67] as a distinct category, but rather as a subset of binary decomposition. Notably, all covered techniques are based on commit-and-prove approaches. Additionally, some recent ZKRP protocols [67, 68] which combine the covered techniques in interesting ways were not included. While the survey was published after the literature review for the current thesis was already established, it backs the choices made in this work.

Although Christ et al. improved upon the earlier work by Morais et al. [69], the latter provides algorithmic descriptions of its covered techniques. It additionally provides a good technical summary on Bulletproof range proofs. There is also the survey by Deng et al. [70] which includes a comparative analysis of different ZKRP approaches, although it does not contain any benchmarks. It also provides a more complete history of range proofs than the survey by Christ et al., while naturally being less current.

It is not uncommon for range proofs to be part of a more complex proof system, such as of ZKSMs as seen in Section 5.2. In fact, Christ et al. [65] attribute the introduction of range proofs to Brickell et al. [71] who merely used it as a building block.

5.3.1 Desirable properties

Principally, the commitment scheme must be compatible with Pedersen/ElGamal commitments in finite fields or an efficient ‘binding’ must exist between the proof system and the cryptosystem. Otherwise, the proof system is not compatible or practical to use with IVXV.

While the transparent setup of a proof system can be viewed as an additional perk, a trusted setup does not pose a problem in IVXV as argued in Section 5.2.1. However, the only trusted data that a ZKRP may rely on is the common reference string, since in IVXV, the election authority does not issue any personalised data to voters. Indeed, the security and trust guarantees of IVXV rely heavily on the Estonian public key infrastructure (PKI),

which is outside the control of election organisers. As such, certain ZKRP approaches such as those based on trusted credentials are not usable with IVXV.

Aggregation of proofs and batch verification are further perks of a ZKRP. Since many ZKRP constructions only allow to prove that a committed value is non-negative, proofs for specific ranges are obtained by combining proofs in a homomorphic manner [65]. As such, the amortised generation and verification of potentially many proofs is beneficial, at least for proofs that do not hold over arbitrary intervals.

5.3.2 Constructions

Christ et al. identified three ZKRP techniques [65]: square decomposition, n-ary decomposition, and hash-chain approach. The hash-chain approach is the only one that is essentially incompatible with IVXV. In this approach, a hash function is applied x times to a nonce r, with x being the commitment. A verifier then checks whether the hash function was applied at least t times, which proves that $x \geq t$, i.e. that x exceeds a threshold t. HashWires is a *credential-based* range proof [72], and is the current state of the art of hash-based ZKRPs. However, this means that HashWires guarantees soundness only if the commitment is well-formed [65, 72], and is therefore not usable with IVXV[1].

Constructions based on square decomposition rely on integer commitments where the binding property must hold over the integers, and not just in the commitment group. The approach remained largely stale from 2005 onwards, save an improvement by Couteau et al. in 2017. However, a newer line of work [74, 75] pioneered by Couteau has improved the approach and led to the development of 'Sharp': short relaxed range proofs [75]. Sharp is competitive with state of the art binary decomposition techniques [65, 75], and is algebraically compatible with IVXV using cross-group binding.

Finally, approaches based on n-ary decomposition are arguably the most widespread and researched approaches to range proofs. The seminal Bulletproofs by Bünz et al. [46] are also based on binary decomposition and referenced in a wide body of literature[2]. Indeed, Bulletproofs have not only received attention regarding various performance improvements [76, 77], but are also commonly incorporated as a subcomponent in more complex schemes as seen in Section 5.2.1. While Bulletproofs are based on Pedersen vector commitments, n-ary decomposition can also be based on general polynomial commitments [78]. Although the scheme in [78] could be instantiated with a Pedersen-compatible polynomial commitment scheme [65] for compatibility with IVXV, this would introduce additional

[1]There is no trusted party that can generate these commitments.
[2]IEEE Xplore claims 1200+ citations since July 2018 as of May 2024.

complexity. Bulletproofs are therefore preferable here due to their more streamlined nature and natural compatibility with Pedersen commitments.

Many lattice-based approaches also fit under the n-ary decomposition paradigm [65]. However, due to their lesser efficiency and the fact that IVXV does not claim post-quantum security, the approaches are not considered here.

5.3.3 Sharp vs. Bulletproofs

By discarding hash-based range proofs and general polynomial commitments, the state of the art narrows down to Sharp [75] and Bulletproofs [46] with its potential improvements. While on paper, variants of Sharp appear more efficient than Bulletproofs [65, 75], there are important nuances to consider because of differing security guarantees. By default, Sharp only provides 'relaxed' soundness, where the prover is only bound to a rational in the target range, instead of an integer [75]. Bulletproofs do not suffer form this limitation. However, Bulletproofs can only directly be used to prove that a number belongs to $[0, 2^n - 1]$ for some integer n while Sharp works for arbitrary ranges. Since Bulletproofs are homomorphic, it is possible to overcome this limitation in practice (Section 6.2).

For Sharp, in the interactive setting, satisfying full soundness with a knowledge error of 2^{-128} would require 128 repetitions of the protocol [75, Table 4]. In the non-interactive setting, Sharp can achieve full soundness with an additional commitment in a hidden order group (class group or RSA group). As such, the performance advantages of Sharp over Bulletproofs are likely lost when requiring full soundness. This remains a speculation, as Couteau et al. did not provide performance benchmarks for their hidden order group versions.

Additionally, Sharp is a very new protocol, and it does not appear to have been implemented outside of research. In fact, while Couteau et al. have benchmarked their implementation of Sharp, they have not made any code publicly available. The existence of a reference and third party implementations[3][4] of Bulletproofs as well as its practical use in protocols[5] are strong arguments for Bulletproofs over Sharp.

[3] `https://crypto.stanford.edu/bulletproofs/`
[4] `https://github.com/dalek-cryptography/bulletproofs`
[5] `https://tlu.tarilabs.com/protocols/mimblewimble-mb-bp-utxo`

5.4 Bulletproofs

Bulletproofs are a zero-knowledge proof protocol without a trusted setup which can be used for range proofs, but also for proofs for arithmetic circuits. Bulletproofs rely only on the discrete logarithm assumption, and are made non-interactive using the Fiat-Shamir heuristic [46]. In this work, only Bulletproof range proofs in the non-interactive setting are considered. For simplicity, from hereinafter, the term 'Bulletproofs' only refers to the range proof variant unless specified otherwise.

Bulletproofs do not directly prove that a value lies in an arbitrary integer range. Rather, given a Pedersen commitment X to x with randomness r, they prove the following relation [46, eq. 36]:

$$\big\{(g, h \in \mathbb{G}, X, n; x, r \in \mathbb{Z}_q) : X = g^x h^r \wedge x \in [0, 2^n - 1]\big\},$$

where $\mathbb{G}$ is a q-element DLOG-group. However, the additive homomorphism of Pedersen commitments allows to combine two Bulletproofs to prove that $x \in [a, b]$ for arbitrary $a, b \in \mathbb{N}$ as required (Section 6.2).

Let $\mathbf{a}_L = (a_1, \ldots, a_n) \in \{0,1\}^n$ be the vector containing the bits of x, so that $\langle \mathbf{a}_L, \mathbf{2}^n \rangle = x$, with $\mathbf{2}^n = (2^0, \ldots, 2^{n-1})$. By proving knowledge of a vector $\mathbf{a}_R$ such that:

1. $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{0}^n$ (component-wise multiplication)
2. $\mathbf{a}_L + \mathbf{a}_R = \mathbf{1}^n$ (component-wise addition)
3. $\langle \mathbf{a}_L, \mathbf{2}^n \rangle = x$ (inner product)

a prover shows that $a_1, \ldots, a_n$ are all in $\{0,1\}$ (1 & 2) and that $\mathbf{a}_L$ contains the binary decomposition of x (3). In turn, $x \in [0, 2^n - 1]$ if all three equations hold. Bootle et al. [47] presented an efficient *inner product argument* which allows to show in zero-knowledge that the inner product of two committed vectors (using Pedersen vector commitments) is some publicly known value. Bünz et al. improved the efficiency of the the argument by showing that the three above equations could be combined into a single inner product argument. Using this fact, they obtained a range proof with a size logarithmic in n [46].

While the original Bulletproofs are already efficient in practice, various optimisations [79, 80] have been proposed since. Bulletproofs+ by Chung et al. [79] are 84% of the size of Bulletproofs for a 32-bit range, however, their performance remains comparable to that of Bulletproofs. Bulletproofs++ by Eagen et al. [80] do not only reduce the size further, but reduce the $O(n)$ scalar multiplications asymptotically needed by Bulletproofs(+) to

$O(n/\log(n))$ multiplications, therefore also improving performance. Eagen et al. claim that their proof verification times are about 3 times faster than those of Bulletproofs [80, p. 4].

However, unlike Bulletproofs+ which still make use of a (weighted) inner product argument, Bulletproofs++ use new techniques altogether. As such, the Bulletproofs++ range proof is in reality encoded into and proven with an arithmetic circuit, rather than being a directly constructed range proof. It is the author's opinion that they are therefore significantly harder to understand than Bulletproofs(+). In fact, Bulletproofs++ are related to the zero-knowledge lookup arguments introduced by Bootle et al. in 2018 [66] which can be used to construct efficient ZKSMs and ZKRPs [81, 82]. While efficient, they are also complex and were therefore not considered here.

6. Proving vote correctness

6.1 Setting

Let f be a safe prime with $p = (f-1)/2$ prime, and let $\mathbb{G}_p$ be the subgroup of quadratic residues in $\mathbb{Z}_f^*$. Let $\mathbb{G}_q$ be a DLOG-group such that $q \ll p$, e.g. an elliptic curve group. Let b_q denote the bit-length of q.

Let x represent a vote and let b_x be the maximal bit-length of a vote such that $b < 2^{b_x} \ll q$. Let a, b be two integers with $0 \leq a < b < 2^{b_x}$. Thus, for any $x \in [a, b]$, it holds that x is the same in both $\mathbb{Z}_p$ and $\mathbb{Z}_q$, and that $x < 2^{b_x}$. Let a (resp. b) represent the lowest (resp. highest) candidate number among candidates who voters can vote for. Without loss of generality, let the candidate numbers be consecutive between a and b. On the off-chance that this is not the case, the available candidate numbers can be mapped onto a consecutive integer range.

