UNIVERSITÉ DU LUXEMBOURG

PhD-FSTM-2025-059
The Faculty of Science, Technology and Medicine

DISSERTATION

Presented on 20/05/2025 in Esch-sur-Alzette

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG
EN INFORMATIQUE

by

Rafieh MOSAHEB

QUANTUM-SAFE ELECTRONIC-VOTING SCHEMES

Dedication

To Prof. Ebad Mahmoodian (1943–2024),
my master's supervisor and mentor in Iran,
whose guidance and wisdom shaped both my research and my life.
You will forever live on in my heart.

Acknowledgements

I would like to acknowledge all those who have supported and inspired me throughout this journey.

First and foremost, I extend my deepest gratitude to my supervisor, Prof. Dr. Peter Y. A. Ryan; it has been a privilege to learn and conduct research under his mentorship. His consistent support and guidance enabled me to attend numerous schools, workshops, and conferences and to build a professional network in the field of electronic voting systems.

I am grateful to Dr. Johannes Müller, my daily supervisor, for his pivotal role in the first two papers that form the foundation of this thesis. His patient instruction from introductory concepts in electronic voting through advanced topics in cryptography and post-quantum cryptography, and his advice on presenting our works at conferences and workshops have deeply strengthened my confidence and expertise.

In the last year of my PhD studies, Dr. Peter Rønne stepped in with his encouragement and insight. His guidance was instrumental in the successful completion of the final research paper of my PhD journey.

I am profoundly grateful to the members of my defense committee. It was an honor to have Professors Véronique Cortier and Olivier Pereira on my defense panel. Special thanks to Prof. Dr. Cortier for her guidance from the very beginning of my doctoral studies: as a CET member, she has been a continual source of inspiration and remains a role model for me as a leading female expert whose contributions have profoundly advanced electronic voting research. I am also deeply grateful to Prof. Dr. Pereira for serving on my defense committee, for asking incisive questions, and for his thoughtful feedback on this thesis. My thanks also go to Dr. Volker Müller for agreeing to chair the defense and for his support on that important day.

I gratefully acknowledge my co-authors, Dr. Thomas Haines, Dr. Ivan Pryvalov, and Sara Sarfaraz, for their contributions to the papers that shaped the outline of this thesis.

I would like to thank the current and former members of APSIA for all the memorable moments over these four years: the shared lunches, TGIF gatherings, and coffee breaks (especially those "awakening" coffees of Dr. Yan Kim). Their professionalism, kindness, and support, both scientific and personal, made me feel an integral part of the group, even as its only female researcher in the last two years of my studies. In particular, I am grateful to Dr. Dimiter Ostrev for generously proofreading sections of this thesis and offering invaluable feedback, and to my officemate, Jan Oupický, for patiently helping me debug the code on my system when nothing else would work.

I am deeply thankful to the Iranian Student Association (ISA) board at the University of Luxembourg. In my final year of doctoral studies, I served as ISA president alongside a dedicated board, organizing events that infused my life beyond academia with community spirit. Those moments of collaboration and connection provided the balance and inspiration that truly enriched my doctoral journey.

Last but not least, I owe my deepest thanks to my lovely family, including our newest member, Kiyanaz, whose arrival during my PhD brought unexpected joy and reminded me of life's most precious moments. Even from afar, my family's video calls, messages, and shared laughter bridged the miles between us and lifted my spirits when I needed it most. Thank you for believing in me and for your continuous support throughout this entire journey.

This research was supported by the Luxembourg National Research Fund (FNR) under the CORE project EquiVox (C19/IS/13643617/EquiVox/Ryan).

To everyone who supported me along the way, my sincere thanks.

Abstract

Electronic voting (e-voting) has emerged as a transformative technology in the modern digital era. Many countries across the world are using e-voting systems in different types of elections, from political to non-political. One of the primary goals of e-voting is ensuring both verifiability and privacy simultaneously, which we refer to as *security*. Verifiability is a security feature that guarantees voters can confirm their vote is reflected in the final election result, while privacy guarantees that no one is able to link a vote to the voter who cast it. Verifiability needs to hold only for the duration of the election, whereas privacy needs to extend beyond the election period, even decades after the election. This property, known as *everlasting privacy* in the literature, ensures that even computationally unbounded adversaries cannot compromise voter privacy, securing elections against future advances in computing, including quantum computing. Researchers have proposed a wide variety of protocols to achieve this ambitious goal in secure e-voting, however, these protocols differ significantly, making the analysis and state-of-the-art complicated.

In this thesis, we first address this fragmentation by systematically analyzing all existing e-voting protocols designed to ensure everlasting privacy. We map out the relationships and dependencies among these protocols, evaluate their security and efficiency under realistic assumptions, and identify unresolved challenges in the field. Our work provides a foundational reference for researchers aiming to design secure e-voting systems with everlasting privacy, paving the way for privacy-preserving elections in the post-quantum era.

Building on these insights, we propose a novel e-voting system that integrates the best practices from prior research while addressing their limitations. Leveraging the Hyperion scheme as a foundation, we develop an enhanced protocol that not only guarantees everlasting privacy but also introduces everlasting receipt-freeness and coercion mitigation. Unlike existing systems like Selene and Hyperion, which rely on computational assumptions for privacy, our protocol offers privacy even against adversaries with unlimited computational power.

In secure electronic voting systems with everlasting privacy, the focus is on future-proofing privacy, while sometimes election verifiability relies on the computational soundness of zero-knowledge proofs (ZKP), which are vulnerable to quantum adversaries. Therefore, a key technical challenge is designing e-voting systems with efficient post-quantum cryptographic primitives to secure both privacy and verifiability against quantum attacks. In this thesis, we advance the state of post-quantum ZKPs by focusing on the ZKPs proposed by Jain et al., which are based on the conservative Learning Parity with Noise (LPN) assumption. We optimize the efficiency of these ZKPs, achieve formal security verification using EasyCrypt, and uncover flaws in existing implementations, demonstrating their vulnerability to malicious provers. Additionally, we construct the first code-based ZKP of shuffle, enabling a verifiable and privacy-preserving e-voting protocol with mixing-based tallying. Our e-voting system ensures both verifiability and vote privacy through the computational difficulty of decoding random linear codes, marking it as the first verifiable code-based e-voting system.

Contents

List of Figures

Chapter 1

Introduction

1.1 Security after Shor's Algorithm

Historically, thousands of years ago, "cryptography" was considered as the *art* of communication between a sender and a receiver in the presence of (passive) adversaries, with the primary objective of providing *privacy*, meaning that the adversary could not decipher the content of the communication [KL14]. The cryptography solution for communication with privacy is Private-Key (Symmetric) Cryptography. These cryptosystems, i.e., systems designed to encrypt messages in the presence of the adversary, are defined over three sets: the set of all possible keys in the key space $\mathcal{K}$, the set of plaintext messages in the message space $\mathcal{M}$, and the set of ciphertexts in the ciphertext space $\mathcal{C}$, all including a combination of letters of an alphabet. The two parties that aim to communicate using symmetric-key encryption, need to share a (secret) key $k \in \mathcal{K}$ before the communication. Then, the sender uses k to create the cipher $c \in \mathcal{C}$ based on the message $m \in \mathcal{M}$ (i.e., scramble the plaintext) and send c to the receiver. The receiver uses the same key k to decipher c (i.e., unscramble the ciphertext) and recover m. The basic informal security here is that the adversary who doesn't have the communication key k but learns the ciphertext c, doesn't obtain any information about m. All the early private-key encryptions, such as the Caesar cipher, are broken today and proven insecure [KL14].

The invention of the radio introduced a new class of adversaries who could eavesdrop on communications remotely, even without physical proximity [Riv90]. As technology progressed, cryptography had to evolve to keep pace with increasingly sophisticated threats. The 19th century marked a turning point with the advent of modern computing, which laid the foundation for what is known today as "modern cryptography". The mathematical analysis of cryptography in Shannon's scientific paper in 1945 (published in 1949) [Sha49] is known as a turning point in cryptographic papers of the 19th century, suggesting formal definitions of security as the first step of designing a cryptographic system [Gol+02].

In 1917, Gilbert Vernam proposed a symmetric cryptosystem named one-time pad (OTP), also referred to as Vernam's cipher [Kah96]. The two most important properties of OTP are that the key size $|k|$ and the message size $|m|$ are equal, and k is used just once for communication. More than two decades later, Shannon introduced the notion of *perfect secrecy* in his work, a.k.a information-theoretical security, and demonstrated that it is attainable by proving that OTP achieves perfect secrecy. Loosely speaking, perfect secrecy means that even a computationally unbounded adversary cannot find which plaintext the ciphertext originates from. This is known as the ciphertext-only attack. Despite the strong security guarantee of OTP, it is not practical for parties communicating long messages, since the key should be the same size. Therefore, the sender needs to know the exact size of the message prior to communication, and if the message is long, find a way to keep the long communication key too. It is proven that any encryption scheme and not only OTP, designed to achieve perfect secrecy, inherits the same drawback: the key size should be at least as long as the message size [KL14].

Modern cryptography is indeed the transformation of cryptography from an intuitive *art* to a structured *science* that emphasizes precise definitions, well-founded assumptions, and security proofs with more extended goals than solely communication with privacy [KL14]. The assumptions are widely-believed yet unproven, enabling cryptographers to provide rigorous security proofs based on them. The Kerckhoffs' principle [KL14], which modern cryptography is based on, asserts that "a system must be practically, if not mathematically, indecipherable". This principle implies that while in theory, an adversary with unlimited time could eventually break a cryptographic system, the goal is to ensure that doing so remains computationally infeasible within any realistic time frame.

In 1976, Diffie and Hellman invented Public-Key Cryptography in [DH76], addressing the drawback of pre-sharing the communication key in symmetric cryptography. In the public-key setting, the receiver should create a pair of public/private keys as pk/sk, share the public-key pk publicly while keeping the private-key sk secret. The public and private keys in this setting are mathematically related and anyone who wants to communicate with the receiver could simply use the pk. The sender uses pk to encrypt her message $m \in \mathcal{M}$, resulting in a ciphertext $c \in \mathcal{C}$, and sends c to the receiver. The receiver then uses sk to recover m from c. Diffie and Hellman in [DH76] observed that, unlike symmetric cryptography, public-key cryptography cannot attain perfect secrecy:

We note that neither public key cryptosystems nor one-way authentication systems can be unconditionally secure because the public information always determines the secret information uniquely among the members of a finite set. With unlimited computation, the problem could therefore be solved by a straightforward search.

Diffie and Hellman's impossibility statement assumes a setting where a public-key is derived from a single secret-key. This leads us to a weaker notion of security in the literature, known as *computational security*, in both public and private-key settings.

Modern cryptographic schemes rely on two computational problems, such as integer factorization (IF) and discrete logarithm (DL), which are infeasible or impractical for classical computers to solve within a reasonable (efficient) time frame. However, the advent of quantum computers has raised significant concerns about the future of cryptographic security, as these machines are expected to outperform classical computers in solving specific types of problems, such as IF and DL. This is where "post-quantum" security (i.e., the security in the presence of quantum computers) emerges.

The story of quantum computers began in 1981 with the physicist Richard Feynman. Recognizing the inherent challenges of simulating quantum mechanics on classical computers, such as the exponential growth in variables needed to represent n particles (2^n variables), Feynman proposed a groundbreaking idea: building a computer that itself operates on quantum principles. Unlike classical computers, which encode information in binary states (0s and 1s), quantum computers utilize quantum states to store information. Leveraging the postulates of quantum mechanics, these machines perform computations by applying quantum operations, or gates, directly to these states.

Quantum computers, while capable of performing all tasks a classical computer can, are distinguished by their ability to leverage quantum mechanics to solve certain problems more efficiently. The key question is not whether quantum computers can replicate classical computations but whether they can outperform classical systems in meaningful ways. This phenomenon, often termed *quantum supremacy*, marks the point where quantum computers solve problems that are practically infeasible for classical computers, even with the most advanced hardware.

A major breakthrough in this regard was achieved in 1994, when Peter Shor introduced a quantum algorithm capable of factoring large numbers exponentially faster than the best-known classical algorithms [Sho94]. This discovery has profound implications for cryptography, particularly for RSA encryption, whose security is based on the assumption that factoring large numbers is computationally infeasible. In theory, Shor's algorithm could render RSA encryption vulnerable by enabling an attacker to derive the private key from the public key, effectively breaking the encryption scheme.

Since Shor's work, the prospect of quantum attacks on modern cryptographic systems has fueled intense research and debate (to date, Google Scholar records around 13,000 citations of [Sho94]). As an example, a recent and controversial claim by researchers at Shanghai University suggests that "military-grade encryption" could be compromised using quantum computers [Wan+24]. In their paper, which is primarily written in Chinese, the authors describe using D-Wave's quantum annealers in a quantum-classical hybrid approach to optimize problem-solving, reportedly allowing them to factor a 50-bit number using Shor's algorithm. While factoring even small numbers using quantum methods represents a step forward in quantum computing, this claim has been met with skepticism. It misrepresents current quantum computing capabilities and overstates the immediate threat to RSA and other cryptographic systems.

Critically, quantum annealers like D-Wave machines operate using quantum annealing rather than gate-based quantum computation. Unlike universal quantum computers, which can execute arbitrary quantum algorithms, annealers are designed to solve specific optimization problems and provide approximate solutions. Their reliance on physical annealing processes and noisy qubits limits their ability to implement precise algorithms like Shor's. Achieving practical attacks on RSA encryption using Shor's algorithm would require a scalable, fault-tolerant universal quantum computer. This is a technology that remains out of reach for the foreseeable future due to challenges in qubit stability, error correction, and coherence (see [SS24]).

Moreover, some articles argue against quantum error correction and the feasibility of quantum computers [Dya18; Kal20]. In simple terms, these arguments highlight that describing the state of a quantum computer with thousands of qubits requires tracking an immense amount of information simultaneously. For instance, a useful quantum computer with 1,000 qubits would require processing $2^{1000} \approx 10^{300}$ continuous parameters, far exceeding the number of subatomic particles in the observable universe. Critics argue that maintaining error correction for such a vast number of parameters is fundamentally unmanageable.

Despite these limitations and skepticism, the ongoing advancements in quantum computing [PKG24], which is mainly the result of massive government research agencies' investments in building quantum computers, underscore the urgency of transitioning to post-quantum cryptographic algorithms designed to resist both classical and quantum attacks. These developments are part of a broader effort to future-proof digital security against the eventual realization of quantum computers capable of executing Shor's algorithm and other quantum attacks. While fully operational quantum computers with the ability to compromise widely used cryptographic systems like RSA and ECC have not yet been realized, experts widely anticipate that their development is only a matter of time [Che+16]. Therefore, the adoption of post-quantum cryptography (PQC) is crucial to ensuring the long-term security of data, as encrypted information transmitted today could be intercepted and decrypted in the future when quantum technology becomes viable. This is especially vital for safeguarding sensitive data with long-term confidentiality requirements, such as government communications, healthcare records, and financial transactions.

1.1.1 The Global Effort Toward Post-Quantum Cryptography Standardization

With the increasing threat posed by quantum computing to traditional cryptographic systems, as explained earlier, nations and organizations worldwide have intensified their efforts to develop and standardize PQC. This movement has led to the establishment of dedicated research initiatives, conferences, and strategic investments aimed at securing data against future quantum-based attacks. One of the primary academic platforms for PQC research is the PQCrypto Conference [BL], founded in 2006 by leading cryptographers Daniel J. Bernstein and Tanja Lange. Since its inception, the conference has hosted 16 global events, fostering discussions on developments in quantum-safe cryptographic algorithms.

NIST's Role in PQC Standardization The United States has taken a proactive stance on quantum security, with the National Institute of Standards and Technology (NIST)[1] leading the charge in PQC standardization. Launched in 2017, NIST's initiative invited cryptographers worldwide to submit candidate algorithms, ensuring a rigorous selection process. Many countries, including those in the EU, have been actively involved in this process, contributing to algorithm development and evaluation. Beyond standardization, the U.S. has enacted the Quantum Computing Cybersecurity Preparedness Act, which mandates the transition of federal IT systems to quantum-resistant cryptography by 2035[2].

Europe's Leadership in Quantum Security The European Union (EU) has emerged as a key player in quantum technologies, investing €1 billion over ten years under the EU Quantum Flagship initiative [Beu+21]. This large-scale program unites approximately 2,000 scientists and industry leaders in a collaborative effort to strengthen

[1]https://www.nist.gov/pqcrypto

[2]https://www.nccoe.nist.gov/crypto-agility-considerations-migrating-post-quantum-cryptographic-algorithms

Europe's position in quantum computing and cryptographic security. The European Union Agency for Cybersecurity (ENISA), alongside other EU institutions, has raised concerns about the potential risks posed by quantum attacks. The European Data Protection Supervisor (EDPS) has particularly emphasized the threats to data privacy and security. Several national authorities, including Germany's Federal Office for Information Security (BSI), have been actively evaluating PQC solutions even before the launch of NIST's PQC standardization process [Beu+21].

While the EU actively participates in NIST's PQC standardization efforts, European researchers have also taken independent steps to secure cryptographic transitions.[3] This includes:

- Funding research initiatives, such as an €11 million investment in PQC development.
- Establishing working groups, including the ETSI Quantum-Safe Cryptography Working Group, which focuses on the standardization and practical implementation of PQC algorithms.
- Publishing official recommendations to encourage EU Member States to adopt a unified approach to PQC migration.