Let $\mathsf{pk} \in \mathbb{G}_p$ be the public key of the lifted ElGamal cryptosystem, and let g_p be a generator of $\mathbb{G}_p$. The encryption of $x \in [a, b]$ with randomness $r_p \xleftarrow{\$} \mathbb{Z}_q$ is then

$$\mathsf{Enc}_{\mathsf{pk}}(x; r_p) = \left(g_p^{r_p}, g_p^x \cdot \mathsf{pk}^{r_p}\right) = (y, X_p).$$

More generally, in this chapter, capital letters denote Pedersen commitments, and indices denote which group or field an element is in. Standalone, X_p can be viewed as a Pedersen commitment of x with randomness r_p in $\mathbb{G}_p$.

Let b_c represent the bit-length of a verifier's challenge in a Σ-protocol. For some security parameter λ, it must hold that $b_c \geq \lambda$ for security with no repetitions of the protocol. In practice, $\lambda \geq 128$. Let $b_\perp$ be a parameter controlling the probability of aborts, i.e. the probability that the Σ-protocol must be restarted to avoid leaking information about the secret. Finally, the parameters $b_x, b_c, b_\perp, b_q$ must satisfy the relation $b_x + b_c + b_\perp < b_q$ which is necessary to avoid modular reductions in $\mathbb{G}_p$ and $\mathbb{G}_q$ for the protocol computations. A discussion on how to pick values for these parameters is given in Section 6.4.

6.2 Range proof for concrete ranges

A Bulletproof range proof can be used to prove that $x \in [0, 2^{b_x} - 1]$ for a commitment C of x. However,

$$a \leq x \leq b < 2^{b_x} \iff 0 \leq x - a < 2^{b_x} \wedge 0 \leq b - x < 2^{b_x}$$

and so, by the conjunction of two Bulletproofs, a range proof for a concrete range can be obtained.

Let g_q, h_q be two generators of $\mathbb{G}_q$ such that $\log_{g_q}(h_q)$ is not known. Let C_q, C'_q be two Pedersen commitments such that

$$\begin{aligned} r_q &\xleftarrow{\$} \mathbb{G}_q, \quad C_q \leftarrow g_q^{x-a} \cdot h_q^{r_q} \\ r'_q &\xleftarrow{\$} \mathbb{G}_q, \quad C'_q \leftarrow g_q^{b-x} \cdot h_q^{-r'_q}. \end{aligned}$$

Let π_{rp_a} be the Bulletproof for $x - a \in [0, 2^{b_x} - 1]$ and the commitment C_q, and let π_{rp_b} be the Bulletproof for $b - x \in [0, 2^{b_x} - 1]$ and the commitment C'_q. To prove that $x \in [a, b]$, it remains to show that x is the same for both $X_q \leftarrow g_q^a \cdot C_q$ and $X'_q \leftarrow g_q^b \cdot (C'_q)^{-1}$, since

$$\begin{aligned} g_q^a \cdot C_q &= g_q^a \cdot g_q^{x-a} \cdot h_q^{r_q} = g_q^x \cdot h_q^{r_q} \\ g_q^b \cdot \left(C'_q\right)^{-1} &= g_q^b \cdot g_q^{x-b} \cdot h_q^{r'_q} = g_q^x \cdot h_q^{r'_q}. \end{aligned}$$

Given C_q, C'_q, anyone can compute X_q, X'_q since g_q, a, b are publicly known values.

A ZKP of discrete logarithm equality is needed to prove that X_q and X'_q are commitments for the same x. Otherwise, there are no guarantees that a dishonest prover did not use a different x value when computing C_q and C'_q since the commitments are perfectly hiding. While Bulletproofs do not offer a way to prove such a conjunction, they do support proof batching for efficiency [46, §4.3]. This means that π_{rp_a} and π_{rp_b} can be proven and verified together with better efficiency than when doing so separately. The discrete logarithm equality is proven in the protocol proposed in Section 6.3.

Since Bulletproofs are a public-coin protocol, they can be made non-interactive using the Fiat-Shamir heuristic [46, §4.4]. Not only are non-interactive Bulletproofs transferable, its authors also claim certain computational speed-ups over the interactive setting. Furthermore, non-interactiveness helps minimise communication complexity between the voting server and voting clients in the IVXV setting, and is easily combined with the non-interactive protocol proposed in Section 6.3.

For practical implementations, the range proofs should therefore be implemented with non-interactive Bulletproofs with aggregation. $\mathbb{G}_q$ should be chosen such that computing and verifying Bulletproofs would be reasonably fast, while also being compatible with other requirements. A more detailed discussion on the parameter choice is provided in Section 6.4.

6.3 Discrete logarithm equality across groups

Let $(y, X_p), (C_q, \pi_{rp_a}), (C'_q, \pi_{rp_b})$ be given to the verifier. By verifying π_{rp_a}, π_{rp_b} and computing X_q, X'_q, the verifier gains assurance that

1. X_q is a commitment for x_q such that $x_q \geq a$,
2. X'_q is a commitment for x'_q such that $x'_q \leq b$.

Furthermore, X_p is a commitment for the vote x_p. Then, to prove that (y, X_p) is an encryption of $x \in [a, b]$, the verifier must additionally be able to verify in zero-knowledge that

$$x_p = x_q = x'_q$$

for the committed values. More formally, to prove that (y, X_p) is an encryption of $x \in [a, b]$, a zero-knowledge proof must be given for the relation

$$\mathcal{R}_{\mathsf{Veq}} = \left\{ \big((y, X_p, X_q, X'_q), (x, r_p, r_q, r'_q)\big) \;\middle|\; \begin{array}{l} y = g_p^{r_p} \wedge X_p = g_p^x \cdot \mathsf{pk}^{r_p} \\ \wedge X_q = g_q^x \cdot h_q^{r_q} \wedge X'_q = g_q^x \cdot h_q^{r'_q} \end{array} \right\}.$$

A ZKP for $\mathcal{R}_{\mathsf{Veq}}$ therefore proves that a vote for an eligible candidate was encrypted without leaking who the vote is for.

When $p = q$, the proof is a variant of proving discrete logarithm equality in a group, for which an efficient approach has been given by Chaum and Pedersen [83]. However, the difficulty lies in proving this efficiently when $p \neq q$, which is the case here, i.e. proving discrete logarithm equality across different groups. For the latter problem, an approach using Pedersen commitments was presented by Chase et al. [84]. More formally, the technique by Chase et al. gives a ZKP for the relation

$$\mathcal{R}_{\mathsf{DLeq}} = \left\{ \big((X_p, X_q), (x, r_p, r_q)\big) \mid X_p = g_p^x \cdot h_p^{r_p} \wedge X_q = g_q^x \cdot h_q^{r_q} \right\}$$

however, a range proof for $x \in \{0, 1^{b_x} - 1\}$ is needed as part of the technique. This range proof must be knowledge-sound, and the authors themselves propose to use Bulletproofs for this [84, p. 5]. The protocol is therefore ideal for the current use case, since such a

range proof is required regardless, i.e, π_{rp_a}, π_{rp_b}. Moreover, since the protocol in [84] is a public-coin protocol, it can also be made non-interactive with the Fiat-Shamir transform.

For efficiency, instead of first proving $x_q = x'_q$ for commitments in $\mathbb{G}_q$ and then proving $\mathcal{R}_{\mathsf{DLeq}}$, the technique by Chase et al. can be extended to prove the entirety of $\mathcal{R}_{\mathsf{Veq}}$. The resulting Σ-protocol Π_{Veq} for proving the relation $\mathcal{R}_{\mathsf{Veq}}$ is given on Figure 1.

Prover Input: x, r_p, r_q, r'_q		**Verifier** Input: $y, X_p, X_q, X'_q, \pi_{rp_a}, \pi_{rp_b}$
$k \xleftarrow{\$} [0, 2^{b_x+b_c+b_\perp} - 1]$ $t_p \xleftarrow{\$} \mathbb{Z}_p$ $(t_q, t'_q) \xleftarrow{\$} \mathbb{Z}_q^2$ $w \leftarrow g_p^{t_p}$ $K_p \leftarrow g_p^k \cdot \mathsf{pk}^{t_p}$ $K_q \leftarrow g_q^k \cdot h^{t_q}$ $K'_q \leftarrow g_q^k \cdot h^{t'_q}$		
	$\xrightarrow{w, K_p, K_q, K'_q}$	
		$c \xleftarrow{\$} [0, 2^{b_c} - 1]$
	$\xleftarrow{c}$	
Check $c \in [0, \ldots, 2^{b_c} - 1]$ $z \leftarrow k + c \cdot x \pmod{\mathbb{Z}}$ Abort if $z \notin [2^{b_x+b_c}, 2^{b_x+b_c+b_\perp} - 1]$ $s_p \leftarrow t_p + c \cdot r_p \pmod{p}$ $s_q \leftarrow t_q + c \cdot r_q \pmod{q}$ $s'_q \leftarrow t_q + c \cdot r'_q \pmod{q}$		
	$\xrightarrow{z, s_p, s_q, s'_q}$	
		Accept iff $z \in [2^{b_x+b_c}, 2^{b_x+b_c+b_\perp} - 1]$, $s_p \in \mathbb{Z}_p$, $(s_q, s'_q) \in \mathbb{Z}_q^2$, $g_p^{s_p} = w \cdot y^c$, (i) $g_p^z \cdot \mathsf{pk}^{s_p} = K_p \cdot X_p^c$, (ii) $g_q^z \cdot h^{s_q} = K_q \cdot X_q^c$, (iii) $g_q^z \cdot h^{s'_q} = K'_q \cdot X_q'^c$ (iv)

Figure 1. Π_{Veq}: a Σ-protocol for proving $\mathcal{R}_{\mathsf{Veq}}$.

6.3.1 Masking and aborts

The mask k is necessary to hide information with random noise, otherwise the verifier could trivially extract x from z with $x = z/c$. Since operations are performed over the integers, the traditional approach of picking the mask uniformly at random from the underlying group is not feasible. As such, not all values of k mask cx 'sufficiently'. Indeed, not all values of z are equally likely to occur for $z < 2^{b_x+b_c}$ or $z \geq 2^{b_x+b_c+b_\perp}$, and so the response may leak information about x. For example, if $k = 0$ and $c = 2^{b_c} - 1$, then $cx < 2^{b_x+b_c}$ for all values of x, but for $k = b_\perp$, this is no longer the case. In such cases, the prover must abort the protocol before sending its response to the verifier to avoid revealing information about x. The prover and verifier must then restart the protocol and must sample a new k and a new c uniformly at random.

The abort condition is motivated by the following lemma [84, 85]:

Lemma 1 ([85, Lemma 1])**.** *In the non-aborting case, the value z in the transcript of an honest protocol execution is uniformly distributed in $\{2^{b_x+b_c}, \ldots, 2^{b_x+b_c+b_\perp} - 1\}$. An honest prover aborts with probability $2^{-b_\perp}$.*

The proof is identical to that of [84, Lemma 2] and is hence not reproduced here.

Notably, the verifier gets no information about k due to the hiding property of the Pedersen commitments transmitted with the first message. Thus, it is infeasible for the verifier to learn anything about cx regardless of whether the prover aborts the protocol.

6.3.2 Non-interactiveness

Interactive proofs have certain shortcomings compared to non-interactive proofs. For example, non-transferability restricts the auditability of i-voting, which is the very situation that this work aims to improve. Moreover, the three move communication might not be practical from an implementation viewpoint, especially when aborts are involved.

Since the verifier is a public-coin verifier, the protocol can be made non-interactive using the Fiat-Shamir transform [52, 53] with aborts [86]. In the following proofs, the random oracle is considered programmable as in [53]. By turning the proposed protocol non-interactive, the proof becomes transferable and some of the abort complexity is mitigated. In the non-interactive version, the prover repeats the proof locally when needed and only outputs a non-aborting transcript, which places the abort burden entirely on the prover.

Therefore, the no-abort zero-knowledge of the interactive protocol becomes regular zero-knowledge in the non-interactive setting [84]. Furthermore, part of the communication overhead is shelved.

However, non-interactive proofs are not without their issues [87, 88], especially in abort-based versions as shown by recent work [89, 90]. More specifically, to prevent a malicious prover being able to prove wrong things, the statement and protocol messages preceding the challenge generation should be included in the challenge seed [87]. For the given protocol, challenges must be obtained from the programmable random oracle $\mathcal{O}$ with $c \leftarrow \mathcal{O}(\mathsf{st}, \alpha)$, where

- st is the statement $\big((y, X_p), (g_p, \mathsf{pk}), (g_q, h_q), (a, b)\big)$,
- α is the first message of the protocol, i.e. (w, K_p, K_q, K'_q).

Additionally, making the proof non-interactive with Fiat-Shamir requires setting $b_c \geq 2\lambda$ for collision resistance due to the birthday attack [91, §7.3]. When instantiating the random oracle with a cryptographic hash function in practice, this security requirement may even be higher [92].

The proofs that follow are based on the proofs in [84] in the interactive setting, but are adapted to the proposed protocol in the non-interactive setting. This simplifies the proofs slightly since the abort cases no longer need to be considered [84, §4.2].

6.3.3 Completeness

Theorem 2. *The non-interactive protocol for the relation $\mathcal{R}_{\mathsf{Veq}}$ is perfectly complete.*

Proof. In the non-interactive setting, the prover never aborts. First, for an honest prover, the equation (i)

$$g_p^{s_p} = g_p^{t_p + c r_p} = g_p^{t_p} \cdot g_p^{c r_p} = w \cdot \left(g_p^{r_p}\right)^c = w \cdot y^c$$

is satisfied. Additionally, equation (ii)

$$g_p^z \cdot \mathsf{pk}^{s_p} = g_p^{k + cx} \cdot \mathsf{pk}^{t_p + c r_p} = g_p^k \cdot \mathsf{pk}^{t_p} \cdot \left(g_p^x \cdot \mathsf{pk}^{r_p}\right)^c = K_p \cdot X_p^c$$

is also satisfied. Equations (iii) and (iv) are proven similarly. □

6.3.4 Soundness

Theorem 3. *Let κ_{rp} be the knowledge error of π_{rp}. The non-interactive protocol for the relation $\mathcal{R}_{\mathsf{Veq}}$ is 2-special sound with knowledge error $\kappa = 2^{-b_c} + \kappa_{rp}$ under the DLOG assumption for $\mathbb{G}_p$ and $\mathbb{G}_q$ in the random oracle model.*

Proof. Let there be an extractor algorithm Ext which is given inputs $\pi_{rp_a}, (w, K_p) \in \mathbb{G}_p^2, (K_q, K'_q) \in \mathbb{G}_q^2$, and two accepting transcripts $((w, K_p, K_q, K'_q), c, (z, s_p, s_q, s'_q))$ and $((w, K_p, K_q, K'_q), \dot{c}, (\dot{z}, \dot{s_p}, \dot{s_q}, \dot{s'_q}))$ with $c \neq \dot{c}$. The extractor also has access to the public parameters, including the public key pk.

The extractor then recovers x_p, x_q, r_p, r_q, r'_q such that $y = g_p^{r_p}, X_p = g_p^{x_p} \cdot \mathsf{pk}^{r_p}$ and $X_q = g_q^{x_q} \cdot h_q^{r_q}, X'_q = g_q^{x'_q} \cdot h_q^{r'_q}$ with the following steps:

1. Bulletproof range proofs are arguments of knowledge [46] and are therefore extractable. By using the knowledge-extractor of π_{rp_a}, Ext can extract $((x_q - a)^*, r_q^*)$, except with some small failure probability κ_{rp}. Since a is publicly known, the extractor can further recover (x_q^*, r_q^*) such that $X_q = g_q^{x_q^*} \cdot h_q^{r_q^*}$ and $x_q^* < 2^{b_x}$.
2. From the two accepting transcripts with distinct challenges $c \neq \dot{c}$, the extractor selects the pairs
   - $\big((K_p, c, z, s_p), (K_p, \dot{c}, \dot{z}, \dot{s_p})\big)$,
   - $\big((K_q, K'_q, c, z, s_q, s'_q), (K_q, K'_q, \dot{c}, \dot{z}, \dot{s_q}, \dot{s'_q})\big)$.

   By defining $x_p = (z - \dot{z}) \cdot (c - \dot{c})^{-1}$ and $r_p = (s_p - \dot{s_p}) \cdot (c - \dot{c})^{-1}$, Ext extracts an opening of $X_p = g_p^{x_p} \cdot \mathsf{pk}^{r_p}$ since

   $$\frac{z - \dot{z}}{c - \dot{c}} = \frac{k + cx_p - (k + \dot{c}x_p)}{c - \dot{c}} = \frac{x_p(c - \dot{c})}{c - \dot{c}} = x_p.$$

   Ext similarly extracts openings for $X_q = g_q^{x_q} \cdot h_q^{r_q}$ and $X'_q = g_q^{x_q} \cdot h_q^{r'_q}$. From Theorem 1, this fails with probability 2^{-b_c}. By the binding property of Pedersen commitments X_q, K_p, K_q, K'_q and of ElGamal,
   (a) $y = g_p^{r_p}$
   (b) $(x_q, r_q) = (x_q^*, r_q^*)$
   (c) $(z - cx_p, s_p - cr_p) = (\dot{z} - \dot{c}x_p, \dot{s_p} - \dot{c}r_p)$
   (d) $(z - cx_q, s_q - cr_q, s'_q - cr'_q) = (\dot{z} - \dot{c}x_q, \dot{s_p} - \dot{c}r_q, \dot{s'_q} - \dot{c}r'_q)$

   must all hold under the DLOG assumption.
3. Finally, the extractor returns $(x_p, x_q, r_p, r_q, r'_q)$.

It remains to show that $x_p = x_q$ over the integers. From verification checks (ii) and (iii),

there must exist $k_p, k_q, u, \dot{u}, v, \dot{v} \in \mathbb{Z}$ such that

$$z = k_p + c \cdot x_p + u \cdot p \qquad z = k_q + c \cdot x_q + v \cdot q$$
$$\dot{z} = k_p + \dot{c} \cdot x_p + \dot{u} \cdot p \qquad \dot{z} = k_q + \dot{c} \cdot x_q + \dot{v} \cdot q \; .$$

Because the verifier must assert that $z \in \{2^{b_x+b_c}, \ldots, 2^{b_x+b_c+b_\perp} - 1\}$ for a transcript to be accepting, it follows that $(u, \dot{u}, v, \dot{v})$ are non-negative. By linear combination with respect to p and q,

$$(z - \dot{z}) = (c - \dot{c})x_p + (u - \dot{u})p \qquad (z - \dot{z}) = (c - \dot{c})x_q + (v - \dot{v})q \; .$$

W.l.o.g., let $z - \dot{z}$ be positive, since if it is negative, then $\dot{z} - z$ is positive instead. From the choice of parameters and verification checks $z - \dot{z} < 2^{b_x+b_c+b_\perp} < 2^{b_q}$. Moreover, π_{rp} ensures that $x_q < 2^{b_x}$, and so $|c - \dot{c}|x_q < 2^{b_x+b_c}$. It follows that $(z - \dot{z}) - |c - \dot{c}|x_q < 2^{b_q}$ and so $(v - \dot{v}) = 0$.

Equating the two representations of $z - \dot{z}$ with $(v - \dot{v}) = 0$ yields

$$(c - \dot{c})x_q = (c - \dot{c})x_p + (u - \dot{u})p$$
$$(c - \dot{c})(x_q - x_p) = (u - \dot{u})p \; .$$

Since p is prime, it must divide $(c - \dot{c})$ or $(x_q - x_p)$, but $p > 2^{b_q} > 2^{b_c}$ and so it cannot divide $|c - \dot{c}|$. Therefore $p|(x_q - x_p)$ and so $x_q \equiv x_p \pmod{p}$. Since $x_q < p$ and $x_p < p$, no modular reduction takes place and so $x_q = x_p$ in $\mathbb{Z}$ as well. □

6.3.5 Zero-knowledge

In a non-interactive protocol execution, an honest prover only outputs non-aborting transcripts. The proof can therefore follow standard reasoning with accepting transcripts and without factoring in aborts.

Theorem 4. *The non-interactive protocol for the relation $\mathcal{R}_{\mathsf{Veq}}$ has computational zero-knowledge in the random oracle model.*

Proof. The simulator takes as input the public parameters, including the group descriptions and the public key pk. It also has access to the random oracle and can program it with input-output pairs.