Other Countries' Growing Role in PQC Development Some other countries are also actively engaged in the race to develop PQC, often establishing their own independent PQC standardization initiatives.[3] These efforts include:

- South Korea: Established a national PQC research group in 2021[4], launching a competition to select standards for post-quantum public-key encryption and digital signatures. The first round concluded in December 2023, narrowing the selection to eight algorithms.
- China: The Chinese Association for Cryptologic Research (CACR) began a PQC standardization competition in 2018,[5] announcing three selected algorithms as winners in 2020.
- Russia: Created Technical Committee 26, which oversees cryptographic standards.[6] In 2019, it launched Working Group 2.5 – Post-Quantum Cryptographic Mechanisms, developing several PQ algorithms, though none have been formally standardized yet.[3]

In Section 2.4, we delve deeper into the existing PQC approaches in the literature and the underlying mathematical problems. Next, in Section 2.4.8, we discuss the public-key cryptographic schemes that were selected as the winners of the NIST competition and are in the process of being standardized.

1.2 Threat posed to electronic voting systems

Two essential security properties in electronic voting systems are *verifiability* and *privacy*. Verifiability ensures that voters or third parties can independently confirm that the election results are correct and reflect the cast votes, without the need to trust the

[3] https://www.europarl.europa.eu/thinktank/en/document/EPRS_BRI(2024)766237
[4] https://kpqc.or.kr/
[5] https://www.cacrnet.org.cn/site/content/259.html
[6] https://tc26.ru/en/

election authority. Privacy guarantees that the choice of each voter remains confidential and cannot be traced back to them, thereby ensuring anonymity throughout the voting process.

Quantum computing introduces significant threats to both verifiability and privacy. For example, in many e-voting systems, verifiability relies on cryptographic techniques to ensure that votes are counted accurately. If these techniques are not quantum-safe, one possibility is that the voting system could allow a dishonest voter to convince third parties (the public) that her (valid-looking) ballot is well-formed based on the election rules without being detected.

Similarly, privacy could depend on classical encryption schemes (such as in Helios [Adi08], that is a widely used voting system) to keep the votes confidential. A sufficiently powerful quantum computer could use Shor's algorithm to break these encryption schemes, compromising voter anonymity and revealing individual choices. The confidentiality of votes is paramount, as any compromise could undermine the democratic process and erode public trust. As highlighted in the previous section, PQC is essential to mitigate these risks. Preserving both verifiability and privacy against current and future threats is critical for maintaining trust in democratic processes.

Everlasting privacy is another security property in e-voting. It emphasizes the need to protect sensitive data, such as the link between voters and their votes, not only in the short term but indefinitely, even against future technological advances such as quantum computing. This principle ensures that vote privacy remains intact regardless of when adversaries gain access to encrypted data or how powerful their computational capabilities become, including potential new methods of computation beyond quantum computing, which we may not yet be aware of today.

To address the concerns raised by quantum computing in the context of e-voting, our main focus in this thesis is on the design of e-voting systems that can withstand these threats. In addition, we provide security analysis discussions and specify the underlying trust/mathematical assumptions. In the subsequent sections, we outline the key contributions of this thesis and provide an overview of its structure.

1.3 Contributions

This thesis is written in the monograph style, based on three research papers that are explained in the Publications section. Building on these studies, the thesis is self-contained and structured to provide a cohesive and comprehensive treatment of the subject. It integrates all content into a unified document with a consistent narrative and flow. First, I list my publications, co-authored with other collaborators during the course of my doctoral studies, while highlighting my specific contributions to each research paper. This is followed by an overview of my conference presentations and other relevant talks delivered during my PhD at the University of Luxembourg.

1.3.1 Publications

I have the three following publications.

1. **SoK: Secure E-voting with Everlasting Privacy** ([Hai+23])

- Co-authors: Thomas Haines (Australian National University), Johannes Müller (University of Luxembourg), Ivan Pryvalov (University of Luxembourg)

- Published in: Proceedings on Privacy Enhancing Technologies (PoPETs) 2023

- Conference and CORE rank: Privacy Enhancing Technologies Symposium (PETS), A

CONTRIBUTION. We conducted a systematization of knowledge on the concept of everlasting privacy in e-voting which resulted in the following contributions.

- We collected 25 papers in the literature within our scope of research, striving to achieve everlasting privacy (see Chapter 3 for a full list). We studied all these papers in a systematic, critical, and detailed manner.
- We classified these protocols into two different classes: B-ANON and B-ID. The protocols in each class use a different approach to achieve everlasting privacy. We identified that achieving everlasting privacy in B-ANON reduces to utilizing anonymous ballot submission channels, while in B-ID it reduces to the privacy-preserving techniques that are used to tally the ballots. Moreover, we argue that the general approach in B-ID is preferred to the general approach in B-ANON.
- We observed that both classes B-ANON and B-ID contain protocols that offer reasonable solutions to achieve (practical) everlasting privacy under respective assumptions made in that class. We distinguish them based on whether they can handle *simple (yes/no)* or *complex (ranking)* ballots. For simple ballot types, there are two approaches in B-ID. One offers everlasting privacy toward the public, while the other one offers everlasting privacy toward a threshold of talliers. For complex ballot types, there is one secure approach in B-ID and two reasonably secure approaches in B-ANON. Moreover, we identified that all these approaches are efficient for large-scale elections (e.g., [CGG19]).
- Based on the previous contributions, we discovered four open problems in this research line, explained in detail in Chapter 3.

My Specific Contribution. In this work, my primary contribution focused on the B-ANON class. While I contributed to identifying papers that achieve EP in general, my efforts were primarily directed toward analyzing those specific to this class. I reviewed the relevant papers, presented and discussed my findings with my co-authors, and documented my analyses. Additionally, I assisted in refining the scope of the study by identifying and excluding papers that fell outside our research objectives. Furthermore, I contributed to identifying and defining the subclasses of B-ANON, namely B-ANON-A and B-ANON-V, observing that they utilize different approaches to achieve EP.

2. **Direct and Transparent Voter Verification with Everlasting Receipt-Freeness** ([Mos+24])

- Co-authors: Peter Rønne (University of Luxembourg), Peter Y. A. Ryan (University of Luxembourg), Sara Sarfaraz (University of Waterloo)
- Published in: Proceedings of the 9th International Joint Conference on Electronic Voting, 2024, Springer LNCS
- Conference: E-Vote-ID

CONTRIBUTION. In this work, we have the following three contributions.

- We introduced *everlasting receipt-freeness (ERF)*, a novel security property in the field of electronic voting, which we defined for the first time in the literature. ERF ensures that the protocol achieves receipt-freeness even against a computationally unbounded adversary.
- Building on this foundation, we designed an electronic voting protocol, named *Everlasting Hyperion*, that achieves the following features: End-to-End Verifiability (E2E), Everlasting Privacy (EP), and ERF. This indeed maintained the transparent verifiability of Hyperion while achieving additional features. The main changes that we applied to Hyperion was using Perfectly Private Audit Trails (PPAT) while keeping the Hyperion terms (tracking terms) perfectly secret. In addition, we used a new re-randomization algorithm for the published terms in order to shuffle them in parallel.
- Everlasting Hyperion is in the B-ID class, which is the preferred approach to achieving EP, making it among the few protocols in this class that achieve both EP and RF at the same time.
- Broader Coercion Model: We consider adversaries who can interact with voters at any point, even before the voting phase, enabling us to defend against a wider range of coercion scenarios.

My Specific Contribution. In this work, I participated in the exchange of ideas and the development of the protocol. I contributed significantly to writing key sections of the paper, including the detailed descriptions of the cryptographic primitives and the election phases. Additionally, I proofread the entire paper to ensure mathematical accuracy and consistency. While preparing this monograph thesis, I further enriched the content by adding detailed explanations and definitions to the game-based security proofs, transforming them from outline-level sketches into comprehensive and rigorous proofs.

3. **Zero-Knowledge Proofs from Learning Parity with Noise: Optimization, Verification and Application** ([Hai+25]):

   - Co-authors: Thomas Haines (Australian National University), Johannes Müller (University of Luxembourg), Reetika Reetika (Indian Institute of Space and Science Technology)
   - Published in: Proceedings of CSF 2025, IEEE

- Conference and CORE rank: Computer Security Foundations (CSF) Symposium, A

CONTRIBUTION. Our research has advanced the understanding and broadened the applicability of the only conservative code-based zero-knowledge proof (ZKP) framework for arbitrary statements. Building on this foundation, we designed an electronic voting protocol. Specifically, we achieved the following.

- Efficiency Improvements:
  We demonstrated how to optimize the verifier in the code-based ZKP protocol to match the efficiency of the prover. This advancement significantly enhances the practicality of the protocol, particularly for resource-constrained environments.
- Formal Security Verification:
  The security of our optimized ZKP was rigorously verified using EasyCrypt, a tool for formal proof of cryptographic systems. This formal verification adds a strong layer of confidence in the correctness and robustness of our approach. As it was not a contribution of the author of this thesis, we refer the reader to our paper [Hai+25].
- Code-Based ZKP of Shuffle:
  We introduced the first code-based ZKP of shuffle, enabling the secure and efficient verification of shuffling a vector of committed messages. This proof could be a critical building block for various privacy-preserving applications in addition to e-voting systems.
- Code-Based Verifiable E-Voting Protocol:
  Leveraging our findings, we designed the first verifiable e-voting protocol based on code-based cryptography. This protocol ensures public verifiability and vote privacy, marking a significant step forward in the field of secure post-quantum electronic voting.

My Specific Contribution. In this work, my primary contributions focused on the implementation of Jain et al.'s basic ZKP and conducting a comparative analysis of our optimized version against Bellini et al.'s implementation. Moreover, I contributed to finding the attack on Bellini et al.'s implementation, where they failed to commit to the permutation in the first phase of the Sigma protocol. I contributed to the design of the e-voting system, but not to the design of the proof of shuffle or formal verification. While writing the corresponding chapter of the thesis, I provided additional details to make the exposition accessible to readers who may not be familiar with the subject.

1.3.2 Conferences and Talks

In this section, I first mention the talks I delivered at different conferences. Next, I mention the talks I delivered in different venues during the course of my PhD studies.

Conferences

- E-VOTE-ID 2022, October 4-7, Bregenz, Austria:
  I presented the results of [Hai+23] in the PhD colloquium track.

- Privacy Enhancing Technologies Symposium (PETS) 2023, July 10-15, Lausanne, Switzerland:
  I presented [Hai+23] in the main conference.

- E-VOTE-ID 2023, October 3-6, Luxembourg, Luxembourg:
  I presented the general idea behind [Hai+25] in the PhD colloquium track. I won **best presentation award** in the corresponding track.

- E-VOTE-ID 2024, October 2-4, Tarragona, Spain:
  I presented [Mos+24] in the main conference, in the Coercion and Receipt Freeness track.

- Computer Security Foundations (CSF) Symposium 2025, June 16-20, Santa Cruz, CA, USA:
  I plan to present [Hai+25] in the main conference.

Talks

- Verifiable Voting Schemes Workshop, University of Luxembourg, March 21-22, 2023:
  I presented [Hai+23], adjusted for experts in e-voting.

- Crossfyre: International workshop on cryptography, robustness, and provably secure schemes for females (affiliated with Eurocrypt conference), Lyon, 23 April 2023:
  I presented [Hai+23], adjusted for a general audience.

- Science Slam Luxembourg: Organized by LuxDoc and Doctoral Education in Science Communication (DESCOM), Neumünster Abbey Cultural Meeting Center, Luxembourg, 15 Sep 2023:
  In 10 minutes, I presented my PhD topic to the general audience with any background. Based on audience votes, I secured **second place** out of six participants.

- Pesto (French abbreviation for proof techniques for security protocols) Seminar in Loria, Nancy, France, 8 December 2023:
  I was an invited speaker in the PESTO seminar series, where I presented our work on everlasting privacy [Hai+23].

- Flashtalk at SnT Partnership Day, European Convention Center, Luxembourg, 8 May 2024:
  I delivered a 5 minutes talk titled "Making Electronic Voting Systems Secure Against Quantum Computer". I shared my insights with over 500 attendees, including the rector of the University of Luxembourg Prof. Jens Kreisel, the director of SnT Prof. Yves Le Traon, the former director of SnT Prof. Bjorn Ottersten, and two special guests: the minister of higher education and research Stéphanie Obertin, and the president of the European Research Council Maria Leptin.

- University of Basel, Switzerland, 4 March 2025:
  I was an invited speaker at their weekly seminars, which are organized for PhD students to present their research work.

1.4 Thesis Outline

This thesis is organized into 6 chapters as described below.

Chapter 1: Introduction We establish the motivation for this thesis by explaining the problem and why addressing it is important.

Chapter 2: Preliminaries We provide the preliminaries, offering essential background information on both cryptographic primitives and key concepts related to e-voting.

Chapter 3: Secure E-voting with Everlasting Privacy This chapter presents a systematization of knowledge (SoK) on the topic of everlasting privacy in e-voting, addressing the limitations and non-transparent state of the art in this critical domain. Drawing from an extensive study of 25 relevant papers that aim to achieve everlasting privacy, we elucidate the relationships between the proposed protocols in these works. To provide a structured perspective, we classify these protocols into two main categories based on their underlying approaches to attaining this property. Through a detailed analysis, we identify the protocols that offer a reasonable solution under realistic assumptions. Furthermore, we identify the research problems that have been successfully addressed, offering a clear understanding of the progress made in the field. At the same time, we pinpoint significant open problems and gaps that remain unresolved, providing a road-map for future research.

Chapter 4: Direct and Transparent Voter Verification with Everlasting Receipt-Freeness We design a verifiable e-voting system that leverages the most promising approach identified in Chapter 3 to achieve everlasting privacy. Our protocol is based on Hyperion, a transparent e-voting system detailed in the Appendix 6.2, using it as a foundation to enhance its functionality. By utilizing Hyperion's tracking terms, we effectively mitigate coercion while maintaining individual and universal verifiability. We introduce a new feature named everlasting receipt-freeness, a security property that our protocol achieves. To demonstrate the usability of our protocol, we present a voter's perspective, showcasing its simplicity in both coercion and non-coercion scenarios. Additionally, we provide a game-based proof to formally establish the security properties achieved by our system.

Chapter 5: Zero-Knowledge Proofs from Learning Parity with Noise: Optimization, Verification and Application In this chapter, we present the first post-quantum e-voting system leveraging code-based cryptography, a conservative approach that has withstood quantum attacks despite its long history. The core building blocks of our system are commitment schemes and zero-knowledge proof (ZKP) based on the learning parity with noise (LPN) problem proposed by Jain et al. We enhance the efficiency of ZKP provided by Jain et al. and use the result to build the first code-based proof of shuffle. These components are then integrated to develop our complete voting system. We explain a flaw that we found in the only open-source implementation of the ZKP and introduce an attack on it. We have formally verified the security of the (optimized) ZKP for the first time in EasyCrypt in our paper; however, this part is beyond the scope of this thesis and is not a contribution of the author of this thesis. Therefore, it is excluded from this chapter. We refer the interested reader to our paper.

Chapter 6: Conclusion We conclude this thesis by highlighting the key contributions of our research and our studies in this field. Additionally, we summarize the remaining open problems and present them as a road-map for future work.

Chapter 2

Preliminaries

2.1 Notations and Definitions

In this thesis, we presume a foundational understanding of linear and abstract algebra but will revisit key concepts and notation as necessary. Although we aim to maintain consistent notation throughout the thesis, each chapter begins with a brief review of relevant notation to ensure clarity and coherence while maintaining notational consistency across the work.

We denote vectors by bold lowercase letters, such as $\mathbf{x}$, and matrices by bold uppercase letters, such as $\mathbf{A}$. Sets are represented by standard uppercase letters, such as X. The inner product of two vectors $\mathbf{x}$ and $\mathbf{y}$ is denoted by $\langle \mathbf{x}, \mathbf{y} \rangle$. For an integer q, $\mathbb{Z}_q$ represents the set $\{0, 1, \ldots, q-1\}$, while $\mathbb{Z}_q^*$ denotes the set of non-zero elements, $\mathbb{Z}_q \backslash \{0\} = \{1, 2, \ldots, q-1\}$.

Definition 2.1. *(Norm $\|\cdot\|$). Norm is a mathematical function that assigns a strictly positive length to a non-zero vector in a vector space (vector $\mathbf{0}$ has length 0). Let V be a vector space over a field $\mathbb{R}$ (real numbers) or $\mathbb{C}$ (complex numbers). A norm on V is a function $\|\cdot\| : V \to \mathbb{R}$ that satisfies the three following properties.*

1. ***Non-negativity:***

$$\|\mathbf{x}\| \geq 0 \quad \textit{for all } \mathbf{x} \in V, \quad \textit{and } \|\mathbf{x}\| = 0 \textit{ if and only if } \mathbf{x} = \mathbf{0}.$$

2. ***Absolute Homogeneity:***

$$\|\alpha \mathbf{x}\| = |\alpha| \|\mathbf{x}\| \quad \textit{for all } \mathbf{x} \in V \textit{ and scalars } \alpha \in \mathbb{R} \textit{ or } \mathbb{C}.$$

3. ***Triangle Inequality:***

$$\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\| \quad \textit{for all } \mathbf{x}, \mathbf{y} \in V.$$

Example 2.1. *The **general** p-**norm** (ℓ^p-norm) for $p \geq 1$ and $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is defined as follows.*

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}.$$

1. *$p = 1$ (Manhattan or ℓ^1-norm):*

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|.$$

2. *$p = 2$ (Euclidean or ℓ^2-norm)*

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2},$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$.

3. *$p = \infty$ (Maximum or ℓ^∞-norm):*

$$\|\mathbf{x}\|_\infty = \max_i |x_i|.$$

In this thesis, we use the $\ell^2 - norm$ and for simplicity, we remove the subscript 2 and write it as $\|\mathbf{x}\|$.