First, the simulator samples uniformly at random

$$c \xleftarrow{\$} \{0, \ldots, 2^{b_c} - 1\}, z \xleftarrow{\$} \{2^{b_x+b_c}, \ldots, 2^{b_x+b_c+b_\perp} - 1\}, s_p \xleftarrow{\$} \mathbb{Z}_p, (s_q, s'_q) \xleftarrow{\$} \mathbb{Z}_q^2.$$

Since K_p satisfies the equation

$$K_p = g_p^k \cdot \mathsf{pk}^{t_p} = g_p^k \cdot \left(g_p^{cx} \cdot g_p^{-cx}\right) \cdot \mathsf{pk}^{s_p - cr_p} = g_p^z \cdot \mathsf{pk}^{s_p} \cdot \left(g_p^x \cdot \mathsf{pk}^{r_p}\right)^{-c},$$

the simulator computes the commitment K_p as

$$K_p \leftarrow g_p^z \cdot \mathsf{pk}^{s_p} \cdot \left(X_p^c\right)^{-1}$$

and commitments K_q and K'_q analogously. It also computes the commitment w as

$$w \leftarrow g_p^{s_p} \cdot \left(y^c\right)^{-1}.$$

Finally, the simulator programs the random oracle $\mathcal{O}$ such that

$$c \leftarrow \mathcal{O}\Big(\big((y, X_p), (g_p, \mathsf{pk}), (g_q, h_q), (a, b)\big), (w, K_p, K_q, K'_q)\Big)$$

and then outputs the transcript $\big((w, K_p, K_q, K'_q), c, (z, s_p, s_q, s'_q)\big)$.

Since non-interactive Bulletproofs are fully zero-knowledge in the ROM [46, §4.4], π_{rp_a} and π_{rp_b} do not affect the zero-knowledge property of the protocol. However, an unbounded adversary can recover r_p from y and therefore decrypt X_p to recover x, thus breaking the zero-knowledge property. This is not feasible for a computationally bounded adversary, and thus the protocol can only satisfy computational zero-knowledge.

It remains to show that the distributions of the real and simulated transcripts are computationally indistinguishable in the programmable random oracle model.

1. In a real protocol execution, the oracle samples c at random and returns it to the prover. In the simulated protocol, the oracle returns the value c which was randomly sampled by the simulator. As such, in both real and simulated protocol executions, c is uniformly distributed.
2. By Lemma 1, z is distributed uniformly in $\{2^{b_x+b_c}, \ldots, 2^{b_x+b_c+b_\perp} - 1\}$ in both real and simulated transcripts since there can be no aborted transcripts.
3. In both transcripts, the values s_p, s_q, and s'_q are uniformly distributed.
4. Since the simulator chooses s_p uniformly from $\mathbb{Z}_p$, $K_p = g_p^k \cdot \mathsf{pk}^{t_p} = g_p^k \cdot \mathsf{pk}^{s_p - cr_p}$ is uniformly distributed in $\mathbb{G}_p$. Uniform distribution can be similarly shown for K_q, K'_q. Furthermore, $w = g_p^{t_p} = g_p^{s_p - cr_p}$, and so w is also uniformly distributed in $\mathbb{G}_p$. $\square$

6.4 Concrete instantiation

From Estonian election statistics [93], the largest number of candidates in an election since 1992 is 15322. However, this is the cumulative candidate count across all local governments. In any concrete municipality, the number of candidates a voter can vote for is much less, and candidate numbers are not global for local elections. The largest number of candidates unified throughout the country is only 1885 [93]. Regardless of the election type, it is reasonable to assume that for the foreseeable future, no election will have more than 2^{16} candidates, and so $b_x = 16$.

IVXV uses finite field ElGamal in Group 15 from RFC3526 [5], and so $b_q = 3071$. Given the current state of classical cryptanalysis, this corresponds to a security level of 128 bits, and so $\lambda = 128$. Since the protocol requires $b_c \geq \lambda$ for soundness without repetitions, $b_c = 128$ satisfies this requirement, but only in the interactive setting. In the non-interactive setting, the hash function must have a range of $\{0,1\}^{2\lambda}$ to achieve a collision resistance of λ bits [91, §7.3]. In practice, $b_c = 256$ is therefore required for a 128-bit security level against confidentiality and soundness attacks.

It remains to find values for $b_\perp$ and b_q such that $b_x + b_c + b_\perp < b_q$ and $b_q \ll b_p$. A popular and performant implementation of Bulletproofs [94] is implemented over ristretto255[1], which is itself built on top of Curve25519. While using ristretto255 would be beneficial for the performance, it has a group size of 2^{252}, which is incompatible with $b_c = 256$. By weakening the soundness guarantees to 112 bits of soundness, the challenge size can be set to $b_c = 224$. Then, $b_\perp = 251 - 16 - 224 = 11$ and the probability of aborts becomes 2^{-11}. This remains small and the computational burden of occasional aborts is only borne by the prover. Since the non-interactive version of the protocol is full computational zero-knowledge, the zero-knowledge guarantees are not impacted and a 128-bit security level against confidentiality attacks is therefore maintained.

To achieve 128-bit soundness guarantees, a larger curve such as P-384 should be used. By setting $b_q = 384$, it follows that $b_\perp < 112$, and so the probability of aborts is 2^{-112}. While $b_\perp$ could be lowered if needed for compatibility with a smaller curve that still accommodates $b_c = 256$, P-384 is a popular curve choice. Notably P-384 is used by the Estonian ID card for digital signatures [95, p. 135].

[1] `https://ristretto.group`

7. Prototype and benchmarks

To analyse and benchmark the computational cost of verification, a prototype of the full protocol was developed in Go [96, 97]. Go was chosen since the IVXV vote collector is also written in Go [98], therefore simplifying the real-system benchmarking. In addition, the prototype can hopefully serve as a template for a production implementation of the protocol should the approach be approved by the SEO.

The Bulletproofs needed as part of the protocol were not implemented from scratch. Rather, the open-source implementation by ING Bank [99]—hereinafter 'library'—was taken as a basis. No implementation of Bulletproofs+ was found for Go. Since the library uses secp2561k as a hardcoded elliptic curve, an abstraction inspired by the CIRCL library [100] was created. The hardcoded curve was subsequently replaced with the abstraction for the ease of testing the protocol with different curves. Additionally, the library code was refactored and consolidated in places for additional clarity and fidelity to the Bulletproofs paper [46].

A shortcoming of the library is that it lacks the batching technique which allows for greater efficiency for proving and verifying two (or more) Bulletproofs. Due to time restrictions, batching was not implemented on top of the library either. There exists an alternative prototype implementation of Bulletproofs in Go (bp-go) which implements the batching technique [101]. However, bp-go seems to mimic the original prototype of Bulletproofs by Bünz in Java [102], rather than following the algorithm descriptions in the paper. As a result, it is hard to verify why the parts of bp-go that deviate from the paper are correct. Moreover, bp-go could not be run due to its use of a version of a package that is no longer available[1]. While a newer version of the package exists[2], it is not usable with bp-go without code changes.

Some prime-field NIST elliptic curves (e.g. P-256, P-384) are available in Go as part of the standard library, however, the direct use of the curve operations has been deprecated [103]. The CIRCL library enhances the P-256 and P-384 implementations provided by the Go standard library, and provides optimised operations on P-384 [100]. CIRCL also supports the use of the ristretto255 group implemented by the go-ristretto library [104]. The versions of P-256, P-384 and ristretto255 provided by CIRCL, and ING Bank's secp256k1 implementation were used for benchmarking the full protocol.

[1] `https://pkg.go.dev/github.com/revolutionchain/btcd/btcec`
[2] `https://pkg.go.dev/github.com/revolutionchain/btcd/btcec/v2`

Initial benchmarks were run on a MacBook Pro with a 2.42 GHz M2 Pro processor. Both the prover and verifier were part of the same program, all data was held in memory and no data was serialised. The average of 1000 runs was taken where the candidate number was fixed to 1500 and the proof range was set to [101, 2000] with $b_x = 16$. Batching was not used for generating or verifying Bulletproofs. The results are presented in Table 1.

					Verifier's work (ms)		
λ	κ	b_c	Curve group	Prover's work (ms)	BP	RP	Total
128	112	224	secp256k1	39.10	10.81	20.34	31.15
128	112	224	ristretto255	41.19	11.08	20.88	31.96
128	112	224	P-256	40.30	10.77	20.01	30.78
128	112	224	P-384	171.8	79.06	22.76	101.8
128	128	256	P-384	169.4	77.89	23.17	101.1

Table 1. Prototype benchmarks on a MacBook Pro with a 2.42 GHz M2 Pro processor. All timings are in milliseconds. 'BP' represents the time required to verify both Bulletproofs, while 'RP' represents the time needed to verify Π_{Veq}.
λ indicates the security level against confidentiality attacks and is upper bounded by the 128-bit security level of the ElGamal group. κ indicates the soundness level of the scheme.

It is clear from the benchmarks that on the prover's side, the proof generation is unlikely to impact the voting experience. While the voting client is written in C++ and not Go, implementation performance in C++ should be comparable. While verification times are lower than proving times, the server must be able to handle and process many concurrent requests. As such, the direct impact is more difficult to assess based on these benchmarks alone. However, it is clear that the curve choice may have a significant impact on the verifier, whereas the impact on the prover is negligible in practice.

To better determine the impact of the protocol on the vote collection server, i.e. the verifier, a standard IVXV benchmark was run with the verification function added in. According to internal SCCEIV data, the peak concurrent load for IVXV has been 12 votes per second. A typical benchmark target is therefore to process 40 votes per second until the target number of votes has been cast [105]. As such, if the vote collector can keep up with this rate with the additional verification added in, the protocol can be deemed practical.

The goal was to run benchmarks on the same servers that are used during the actual elections, which are physical servers in Estonia hosted by the Estonian Information System Authority (RIA). The servers run Ubuntu 22.04, with 8 CPU cores with a base frequency of 2.90 GHz, and 16 GB of RAM. The processor itself is the Intel® Xeon® Gold 6326 Processor with 16 physical cores, however the vote collector is virtualised, with 8 CPU cores available to the virtualised container. For load balancing and redundancy, the vote

collector is comprised of three servers. Unfortunately, due to the proximity in time to the 2024 elections, there was little availability for benchmarking on this hardware. As a result, only two benchmarks could be run for which P-256 and P-384 were chosen.