Definition 2.2. *($dist(\mathbf{x}, \mathbf{y})$). A distance function or metric is a function $dist : X \times X \to \mathbb{R}$ for a set X such that for $\mathbf{x}, \mathbf{y}, \mathbf{z} \in X$ it satisfies the following properties.*

1. ***Non-negativity:***

$$dist(\mathbf{x}, \mathbf{y}) \geq 0, \quad \textit{and } dist(\mathbf{x}, \mathbf{y}) = 0 \textit{ if and only if } \mathbf{x} = \mathbf{y}.$$

2. ***Symmetry:***

$$dist(\mathbf{x},\mathbf{y}) = dist(\mathbf{y},\mathbf{x}).$$

3. ***Triangle Inequality:***

$$dist(\mathbf{x},\mathbf{z}) \leq dist(\mathbf{x},\mathbf{y}) + dist(\mathbf{y},\mathbf{z}).$$

A set X equipped with a metric dist is called a **metric space**, denoted as (X, dist).

Example 2.2. *The Euclidean Distance for vectors $x, y \in \mathbb{R}^n$ is defined as follows.*

$$dist(\mathbf{x},\mathbf{y}) := \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2},$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$.

In this thesis, alongside the Euclidean metric, we also use the Hamming and Rank metrics. While the Hamming metric is a measure of distance between two vectors of equal length, the Rank metric is a measure of distance between matrices, particularly over finite fields. We define both these metrics below.

Definition 2.3. *(Hamming metric). For two vectors $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ of length n, the Hamming distance $d_H(\mathbf{x},\mathbf{y})$ is defined as follows.*

$$d_H(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} \delta(x_i, y_i),$$

where:

$$\delta(x_i, y_i) = \begin{cases} 1 & \text{if } x_i \neq y_i, \\ 0 & \text{if } x_i = y_i. \end{cases}$$

Definition 2.4. *(Rank metric). For two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{F}_q^{m \times n}$, where $\mathbb{F}_q$ is a finite field, the rank distance $d_R(\mathbf{A}, \mathbf{B})$ is defined as follows.*

$$d_R(\mathbf{A}, \mathbf{B}) = rank(\mathbf{A} - \mathbf{B}),$$

where $rank(\mathbf{A} - \mathbf{B})$ is the rank of the matrix $\mathbf{A} - \mathbf{B}$.

The Hamming and Rank metrics satisfy the axioms of a metric: non-negativity, symmetry, and triangle inequality.

A *probability distribution* describes how the probabilities of a random variable are distributed across the possible values it can take. We use χ to denote a general probability distribution. In this thesis, we use Bernoulli and Gaussian probability distributions, which we explain next.

Definition 2.5. *(Bernoulli probability distribution). The probability mass function of a Bernoulli random variable X is given as follows.*

$$Pr(X = x) = \begin{cases} p & \text{if } x = 1, \\ 1 - p & \text{if } x = 0, \end{cases}$$

where p is the probability of success ($X = 1$), and $1 - p$ is the probability of failure ($X = 0$).

We denote this probability distribution by $\mathcal{B}_p$.

Definition 2.6. *(Gaussian or Normal probability distribution). The probability mass function of a Gaussian random variable X is given as follows.*

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),$$

where μ is the mean value (the distribution's center), σ^2 is the variance (the distribution's spread), and σ is the standard deviation.

2.2 Secure E-Voting

The poll tax, also known as capitation, was historically used in many countries, including the United States, Canada, South Africa, Australia, Russia, France (before the French Revolution), and Luxembourg. It was often employed as a method of racial and class-based voter suppression, disenfranchising specific groups of people, particularly those from lower socioeconomic backgrounds. Over time, the poll tax was abolished in various countries at different points in history.

In Luxembourg, a significant turning point came with the Constitutional Review of 1919, which marked the abolition of the poll tax and the introduction of universal suffrage. This reform granted women the right to vote and removed any wealth-based restrictions on voting rights [Dep]. The 1919 reform was a critical step toward establishing a democratic society that recognized the importance of equal voting rights for all citizens, regardless of their financial situation, race, or gender.

Today, voting in Luxembourg is primarily conducted through traditional paper ballots for political and national elections. On 10 July 2024, elections were held to select 16 municipal representatives via a 24-hour online voting platform. The results showed a 60.27% participation rate, with 719 voters casting 5,202 votes [Gov24], marking a step towards embracing e-voting. In paper-based voting, voters are required to be physically present at voting booths to cast their ballots. However, those living abroad or not present in Luxembourg within the voting period have the option of voting by mail, provided they declare their intention to do so in advance.

While paper ballots are generally considered secure, they present certain challenges. For instance, once voters cast their votes, there is no receipt or tracker to verify that their vote was counted correctly, making it impossible for voters to ensure their vote was included in the tally unless they completely trust the election authorities. Moreover, the paper ballots could be marked without the voters noticing, putting ballot privacy in danger. There is always a risk of fraud or ballot tampering, despite the security of the paper system. Furthermore, voters abroad face additional complications with postal voting. A significant number of ballots are often not returned on time, leading to their exclusion from the final tally. This is particularly important in Luxembourg, where the margin of competition per seat can be narrow, and such excluded votes could potentially impact the final election results.

In contrast, electronic voting could help mitigate these issues. E-voting comes in two main forms: voting machines (Electronic Voting Machines, EVM) and internet voting. This thesis focuses on the latter, internet voting, while we still use the e-voting terminology. E-voting systems offer several advantages: they enable fast result computation, minimize the risk of human error in the computation process, and are accessible from anywhere in the world. Moreover, election data can be stored more efficiently. However, e-voting must meet a range of criteria to ensure a secure system that everyone can trust. A secure e-voting system requires a combination of basic and additional properties that work together to create a trustworthy process.

In his groundbreaking 1981 paper on anonymous communication [Cha81], David Chaum was the first to propose the use of cryptography to secure elections. He introduced novel cryptographic methods such as anonymizing mixes, which are foundational to modern e-voting systems. Since then, numerous secure e-voting systems have been proposed and developed, with Chaum's work influencing over 7,000 academic citations on the topic [AM16].

In what follows, we will first explain the various parties involved in a typical e-voting system. Next, we will outline the common phases of an election process, and finally, we will discuss the security properties that are essential for ensuring a secure and reliable e-voting system. For cryptographic primitives, we refer to Section 2.3.

2.2.1 Entities or Parties

Depending on the design and purpose of the e-voting system, it may involve different parties or entities. Based on specific needs, a new party may be introduced or an existing party may take on additional responsibilities. Below are the most commonly cited parties involved in e-voting, which are also referenced throughout this thesis, along with their respective roles.

- Voter (V): In each election, there is a set of eligible voters who wish to cast their ballots. We usually denote them by $\mathsf{V}_1, \ldots, \mathsf{V}_n$, each associated with a unique identification number $\mathsf{ID}_1, \ldots, \mathsf{ID}_n$. We use v_i to denote the vote of V_i and b_i to denote her ballot. Each voter is responsible for preparing their ballot in accordance with the election rules. To do so, voters typically use a voting device, such as a computer or smartphone, to execute the necessary cryptographic primitives. Once the ballot is prepared, it must be securely transmitted to the designated authority or party responsible for collecting and processing the votes.

- Election Authority (EA) or Admin (Adm): EA sometimes referred to as Adm, oversees key aspects of the election process. This includes setting up the election parameters, encompassing both the mathematical parameters required for cryptographic schemes and operational parameters such as the election date, time, and voting rules. In addition to its role in initialization, EA may also be tasked with verifying voter eligibility to ensure that only authorized participants can cast their ballots. Once ballots are submitted, EA may be responsible for receiving them, verifying their correctness, and anonymizing them as needed to preserve voter privacy. Following this, the processed ballots are forwarded securely to the designated tallier for the tally phase.

- Tallier (T): In threshold voting systems, more than one tallier is typically required to distribute trust among them and ensure fault tolerance. These parties collaborate during the tallying phase, which involves executing the vote-counting function $\rho(v_1, \ldots, v_n)$ in accordance with the rules.

- Mix Servers ($\mathsf{M_i}$): In the e-voting systems that use mix-servers, each M_i is responsible for re-randomizing the voters' ballots (either their encryption or commitment), shuffling the results using a secret permutation, and publishing the final shuffled output. Along with the output, the mix server must also provide proof of knowledge of the secret permutation and a proof of shuffle to demonstrate that the operation was performed correctly without altering the contents of the ballots. Depending on the specific voting system, each mix server can handle the shuffled results in one of three ways: it can send the information directly to the next mix server, publish it on a secret board accessible only to mix servers, or post it on a public board where any party can independently verify the correctness of the operation.

- Bulletin Board: In many voting systems, a public bulletin board (PBB) serves as a central, transparent broadcast channel with memory for election-related data [Pet05]. The PBB is designed to be append-only, meaning that once information is published, it cannot be modified or deleted. This immutability ensures the integrity of the election process by making any tampering attempts evident. The PBB is accessible to all parties, including voters, auditors, and observers, enabling them to independently verify the election's correctness and transparency. It typically stores data such as encrypted ballots, cryptographic proofs, and final tallies.
  Some voting systems introduce an additional, more restricted bulletin board known as the secret bulletin board (SBB). The SBB is shared only among specific trusted parties, such as mix servers or talliers. It is indeed used to facilitate intermediate operations that require privacy, such as shuffling and re-randomization of ballots.

To ensure clarity, we will detail the specific parties responsible for operating the election system and their respective roles in subsequent chapters, particularly if they differ from the descriptions provided here.

2.2.2 Election Phases

A typical election has the following phases.

- *Setup Phase*: During this phase, the election authority (EA) or administrator (Adm) generates the election parameters *prm*, which include both cryptographic parameters (e.g., keys and algebraic group settings) and administrative details (e.g., the election's start and end times, rules, and procedures). The public parameters are then published on the PBB to ensure it's accessible to all voters. Additionally, a list of candidates or options is made available, allowing voters to familiarize themselves with their choices prior to casting their ballots.

- *Vote Submission Phase*: In this phase, each voter V_i prepares their ballot following the provided instructions. Typically, this involves either committing to their vote or encrypting it using the election's public key via their voting device. Once the ballot is prepared, the voter sends it to the designated party, such as the PBB or the EA.

- *Tally Phase*: During the tallying phase, the tallier is responsible for counting the ballots in accordance with the rules of the election. There are two main approaches to tallying: homomorphic tallying and mix-net tallying.

  - **Homomorphic Tallying**:
    In this method, the vote-counting function leverages the homomorphic properties of the underlying commitment or encryption scheme. Using these properties, the tallier additively aggregates the encrypted or committed votes into a single value without decrypting or opening individual

ballots. Once the aggregation is complete, the tallier uses the election's secret key to decrypt or aggregate opening values to open the result, revealing the final tally. To ensure that individual votes remain private throughout the aggregation process, the tallier needs to first check that each ballot is independent of the other submitted ballots. This step is crucial in preventing one of the most significant attacks on vote privacy, known as the *replay attack*. In this attack, if the prevention measure is not taken in the tally phase, the adversary could amplify the vote of a target voter through the votes of other corrupted voters by replaying the target voter's vote, resulting in information leakage on the vote of that voter. As shown in [MMR23], this attack poses a serious threat against vote privacy even if the adversary possesses limited resources.

  – **Mix-Net Tallying**:
  In a mix-net tally, the ballots are processed by a series of mix servers. Each server takes the ballots as input, applies a secret permutation (shuffle), and re-randomizes the data to produce an anonymized output vector. When performed securely, this process ensures that the resulting vector cannot be linked back to individual voters, preserving voter anonymity. After the anonymization, the commitments or encryptions in the vector are opened or decrypted, revealing the plaintext votes. These votes can then be used to compute the final election result.

- *Verification Phase*: After the tallying phase, voters and third parties are provided with the opportunity to verify the integrity of the election process. Voters can confirm that their individual votes were accurately included in the final tally, while third parties or observers can verify that the overall election result was computed correctly. The specific procedures for verification depend on the design of the voting system and the type of verification information available to voters. However, the general approach in the literature is to publish encryption of the vote on PBB which could potentially provide both verifiability and privacy (see Section 2.2.4 and 2.2.3). This phase is crucial for ensuring transparency and trust in the election process.

As observed in the typical tally phase, a tallier T holds critical information, including the election secret key and the opening values for commitments. Without additional safeguards, this level of access poses significant risks. A malicious T could tamper with ballots, alter vote counts, or even introduce ballot stuffing (fraudulent ballots) without detection. Such vulnerabilities undermine the integrity and trustworthiness of the e-voting system.

To address these concerns, we outline the fundamental security properties required for a secure e-voting system. These properties ensure that every vote is counted in the final tally, while preventing unauthorized manipulation or breaches of voter privacy. Furthermore, we will later demonstrate how our proposed e-voting systems incorporate countermeasures to mitigate this threat. These measures include but are not limited to cryptographic techniques such as zero-knowledge proofs, verifiable mix-nets, and threshold primitives, which enhance transparency and minimize reliance on a single trusted entity. Moreover, the role of EA or T in some systems is distributed to reduce the risk of fraud. These systems use threshold cryptography such as threshold SSS (see Section 2.3.6) to share the tallier's secret key or threshold ElGamal encryption (see Section 4.4) to threshold share the encrypted secret. This ensures that less than a threshold number of secret holders cannot recover the secret.

2.2.3 Privacy

Ballot privacy, or simply privacy, is a fundamental property of secure e-voting systems. It ensures that a voter's choice remains confidential and that no one should be able to link a specific ballot to its corresponding voter. This requirement implies that no data published during the various voting phases (see Section 2.2.2), including the verification phase, should not leak any information about the vote v_i cast by the voter V_i. To establish and formally prove that ballot privacy is upheld in an e-voting system, researchers have proposed several approaches in the literature, as discussed in [Ber+15]. These include simulation-based models, entropy-based measures, and game-based definitions, as we detail below.

- *Simulation-Based Models*: This model constructs an ideal functionality that serves as a benchmark to simulate a system where ballot privacy is preserved, allowing comparisons with the real system.
- *Entropy-Based Measures*: This method quantifies the amount of information leakage by measuring how much uncertainty remains about a vote after observing the system's output.
- *Game-Based Definitions*: This model defines a series of adversarial game hops where attackers attempt to distinguish between different voting scenarios, ensuring the system resists such attempts.

As an example, in Section 4.7, we adopt the game-based definition to demonstrate that our proposed e-voting system achieves ballot privacy.

A stronger notion of privacy is referred to as everlasting privacy (EP). This notion assumes that the adversary $\mathcal{A}$ has unbounded computational power and guarantees that even with such power, the privacy of the voter's choice is preserved indefinitely. In Chapter 3, we provide an in-depth discussion of EP, presenting a systematic study of its treatment in the literature as part of our systematization of knowledge (SoK) on the subject.

2.2.4 Verifiability

In general, verifiability ensures that all participants, including voters, auditors, and election officials, can verify the correctness of the election process. The ultimate goal of verifiability is to provide transparency, detect any potential manipulation, and build trust. Verifiability enables voters to confirm their vote is accurately reflected in the final election result, a property known as *individual verifiability*. It also allows third parties or observers to ensure that the final result correctly reflects the votes cast in the election, a property known as *universal verifiability*.

Verifiability can be achieved at different stages of the voting process, and the level of verifiability varies depending on the specific mechanism in place. In some systems, voters are provided with mechanisms, such as the Benaloh challenge [Ben06] (see [Est+20] for attacks on Benaloh challenge), BeleniosCaI [Cor+23], the work in [RRS21], MarkPledge[1] [Nef04] to ensure that their vote is cast as they intended. This is referred to as *cast-as-intended (CaI)* or ballot assurance. These mechanisms typically come into play before the voter casts their real ballot, providing a way to verify that their vote is correctly encoded/committed, and transmitted. This approach prevents manipulation during the vote-casting phase, such as a malicious voting device

[1]Unlike Benaloh challenge that the audited ballot cannot be cast, and MarkPledge proves correctness of the actually cast ballot.

altering the voter's choice v_i for the voter V_i. Once the vote is cast, voters can then verify that their ballot has been properly recorded by the system as intended by checking the public bulletin board PBB, ensuring the *recorded-as-cast (RaC)* property. This guarantees that no votes have been lost or altered after being cast. Finally, after the tally phase, verifiability extends to the process of *tallied-as-recorded (TaR)*.

Using cryptographic methods (we will see in detail in Section 2.3) such as providing zero-knowledge proof of correct decryption/opening (in homomorphic tally systems) or proof of shuffle (in mix-net tally systems), voters can confirm that the final tally reflects their recorded vote. These proofs, which are often published on the public bulletin board PBB, ensure that the tallying phase is carried out correctly and that no manipulation has occurred. The cast-as-intended property directly supports individual verifiability by ensuring each voter's choice is accurately captured and represented. On the other hand, the recorded-as-cast and tallied-as-recorded properties support universal verifiability by ensuring the integrity of both the vote transmission and tallying processes.

As with privacy (see Section 2.2.3), various approaches have been proposed in the literature to achieve verifiability and to prove that a system meets these requirements. The first formal definition of verifiability was proposed by Benaloh in 1987 [Ben87], which laid the foundation for much of the modern research in verifiable voting systems. For a comprehensive explanation of the different notions of verifiability in the literature, we refer readers to the work by Cortier et al. [Cor+16].

2.2.5 End-to-End Verifiability

Chaum's work [Cha04] is widely regarded as one of the foundational works in the development of what we now refer to as End-to-End (E2E) verifiability and serves as the groundwork for many E2E verifiable voting systems. From this point, other systems such as Prêt à Voter, introduced in 2005 by Peter Ryan [Rya05; CRS05], and Helios, developed in 2008 by Ben Adida [Adi08], built on this foundation to formalize and advance the concept of end-to-end verifiability. Prêt à Voter introduces a voter-friendly approach to E2E verifiability and Helios applied E2E verifiability to web-based elections, making it one of the first practical implementations for real-world online voting.

End-to-end verifiability is a property of e-voting systems that ensures one can check that not only the correct election result is produced, but also that voters and third parties can independently verify its accuracy. Instead of relying on the trustworthiness of election authorities or software, E2E systems provide solid, transparent evidence that the votes were cast, recorded, and tallied correctly. This allows voters and external auditors to confidently trust the election outcome, even in the presence of potential errors or malicious actors. Therefore, an electronic voting system satisfying CaI, RaC, and TaR achieves E2E verifiability.

2.2.6 Receipt-Freeness (RF)

Vote buying is a significant threat to the integrity of elections, whether conducted through traditional paper-based voting or e-voting systems. In vote buying, a buyer incentivizes voters with money or other benefits in exchange for proof of voting for a specific candidate. This proof could take various forms: in paper-based voting, it might involve a photograph of the marked ballot, while in e-voting, it could include cryptographic-based receipts such as credentials, secret keys, random coins used in cryptographic primitives, or even trackers. Such practices not only violate ballot privacy but also undermine the integrity and fairness of the election process.

The notion of RF, which addresses this issue, was first introduced in [BT94] and formally defined for the first time in [DKR06]. RF ensures that voters cannot generate a verifiable receipt of their vote, thereby preventing them from proving their vote to a coercer or vote-buyer. Achieving RF is crucial in the design of secure e-voting systems, as it ensures that verification terms provided to voters cannot be misused as a receipt for coercion or vote-selling.

In Chapter 4, we introduce the novel notion of everlasting receipt-freeness (ERF). ERF extends the principles of RF by aiming to protect against an adversary with unlimited computational power after the election. We achieve ERF in the e-voting system proposed in that chapter, ensuring that even if an adversary gains access to computational resources far beyond current capabilities, the system remains secure against vote-buying. Additionally, we provide a game-based proof to demonstrate that our system satisfies ERF.

2.2.7 Coercion-Resistance (CR)

A coercer may seek to bribe or intimidate a voter into following specific instructions, such as voting for or against a particular candidate, abstaining from voting, or being forced to cast a ballot regardless of preference. In such scenarios, the coerced voter is typically required to provide evidence of compliance whether by showing a receipt, disclosing cryptographic credentials, or even cooperating with the coercer during the voting process. This threat is especially prevalent in countries with limited democratic freedoms, where coercion can take various forms, including threats of job loss, denial of civil rights or social benefits, or outright violence if voters do not comply. The risk of coercion becomes particularly pronounced in economically vulnerable populations or communities where individuals are dependent on local leaders, who may act as coercers. Therefore, in these cases, it is crucial to design an e-voting system that is coercion-resistance, letting voters follow their own will without fear of punishment.

A coercion-resistant system makes it impossible for the coercer to distinguish whether a voter has complied with their instructions or voted freely. This is achieved by enabling voters to employ a so-called *counter-strategy*, a mechanism that renders the coercer's strategy ineffective. In the literature, coercion resistance is typically achieved through three main approaches: fake credentials, masking techniques, and deniable vote updating [HMQ23]. In Chapter 4, we discuss how to mitigate coercion by designing an electronic voting system providing a strategy for the voters to fake their tracking terms (fake credential approach) in the event of coercion.

Relation between BP, RF, and CR. The relationship between ballot privacy (BP), receipt-freeness (RF), and coercion-resistance (CR) remains a topic of debate in the literature. While no consensus exists, some researchers argue that the informal definitions of these properties, as presented earlier, suggest a hierarchical structure. According to this perspective, RF and CR represent progressively stronger guarantees compared to BP. The rationale is as follows: if BP is violated, a voter could easily provide proof of their vote, thereby breaching RF. Consequently, this would enable a coercer to verify whether the voter followed specific instructions, undermining RF's protection against coercion. Furthermore, CR is considered a stronger property than RF, as it assumes a more powerful adversary capable of actively communicating with the voter to issue voting instructions. This makes CR a more comprehensive security notion, addressing scenarios where the coercer attempts to directly influence the voter, rather than merely seeking post-vote evidence of compliance [DKR06]. In contrast, K"usters et al. argue in [KTV11] that the relationship between these properties is not strictly hierarchical but rather more nuanced. Based on case studies, they demonstrate that achieving one property does not necessarily imply the satisfaction of another. Their work provides both theoretical proofs and practical examples illustrating that BP, RF, and CR can exist independently without a strict implication between them.

For the purpose of this thesis, it is essential to emphasize the distinction between RF and CR. As a reminder, RF ensures that the voting system does not provide any evidence that a voter could use to prove how they voted. In contrast, CR guarantees that the system offers a strategy for the voter to resist coercion and cast their vote according to their own will, even when under external pressure.

Trust Assumptions. In the literature, achieving each of the aforementioned security properties in an e-voting system requires specific trust assumptions tailored to that system. Some systems rely on a single trusted election authority, while others distribute trust among multiple talliers. Certain models account for a malicious BB, whereas others assume BB is trustworthy. It is essential to clarify which security property each election party is trusted to uphold. Ideally, based on the definition of verifiability, we expect this property to be achieved without relying on trust in any party, meaning no trust assumptions should apply in this case, however, this is not the case in reality. For a detailed and extensive example of the underlying trust assumptions in different e-voting schemes to achieve a combination of security properties, see Table 3.2 in Chapter 3.

Last but not least, note that each security property can be achieved in a trivial manner independently. The challenge is to achieve two or more of them simultaneously. Below are examples of voting systems that satisfy only one security property while failing to achieve the others:

- **Privacy:** A system where each voter encrypts their vote with a public key and submits it to the authority, but the authority does not provide any proof of correct decryption or tallying. This ensures privacy but lacks verifiability, receipt-freeness, and coercion-resistance.

- **Verifiability:** A system where each voter's plaintext vote is published on a public bulletin board. This provides perfect verifiability but completely compromises privacy, receipt-freeness, and coercion-resistance.

- **Receipt-Freeness:** A system where the voter casts her ballot in an isolated voting booth, without any information or receipt that could prove their vote to a third party, but there is no way for the voter to check if their vote is correctly included in the tally. This satisfies receipt-freeness but lacks verifiability.

- **Coercion-Resistance:** A system where the voter submits their vote via an anonymous channel without obtaining any receipt or proof of their vote enables the voter to cast their true will secretly, but there is no verifiability mechanism since the published ballot has no information about the voter's identity on it. If the plain vote is submitted, then privacy is endangered too.

2.3 Cryptographic Primitives

In this section, we introduce the cryptographic primitives that we briefly referred to in Section 2.2 and we will use throughout the rest of this thesis. This includes Zero-Knowledge Proofs, Public-Key Encryption (PKE) Schemes, Commitment Schemes (CS), Digital Signatures, and Threshold Shamir Secret Sharing (SSS).

2.3.1 Zero-Knowledge Proofs

Throughout this subsection, we mainly use the content from Goldwasser's 2023 MOOC (Massive Open Online Course) on zero-knowledge proofs [Gol23], adjusting her notation to align with that used in this thesis. Zero-Knowledge Proof (ZKP), first introduced by Goldwasser et al. in [GMR89], is a cryptographic primitive that allows one party (the prover P) to convince another party (the verifier V) that a statement is true i.e. they know a value or possess certain information, without revealing any information beyond the correctness of the statement. Indeed, ZK means that for every V and every true statement, what V can compute after the interaction with P is the same as what he could have computed before the interaction, meaning that the interactions did not reveal any additional information to V.

To mathematically model this concept, Goldwasser et al. defined a *Simulation Paradigm*. Let's define $T := \{a_1, q_1, a_2, q_2, \ldots\}$ as a sequence of answer/question transactions between P and V. Then, we say the verifier V's view gives him zero knowledge about the secret if he could have simulated T on his own without anyone's help, including the prover P. More precisely, the real view and the simulated view are computationally indistinguishable (i.e. is not better than a random guess). Suppose that a language $\mathcal{L}$ is a set of binary instances x (input) such that we want to find a witness w (secret) to satisfy a specific poly-time relation $\mathcal{R} : \mathcal{R}(x, w) = 1$. That is:

$$\mathcal{L} = \{x : \exists\, w \text{ s.t. } \mathcal{R}(x, w) = 1\}.$$

Then, we have the following definition for ZK with an honest V.

Definition 2.7. *An interactive protocol* (P, V) *with security parameter* λ *is honest-verifier zero-knowledge (HVZK) for a language* $\mathcal{L}$ *if there exists a PPT simulation algorithm* Sim *such that for every* $x \in \mathcal{L}$*, the two following probability distributions are indistinguishable by a polynomial time distinguisher over the coin choices of the parties:*

1. $\mathsf{Real}(\mathsf{P}, \mathsf{V})[x]$
2. $\mathsf{Sim}(x, 1^\lambda)$.

To define ZKP for any possible verifier, even a dishonest one indicated by V^*, we have the following definition.

Definition 2.8. *An interactive protocol* (P, V) *with security parameter* λ *is zero-knowledge (ZK) for a language* $\mathcal{L}$ *if for every PPT* V^**, there exists a PPT simulation algorithm* Sim *such that for every* $x \in \mathcal{L}$*, we have the following indistinguishability over the coin choices of the parties:*

$$\mathsf{Real}(\mathsf{P}, \mathsf{V})[x] \approx \mathsf{Sim}(x, 1^\lambda).$$

A zero-knowledge protocol must fulfill the following three properties.

Definition 2.9. (P, V) *is a zero-knowledge interactive protocol if it is:*

1. *Complete: if $x \in \mathcal{L}$,* V *always accepts the proof.*
2. *Sound: if $x \notin \mathcal{L}$,* V *always rejects the proof.*
3. *Zero-knowledge based on Definition 2.8*

By their definition, these three properties address scenarios involving honest P, malicious P, and malicious V, respectively. In the context of ZKP, a proof is different than what we know as a mathematical proof; indeed, a zero-knowledge proof is a probabilistic proof rather than deterministic. That is why in practice, it is necessary to repeat it enough number of times to reduce the chance of a false statement being verified while keeping the zero-knowledge property. This is called the *soundness error*.

Perfect zero-knowledge, which is a stronger assumption, is defined for the case that there exists a Sim such that the real and the simulated views are identical.

There is another definition in the literature named *proof of knowledge* (PoK) where P wants to prove to V that she knows a secret.

Definition 2.10. *An interactive protocol* (P, V) *is a proof of knowledge (PoK) for a language $\mathcal{L}$ if there exists a PPT knowledge extractor algorithm* Ext *such that for every $x \in \mathcal{L}$,* $\mathsf{Ext}^{\mathsf{P}}(x)$ *it outputs w with* $\mathsf{Verify}(x, w) = 1$ *in expected polynomial time.*

The term $\mathsf{Ext}^{\mathsf{P}}(x)$ means that the knowledge extractor algorithm Ext has access to P as an oracle with the input x and may run P repeatedly on the same randomness with different challenges from V. This technique is called *rewinding*. This is the technique that is used all the time to prove something is ZK.

Now there is an interesting question here: do all non-deterministic polynomial (NP) time languages (i.e. the relation $\mathcal{R}$ is poly-time verifiable) have ZK interactive proofs? The following theorem proved in 1986 by Goldreich et al. [GMW86], answers this question with yes. However, it depends on the existence of one-way functions, defined bellow.

Definition 2.11. *A function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ is a one-way function if it satisfies the following properties:*

1. *(Efficient Computability) Given an input x, it is easy (poly-time computable) to compute $f(x)$.*
2. *(Hard to Invert) Given the output $y = f(x)$, without additional information, it is computationally infeasible to find any input x' with $f(x') = y$. This means that it is hard on average to find an inverse.*

The existence of one-way functions is a widely held belief in cryptography, but it has not been proven. If one-way functions exist, then we have P≠NP which is a major unsolved problem in computer science, introduced in 1971 by Stephen Cook [Coo23]. In other words, the existence of one-way functions implies that certain problems in NP cannot be solved in polynomial time, supporting the conjecture that P is not equal to NP.

Theorem 2.1. *If one-way functions exist, then every language $\mathcal{L}$ in NP has computational zero-knowledge interactive proofs.*

A hash function H, of range n, is a one-way function, as defined in Definition 2.11, that takes an arbitrary length message as an input and produces a (fixed-length) output of size n, known as a *hash value*. This output is uniquely determined by the given input, meaning that even a minor change in the input will produce a completely different hash value. There are two types of hash functions: keyed hash function and unkeyed hash function [KL14]. In what follows, we define a keyed hash function.

Definition 2.12. *(Keyed Hash Function). A keyed hash function is a pair of PPT algorithms* (Gen, H)*, with* $H : \mathcal{K} \times \{0,1\}^* \to \{0,1\}^n$*, satisfying the following properties:*

- *The PPT algorithm* Gen *takes as input the security parameter* 1^λ *and outputs a key* $k \in \mathcal{K}$*.*
- *The function* H *takes as input the key* k *and a message of arbitrary length* $m \in \{0,1\}^*$*, and outputs the hash value* $H^k(m) := H(k, m) \in \{0,1\}^n$*, where* n *is dependent on* k *and* λ*.*

The primary requirement of a hash function is collision resistance. A *collision* happens if the hash of two different messages results in the same hash value. Formally, given the key $k \in \mathcal{K}$, a PPT adversary $\mathcal{A}$ outputs two messages m, m'. Then, $\mathcal{A}$ wins the game if and only if $m \neq m'$ and $H^k(m) = H^k(m')$. If the domain of H is greater than its range, then a collision will happen. Therefore, a collision-resistant hash function requires that the probability of an adversary $\mathcal{A}$ winning the game is negligible in the security parameter λ for all PPT adversaries $\mathcal{A}$. For additional properties of hash functions and details on how to construct one, we refer the interested reader to [KL14].

In certain applications, a party may need to prove a statement in zero-knowledge without interacting directly with a verifier and to enable anyone to verify the proof independently. For example, a voter might need to demonstrate that their encrypted ballot contains a valid message, without disclosing the message itself, and then publish the proof on PBB. This is called a *non-interactive zero-knowledge proof (NIZKP)*. In what follows, we first define a Non-Interactive proof system.

Definition 2.13. *(Non-interactive Proof System). Let* $\mathcal{R}$ *be a relation defined on the language* $\mathcal{L}$*. A non-interactive proof system for* $\mathcal{R}$ *consists of a pair of algorithms* $(\mathsf{Gen}, \mathsf{Verify})$*, on the set* $\mathcal{PS}$ *of proof space, such that:*

- *The probabilistic algorithm* Gen *takes the pair of statement and witness* $(x, w) \in \mathcal{R}$ *and outputs a proof* $\pi \in \mathcal{PS}$*.*
- *The deterministic algorithm* Verify *takes a statement-proof pair* (x, π)*, as input and outputs 1 if* π *is a valid proof for* x*, and 0 otherwise.*

A non-interactive proof system should be complete and sound, as defined earlier for the case of interactive proof systems.

The zero-knowledge property for the non-interactive proof system is tricky, since unlike the interactive proof system in Definition 2.8, it is impossible to create a simulator that on the input of $x \in \mathcal{L}$, simulates the real proof. To address this problem, the simulator is granted an "insider advantage" along with access to a hash function H with domain $\mathcal{DM}$ and range $\mathcal{RN}$ as $H : \mathcal{DM} \to \mathcal{RN}$ [BS20]. The simulator algorithm Sim responds to the following two types of queries:

1. Unjustified Proof Query: For the input $x \in X$ for a set X, Sim outputs $\pi \in \mathcal{PS}$.
2. Random Oracle Query: For the input $x \in \mathcal{DM}$, Sim outputs $y \in \mathcal{RN}$.

Now, we can define a non-interactive zero-knowledge (NIZK) proof system.

Definition 2.14. *(NIZK). Let* $\Pi = (\mathsf{Gen}, \mathsf{Verify})$ *be a non-interactive proof system defined in Definition 2.13 for a relation* $\mathcal{R}$ *and proof space* $\mathcal{PS}$*. Let the hash function* H *and the simulator* Sim *be as explained above. We define two real-world and simulated experiments, where* $\mathcal{A}$ *can make the following queries to the challenger:*

1. *Justified Proof Query: For the input* $(x, w) \in \mathcal{R}$*, the challenger outputs* $\pi \in \mathcal{PS}$*.*
2. *Random Oracle Query: For the input* $x \in \mathcal{DM}$*, the challenger outputs* $y \in \mathcal{RN}$*.*

Let $F_{\mathcal{DR}}$ *be the set of all functions with domain* $\mathcal{DM}$ *and range* $\mathcal{RN}$*. The experiments are as follows.*

- *Real-World: The challenger picks a random* $f \in F_{\mathcal{DR}}$ *(i.e., performs as a random oracle), and replies to the two queries as follows:*
    1. *Justified Proof: For query on* $(x, w) \in \mathcal{R}$*, run* $\pi \leftarrow \mathsf{Gen}(x, w)$ *using* f *instead of* H *and output* $\pi \in \mathcal{PS}$*.*
    2. *Random Oracle: For query on* $x \in \mathcal{DM}$*, output* $f(x) \in \mathcal{RN}$*.*
- *Simulated World: The challenger replies to the queries as follows:*
    1. *Justified Proof: For query on* $(x, w) \in \mathcal{R}$*, run* Sim *with unjustified query* x *to output a* $\pi \in \mathcal{PS}$*.*
    2. *Random Oracle: For query on* $x \in \mathcal{DM}$*, run* Sim *with random oracle query on* x *to output a* $y \in \mathcal{RN}$*.*

We say the non-interactive proof system Π *is zero-knowledge if the adversary* $\mathcal{A}$ *can distinguish the real-world and simulated experiments with a negligible probability.*

To transform a ZKP into NIZKP, the common approach is to use the Fiat–Shamir (FS) heuristic [FS86]. This heuristic replaces the verifier's random challenge in an interactive protocol with a deterministic challenge derived from a cryptographic hash function H. The basic idea is as follows: Let a be P's message to V (usually a commitment) and x be the statement of the interactive system. To make it non-interactive, the challenge will be computed as $ch := H^k(a, x)$ by P herself and plugged into her proof, together with the response that she computes based on that. In this case, anyone can confirm the validity of this NIZK proof.