For the benchmarks, a proof of vote correctness was serialised into JSON and stored as a Go variable in the server-side code. For each received vote, after performing the habitual verification checks, the server de-serialised the JSON proof and verified it, therefore simulating the actual work of the verifier. In a production setting, the proof and the ballot will need to be read from the ASiC-E container containing the encrypted ballot instead. However, this overhead is likely to be marginal. Additionally, for the benchmarking, the server re-generated the public parameters before verifying the proof since this allowed easily hooking the prototype to the vote collector. This can easily be avoided in production since the public parameters are known in advance and are the same for every proof.

A total of 882366 ballots were cast, which was the size of the list of eligible voters in the testing environment, and for which a baseline benchmark already existed. The results are presented in Table 2. While the impact of the proof verification on the processing rate of received ballots is negligible, some overhead is still introduced as shown by the increased database error rate.

Category	Duration	Processing rate	DB error count
Reference	06:21:03	38.59	16
P-256	06:20:28	38.66	228
P-384	06:24:47	38.22	1205

Table 2. Load tests with proof verification on the IVXV vote collector with 882366 ballots cast. The duration is in hh:mm:ss format, the processing rate is in ballots per second, and the benchmark target was 40 ballots/s. DB error count represents the number of errors related to the etcd database of the vote collector.

The vote collector uses etcd[3] as its database for storing ballots. While the common practice is to allocate dedicated resources to etcd [106], the vote collector's etcd shares its resources with the rest of the collector's services, including the verifier service. The higher etcd error count could therefore be explained by the proof verification requiring some of the processing power that was previously used by etcd only. However, it was not possible to determine with certainty whether this was the case, or whether the etcd errors were caused by unrelated state-changes to the system in-between benchmarks.

If an etcd error happens before the received ballot is stored, the collector will retry the

[3] `https://etcd.io`

operation. However, if the ballot cannot be stored before the configured timeout—typically between 5–10 seconds—is reached, the voting process will fail and the voter will have to restart the voting process. While the number of etcd errors is higher with the proof verifications added in, the error rates remain marginal compared to the total number of votes cast. Furthermore, processing 40 ballots per second is three to eight times more than the expected voting rate during the elections. Even if such a rate is achieved during a peak, it is unlikely to be sustained for a period of several hours and the real impact is therefore likely to be negligible. This remains a speculation however, since a benchmark with a target rate of 10 ballots per second could not be performed due to the unavailability of the infrastructure.

A possible improvement would be to deploy etcd on its own dedicated resources. Not only could this improve or solve the performance problem that caused etcd transactions to fail, it could be a worthwhile architecture improvement in general. While this may require purchasing or leasing additional hardware, the resource requirements for etcd are modest [107] and the cost should therefore not be prohibitive. Three 4-core machines with 16 GB of RAM each should be sufficient for IVXV's etcd needs. Improvements regarding vote verification itself are discussed in Section 8.3.

Since no proofs were stored by the collector during the benchmarks, there is no empirical data available regarding the additional space needed for storing the proofs. One reason for this is due to the suboptimal serialisation of the internal data structures directly into JSON. In practice, ASN.1 DER should be used as the encrypted ballot is also encoded into an ASN.1 structure. To minimise space requirements further, elliptic curve point compression could be used. Regardless, as space is easily extensible, it was not deemed to be a potentially limiting factor.

8. Discussion

8.1 Protocol complexity

While the protocol for proving vote correctness was chosen with simplicity in mind, the resulting protocol is not exactly simple. The complexity is twofold: there is the complexity due to the underlying range proof protocol, but also the complexity due to the group switching strategy.

Group switching is seemingly unavoidable as long as the ElGamal ciphertext is generated in the prime-order group of quadratic residues. Because operations in this group are computationally expensive, it is unlikely that range proofs could be efficiently generated in this group, regardless of the technique used. Range proofs must therefore be generated in a group different than the one used for the ElGamal encryption, and a group switching strategy must be used to bind the proof to the ciphertext. Regarding range proofs, the author's opinion is that Bulletproofs(+) are the most understandable of the considered approaches.

8.2 Elliptic curve ElGamal

To avoid the need for group switching, the ElGamal ciphertext could be generated in a computationally more performant group instead. In practice, this means using lifted ElGamal in an elliptic curve group instead of the prime-order group of quadratic residues modulo a prime. In lifted elliptic curve ElGamal, the message is represented as an integer which is used as a scalar and is multiplied with the group's base point (the generator). In additive notation, this gives

$$\mathsf{Enc}_H(m; r) = (rG, mG + rH)$$

where G is the group's base point, H is the public key (a point on the curve), m is the message scalar, and r is a random scalar. By only encrypting an integer in the range $[0, 2^{16} - 1]$ it is certain that the message scalar will not overflow the group order.

By picking an elliptic curve compatible with the range proof, the encryption and the range proofs could then be generated in the same group. While this may not replace the need for generating a range proof separately from the ciphertext, proving the equality of discrete

logarithms in the same group is simpler than using the cross-group proof, e.g. as in [83]. Since there are less operations to do, this will also improve overall performance on top of simplifying the vote correctness protocol. However, a complete migration to elliptic curves involves requires re-implementing the current IVXV cryptosystem, thus losing the 'tried and tested' advantage of the current implementation. The potential development effort is therefore much higher than solely using elliptic curves for the vote correctness proof.

8.3 Improving performance

While the benchmark results show that the current prototype should already be practical performance-wise, there are a number of ways to potentially improve the performance. Performance improvements are especially important if the SEO decides that the current buffer zone between real loads and test loads is not large enough.

The first change would be to implement Bulletproof batching to verify the two range proofs simultaneously. This would replace the $4\lceil\log_2(n)\rceil + 8$ group elements needed for verifying the proofs separately with $2\lceil\log_2(n) + 1\rceil + 4$ elements, thus shortening the proofs. While the exact time-improvement is hard to estimate, verification time should improve at least by a quarter judging by data from the original paper [46, §6.3]. Second, since Bulletproofs+ can be used as a drop-in replacement to Bulletproofs, Bulletproofs+ could be used instead, although this would improve the proof size rather than verification time. While Bulletproofs++ are also a drop-in replacement to Bulletproofs(+), the protocol itself is arguably more complicated than the former two. As such, this change would not be recommended unless absolutely needed for performance. It might also be possible to solely replace the inner product argument with future improvements in the literature without replacing the Bulletproofs(+) themselves, however this is not guaranteed.

Finally, other than the potential improvements that could be obtained by optimising the Go code, a more radical change would be to use another programming language for the proof verification. In particular, Rust[1] could be used since many production-grade open source proof system implementations have been written in Rust (e.g. [94, 108–110]). In turn, this would remove the need for an in-house implementation of the underlying proof system and is likely to also improve performance. Developers could therefore treat the implementation as a black-box, although any third-party library should still be audited.

[1] `https://www.rust-lang.org`

8.4 Range vs set-membership proofs

While range proofs were chosen in this work due to their arguable simplicity compared to the identified set-membership schemes, ZKSMs have advantages over ZKRPs as well. Since ZKSMs do not require a continuous range of allowed values, but only a public set, they offer additional flexibility over ZKRPs. One potential issue of using a range proof for proving vote correctness is the uncertainty surrounding whether it is possible for the candidate range not to be continuous. It is not clear from the law what happens to candidate numbers if a candidate's eligibility is revoked after the numbers have been issued. Likely, this candidate number will not be available to vote for, at least not in the digital voter lists. While the author sent a request to the SEO for clarifications, he unfortunately did not receive a response.

The issue of continuous ranges can be solved by mapping the set of available candidates to integers, and so the range proof approach remains applicable, although this would add another layer of complexity. However, there is another edge case which is interesting to consider.

1. In local government elections, the candidates are different per municipality, and voters can only vote for candidates in their municipality. The candidate numbers are unique per municipality, but not at the national level.
2. In European Parliament elections, the candidates are the same across Estonia, and voters can vote for any candidate.
3. In parliamentary elections, the candidates are the same across Estonia, however voters can only vote for the candidate in their electoral district (their county).

As such, for the first two cases, the range proofs can also be used to verify that the voter votes for a candidate they are allowed to vote, not just for an eligible candidate. As an example, if a resident of Tartu attempts to create an illegitimate ballot for a candidate in Tallinn, the vote collector could identify this immediately and refuse to accept the vote. Currently, this is mitigated by sorting ballots into separate ballot boxes depending on the municipality associated with each voter. The ballot boxes are then decrypted separately, and since the ballots contain also candidate names (Section 2.4), the outliers can be discarded.

In the third case, the range proof does not allow for this by default, whereas a set membership proof would. Once more, it could then be possible to map all the eligible candidates under the hood such that numbers for candidates in the same district are consecutive. Changing the procedure for assigning candidate numbers itself is likely not a good idea since the drawing by lots system has been used for a long time. Allocating specific ranges

to districts is likely to encounter push-back due to fairness claims (smaller numbers are higher in lists), although district ordering could theoretically also be drawn by lots.

8.5 Proof verification

The proofs of vote correctness must be included in the digitally singed ASiC-E container alongside the encrypted ballot. The proofs of vote correctness must be verified by the vote collector upon receiving each vote, as this enables it to provide immediate feedback to the voter when something is amiss. It remains to determine what should the verification application, the processing application, the key application, and the auditor do.

Since the verification application has the capability to decrypt the vote by using the randomness used during encryption it can verify the range directly, without needing the range proofs. In fact, the verification application currently already verifies whether the ballot contains an eligible candidate. A necessary change to the verification application, however, is to display the full candidate information as usual. That is, the verification application should map the integer in the ballot back to the candidate list, particularly if candidate numbers were remapped to avoid gaps in the range. In principle, it should not be possible to cast a vote with an invalid range proof without the collector refusing to accept the vote. However, the purpose of the verification application is to determine whether both the voting client and the collector behaved correctly. Since the verification application can only be used if a vote has been successfully cast, the verification application should still verify the range proofs. If the range proofs do not validate, the application should alert the voter that the collector misbehaved.