2.3.2 Public-Key Encryption (PKE) Schemes

In the Introduction of Chapter 1 we briefly introduced PKE. Here, we provide more details. An encryption scheme is a mechanism which involves encoding a plaintext (readable data) into a ciphertext (unreadable data) using mathematical algorithms and a pair of public/private key. Only the person holding the secret-key should be able to transform the ciphertext back to its plaintext. More formally, an encryption scheme $\mathcal{E}$ consists of three algorithms $\mathcal{E} = (\mathsf{KeyGen}_{\mathcal{E}}, \mathsf{Enc}, \mathsf{Dec})$, that satisfy the properties in the following definition.

Definition 2.15. *(Public-Key Encryption Scheme). We denote a PKE scheme by three algorithms* $\mathcal{E} = (\mathsf{KeyGen}_{\mathcal{E}}, \mathsf{Enc}, \mathsf{Dec})$*, operating on three sets, the key space* $\mathcal{K}$*, the message space* $\mathcal{M}$*, and the ciphertext space* $\mathcal{C}$*, satisfying the following properties.*

- *The probabilistic key generation algorithm* $\mathsf{KeyGen}_{\mathcal{E}}$ *takes as input the security parameter* 1^{λ} *and outputs a pair of key* (pk, sk) *in the key space* $\mathcal{K}$*.*

- *The probabilistic encryption algorithm* Enc *takes as input a message* m *from the message space* $\mathcal{M}$*, and the public-key* pk*, and outputs a ciphertext* $e \in \mathcal{C}$*:* $e \leftarrow \mathsf{Enc}(pk, m)$*.*
- *The deterministic decryption algorithm* Dec *takes as input a ciphertext* e*, and the secret-key* sk*, and outputs the message* $m' \in \mathcal{M}$*:* $m' \leftarrow \mathsf{Dec}(sk, e)$*.*

A PKE scheme has to satisfy two important properties.

1. *Correctness:* For every plaintext $m \in \mathcal{M}$, and key pairs $(pk, sk) \in \mathcal{K}$ generated by $\mathsf{KeyGen}_{\mathcal{E}}$, it has to satisfy the following relation: $\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m$.
2. *Security*: The basic security notion for PKE is *semantic security*. This notion asserts that, given two plaintexts m_0 and m_1 and a ciphertext e, a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ cannot guess better than a random guess that e is an encryption of which plaintext. Depending on the adversary's capabilities (i.e., adversarial power) such as access to encryption or decryption oracles, or whether it operates passively or actively, stronger security notions for PKE have been defined in the literature. These include IND-CPA, IND-CCA, NM-CPA, and others. For definitions of these security notions, we refer the interested reader to [Bel+98].

The history of encryption goes back to 100 BCE, when Caesar's cipher was used by Julius Caesar to encrypt military-related data. Modern encryption dates back to the 20th century, when the Enigma machine was used by Germany during World War II. The asymmetric (public-key) cryptography was invented by Whitfield Diffie and Martin Hellman in 1976 [DH76], leading to the famous ElGamal encryption scheme [Gam85], which we explain in Example 2.3. The message space in the ElGamal encryption scheme is a (multiplicative) cyclic group $\mathbb{G}$, where *decisional Diffie-Hellman (DDH)* assumption holds. In what follows, we first explain the DDH assumption and then the ElGamal encryption scheme.

Definition 2.16. *(DDH Assumption). Let* $\mathbb{G}$ *be a finite cyclic group of order* q *with generator* g*. DDH assumption asserts that for every* $a, b \in \mathbb{Z}_q^*$ *the probability distribution of the tuples* (g^a, g^b, g^{ab}) *and* (g^a, g^b, g^c) *for* $c \in_R \mathbb{Z}_q^*$ *is computationally indistinguishable in the security parameter* λ*. Loosely speaking,* g^{ab} *looks like a random element in the group* $\mathbb{G}$*.*

If one could efficiently solve the discrete logarithm (DL) problem, they could determine a by computing $\log_g(g^{ab})$. This enables the computation of $(g^b)^a$, making the aforementioned decision easy. This implies that the DDH assumption is stronger than the DL assumption.

ElGamal encryption scheme, explained in the following, has IND-CPA security under the DDH assumption.

Example 2.3. *(ElGamal Encryption). Let* $\mathbb{G}$ *be a cyclic group of prime order* q *with group generator* g*, where the DDH assumption (Definition 2.16) holds. The key generation, encryption, and decryption algorithms for the sets* $\mathcal{M} = \mathbb{G}$ *and* $\mathcal{C} = \mathbb{G}^2$ *work as follows.*

- $\mathsf{KeyGen}_{\mathcal{E}}(1^\lambda) = (pk, sk) := ((\mathbb{G}, q, g, h), x)$*, where* $x \in_R \mathbb{Z}_q^*$ *and* $h := g^x$*.*
- $\mathsf{Enc}(pk, m) = (e_1, e_2) := (g^r, m.h^r) \pmod q$*, where* $r \in_R \mathbb{Z}_q^*$*.*
- $\mathsf{Dec}(x, (e_1, e_2)) = (e_2.(e_1^x)^{-1}) \pmod q$*.*

The original version of ElGamal is multiplicative, meaning the encryption algorithm satisfies the property $\mathsf{Enc}(m_1) \times \mathsf{Enc}(m_2) = \mathsf{Enc}(m_1 \times m_2)$. In contrast, the additive version encodes the plaintext m in the exponent of the second component of the ciphertext rather than as a coefficient. Specifically, the encryption function is defined as $\mathsf{Enc}(pk, m) := (g^r, g^m h^r)$, such that it satisfies the *additively homomorphic* property: $\mathsf{Enc}(m_1) \times \mathsf{Enc}(m_2) = \mathsf{Enc}(m_1 + m_2)$

2.3.3 Commitment Schemes (CS)

A commitment scheme allows a party to securely lock a value, called the message, in a commitment, ensuring two essential properties: *hiding*, which guarantees that the commitment conceals the message until it is revealed, and *binding*, which ensures that the committed value cannot be altered after the commitment is made. These two properties of CS can be compared to an envelope containing a letter: it conceals the letter until the envelope is opened, and once the letter is placed inside and sealed, its contents cannot be altered. Formally, a CS is defined by three algorithms $\mathcal{CS} = (\mathsf{KeyGen}_\mathcal{C}, \mathsf{Com}, \mathsf{Verify}_\mathcal{C})$, such that they satisfy the properties in the following definition.

Definition 2.17. *(Commitment Scheme). We denote a commitment scheme CS by three algorithms* $\mathcal{CS} = (\mathsf{KeyGen}_\mathcal{C}, \mathsf{Com}, \mathsf{Verify}_\mathcal{C})$*, such that they satisfy the following properties.*

- *The key generation algorithm* $\mathsf{KeyGen}_\mathcal{C}$ *takes as input* 1^λ *and outputs a set of public parameters prm, including the message space* $\mathcal{M}$*, the commitment space* $\mathcal{C}$*, the opening space* $\mathcal{O}$*, and more.*

- *The probabilistic commitment algorithm* Com *takes as input the public parameters* $prm \leftarrow \mathsf{KeyGen}_\mathcal{C}$ *and a message* $m \in \mathcal{M}$*, and outputs a commitment/opening pair* (c, d)*:* $(c, d) \leftarrow \mathsf{Com}(prm, m) \in \mathcal{C} \times \mathcal{O}$*.*

- *The verification algorithm* $\mathsf{Verify}_\mathcal{C}$ *takes as input* $prm \leftarrow \mathsf{KeyGen}_\mathcal{C}$*, a message* $m \in \mathcal{M}$*, a commitment* $c \in \mathcal{C}$ *and an opening value* $d \in \mathcal{O}$*, and outputs a binary b of value 1 (valid) or 0 (invalid):* $b \leftarrow \mathsf{Verify}_\mathcal{C}(prm, c, m, d)$*.*

A commitment scheme as $\mathcal{CS} = (\mathsf{KeyGen}_\mathcal{C}, \mathsf{Com}, \mathsf{Verify}_\mathcal{C})$ has to satisfy three important properties:

1. Correctness: $\mathcal{CS}$ is correct if for all messages $m \in \mathcal{M}$, let $prm \leftarrow \mathsf{KeyGen}_\mathcal{C}$ and $(c, d) \leftarrow \mathsf{Com}(prm, m)$, then: $\mathsf{Verify}_\mathcal{C}(prm, c, m, d) = 1$.

2. Hiding: Consider an experiment between a PPT adversary $\mathcal{A}$ and the challenger. The challenger sends $prm \leftarrow \mathsf{KeyGen}_\mathcal{C}$ to $\mathcal{A}$. Then, adversary chooses two messages $(m_0, m_1) \leftarrow \mathcal{A}(prm, 1^\lambda)$ and sends them to the challenger. The challenger chooses a random bit $b \in_R \{0, 1\}$ and commits to m_b as $(c, d) = \mathsf{Com}(prm, m_b)$ and sends the commitment c to $\mathcal{A}$. Finally, the adversary has to output b^*, which is his guess of which message was committed. Adversary wins if $b^* = b$, and the advantage of the adversary is defined as $\mathrm{Adv}_\mathcal{A} = |\Pr[b = b^*] - \frac{1}{2}|$. The $\mathcal{CS}$ provides hiding if $\mathrm{Adv}_\mathcal{A}$ is negligible in terms of the security parameter λ.

3. Binding: A $\mathcal{CS}$ provides binding if for every PPT adversary $\mathcal{A}$, the probability of finding two pairs (m_0, r_0) and (m_1, r_1) such that $\mathsf{Com}(prm, m_0; r_0) = \mathsf{Com}(prm, m_1; r_1)$ with $m_0 \neq m_1$ is negligible in terms of the security parameter λ.

There are two types of CS:

1. Perfectly (Unconditionally) Binding, Computationally hiding: Hiding holds against computationally bounded adversaries, while binding is absolute.
2. Perfectly (Unconditionally) Hiding, Computationally Binding: Binding holds against computationally bounded adversaries, while hiding is absolute.

The existence of only the two aforementioned types of commitment schemes stems from the inherent impossibility of achieving both hiding and binding properties unconditionally. Perfect hiding implies that the commitment could correspond to multiple possible messages (because the adversary cannot tell which one was committed). In contrast, perfect binding implies that the commitment corresponds to only one possible message. Therefore, a commitment scheme cannot be both perfectly hiding and perfectly binding simultaneously.

Next, we explain one of the most well-known commitment schemes, proposed by Pedersen in 1991 [Ped91].

Example 2.4. *(Pedersen Commitment). Let $\mathbb{G}$ be a cyclic group of prime order q, and let $g, h \in \mathbb{G}$ be group generators where the discrete logarithm of h with respect to g is unknown. Then, the Pedersen CS is defined by the following algorithms.*

- $\mathsf{KeyGen}_{\mathcal{C}}(1^\lambda) = (\mathbb{G}, q, g, h)$.
- $\mathsf{Com}(prm, m) = (g^m h^r \bmod q, r) := (c, d)$, *where* $r \in \mathbb{Z}_q^*$.
- $\mathsf{Verify}_{\mathcal{C}}(c', m', d') = (c' == g^{m'} h^{d'} \bmod q)$.

Pedersen commitment, explained in Example 2.4, is a well-known example in the second type of CS: it is perfectly hiding, and computationally binding under the DL assumption.

We would like to note here that as we explained in the Introduction of Chapter 1, PKE cannot attain perfect or unconditional secrecy. This means that if in an e-voting system, the votes of the voters are encrypted using PKE and published on the PBB, vote privacy will be computational. On the other hand, if the commitments of the votes are published on the PBB using a CS of type 2 with perfectly hiding property, vote privacy will be unconditional or everlasting.

Moreover, certain e-voting protocols utilize *homomorphic* commitment schemes. In these protocols, individual commitments $c_i = \mathsf{Com}(m_i; r_i)$ can be combined, without knowing the opening values $(m_i; r_i)$, to form a new commitment c as their aggregation: $c = \mathsf{Com}(\sum_i m_i; \sum_i r_i)$. Pedersen CS, defined in Example 2.4, is a widely used and popular homomorphic CS in the literature with applications in e-voting, mainly to achieve everlasting privacy.

2.3.4 Digital Signature

To prove eligibility in paper-based voting, voters need to prove their identities to the election authorities by showing their ID card or passport. Equivalently, to do the same in e-voting, in some systems, voters use a digital signature scheme to show that the ballot they cast is from an eligible voter and also that the ballot has not been altered in transit. In addition, in some e-voting systems, a digital signature scheme is used by the election authorities to ensure the integrity and authenticity of their message.

Formally, a digital signature scheme is typically defined by three algorithms $\mathcal{DS} = (\mathsf{KeyGen}_{\mathcal{S}},\ \mathsf{Sign}, \mathsf{Verify}_{\mathcal{S}})$, where they satisfy the properties in the following definition.

Definition 2.18. *(Digital Signature). We denote a digital signature scheme by three algorithms $\mathcal{DS} = (\mathsf{KeyGen}_{\mathcal{S}}, \mathsf{Sign}, \mathsf{Verify}_{\mathcal{S}})$, where they satisfy the following properties.*

1. *The key generation algorithm $\mathsf{KeyGen}_{\mathcal{S}}$ takes as input 1^{λ} and outputs a pair of signing/verifying key (sk, pk). We sometimes denote this pair by (ssk, vk), which stands for secret signing key and verification key, respectively.*

2. *The signature on message m with secret-key sk is computed as $\sigma = \mathsf{Sign}_{\mathcal{S}}(sk, m)$.*

3. *The verification of signature σ on message m using the public-key pk is performed as $b := \mathsf{Verify}_{\mathcal{S}}(pk, m, \sigma)$ where b is a bit. If the signature is valid, $b = 1$; otherwise, $b = 0$.*

A digital signature scheme must satisfy two primary properties.

- *Correctness*: A valid signature created by an honest signer on a message m with the signer's secret-key can be verified as valid using the signer's public key.

- *Unforgeability*: It must be computationally infeasible for an adversary to generate a valid signature (forge a signature) for any message that was not signed by the legitimate signer, without the knowledge of sk.

Example 2.5. *One of the most famous digital signature schemes is RSA, invented by Rivest, Shamir, and Adleman, which works as follows.*

1. $\mathsf{KeyGen}_{RSA}(1^{\lambda}) = ((n, e), (n, d))$, *with $n = p \cdot q$ where p and q are large prime numbers, e be the public exponent, and d be the private exponent such that $d \cdot e \equiv 1 \pmod{\phi(n)}$ for $\phi(n) = (p-1) \cdot (q-1)$.*

2. $\mathsf{Sign}(m, (n, d)) = m^d \pmod{n} := \sigma$.

3. $\mathsf{Verify}_{RSA}((n, e), m', \sigma) = 1$ *iff* $\sigma^e = m'$.

2.3.5 Bilinear Pairing and Elliptic Curves

Pairing-based cryptography uses pairing functions to map pairs of points on groups (or elliptic curves) into a finite field. This gives the advantage of making efficient calculations in the target group $\mathbb{G}_T$ while maintaining security in the source groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

Definition 2.19. *(General Bilinear Pairing). Let $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$ be additive and $(\mathbb{G}_T, \cdot)$ be multiplicative cyclic groups of (prime) order q. A bilinear pairing map is a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that satisfies two properties.*

1. *Nondegeneracy: for all $g_2 \in \mathbb{G}_2$, $\hat{e}(g_1, g_2) = 1_{\mathbb{G}_T}$ iff $g_1 = 1_{\mathbb{G}_1}$. Similarly, for all $g_1 \in \mathbb{G}_1$, $\hat{e}(g_1, g_2) = 1$ iff $g_2 = 1_{\mathbb{G}_2}$.*

2. *Bilinearity: for all $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}$: $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$.*

According to [Lyn07], $\mathbb{G}_2$ does not necessarily need to be cyclic and q does not need to be prime.

Depending on the application, we may consider hardness assumptions in a combination of the source groups. This gives flexibility and has a wider application compared to when a pairing is not used (see Section 4.4). However, at the same time, it opens doors for attacks such as sub-exponential index calculus attacks. Therefore, careful parameter selection for the target group is necessary.

Pairing can be classified into three types:

1. Symmetric pairing: $\mathbb{G}_1 = \mathbb{G}_2$.
2. There is an efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$, where $\mathbb{G}_1 \neq \mathbb{G}_2$.
3. There is no efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$, where $\mathbb{G}_1 \neq \mathbb{G}_2$.

The source groups are realized with elliptic curves, which we define next.

Definition 2.20. *(Elliptic Curve). An elliptic curve is a curve over a finite field $\mathbb{F}_q$, where its points (x, y) satisfy the equation $y^2 = x^3 + ax + b$ for coefficients $a, b \in \mathbb{F}_q$, as well as the point at infinity ∞.*

To clarify the pairing-based application in e-voting, we refer to an example from [CPP13]. Using the elliptic curve $y^2 = x^3 + b(\text{with} a = 0)$, a voter's choice, represented by y, can be encoded in $\mathbb{Z}_q$ and mapped to the point $(x, y) = ((y^2 - b)^{\frac{1}{3}}, y)$.

2.3.6 Threshold Shamir Secret Sharing (SSS)

Shamir's Secret Sharing (SSS), introduced by Adi Shamir in 1979 [Sha79], is a cryptographic method designed to securely distribute a secret (e.g., a secret key or a confidential message) among a group of participants, known as shareholders. The SSS algorithm mathematically divides the secret into shares, corresponding to the number of shareholders. A predetermined threshold number t of these shareholders must collaborate to reconstruct the secret; fewer than the threshold are unable to do so. The SSS is based on Lagrange polynomial interpolation to reconstruct the secret based on a threshold of shares. A trusted authority or dealer is needed to set up the scheme. Formally, the threshold SSS algorithm works as follows.

Definition 2.21. *(Threshold Shamir Secret Sharing). The $(t, k)-$threshold SSS algorithm works as follows. Let s be the secret, t the threshold, and k the total number of shareholders such that $t \leq k$.*

1. *Polynomial Construction: Dealer chooses random coefficients $a_1, \cdots, a_{t-1} \in \mathbb{F}_q$. Compute the polynomial $p(x)$ of degree $t-1$ as $p(x) = s + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1}$*
2. *Create Shares: Dealer chooses k distinct non-zero points $x_1, \ldots, x_k \in \mathbb{F}_q$ and evaluates $p(x)$ on these points. Then she distributes the shares $\{(x_i, p(x_i))\}_{i=1}^{k}$ to the corresponding shareholders.*
3. *Secret Reconstruction: let $\{(x_i, p(x_i))\}_{i=1}^{t}$ be a set of t distinct shares. We use the Lagrange interpolation formula as follows: $p(x) = \sum_{i=1}^{t} p(x_i) \lambda_{x,i}^{S_t}$ where $S_t := \{1, \ldots, t\}$ and $\lambda_{x,i}^{S_t}$ is the Lagrange coefficient. The secret s is equal to $p(0)$; therefore, we need to compute $\lambda_{x,i}^{S_t}$ for $x = 0$ which is equal to $\prod_{j \in S_t \setminus \{i\}} \frac{j}{j-i}$. Now, using the full Lagrange formula, we can reconstruct the secret.*

2.4 Post-Quantum Security

As explained in detail in Chapter 1, Post-Quantum Cryptography (PQC) is an emerging field of critical importance in our modern era with the main concern known in the literature as "store now, decrypt later". PQC encompasses a diverse range of approaches aimed at providing security for public-key cryptography against the advanced computational capabilities of quantum computers. In the following, we outline a brief explanation of the mathematical problems that serve as the foundation of each approach, accompanied by the most prominent examples from the field. These problems are widely regarded as resistant to quantum attacks under current knowledge and assumptions (no full cryptanalysis yet, just partial cryptanalysis), forming the backbone of quantum-safe cryptographic protocols.

2.4.1 Lattice-Based Cryptography (LBC)

A lattice is a mathematical structure that can be visualized as a discrete set of points (e.g., a grid in 2D visualization) in $m-$dimensional Euclidean space.[2] More formally, a lattice L of rank n is defined by a (non-unique) basis of n linearly independent *basis vectors* $B := \{b_1, \ldots, b_n\} \in \mathbb{R}^m$, such that any vector $y \in L$ can be expressed as an integer linear combination of the basis vectors b_i. This can be written as follows.

$$L(B) := \{y = \textstyle\sum_{i=1}^{n} c_i b_i | \forall i : c_i \in \mathbb{Z}\}.$$

Some problems in lattices can be solved in polynomial time and are therefore not secure enough in the field of PQC: testing equality of lattices and intersecting two lattices. The algorithms to solve these two problems are simple and based on the Hermite Normal Form (HNF) of the basis matrix. One solution can be seen in [Bar22]. However, the most important problems in lattices that are *believed* to be resistant against both classical and quantum attacks, when $n = m$ is large, [3] are as follows. For simplicity, we describe the search version of these three problems:

1. Shortest Vector Problem (SVP): The objective is to find the shortest non-zero vector $x \in L$, where the vector norm of x is minimized with respect to the corresponding norm. More formally, find $x \in L$ such that $||x|| = \min_{y \in L \setminus \{0\}} ||y||$.

2. Closest Vector Problem (CVP): This problem involves finding the lattice point closest, i.e., the shortest vector distance, to a given target point t, which does not necessarily belong to the lattice L. More formally, given a target point $t \notin L$, find $x \in L$ such that $||x - t|| = \min_{y \in L} ||y - t||$.

3. Learning With Errors (LWE): This problem was introduced by Regev in 2005 in his groundbreaking paper [Reg05], which was awarded the Gödel prize in 2018 for its significant impact on theoretical computer science. [4] At a high-level, the objective of LWE is to recover a secret in a noisy system of linear equations. In the absence of noise, the solution of a system of linear equations can be efficiently found by the Gaussian elimination algorithm in $O(n^3)$ time for an $n \times n$ system. However, the presence of noise makes it more complex.

[2]Euclidean norm is the usual norm used in the literature; however, in the lattice applications, other non-Euclidean norms such as ℓ_1 and ℓ_∞ are used.

[3]Unfortunately, public-key cryptanalysis, especially quantum cryptanalysis, has relatively few active researchers. Globally, only about 20 experts specialize in quantum cryptanalysis for lattice-based cryptography [Hen24].

[4]Gödel award recognizes his major contributions to mathematical logic and the foundations of computer science: https://eatcs.org/.

More formally, consider an error distribution χ on $\mathbb{Z}_q$ for a positive integer (not necessarily prime) number q. Let $A \in \mathbb{Z}_q^{m \times n}$ be a uniformly random matrix, $x \in \mathbb{Z}_q^n$ be a uniformly random secret, and $\mathbf{e} = (e_1, \ldots, e_m)$ be an error vector such that each e_i is sampled independently from χ. The LWE problem is to find the secret x given A and the noisy vector $y := Ax + e \bmod q$. It is considered related to lattices because it can be viewed as a noisy version of CVP, where the lattice L is generated by the matrix A. If q is too large compared to n, LWE is not a hard problem since the famous Lenstra–Lenstra–Lovász (LLL) classical algorithm from 1982 [LLL82] can find x in polynomial time $O(n^4)$. The Block-wise Korkine-Zolotarev (BKZ) [Sch87] classical algorithm can also solve LWE more efficiently than LLL for large q, as demonstrated in [Che+20].

Various results are known comparing the complexity of the three problems. For example, CVP is hard if LWE is hard. When comparing the complexities of SVP and CVP, it is evident that CVP is at least as difficult as SVP. Furthermore, the decision variant of CVP is NP-complete in ℓ_2-norm. Consequently, solving CVP in deterministic polynomial time is not feasible unless P=NP [MG02]. Moreover, CVP and SVP are NP-hard in ℓ_p-norm and ℓ_∞-norm, respectively. LWE is also conjectured to be hard and hence suitable approach in PQC [Reg09]. The complexity comparison of LWE with SVP and CVP can also be found in the provided citation.

2.4.2 Code-Based Cryptography (CBC)

Code-based cryptography derives its security from the computational difficulty of hard problems in algebraic coding theory, particularly the prominent NP-complete problem of decoding random linear codes [BMV78]. A linear code, often denoted as an $[n, k, d]$-code $\mathcal{C}$ for $0 < k \leq n$ is a linear subspace of the vector space $\mathbb{F}_q^n$,[5] where: n is the length of the *codewords*, k is the dimension of the subspace, and d is the minimum distance (see Section 2.1) between two codewords measured using a chosen metric such as Hamming, Rank, or Lee metric [Rot06] (the later is less common.).

Definition 2.22. *(Dual Code). The dual code of the code $\mathcal{C}$ is defined as $\mathcal{C}^* := \{\mathbf{c}^* \in \mathbb{F}_q^n : \forall \mathbf{c} \in \mathcal{C}, \langle \mathbf{c}, \mathbf{c}^* \rangle = 0\}$, where $\langle \mathbf{c}, \mathbf{c}^* \rangle \in \mathbb{F}_q$ is the inner product of c and c*.*

A $[n, k, d]$-code can be represented in two ways:

1. Using a generator or a basis matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$, which its rows shape a basis for the code $\mathcal{C}$. The code is represented as:

$$\mathcal{C} := \{\mathbf{mG} : \mathbf{m} \in \mathbb{F}_q^k\},$$

meaning that each codeword is a linear combination of the rows of $\mathbf{G}$.

2. Alternatively, using a parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k) \times n}$, which its rows shape a basis for the dual code $\mathcal{C}^*$. The code is represented as:

$$\mathcal{C} := \{\mathbf{m} \in \mathbb{F}_q^n : \mathbf{Hm}^T = \mathbf{0}\},$$

with $\mathbf{m}^T$ being the vector transpose of $\mathbf{m}$. For any vector $\mathbf{y} \in \mathbb{F}_q^n$, the vector $\mathbf{yH}^T$ is called a *syndrome*.

[5]For the prime number $q = 2$, the code is called *binary* linear code.

Using these definitions, the *decoding random linear codes* problem can be defined in the two following ways, which are equivalent. The difference is how to represent the codewords.

1. Noisy codeword decoding: given a random generator matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$, an error weight $w \in [1, n)$, and a noisy codeword $\mathbf{y} \in \mathbb{F}_q^n$ with $\mathbf{y} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c} = \mathbf{mG}$ for some $\mathbf{m} \in \mathbb{F}_q^k$, and $\mathbf{e} \in \mathbb{F}_q^n$, find the error vector $\mathbf{e}$ with $\|\mathbf{e}\| \leq w$.

2. Syndrome decoding: given a random parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k) \times n}$, an error weight $w \in [1, n)$, and a syndrome vector $\mathbf{s} \in \mathbb{F}_q^{n-k}$ with $\mathbf{He}^T = \mathbf{s}^T$, for some $\mathbf{e} \in \mathbb{F}_q^n$, find the error vector $\mathbf{e}$ with $\|\mathbf{e}\| \leq w$.

Both of these equivalent problems have been utilized in the CBC literature to construct quantum-safe schemes. In 1978, Robert McEliece introduced the McEliece encryption scheme that relies on the hardness of decoding random linear codes [McE78]. This encryption scheme uses Goppa code [Ber73], which is a binary linear error-correcting code. More precisely, let $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ be the generator matrix of the Goppa code. To compute the public key $\mathbf{G}'$, we disguise $\mathbf{G}$ with two random matrices $\mathbf{S} \in \mathbb{F}_2^{k \times k}$ (scrambler) and $\mathbf{P} \in \mathbb{F}_2^{n \times n}$ (permuter) as $\mathbf{G}' = \mathbf{S} \cdot \mathbf{G} \cdot \mathbf{P}$. In fact, the secret key consists of the Goppa code (e.g., the decoding algorithm) and the matrices $(\mathbf{G}, \mathbf{S}, \mathbf{P})$. The encryption and decryption algorithms work as follows:

- Encryption: to encrypt a message $\mathbf{m} \in \mathbb{F}_2^k$, we compute the codeword $\mathbf{c} := \mathbf{mG}' \oplus \mathbf{e}$ for a randomly chosen error vector $\mathbf{e} \in \mathbb{F}_q^n$ with small weight. [6]
- Decryption: to decrypt a ciphertext $\mathbf{c}$, we take the following steps:
  1. Multiply $\mathbf{c}$ by the inverse permutation: $\mathbf{c}' := \mathbf{cP}^{-1} = \mathbf{mSG} \oplus \mathbf{eP}^{-1}$. Since the error here is just permuted, it has the same weight as $\mathbf{e}$.
  2. Use the decoding algorithm of the Goppa code to recover $\mathbf{mS}$.
  3. Remove scrambling to recover the original message: $(\mathbf{mS})\mathbf{S}^{-1} = \mathbf{m}$

The main drawback of this scheme lies in its large key size, a common challenge among many primitives in code-based cryptography. Despite this, the underlying Goppa code offers strong resistance to structural attacks that aim to exploit specific patterns or weaknesses in the code structure. However, the scheme remains vulnerable to other types of attacks, such as information set decoding (ISD) and improved Stern's attack [BLP08], which continues to see algorithmic advancements, necessitating careful parameter selection to maintain security levels, but the underlying problem is still quantum-resistant.

Another famous hard problem in algebraic coding theory is *Learning Parity with Noise (LPN)*. This problem was first introduced in the paper by Alekhnovich in 2003 [Ale03]. In this work, Alekhnovich studied the hardness of decoding random linear codes in the presence of noise. This contribution led to the groundwork for the LPN problem. Formally, LPN problem is defined as follows: let $\mathbf{A} \in \mathbb{Z}_2^{m \times n}$ be a uniformly random matrix, $\mathbf{s} \in \mathbb{Z}_2^n$ be a secret, and $\mathbf{e} \in \mathcal{B}_\mu^m$ be the noise sampled uniformly at random with respect to $\mathcal{B}_\mu$, the Bernoulli distribution (see Section 2.1) with success probability $\mu \in (0, \frac{1}{2})$. The goal is to find the secret $\mathbf{s}$ given $\mathbf{A}$ and the noisy system $\mathbf{b} := \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod 2$.

The security of the LPN problem is equivalent to the hardness of decoding random linear codes problem, which is considered a conservative assumption in PQC

[6]The weight of error vector needs to be within the error-correcting capacity of the Goppa code.

due to the lack of efficient classical or quantum algorithms capable of solving the problem in polynomial time. As such, the LPN problem is a robust candidate for quantum-resistant cryptographic systems.

Two additional codes are worth mentioning, as they will be referenced later in this thesis: Hamming Quasi-Cyclic (HQC) and Bit Flipping Key Encapsulation (BIKE). HQC is a public-key encryption scheme based on the hardness of decoding random quasi-cyclic codes in the Hamming metric [Agu+18], and BIKE utilizes bit-flipping techniques for encryption and key exchange, relying on the difficulty of solving specific decoding problems [Ara+22].

2.4.3 Isogeny-Based Cryptography (IBC)

Isogeny-based cryptography originated with the work of Rostovtsev and Stolbunov in 2006 [RS06a]. They constructed ElGamal public-key encryption and Diffie-Hellman key exchange schemes based on isogenies. An isogeny, also known as an algebraic homomorphism of elliptic curve groups, is a map φ defined over a finite field $\mathbb{F}_q$ between two elliptic curves such that the group operation (e.g., addition) is maintained. More formally, $\varphi := E_1(\mathbb{F}_q) \to E_2(\mathbb{F}_q)$ is a *rational* map[7] where for two points $P, Q \in E_1(\mathbb{F}_q)$, it satisfies $\varphi(P + Q) = \varphi(P) + \varphi(Q)$.

IBC cryptosystems rely on the properties of isogeny graphs of elliptic curves (see Section 2.3), such as the hardness of finding an isogeny (or morphism) between elliptic curves over finite fields. In general, elliptic curves are classified into two groups: ordinary and supersingular.

The ordinary elliptic curves, used in the former constructions, are vulnerable to efficient attacks that leverage the commutative structure of ordinary isogenies, which makes the security of these two constructions weak.

In contrast, quantum attacks against construction based on supersingular curves are supposed to still be exponential since the underlying endomorphism ring is non-commutative, as outlined in [JD11]. Although quantum attacks on supersingular-based schemes are theoretically exponential, isogeny-based cryptography is a relatively young field compared to lattice- and code-based approaches. Its long-term security still requires thorough cryptanalysis. For more information on the mathematics of IBC, we refer interested readers to [De 17].

2.4.4 Hash-Based Cryptography (HBC)

HBC relies on the security of hash functions. We defined hash functions earlier in Definition 2.12. Two famous hash functions used in the literature are MD5 (Message Digest Algorithm 5) [Riv92] and a family of SHA (Secure Hash Algorithms) [SN93; SN95]. Security of one-way functions is, in general, not susceptible to Shor's algorithm; however, the Grover search algorithm can quadratically reduce the security, e.g. for $n = 256$, an ideal HBC would offer 128-bit security against Grover.

The hash-based cryptography submissions to the NIST competition were both digital signature schemes: Gravity-SPHINCS[8] and SPHINCS+ [Ber+19]. The most famous hash-based signature scheme is the Merkle Signature Scheme (MSS), which extends the one-time signature (OTS) of Lamport [Lam79] using a Merkle tree to allow multiple signatures [Mer79]. To avoid attacks against MSS, it is important not to use the same OTS in the tree twice.

[7]A rational map φ is expressed as a pair of polynomial fraction functions $(\varphi_x(x, y), \varphi_y(x, y))$ where the index show the effect of the map φ on the corresponding coordinate x or y.

[8]https://github.com/gravity-postquantum/gravity-sphincs

2.4.5 Multivariate-Based Cryptography (MBC)

MBC[9] relies on the NP-complete problem of solving systems of multivariate polynomial equations over a finite field $\mathbb{F}_q$ [Har82]. The hardness of this problem, related to algebraic geometry, forms the basis for the design of two cryptographic primitives such as public-key encryption and digital signature scheme [DY09]. More formally, for the case of an encryption scheme, the public-key is represented by a set of m polynomials in n variables:

$$G(x_1, \dots, x_n) := (\mathrm{poly}_1(x_1, \dots, x_n), \dots, \mathrm{poly}_m(x_1, \dots, x_n)),$$

where each poly_i is a non-linear polynomial (usually quadratic) in $\mathbf{x} := (x_1, \dots, x_n)$, with the following general form:

$$\mathrm{poly}_i(\mathbf{x}) := \sum_j P_{ji} x_j + \sum_j Q_{ji} x_j^2 + \sum_{j>k} R_{jki} x_j x_k,$$

where all the coefficients P, Q, R and variables $\mathbf{x}$ are in $\mathbb{F}_q$.