Similarly, the processing application should verify the range proofs to also make sure that the collector did not misbehave. The after the fact verification of proofs is the main reason why the proofs should be non-interactive and transferable. If the processing application does not verify that all range proofs are valid, ballots with illegitimate contents could still reach the decryption phase if they slipped by the collector. Since the processing application performs all necessary verifications and is operated by the organiser, the output of the application can be considered trusted. As such, the key application needs not verify the range proofs, similarly to how the key application does not currently verify whether ballots are well formed, as this is already done by the processor. In fact, it is not possible to verify the vote correctness proofs after ballots have been mixed since the ballot randomness is altered. Therefore, the proofs can only be verified with pre-mixnet ballots. The auditor should verify all the proofs as well.

The processing application and the Android verification application are written in Java

while the iOS verification application is written in Objective-C. This means that Bulletproofs will also have to be implemented in these other languages, which increases the development effort significantly. While this development effort may outweigh the benefits of proof verification by the verification applications, the verification by the processing application is a must. For Java, the implementation by Bünz [102] could be taken as a basis.

9. Conclusion

In the current version of IVXV, there is no way to verify whether an encrypted ballot contains a vote for an eligible candidate until the ballot is decrypted. As a result, using a modified voting client, it is possible to cast a ballot containing arbitrary data, and this ballot will be decrypted during the tallying process. The key application that decrypts the ballots issues ZKPs of correct decryption to prove that it correctly decrypted the ballots.

Previously, it was not possible to generate proofs of correct decryption for invalid ballots, however, the author has remediated this in the key application for the 2024 elections. Regardless of whether proofs of correct decryption are generated for invalid ballots, election observers are not allowed to verify those proofs, only a designated auditor is. The rationale behind this is that invalid ballots may contain information that can identity the original voter. As a result, observers cannot verify whether the key application has falsely declared ballots as invalid as a means to manipulate the election result. By ensuring that invalid ballots cannot reach the final ballot box and the decryption stage, all decryption proofs could be made publicly available, e.g. online. Thus, anyone could verify that the key application behaved correctly, and that the individual votes sum up to the claimed tally.

To this end, the goal of this work was to introduce ZKPs of ballot validity into IVXV. That is, voters would have to prove that they have voted for a valid candidate, but without leaking who they voted for. After considering different state of the art proof systems, the Bulletproofs range proof was chosen. However, by themselves, Bulletproofs only allow to prove that a committed integer is non-negative, and less than some maximum. That is, they do not support range proofs for arbitrary ranges by default. Moreover, for performance reasons, Bulletproofs must be generated in an elliptic curve group, which is separate from the algebraic group where the ballot is encrypted in. To mitigate both issues, a group switching strategy was proposed to bind the Bulletproofs in the elliptic curve group to the group used for encryption. The group switching strategy was also enhanced to combine two Bulletproofs such that they would form a range proof for an arbitrary range. As a result, a system for proving that the ballot encrypts a number in an given range was created. In cases where the candidate numbers are not consecutive, the proof system remains usable by mapping the candidate numbers to a consecutive integer range.

To verify whether the proposed scheme is practically usable, a prototype was implemented in Go, the same language that the vote collector is written in. This prototype demon-

strates that the system is realisable in practice and can serve as a reference for a future production implementation. Additionally, the proof verification was benchmarked on the same infrastructure used during real elections to see whether the system is practical performance-wise. While the results show a slight performance impact for the tested configurations, the proposed system remains practical. The prototype is publicly available and can be used to benchmark other configurations. Whether to implement the proposed scheme into IVXV or to further research the subject beforehand will be determined by discussions with the SEO.

Several potential optimisations were proposed to the scheme, the analysis of which is left for future work. Additionally, some schemes that were discarded in this work but were identified as potentially usable could be explored in future work to see how they compare to the proposed scheme. Finally, whereas research on a post-quantum secure voting system to use in Estonia is ongoing, porting such a range proof to a post-quantum scheme would also be valuable.

References

[1] Ülle Madise and Tarvi Martens. 'E-voting in Estonia 2005. The first Practice of Country-wide binding Internet Voting in the World'. In: *Electronic Voting 2006: 2nd International Workshop, Co-organized by Council of Europe, ESF TED, IFIP WG 8.6 and E-Voting.CC*. Ed. by Robert Krimmer. Vol. P-86. LNI. GI, 2006, pp. 15–26.

[2] State Electoral Office. *Statistics about Internet voting in Estonia*. 2023. URL: https://www.valimised.ee/en/archive/statistics-about-internet-voting-estonia (visited on 12/05/2024).

[3] Sven Heiberg et al. 'Improving the Verifiability of the Estonian Internet Voting Scheme'. In: *E-Vote-ID 2016, Proceedings*. Ed. by Robert Krimmer et al. Vol. 10141. Lecture Notes in Computer Science. Springer, 2016, pp. 92–107. DOI: 10.1007/978-3-319-52240-1_6.

[4] *Elektroonilise hääletamise üldraamistik ja selle kasutamine Eesti riiklikel valimistel*. Tech. rep. Version 1.1. 3rd Feb. 2023, IVXV-ÜK-1.1.

[5] Jan Willemson. 'Creating a Decryption Proof Verifier for the Estonian Internet Voting System'. In: *Proceedings of the 18th International Conference on Availability, Reliability and Security, ARES 2023*. ACM, 2023, 58:1–58:7. DOI: 10.1145/3600160.3605467.

[6] Douglas Wikström et al. 'How Could Snowden Attack an Election?' In: *E-Vote-ID 2017, Proceedings*. Ed. by Robert Krimmer et al. Vol. 10615. Lecture Notes in Computer Science. Springer, 2017, pp. 280–291. DOI: 10.1007/978-3-319-68687-5_17.

[7] Johannes Müller. 'Breaking and Fixing Vote Privacy of the Estonian E-Voting Protocol IVXV'. In: *FC 2022 International Workshops, Revised Selected Papers*. Ed. by Shin'ichiro Matsuo et al. Vol. 13412. Lecture Notes in Computer Science. Springer, 2022, pp. 325–334. DOI: 10.1007/978-3-031-32415-4_22.

[8] Véronique Cortier et al. 'SoK: Verifiability Notions for E-Voting Protocols'. In: *IEEE Symposium on Security and Privacy, SP 2016, Proceedings*. IEEE Computer Society, 2016, pp. 779–798. DOI: 10.1109/SP.2016.52.

[9] Eesti Vabariik. *Eesti Vabariigi põhiseadus*. Riigi Teataja. 2015. RT: 115052015002.

[10] Smartmatic-Cybernetica. *IVXV Processor Application*. Version 1.8.2-RK2023. 2023. URL: https://github.com/valimised/ivxv/tree/master/processor (visited on 12/05/2024).

[11] Riigikogu. *Riigikogu valimise seadus*. Riigi Teataja. 2020. RT: 103012020013.

[12] Smartmatic-Cybernetica. *IVXV*. Version 1.8.2-RK2023. 2023. URL: https://github.com/valimised/ivxv/tree/master (visited on 12/05/2024).

[13] *IVXV seadistuste koostamise juhend*. Tech. rep. Version 1.8.2. 23rd Jan. 2023, IVXV-JSK-1.8.2.

[14] *IVXV: E-hääletamise käsiraamat*. Tech. rep. Version 0.7. 13th Feb. 2023, IVXV-KR-0.7.

[15] Jaan Oruaas. *Elektroonilise hääletamise protsessi audit. Lõpparuanne*. FocusIT, 8th Dec. 2017.

[16] Mihkel Kukk. *Riigikogu valimised 2019. Elektroonilise hääletamise protsessi auditeerimine. Lõpparuanne*. KPMG Baltics OÜ, 17th Apr. 2019.

[17] Mihkel Kukk. *2019. aasta Euroopa Parlamendi valimiste elektroonilise hääletamise protsessi auditeerimine. Lõpparuanne*. KPMG Baltics OÜ, 17th Sept. 2017.

[18] Vitali Jekimtsev. *Elektroonilise hääletamise protsessi auditeerimine. Lõpparuanne*. Valimised: 2021. aasta kohaliku omavalitsuse volikogude valimised. KPMG Baltics OÜ, 7th Feb. 2022.

[19] Risto Jürisson. *Elektroonilise hääletamise protsessi auditeerimine. Lõpparuanne*. Valimised: 2023. aasta riigikogu valimised. KPMG Baltics OÜ, 16th May 2023.

[20] Vabariigi Valimiskomisjon. *Hääletamissedeli ja elektroonilise hääle vormi kehtestamine*. Riigi Teataja. 2023. RT: 322092023001.

[21] *IVXV protokollide kirjeldus*. Tech. rep. Version 1.8.2. 23rd Jan. 2023, IVXV-PR-1.8.2.

[22] Smartmatic-Cybernetica. *IVXV Key Application*. Version 1.8.2-RK2023. 2023. URL: https://github.com/vvk-ehk/ivxv/tree/master/key (visited on 12/05/2024).

[23] Josh D. Cohen and Michael J. Fischer. 'A Robust and Verifiable Cryptographically Secure Election Scheme'. In: *26th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 1985, pp. 372–382. DOI: 10.1109/SFCS.1985.2.

[24] Josh Cohen Benaloh and Moti Yung. 'Distributing the Power of a Government to Enhance the Privacy of Voters'. In: *Proceedings of the Fifth Annual ACM Symposium on Principles of Distributed Computing*. Ed. by Joseph Y. Halpern. ACM, 1986, pp. 52–62. DOI: 10.1145/10590.10595.

[25] Josh Cohen Benaloh. ‘Verifiable secret-ballot elections’. PhD thesis. Yale University, 1987.

[26] Ronald Cramer, Rosario Gennaro and Berry Schoenmakers. ‘A Secure and Optimally Efficient Multi-Authority Election Scheme’. In: *EUROCRYPT ’97, Proceedings*. Ed. by Walter Fumy. Vol. 1233. Lecture Notes in Computer Science. Springer, 1997, pp. 103–118. DOI: 10.1007/3-540-69053-0_9.

[27] Henri Devillez, Olivier Pereira and Thomas Peters. ‘How to Verifiably Encrypt Many Bits for an Election?’ In: *ESORICS 2022, Proceedings, Part II*. Ed. by Vijayalakshmi Atluri et al. Vol. 13555. Lecture Notes in Computer Science. Springer, 2022, pp. 653–671. DOI: 10.1007/978-3-031-17146-8_32.

[28] Michael R. Clarkson, Stephen Chong and Andrew C. Myers. ‘Civitas: Toward a Secure Voting System’. In: *2008 IEEE Symposium on Security and Privacy (SP 2008)*. IEEE Computer Society, 2008, pp. 354–368. DOI: 10.1109/SP.2008.32.