The secret-key is the hidden structure of G. Now, to encrypt a message $M = (m_1, \dots, m_n)$, one needs to evaluate $G(M) = (y_1, \dots, y_m)$.

To decrypt the ciphertext $(y_1, \dots, y_m)$, one needs to invert the multivariate map using the (trapdoor) secret-key, which is equivalent to solving the systems of quadratic equations $\mathrm{poly}_i(\mathbf{x}) = y_i$ over $\mathbb{F}_q$ for each $i \in \{1, \dots m\}$. Without the trapdoor, decryption is computationally intractable, highlighting the critical role of the trapdoor in MBC. However, the existence of such a trapdoor cannot be guaranteed in a *random* system of multivariate polynomials. This means that the NP-hardness of the underlying mathematical problem does not directly translate to guaranteed security, as specific attacks might exploit the particular structure of the trapdoor. Consequently, the design and analysis of trapdoor mechanisms remain a significant focus of this PQC approach to ensure the required level of security.

2.4.6 Group-Based Cryptography (GBC)

This is one of the traditional approaches in PQC that has received less attention compared to the former ones. However, it regained prominence in 2023 [KFN+23], to highlight the importance of greater diversity in the field, after the introduction of the new attacks on multivariate-based and isogeny-based cryptographic algorithms.

GBC relies on problems in group theory that leverage mathematical properties and operations of algebraic groups (often partial or non-abelian[10]). The focus of recent research in GBC is on polycyclic groups and graph groups.

Cyclic groups, used in modern cryptosystems such as RSA and Diffie-Hellman, are special cases of polycyclic groups which can accommodate more complex cryptographic algorithms. A graph group is generated by the (finite) graph vertices and the only group operation between them is based on whether there is an edge between those specific vertices or not. This is a partially abelian group, since not all two vertices are connected in a general (non-complete) graph.

Various decisional problems in combinatorial group theory need further exploration to evaluate their resistance to quantum adversaries. For more information in this regard, we refer interested readers to [KFN+23; Bat+24; Deh11].

[9]Sometimes abbreviated as MPKC, standing for Multivariate Public Key Cryptography.

[10]Two famous examples of non-abelian groups in mathematics are matrix groups ($A \times B \neq B \times A$) and braid groups denoted by B_n[Ada04].

2.4.7 Symmetric Cryptography

We briefly explained symmetric cryptography in the Introduction of Chapter 1. Unlike public-key cryptography, which relies on two different keys for encryption and decryption, symmetric-key cryptography requires the involved parties to use the same key for a specific cryptographic algorithm (e.g., Advanced Encryption Standard (AES) or Data Encryption Standard (DES)) [KL14]. Unlike public-key cryptography, which is vulnerable to Shor's algorithm due to its reliance on integer factorization and discrete logarithm, symmetric-key cryptography remains unaffected by this threat.

To date, the only known generic quantum attack against symmetric-key cryptography is Grover's search algorithm. This algorithm provides a quadratic speed-up over classical search algorithms. In particular, Grover's search can search an unsorted database of size N in $O(\sqrt{N})$ time, compared to the classical $O(N)$ time, which means a quadratic speed-up. Moreover, in [Zal99], Zalka proved that Grover's search algorithm on unstructured database is exactly (and not asymptotically) optimal in the circuit model setting, meaning that no other algorithm can solve the problem with fewer queries than Grover (i.e., $\frac{\pi}{4}\sqrt{N}$). Therefore, as highlighted by NIST in a 2016 report [Che+16], symmetric cryptographic schemes are considered quantum-safe, provided that sufficiently large key sizes are used.

2.4.8 NIST

In the previous sections, we provided an overview of the PQC approaches discussed in the literature and highlighted their significance in the introduction of this thesis. We also introduced NIST and its effort in standardizing the PQC algorithms. NIST organized a four-round competition, a rigorous process designed to evaluate and select post-quantum cryptographic algorithms. The competition commenced with Round 1 in 2017, featuring 69 initial candidates. These submissions spanned various cryptographic primitives and approaches, including LBC (e.g., Ring-LWE), CBC (e.g., McEliece, Niederreiter), MBC (e.g., Rainbow), HBC (e.g., SPHINCS+), and IBC (e.g., SIKE). Round 2, in 2019, narrowed the field to 26 candidates, reflecting the elimination of those deemed insecure or less promising. Round 3, in 2020, further refined the selection to 15 finalists. Finally, Round 4, in 2022, concluded the evaluation process. Throughout these rounds, cryptanalysis played a crucial role, with vulnerabilities discovered and exploited, leading to the removal of candidates that couldn't be adequately patched.

Integral to the NIST PQC standardization process was the establishment of five security levels, designed to provide varying degrees of resistance against quantum attacks. These levels were defined to match or exceed the security strength of existing symmetric key algorithms:

1. Level 1: Security equivalent to AES-128
2. Level 2: Security equivalent to AES-192
3. Level 3: Security equivalent to AES-256
4. Level 4: Security equivalent to AES-256
5. Level 5: Security exceeding AES-256, providing the highest level of resistance

Candidates were required to specify their security level claims, and NIST evaluated them accordingly. The chosen algorithms were selected based on their security strength and performance, with the aim of providing a diverse set of options for different applications.

The culmination of this multi-year effort resulted in the selection of the first set of standardized post-quantum cryptographic algorithms. For general-purpose encryption and key establishment, CRYSTALS-Kyber [Sch+21], a lattice-based key-encapsulation mechanism (KEM) based on Module-LWE [BGV14], was chosen, meeting the security level 1 and 3 requirements. For digital signatures, CRYSTALS-Dilithium [Lyu+21], another lattice-based scheme based on Module-LWE, FALCON [Pre+20], a lattice-based signature scheme based on the Shortest Integer Solution (SIS) problem, and SPHINCS+, a hash-based signature scheme, were selected. They also meet levels 1, 3 and 5 security requirements. Additionally, NIST announced that BIKE [Ara+22], HQC, and Classic McEliece [Ber+22] code-based KEMs, and SIKE [Jao+22] an isogeny-based KEM, would advance to a fourth round of consideration for potential inclusion in a future standard, but SIKE was later broken [CD23; Rob23]. Therefore, at the end of the competition, the winners selected for standardization in the first round were CRYSTALS-Kyber, CRYSTALS-Dilithium, FALCON, and SPHINCS+, all of which met the required NIST security levels.

To conclude this section, we examine the trajectory of 'Unconditionally Secure Public-Key Encryption (with Possible Decryption Errors),' a first-round NIST competition candidate proposing the 'Guess Again' public-key encryption scheme. After the impossibility statement of attaining a perfectly secret public-key encryption scheme that we explained in Section 1.1, the title sounds ambitious. The authors of Guess Again claim that the impossibility statement is true only if the decryption algorithm of the PKE works with a success probability of one. Therefore, their construction allows decryption error to achieve perfect secrecy. In other words, the authors of Guess Again attempt to make the secret information non-uniquely identifiable from the public information. In [Pan20], Panny introduces an attack on the Guess Again PKE and demonstrates that a brute-force attack on the set of possible messages is always feasible, allowing an adversary with unlimited resources to succeed without knowing the secret key.

2.4.9 Comments on PQC

To conclude the post-quantum cryptography section, note that while some PQC approaches mentioned previously have been studied extensively for years, others are relatively new and are still emerging. Overall, PQC algorithms remain immature in several critical areas. First, the underlying mathematical problems, essential for the security of PQC, are still being analyzed in both classical and quantum computation models, leaving some uncertainty about their true difficulty. Second, there is a notable lack of formal proof of how these algorithms integrate securely within cryptographic protocols, which is crucial for reliable implementation. Third, the selection of optimal parameters, which directly impact security and efficiency, requires further rigorous research and validation. Finally, secure implementation of PQC algorithms demands greater focus, particularly in mitigating vulnerabilities to side-channel attacks. These attacks exploit physical leakage, such as timing or power consumption, making robust countermeasures a necessity for practical deployment. Addressing these challenges is essential for transitioning PQC from theoretical constructs to reliable cryptographic standards.

Despite these challenges, there is higher confidence in lattice-based and code-based schemes, as they are well-studied and have withstood attacks so far. Among these, the code-based approach is particularly important in this thesis, as it serves as the backbone of our PQ e-voting system. Detailed analysis of two cryptographic primitives in this approach can be found in Chapter 5. Despite the quantum resilience of symmetric cryptography, it is not practical for e-voting systems, as the election admin would need to securely distribute the encryption key to each voter, which introduces significant security challenges and is not efficient time-wise. For more detailed information on PQC, we refer interested readers to [BL17].

Chapter 3

Secure E-voting with Everlasting Privacy

3.1 Introduction

We begin the introduction of this chapter by introducing the concept of Systematization of Knowledge (SoK) and explaining why we chose to conduct an SoK on the topic of ”Everlasting Privacy“ in secure e-voting systems.

The role of SoK in Academic Advancement Systematization of Knowledge (SoK) plays a pivotal role in the advancement of any academic discipline. By consolidating, organizing, and critically analyzing existing research, SoKs provide a comprehensive foundation for understanding the current state of the field, identifying the research gaps, and paving the way for future advancements. In the field of cryptography, SoKs not only illuminate the theoretical foundations of critical topics but also establish frameworks for their practical applications. This chapter explores the concept of *everlasting privacy* and underscores its critical role in shaping a comprehensive SoK.

What is Everlasting Privacy? Everlasting privacy refers to a security paradigm where the confidentiality of sensitive information is guaranteed indefinitely, even against adversaries equipped with advanced future technologies, giving them unlimited computing power. Mechanisms ensuring everlasting privacy aim to protect data regardless of breakthroughs in computation or cryptanalysis.

This concept reminds us of information-theoretic security or perfect secrecy, which we explained earlier in Chapter 1. It was first introduced in the literature by Shannon in 1949 to prove the perfect secrecy of one-time pad (OTP) encryption [Sha49]. While OTP guarantees unconditional privacy, it is impractical for e-voting due to its reliance on symmetric-key encryption. This requires a unique encryption algorithm key to be generated and securely shared with each voter before vote submission, posing significant logistical challenges for the election authority (EA), especially in large-scale elections. Additionally, OTP's general applicability is limited by its key size requirement, as discussed in Chapter 1. Furthermore, we mentioned that public-key encryption schemes cannot attain perfect secrecy, meaning that publishing the encryption of the vote publicly, breaks the everlasting privacy property immediately. This is unlike hiding the votes using perfectly hiding commitments and publishing them publicly (see Section 2.3).

Importance of Everlasting Privacy in E-Voting Systems Everlasting privacy is particularly valuable in scenarios where sensitive information must remain confidential for decades or longer, such as personal health records, financial data, and electoral votes. Among the applications of everlasting privacy, electronic voting (e-voting), which is the main concern of this thesis, stands out as a domain where its principles are particularly crucial. E-voting systems must balance transparency, verifiability (see Section 2.2.4), and information accessibility with the critical necessity for voter privacy (see Section 2.2.3) and ballot secrecy. Anonymity and confidentiality are not only ethical imperatives but also prerequisites for fair and unbiased elections. If a voter's choices could be uncovered at some point in time, e.g., through technological advancements, the integrity of the electoral process would be compromised as the voter fears to express his/her true will.

Everlasting privacy addresses this concern by ensuring that ballots remain confidential for all time. In addition to privacy, verifiability is a critical requirement for e-voting systems. Verifiability ensures that all (internal and external) parties can confidently verify the correctness of election outcomes. However, verifiability often requires making some information public, such as vote encryptions, cryptographic proofs, or partial results, to allow external validation. This need for transparency may appear to contradict the principles of everlasting privacy, as it introduces the potential for sensitive information to be exposed.

To reconcile these conflicting goals, e-voting systems must carefully design protocols that provide strong privacy guarantees while enabling robust verifiability. We refer to these systems as *secure*. In secure systems, verifiability matters till end of the election, while privacy needs to last longer, even years after the election. If voters fear that their vote might be revealed in the future, at any time after the election, they might not show their true will to avoid future consequences or threats. This shows the importance of everlasting (long-term) privacy in the context of e-voting, however, as mentioned in Section 2.2, e-voting requires public-key cryptography and the symmetric-key solutions mentioned before are not a realistic option. On the other hand, the general approach to achieve a secure system, publishes the vote encryption on PBB, while the security of the encryption schemes used in the general approach (see Section 2.2.2), are based on computational assumptions. For instance, the security of ElGamal encryption which is used in Helios [Adi08], Selene [RRI16], and Hyperion [Dam+24] is based on DDH assumption. An adversary with unlimited computational power in the future, could save the public data of election on PBB now, and break the voter privacy later by breaking the DDH assumption.

Now, the question is how we could achieve EP in e-voting then? There are several protocols offered in the literature which strive for everlasting privacy (e.g., [Cra+96; CPP13; MN07; BDG13; LH15]). Most of them aim to achieve a weaker notion of everlasting privacy, named *practical* everlasting privacy. To make the strong notion of EP more realistic, Arapinis et al. in [Ara+13] assume a weaker adversary. An adversary that could monitor the communication channel between V and EA, is able to recover the vote and there is no privacy in this case. Therefore, for practical everlasting privacy, Arapinis et al. assume that the adversary is still computationally unbounded, however, does not have access to this communication channel (referred to as untappable or private channel) and does not collaborate with the EA (i.e., EA is honest). This means that the adversary is computationally unbounded towards the public election data. This relaxed notion still captures the threat of "store now, decrypt later" and at the same time, is more realistic to achieve in practice.

The existing protocols in the literature that strive to achieve EP differ in many aspects, resulting in a confusing state-of-the-art. They are different in the following aspects.

1. Desired Security Properties: In addition to the basic security properties, namely verifiability and privacy, that some protocols aim for (e.g.,[Cra+96; LH15]), some protocols aim to achieve additional properties such as receipt-freeness (e.g., [MN07]) and accountability (e.g., [CPP13]) (see Section 2.2 for definitions).

2. Threat Scenarios: Most of the protocols aim to achieve "practical" everlasting privacy (privacy against external adversaries), while some aim for everlasting privacy (privacy against both external and internal adversaries) [BDG13]. The threat scenario is sometimes mentioned implicitly.

3. Underlying Techniques: Several privacy-preserving techniques have been used in general to achieve the desired security properties. In most of the works, it is not clear which privacy-preserving technique is supposed to provide which security property. Some works use a single technique (e.g., [CPP13]), while others use two or more (e.g., [MN07]).

This confusing state-of-the-art among the protocols striving for EP brought up the following four questions, which was the motivation for conducting an SoK on this topic to provide reasonable answers to each.

- **Q1.** What is the relationship among these protocols and how do they relate to each other? Can we classify them to make them easier to understand?

- **Q2.** Which ones provide a secure and reasonable (efficient) solution under realistic (practical) assumptions to achieve EP?

- **Q3.** Which challenges in this field have been successfully addressed, and which continue to require solutions?

- **Q4.** Are there secure and practical solutions available today that ensure EP in real-world e-voting systems? If not, what gaps must be addressed to achieve this?

Structure of the Chapter. This chapter is organized into six sections. Section 3.2 provides a brief overview of the relevant preliminaries and cryptographic schemes. Section 3.3 outlines the scope of our study and the methodology employed to address the research questions introduced in the Introduction. Based on this methodology, Section 3.4 presents the classification of the collected papers and explains how this classification contributes to answering the main questions. Section 3.5 summarizes the key findings of our study. Finally, Section 3.6 concludes the chapter by discussing newly proposed protocols with EP that emerged in the literature following the publication of our paper [Hai+23].

3.2 Preliminaries and Cryptographic Schemes

We direct readers to Sections 2.1, 2.2, and 2.3 for the notations, foundational definitions of secure e-voting, and the cryptographic primitives used throughout this thesis. In instances where a new entity is introduced within an e-voting system, or a cryptographic primitive is employed that is not covered in Section 2.3, we provide an explanation in the relevant section itself. This approach is taken because these elements are unique to specific contexts and are not used elsewhere in the thesis. By explaining where they are applied directly, we maintain simplicity and ensure clarity for the reader.

3.3 Scope and Methodology

In this section, we begin by outlining the scope of our study, followed by a detailed explanation of the methodology employed to address the research questions posed in the Introduction of this Chapter 3.

3.3.1 Scope

Our paper [Hai+23], which forms the basis of this chapter, focuses exclusively on systematizing *secure* e-voting protocols with everlasting privacy that have been published in the academic literature. We provide post-publication updates for our paper up until the time of writing this thesis in Section 3.6. Our emphasis in both is on concrete implementations and constructions that directly contribute to the design and analysis of secure e-voting systems meeting the everlasting privacy criterion. We deliberately exclude the following categories of work from our analysis to maintain a precise and manageable scope.

- Paper-Based Voting Protocols:
  Although secure paper-based voting methods, such as those in [Smi07], are useful in certain contexts and can offer a good level of privacy, they fall outside the realm of secure electronic voting and are therefore beyond the focus of this work.

- E-Voting Protocols with Post-Quantum Privacy:
  Protocols such as Epoque [BHM21] and the one in [BHM20], aim to protect privacy conditionally by relying on post-quantum hardness assumptions (lattice-based cryptography here (see Section 2.4)) but cannot ensure indefinite protection if those assumptions are broken in the future.

- Non-Verifiable E-Voting Protocols:
  We exclude protocols that do not guarantee public verifiability, such as [YC17], since verifiability, as mentioned before, is a core requirement for secure and trustworthy e-voting systems.