[29] Daniel Sandler, Kyle Derr and Dan S. Wallach. ‘VoteBox: A Tamper-evident, Verifiable Electronic Voting System’. In: *Proceedings of the 17th USENIX Security Symposium*. Ed. by Paul C. van Oorschot. USENIX Association, 2008, pp. 349–364.

[30] Josh Benaloh et al. ‘STAR-Vote: A Secure, Transparent, Auditable, and Reliable Voting System’. In: *2013 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections*. USENIX Association, 2013.

[31] Véronique Cortier, Pierrick Gaudry and Stéphane Glondu. ‘Belenios: A Simple Private and Verifiable Electronic Voting System’. In: *Foundations of Security, Protocols, and Equational Reasoning - Essays Dedicated to Catherine A. Meadows*. Ed. by Joshua D. Guttman et al. Vol. 11565. Lecture Notes in Computer Science. Springer, 2019, pp. 214–238. DOI: 10.1007/978-3-030-19052-1_14.

[32] Josh Benaloh and Michael Naehrig. *Election Guard Design Specification*. Tech. rep. Version 2.0.0. Microsoft Reserach, 18th Aug. 2023.

[33] Arnis Parsovs. *Homomorphic Tallying for the Estonian Internet Voting System*. Cryptology ePrint Archive. 2016. IACR: 2016/776.

[34] Martin Hirt and Kazue Sako. ‘Efficient Receipt-Free Voting Based on Homomorphic Encryption’. In: *EUROCRYPT 2000, Proceedings*. Ed. by Bart Preneel. Vol. 1807. Lecture Notes in Computer Science. Springer, 2000, pp. 539–556. DOI: 10.1007/3-540-45539-6_38.

[35] Rui Joaquim. ‘How to prove the validity of a complex ballot encryption to the voter and the public’. In: *Journal of Information Security and Applications* 19.2 (2014), pp. 130–142. DOI: 10.1016/J.JISA.2014.04.004.

[36] Stefan Brands. 'Rapid Demonstration of Linear Relations Connected by Boolean Operators'. In: *EUROCRYPT '97, Proceedings*. Ed. by Walter Fumy. Vol. 1233. Lecture Notes in Computer Science. Springer, 1997, pp. 318–333. DOI: 10.1007/3-540-69053-0_22.

[37] Nicolas Huber et al. 'Kryvos: Publicly Tally-Hiding Verifiable E-Voting'. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022*. Ed. by Heng Yin et al. ACM, 2022, pp. 1443–1457. DOI: 10.1145/3548606.3560701.

[38] Jens Groth. 'On the Size of Pairing-Based Non-interactive Arguments'. In: *EUROCRYPT 2016, Proceedings, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. Lecture Notes in Computer Science. Springer, 2016, pp. 305–326. DOI: 10.1007/978-3-662-49896-5_11.

[39] Wouter Lueks, Iñigo Querejeta-Azurmendi and Carmela Troncoso. 'VoteAgain: A scalable coercion-resistant voting system'. In: *29th USENIX Security Symposium, USENIX Security 2020*. Ed. by Srdjan Capkun and Franziska Roesner. USENIX Association, 2020, pp. 1553–1570.

[40] Stephanie Bayer and Jens Groth. 'Zero-Knowledge Argument for Polynomial Evaluation with Application to Blacklists'. In: *EUROCRYPT 2013, Proceedings*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 646–663. DOI: 10.1007/978-3-642-38348-9_38.

[41] Rolf Haenni et al. *CHVote Protocol Specification*. Cryptology ePrint Archive. 2017. IACR: 2017/325.

[42] Torben P. Pedersen. 'Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing'. In: *CRYPTO '91, Proceedings*. Ed. by Joan Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Springer, 1991, pp. 129–140. DOI: 10.1007/3-540-46766-1_9.

[43] Liina Kamm. 'Homological classification of commitment schemes'. MA thesis. University of Tartu, 2007.

[44] Taher El Gamal. 'A public key cryptosystem and a signature scheme based on discrete logarithms'. In: *IEEE Trans. Inf. Theory* 31.4 (1985), pp. 469–472. DOI: 10.1109/TIT.1985.1057074.

[45] Shafi Goldwasser, Silvio Micali and Charles Rackoff. 'The Knowledge Complexity of Interactive Proof Systems'. In: *SIAM J. Comput.* 18.1 (1989), pp. 186–208. DOI: 10.1137/0218012.

[46] Benedikt Bünz et al. 'Bulletproofs: Short Proofs for Confidential Transactions and More'. In: *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings*. IEEE Computer Society, 2018, pp. 315–334. DOI: 10.1109/SP.2018.00020.

[47] Jonathan Bootle et al. 'Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting'. In: *EUROCRYPT 2016, Proceedings, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. Lecture Notes in Computer Science. Springer, 2016, pp. 327–357. DOI: 10.1007/978-3-662-49896-5_12.

[48] Jens Groth and Markulf Kohlweiss. 'One-Out-of-Many Proofs: Or How to Leak a Secret and Spend a Coin'. In: *EUROCRYPT 2015, Proceedings, Part II*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. Lecture Notes in Computer Science. Springer, 2015, pp. 253–280. DOI: 10.1007/978-3-662-46803-6_9.

[49] Emilia Käsper, Sven Laur and Helger Lipmaa. 'Black-Box Knowledge Extraction Revisited: Universal Approach with Precise Bounds'. In: (2006). IACR: 2006/356.

[50] Ivan Damgård. *On Σ-protocols*. 2010. URL: https://cs.au.dk/%7Eivan/Sigma.pdf (visited on 12/05/2024).

[51] ZKProof. *ZKProof Community Reference, Version 0.3*. Ed. by D. Benarroch et al. Updated versions at https://docs.zkproof.org/reference. 2022.

[52] Amos Fiat and Adi Shamir. 'How to Prove Yourself: Practical Solutions to Identification and Signature Problems'. In: *CRYPTO '86, Proceedings*. Ed. by Andrew M. Odlyzko. Vol. 263. Lecture Notes in Computer Science. Springer, 1986, pp. 186–194. DOI: 10.1007/3-540-47721-7_12.

[53] Mihir Bellare and Phillip Rogaway. 'Random Oracles are Practical: A Paradigm for Designing Efficient Protocols'. In: *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*. Ed. by Dorothy E. Denning et al. ACM, 1993, pp. 62–73. DOI: 10.1145/168588.168596.

[54] Ran Canetti et al. 'Universally composable two-party and multi-party secure computation'. In: *Proceedings on 34th Annual ACM Symposium on Theory of Computing*. Ed. by John H. Reif. ACM, 2002, pp. 494–503. DOI: 10.1145/509907.509980.

[55] Matteo Campanelli, Dario Fiore and Anaïs Querol. 'LegoSNARK: Modular Design and Composition of Succinct Zero-Knowledge Proofs'. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019*.

Ed. by Lorenzo Cavallaro et al. ACM, 2019, pp. 2075–2092. DOI: 10.1145/3319535.3339820.

[56] Polygon Zero Team. *Plonky2: Fast recursive arguments with PLONK and FRI*. 2022. URL: https://github.com/0xPolygonZero/plonky2/blob/main/plonky2/plonky2.pdf (visited on 12/05/2024).

[57] David Chaum and Eugène van Heyst. 'Group Signatures'. In: *EUROCRYPT '91, Proceedings*. Ed. by Donald W. Davies. Vol. 547. Lecture Notes in Computer Science. Springer, 1991, pp. 257–265. DOI: 10.1007/3-540-46416-6_22.

[58] Lidong Chen and Torben P. Pedersen. 'New Group Signature Schemes'. In: *EUROCRYPT '94, Proceedings*. Ed. by Alfredo De Santis. Vol. 950. Lecture Notes in Computer Science. Springer, 1994, pp. 171–181. DOI: 10.1007/BFB0053433.

[59] Jan Camenisch, Rafik Chaabouni and Abhi Shelat. 'Efficient Protocols for Set Membership and Range Proofs'. In: *ASIACRYPT 2008, Proceedings*. Ed. by Josef Pieprzyk. Vol. 5350. Lecture Notes in Computer Science. Springer, 2008, pp. 234–252. DOI: 10.1007/978-3-540-89255-7_15.

[60] Daniel Benarroch et al. *Zero-Knowledge Proofs for Set Membership: Efficient, Succinct, Modular*. Cryptology ePrint Archive. 2019. IACR: 2019/1255.

[61] Josh Cohen Benaloh and Michael de Mare. 'One-Way Accumulators: A Decentralized Alternative to Digital Sinatures (Extended Abstract)'. In: *EUROCRYPT '93, Proceedings*. Ed. by Tor Helleseth. Vol. 765. Lecture Notes in Computer Science. Springer, 1993, pp. 274–285. DOI: 10.1007/3-540-48285-7_24.

[62] Matteo Campanelli, Mathias Hall-Andersen and Simon Holmgaard Kamp. 'Curve Trees: Practical and Transparent Zero-Knowledge Accumulators'. In: *32nd USENIX Security Symposium, USENIX Security 2023*. Ed. by Joseph A. Calandrino and Carmela Troncoso. USENIX Association, 2023, pp. 4391–4408.

[63] Daniel Benarroch et al. 'Zero-knowledge proofs for set membership: efficient, succinct, modular'. In: *Des. Codes Cryptogr.* 91.11 (2023), pp. 3457–3525. DOI: 10.1007/S10623-023-01245-1.

[64] Samuel Dobson, Steven D. Galbraith and Benjamin Smith. 'Trustless unknown-order groups'. In: *CoRR* abs/2211.16128 (2022). DOI: 10.48550/ARXIV.2211.16128. arXiv: 2211.16128.

[65] Miranda Christ et al. *SoK: Zero-Knowledge Range Proofs*. Cryptology ePrint Archive. 2024. IACR: 2024/430.

[66] Jonathan Bootle et al. ‘Arya: Nearly Linear-Time Zero-Knowledge Proofs for Correct Program Execution’. In: *ASIACRYPT 2018, Proceedings, Part I*. Ed. by Thomas Peyrin and Steven D. Galbraith. Vol. 11272. Lecture Notes in Computer Science. Springer, 2018, pp. 595–626. DOI: 10.1007/978-3-030-03326-2_20.

[67] Kwantae Cho, Sangrae Cho and Soohyung Kim. ‘Lightweight Signature-based Range Proof’. In: *13th International Conference on Information and Communication Technology Convergence, ICTC 2022*. IEEE, 2022, pp. 1862–1865. DOI: 10.1109/ICTC55196.2022.9952590.

[68] Cong Deng et al. ‘Cuproof: Range Proof with Constant Size’. In: *Entropy* 24.3 (2022), p. 334. DOI: 10.3390/E24030334.

[69] Eduardo Morais et al. ‘A Survey on Zero Knowledge Range Proofs and Applications’. In: *CoRR* abs/1907.06381 (2019). arXiv: 1907.06381.

[70] Chunhua Deng et al. ‘A Survey on Range Proof and Its Applications on Blockchain’. In: *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2019*. IEEE, 2019, pp. 1–8. DOI: 10.1109/CYBERC.2019.00011.

[71] Ernest F. Brickell et al. ‘Gradual and Verifiable Release of a Secret’. In: *CRYPTO ’87, Proceedings*. Ed. by Carl Pomerance. Vol. 293. Lecture Notes in Computer Science. Springer, 1987, pp. 156–166. DOI: 10.1007/3-540-48184-2_11.

[72] Konstantinos Chalkias et al. ‘HashWires: Hyperefficient Credential-Based Range Proofs’. In: *Proc. Priv. Enhancing Technol.* 2021.4 (2021), pp. 76–95. DOI: 10.2478/POPETS-2021-0061.

[73] Geoffroy Couteau, Thomas Peters and David Pointcheval. ‘Removing the Strong RSA Assumption from Arguments over the Integers’. In: *EUROCRYPT 2017, Proceedings, Part II*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10211. Lecture Notes in Computer Science. 2017, pp. 321–350. DOI: 10.1007/978-3-319-56614-6_11.

[74] Geoffroy Couteau et al. ‘Efficient Range Proofs with Transparent Setup from Bounded Integer Commitments’. In: *EUROCRYPT 2021, Proceedings, Part III*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12698. Lecture Notes in Computer Science. Springer, 2021, pp. 247–277. DOI: 10.1007/978-3-030-77883-5_9.

[75] Geoffroy Couteau et al. ‘Sharp: Short Relaxed Range Proofs’. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022*. Ed. by Heng Yin et al. ACM, 2022, pp. 609–622. DOI: 10.1145/3548606.3560628.

[76] HeeWon Chung et al. *Bulletproofs+: Shorter Proofs for Privacy-Enhanced Distributed Ledger*. Cryptology ePrint Archive. 2020. IACR: 2020/735.

[77] Liam Eagen. *Bulletproofs++*. Cryptology ePrint Archive. 2022. IACR: 2022/510.

[78] Dan Boneh et al. *A Simple Range Proof From Polynomial Commitments*. 2020. URL: https://hackmd.io/@dabo/B1U4kx8XI (visited on 12/05/2024).

[79] HeeWon Chung et al. 'Bulletproofs+: Shorter Proofs for a Privacy-Enhanced Distributed Ledger'. In: *IEEE Access* 10 (2022), pp. 42067–42082. DOI: 10.1109/ACCESS.2022.3167806.

[80] Liam Eagen et al. *Bulletproofs++: Next Generation Confidential Transactions via Reciprocal Set Membership Arguments*. Cryptology ePrint Archive. 2022. IACR: 2022/510.

[81] Arantxa Zapico et al. 'Caulk: Lookup Arguments in Sublinear Time'. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022*. Ed. by Heng Yin et al. ACM, 2022, pp. 3121–3134. DOI: 10.1145/3548606.3560646.

[82] Liam Eagen, Dario Fiore and Ariel Gabizon. *cq: Cached quotients for fast lookups*. Cryptology ePrint Archive. 2022. IACR: 2022/1763.

[83] David Chaum and Torben P. Pedersen. 'Wallet Databases with Observers'. In: *CRYPTO '92, Proceedings*. Ed. by Ernest F. Brickell. Vol. 740. Lecture Notes in Computer Science. Springer, 1992, pp. 89–105. DOI: 10.1007/3-540-48071-4_7.

[84] Melissa Chase et al. *Proofs of discrete logarithm equality across groups*. Cryptology ePrint Archive. 2022. IACR: 2022/1593.

[85] Michel Abdalla et al. 'Tightly Secure Signatures From Lossy Identification Schemes'. In: *J. Cryptol.* 29.3 (2016), pp. 597–631. DOI: 10.1007/S00145-015-9203-7.

[86] Vadim Lyubashevsky. 'Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures'. In: *ASIACRYPT 2009, Proceedings*. Ed. by Mitsuru Matsui. Vol. 5912. Lecture Notes in Computer Science. Springer, 2009, pp. 598–616. DOI: 10.1007/978-3-642-10366-7_35.

[87] David Bernhard, Olivier Pereira and Bogdan Warinschi. 'How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios'. In: *ASIACRYPT 2012, Proceedings*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. Lecture Notes in Computer Science. Springer, 2012, pp. 626–643. DOI: 10.1007/978-3-642-34961-4_38.

[88] Eleanor McMurtry, Olivier Pereira and Vanessa Teague. 'When Is a Test Not a Proof?' In: *ESORICS 2020, Proceedings, Part II*. Ed. by Liqun Chen et al. Vol. 12309. Lecture Notes in Computer Science. Springer, 2020, pp. 23–41. DOI: 10.1007/978-3-030-59013-0_2.

[89] Julien Devevey et al. 'A Detailed Analysis of Fiat-Shamir with Aborts'. In: *CRYPTO 2023, Proceedings, Part V*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14085. Lecture Notes in Computer Science. Springer, 2023, pp. 327–357. DOI: 10.1007/978-3-031-38554-4_11.

[90] Manuel Barbosa et al. 'Fixing and Mechanizing the Security Proof of Fiat-Shamir with Aborts and Dilithium'. In: *CRYPTO 2023, Proceedings, Part V*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14085. Lecture Notes in Computer Science. Springer, 2023, pp. 358–389. DOI: 10.1007/978-3-031-38554-4_12.

[91] Douglas R. Stinson. *Cryptography - theory and practice*. Discrete mathematics and its applications series. CRC Press, 1995. ISBN: 978-0-8493-8521-6.

[92] Mihir Bellare and Tadayoshi Kohno. 'Hash Function Balance and Its Impact on Birthday Attacks'. In: *EUROCRYPT 2004, Proceedings*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. Lecture Notes in Computer Science. Springer, 2004, pp. 401–418. DOI: 10.1007/978-3-540-24676-3_24.

[93] State Electoral Office. *Statistics*. 2023. URL: https://www.valimised.ee/en/archive/statistics (visited on 12/05/2024).

[94] Henry de Valence, Cathie Yun and Oleg Andreev. *Bulletproofs (Dalek Cryptography)*. URL: https://github.com/dalek-cryptography/bulletproofs (visited on 12/05/2024).

[95] Arnis Parsovs. 'Estonian electronic identity card and its security challenges'. PhD thesis. University of Tartu, 2021.

[96] Taaniel Kraavi. *ZKRPs of ballot correctness*. 2024. URL: https://github.com/takakv/msc-poc (visited on 12/05/2024).

[97] Taaniel Kraavi. *ZKRPs of ballot correctness*. 2024. URL: https://gitlab.cs.taltech.ee/takraa/msc-poc (visited on 12/05/2024).

[98] Smartmatic-Cybernetica. *IVXV Voting Service*. Version 1.8.2-RK2023. 2023. URL: https://github.com/valimised/ivxv/tree/master/voting (visited on 12/05/2024).

[99] ING Bank. *zkrp*. 2021. URL: https://pkg.go.dev/github.com/ing-bank/zkrp (visited on 12/05/2024).

[100] Armando Faz-Hernández and Kris Kwiatkowski. *Introducing CIRCL: An Advanced Cryptographic Library*. Version 1.3.7. July 2019. URL: https://github.com/cloudflare/circl/ (visited on 12/05/2024).

[101] Willy R. Vasquez. *bp-go*. 2018. URL: https://github.com/wrv/bp-go (visited on 12/05/2024).

[102] Benedikt Bünz. *BulletProofLib*. 2017. URL: https://github.com/bbuenz/BulletProofLib (visited on 12/05/2024).

[103] The Go Authors. *Package elliptic*. URL: https://pkg.go.dev/crypto/elliptic (visited on 12/05/2024).

[104] Bas Westerbaan. *go-ristretto*. URL: https://github.com/bwesterb/go-ristretto (visited on 12/05/2024).

[105] Sven Heiberg. Private communication. May 2024.

[106] etcd Authors. *etcd FAQ*. 2023. URL: https://etcd.io/docs/v3.5/faq/#deployment (visited on 12/05/2024).

[107] etcd Authors. *etcd Hardware recommendations*. 2023. URL: https://etcd.io/docs/v3.5/op-guide/hardware/ (visited on 12/05/2024).

[108] The Tari Project. *Bulletproofs+*. URL: https://github.com/tari-project/bulletproofs-plus (visited on 12/05/2024).

[109] Polygon Zero Team. *plonky2*. URL: https://github.com/0xPolygonZero/plonky2 (visited on 12/05/2024).

[110] 0xProject. *OpenZKP*. URL: https://github.com/0xProject/OpenZKP (visited on 12/05/2024).

Appendices

Appendix 1 – Non-exclusive license for reproduction and publication of a graduation thesis[1]

I Taaniel Kraavi

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis 'Proving Vote Correctness in the Estonian Internet Voting System', supervised by Prof. Ahto Buldas and Jan Willemson
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

12.05.2024

[1]The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 – Plaintext ballot format

The Augmented Backus-Naur Form (ABNF) specification of the digital ballot format defined in decision RT III, 22.09.2023, 1 of the NEC [20] is the following:

```
ehak-district = 4DIGIT
choice-no = 1*11DIGIT
district-choice = ehak-district %x2E choice-no
choicelist-name = 1*100UTF-8-CHAR
choice-name = 1*100UTF-8-CHAR
ballot = district-choice %x1F choicelist-name %x1F choice-name
```

The ABNF form of the padded ballot [21] is

```
padded-ballot = %x00 %x01 *%xFF %x00 ballot
```

where `%xFF` is repeated as many times as needed for the padded-ballot to occupy as many bytes as the ElGamal modulus.