- Generic Information-Theoretically Secure Multiparty Computation (MPC):
  While generic MPC protocols such as [DER21] can theoretically achieve everlasting privacy, they are not specifically tailored for e-voting use cases and lack the application-specific optimizations needed to address challenges unique to e-voting systems. For instance, as mentioned in [Küs+20], these protocols lack a suitable solution for verifiable e-voting: To achieve public verifiability in e-voting, we require not only the voters, but also any external party or observer to be able to verify the outcome. This is not an important property in a generic MPC, as privacy and integrity of the computations matter more.

- Theoretical Limits of Secure E-Voting and MPC:
  We exclude studies focused on the theoretical boundaries and feasibility of secure e-voting or MPC with everlasting privacy such as [HV08; MU10; Che+10]. These works do not provide concrete implementable constructions. Our focus lies in practical solutions that can be evaluated, compared, and potentially deployed.

3.3.2 Methodology

To systematically analyze the state-of-the-art in secure e-voting protocols with everlasting privacy, we adopted a structured, multi-step approach. This methodology ensures a comprehensive understanding of the field and addresses the questions raised in Chapter 3. The steps are as follows:

- **Step 1. Literature Review:** We conducted an extensive review of academic literature to identify all relevant protocols in the field, within the scope that we explained.

- **Step 2. Classification of Protocols:** We classified the identified protocols based on the technical mechanisms they use to achieve everlasting privacy. This classification highlights the dependencies among different protocols and reveals the evolution of ideas in the field.

- **Step 3. Evaluation of Practicality and Security Guarantees:** We critically evaluated the protocols for practical efficiency and their ability to guarantee both public verifiability and (practical) everlasting privacy under realistic assumptions. This step involved: Verifying whether the protocols meet their claimed properties, identifying any limitations or vulnerabilities that may arise in real-world scenarios, and examining the assumptions made by each protocol, such as trust assumptions on the internal parties, or other implicit assumptions involved.

- **Step 4. Identification of Research Gaps:** Based on our findings, we identified areas where existing protocols succeed and where they fall short. This step provided a clear road-map for open research problems and highlighted opportunities for future work in the field.

By following this methodology, we ensure a rigorous and systematic analysis of secure e-voting protocols with everlasting privacy, providing a foundation for further research and development in this pressing topic.

3.4 Classification

In Step 1. of our methodology explained in Section 3.3, we gathered and selected 25 research papers that align with the scope of our study as outlined in Section 3.3.1. The complete list of these papers includes: [FOO92; Gra10; Gro+18; Iov+17; Que+20; Kai+21; HB16; LH15; Kil+21; LH16; LHK16; CGG19; Ara+13; Ge+19; Cra+96; Gra09; MN06; MN07; CPP13; PR19; GHS20; BDG13; Hai19; DGS12; Dem+13]. To organize and analyze these papers effectively, we categorized them into two main classes: B-ANON and B-ID. Here, the letter B represents "Ballot," ANON signifies "Anonymous," and ID stands for "Identifiable". The B-ANON class consists of protocols that use anonymous ballots, meaning the ballots contain no information that could be used to trace them back to the corresponding voter. In contrast, the B-ID class comprises protocols with identifiable ballots, where each ballot includes some information that can be used by anyone to determine the voter who cast it. A detailed explanation of these two classifications is provided in Sections 3.4.1 and 3.4.2, respectively.

Each of these main classes is further divided into two subclasses. For B-ANON, the subclasses are based on the party responsible for generating the voter credentials. In contrast, for B-ID, the subclasses are determined by the privacy-preserving technique employed to tally the ballots. Our classification not only captures all these 25 papers but also provides a structured framework to better understand the underlying protocols and approaches used to achieve EP. Furthermore, this classification plays a crucial role in addressing the research questions raised in the Introduction (Section 3.1).

In Section 3.4.1, we first delve into the B-ANON class, outlining its two subclasses. Each protocol within these subclasses is analyzed in detail, highlighting its strengths, limitations (if any), and the explicit and implicit trust assumptions they involve. Following this, Section 3.4.2 provides a similar analysis for the B-ID class, with equal emphasis on its two subclasses.

For readers unfamiliar with the fundamentals of secure e-voting, including basic and additional security properties, election entities, and election phases, we strongly recommend reviewing Section 2.2 before proceeding. This section serves as a prerequisite, providing the necessary context to fully understand the methodologies and classifications discussed here.

3.4.1 B-ANON

In the B-ANON class, the focus is on maintaining the anonymity of voters by ensuring that the ballots published publicly on PBB contain no identifiable information (e.g., voter's ID, signature, or public-key), making the ballots unlinkable to individual voters. In this approach, voters need to submit their ballots via an anonymous channel. Therefore, achieving EP reduces to the assumption that this channel is unconditionally anonymous, ensuring that the unlimited adversary cannot observe or infer any information about which voter submitted which ballot.

Among the 25 papers we collected, 13 belong to the B-ANON class: [FOO92; Gra10; Gro+18; Iov+17; Que+20; Kai+21; HB16; LH15; Kil+21; LH16; LHK16; CGG19;

Ara+13]. The B-ANON class is further divided into two subclasses based on the entity responsible for generating voters' credentials during the election process. If the election authority (EA) generates the credentials, the protocol is classified as B-ANON-A. Conversely, if the voter (V) generates her own credentials, it falls under B-ANON-V. This distinction is significant because the approaches employed in these two subclasses differ in a critical aspect, which justifies this classification. We elaborate on this key difference in Section 3.5. Note that, in general, a voter's credential can be jointly generated by both the voter and the EA, as demonstrated in [Roe16]. However, none of the 25 protocols we studied adopt this approach. In the following, we describe the protocols in each subclass in detail.

3.4.1.1 B-ANON-A

The protocols that fall under the B-ANON-A class include the following papers: [Ara+13; CGG19; FOO92; Gro+18; Iov+17; Kai+21; Que+20; Gra10]. In this subclass, EA is responsible for generating the voter credentials, ensuring that only eligible voters can submit a ballot via anonymous channel (see Section 2.2.5 for the importance of voter eligibility). This is done using three different approaches in B-ANON-A: no identities, blind signatures, and pseudonyms. Next, we explain each approach and the protocols that follow them.

1. No Identities

This approach contains a single protocol [Ara+13]. As mentioned in Section B-ANON 3.4.1, the goal is to remove identifiable information from the voter's ballot, so that it remains unlinkable to the corresponding voter. This approach uses the simplest way, which is to remove the voter's identity from the ballot. As we discussed earlier in this thesis, Helios [Adi08] publishes the pair of voter identity and encrypted vote as $(\mathsf{ID}_i, \mathsf{Enc}(v_i))$ for voter V_i on PBB. Arapinis et al. in [Ara+13] offer and analyze a version of Helios that removes ID from the ballot, which they call *Helios without identities*. To provide better clarity on Helios-without-identities, we first outline the high-level idea of the original Helios protocol.

Original Helios:

1. *Setup Phase:* Each V_i receives an individual password denoted by psw_i from EA.
2. *Submission Phase:* V_i performs the following tasks.
   - Encrypts v_i using the tallier's public key pk_T: $ct_i := \mathsf{Enc}(pk_T, v_i)$.
   - Computes NIZKP of well-formedness of ct_i denoted by π_i.
   - Authenticates to PBB using psw_i and sends her ballot $b_i = (\mathsf{ID}_i, ct_i, \pi_i)$.

   PBB performs the following checks before publishing b_i. If all pass, b_i will be published on PBB.

   - Check psw_i matches V_i's identity (ID_i).
   - Check validity of π_i.
   - Check π_i is not in any other ballot b_j.
3. *Tally Phase:* Votes are tallied using either homomorphic aggregation or a mixnet, depending on the implementation.

As we mentioned earlier, since the pair (ID_i, ct_i) is on PBB, and ElGamal encryption is used to mask the vote, a future adversary could break the privacy. Arapinis et al. address this issue by removing ID_i from b_i.

Evaluation of Helios Without Identities:

Privacy Guarantees:
Arapinis et al. claim that Helios without identities achieves practical everlasting privacy. They formally validate this claim using two formal verification software: AKISS [CCK12] and ProVerif [BAF08]. Their analysis confirms that the protocol maintains privacy even against adversaries with unbounded computational power. Our analysis did not reveal any issues that would challenge or contradict their privacy findings.

Limitations:
No Public Verifiability: While the protocol ensures voter anonymity by removing ID from b_i, it sacrifices public verifiability. There is no way for an observer to verify that the ballots b_i on PBB are sent by eligible voters and confirm the election outcome. In this setting, a corrupted EA could manipulate the election outcome by adding fraudulent ballots for abstaining voters or even non-existent voters. This lack of transparency makes it unsuitable for applications where public trust and verifiability are critical.

2. Blind Signatures

This approach encompasses four protocols: [FOO92; Kai+21; Gra10; Gro+18]. In this approach, to prove voter eligibility without linking b_i to ID_i, the idea to set the role of EA to issue a blind signature on voter's blinded ballot, add this signature to b_i. The voter then casts b_i via an anonymous ballot submission channel. The idea of using a blind signature in e-voting originates from the work [FOO92] by Fujioka, Okamoto, and Ohta. The e-voting protocol offered in this work is commonly referred to as FOO protocol in the literature, named after its authors.

While the FOO protocol was not initially designed to provide EP, Van de Graaf was the first to observe its potential in this regard [Gra10]. In his work, he proposed a technique to implement an anonymous ballot submission channel in the FOO protocol, laying the groundwork for its use in achieving EP.

Building upon the FOO protocol, two additional protocols have been proposed with specific enhancements, aiming to achieve EP: [Gro+18] additionally offers coercion-resistance (CR), and [Kai+21] aims to reduce the trust on EA who generates the blind signatures. We distinguish between the first three works [FOO92; Kai+21; Gra10] and the last one [Gro+18]. The reason is that the last work aims to achieve CR. To this end, it uses an advanced version of blind signature, compared to the basic version used in the other three works.

In what follows, we first explain the basic version of blind signature, then we elaborate on the advanced version used in the last work. Finally, we conclude by evaluating all these protocols.

Basic Version of Blind Signature:

The concept of blind signature was first introduced in 1984 by David Chaum in [Cha83]. In a standard digital signature scheme (see Section 2.3.4), the signer has full knowledge of the message they are signing. This characteristic, however, poses a significant privacy risk in scenarios like e-voting. If a voter (V) requires the election authority (EA) to sign their ballot as proof of eligibility using a digital signature, the ballot contains the plain vote. This linkage between the signature and the vote would compromise ballot privacy. To address this issue, the "blinding" technique introduced by Chaum could be used. This technique consists of two algorithms: $(\mathsf{Blind}, \mathsf{Unblind})$.

In what follows, we explain the basic version of the blind signature approach using an e-voting scenario, as applied in the first three referenced papers: Using this basic version of the blind signature approach, V_i blinds her vote as $bv_i = \mathsf{Blind}(v_i)$, and sends the blinded vote bv_i to EA via a private channel. EA first checks V_i's eligibility, and if verified, signs the bv_i, without knowing the blinded content, and sends the signature back to V_i as σ_i. Then, V_i unblinds EA's signature by computing $\mathsf{Unblind}(\sigma_i)$, leading to EA's signature on the plain vote v_i. This signature is added to b_i and submitted to PBB via the anonymous channel by V_i.

Advanced Version of Blind Signature:

The advanced version of the blind signature by Chaum is *conditional blind signature* scheme, introduced in [ZGP17]. This version builds on the basic blind signature scheme but adds a condition that introduces the ability to achieve CR. Indeed, in their work [Gro+18], Grontas et al. combine the fake credential technique (see Section 2.2.7) with the conditional blind signature approach to achieve CR.

The key idea is that the voter's eligibility to receive a valid signature from the EA depends on specific conditions that are designed to detect and prevent coercion: The conditional blind signature technique, issues a valid signature on the blinded message just in a special case, when the voter is under coercion. Otherwise, the signature is invalid. This is done by embedding a bit into the blind signature, which shows the validity of the signature only if the voter provides her real credential to EA (she is not coerced). If she provides any other credential not matching her real one, EA will know she is coerced and will issue an invalid signature. This embedded bit is not known to the voter for the sake of CR; therefore, she cannot verify the validity of the issued signature.

To address this verifiability issue, they introduce a publicly auditable version of the conditional blind signature in their work [Gro+18]. In the tally phase, when EA has to remove the ballots with invalid signatures (it could be multiple ballots per voter), she must perform this task without revealing the link between the individual ballots. To this end, Grontas et al. use the famous JCJ technique in the literature [JCJ05].

Evaluation of the Blind Signature Technique

In general, we observed that neither the basic nor the advanced blind signature approach, do not offer a secure solution under realistic assumptions to achieve EP. To justify this, we first start by explaining the privacy guarantees each approach provides, and then their limitations.

Privacy Guarantees:

- Basic Version (Chaum's Blind Signature):

1. The use of unconditionally blinding signature schemes in the FOO protocol and its variants [Kai+21; Gra10] ensures that the voter's identity is decoupled from their ballot, offering a foundation for everlasting privacy if the ballot submission channel is perfectly anonymous.
2. Van de Graaf's work [Gra10] extends this by proposing the use of a variant of Chaum's Dining Cryptographers Network (DC-net) [Cha88] to achieve unconditionally anonymous submission channels, enabling everlasting privacy for ballots.

- Advanced Version (Conditional Blind Signature):
  Grontas et al. [Gro+18] introduce (publicly auditable) conditional blind signature scheme, which incorporates mechanisms to safeguard against coercion by embedding a validity condition into the signature. This aims to combine CR with privacy guarantees under the assumption of a trustworthy EA.

Limitations:

- Basic Version (Chaum's Blind Signature):
  1. Corrupted EA: In the original FOO protocol, the EA can issue blind signatures even for voters who have not requested them, enabling ballot stuffing with illegitimate votes. To address this issue, Kaim et al. [Kai+21] use the lattice-based threshold blind signature proposed by Bouaziz et al. [Bou+20] to distribute the EA's role. However, according to [Hau+20] this lattice-based blind signature lacks a valid security proof.
  2. Reliance on Anonymous Channels: The FOO protocol's EP entirely depends on the assumption of perfectly anonymous submission channels. Without such channels, privacy is compromised.
  3. Practicality Issues: The realization of unconditionally anonymous channels using DC-nets, as suggested by Van de Graaf, is unrealistic for large-scale elections due to its requirement for the collective participation of all voters and its lack of robustness to voter dropout.
  4. Lack of Computational Safeguards: Unlike other B-ANON-A protocols, the FOO protocol does not include computationally secure tallying mechanisms, leaving privacy solely dependent on the anonymity of the submission channel. More precisely, voters send the opening values of their committed (blinded) votes after the voting phase to PBB through an anonymous channel. Then, the votes are opened using these values and the election result is announced.

- Advanced Version (Conditional Blind Signature):
  1. Excessive Trust in EA: The protocol in [Gro+18] assumes that EA creates and therefore, knows the secret credentials of all voters. This means that, as with the basic version, corrupted EA can impersonate voters and submit ballots on their behalf
  2. Quadratic Tallying Complexity: The protocol inherits the high computational cost of the JCJ tallying phase [JCJ05], making it impractical for large-scale elections.
  3. Dummy Votes and Coercion: The use of dummy votes to protect against coercion, as mentioned in [Gro+18], introduces additional vulnerabilities, as EA can manipulate these votes without detection.

4. All together, this means that unlike claimed in [Gro+18], EA needs to be trusted for all security properties the system achieves, namely public verifiability, EP, and CR (absolute trust). Therefore, its security level is the same as a basic e-voting system with a single trusted authority (see Section 2.2). To see the role of EA in the last two properties, consider the following scenario: a corrupted EA impersonates all voters but one, let's say V_i. Then, when the final election result is available, he can easily learn the vote of V_i and see if she followed the coercion instruction or not.

3. Pseudonyms

Among the 13 papers in B-ANON-A, three fall into the category of pseudonym-based approaches: [CGG19; Iov+17; Que+20]. In this approach, the EA assigns a pseudonym (a pseudo-random number) to each voter, replacing voter identities with pseudonyms on PBB. This mechanism removes the direct link between voter identities and their ballots, which, as noted by [Iov+17], is a "quick and dirty way" to achieve EP compared to more sophisticated cryptographic techniques. The privacy guarantee hinges on the secrecy of the mapping between ID_i and the pseudonym assigned to V. Unlike Belenios [CGG19], the other two works aim to achieve CR as an additional security property. This distinction introduces varying trust assumptions across the election entities involved. Below, we outline the high-level ideas in Belenios, followed by an overview of the other two works.

Belenios: Belenios [CGG19] aims to mitigate trust on PBB in Helios (see Section 3.4.1.1), as there, a malicious PBB could stuff ballot without being caught. Belenios adds eligibility verifiability to it. To mitigate this risk, Belenios introduces eligibility verifiability through a new entity called the Registrar (R), a specialized version of EA. The key steps in the Belenios protocol are as follows:

1. *Setup Phase:* R generates a signing key pair (sk_i, vk_i) for each voter V_i. It sends the private signing key sk_i to V_i and publishes the public verification key vk_i as the pseudonym of V_i on PBB.

2. *Submission Phase:* V_i authenticates to PBB using her password psw_i and submits her signed ballot, encrypted with sk_i.

3. *Tally Phase:* During tallying, the tallier verifies the validity of all submitted ballots using the corresponding verification keys vk_i and removes any ballots with invalid signatures.

Belenios guarantees EP as long as R securely maintains the secrecy of the mapping between ID_i and vk_i. This design adds an eligibility verifiability feature absent in Helios while reducing reliance on PBB.

The other two papers [Iov+17; Que+20], extend the pseudonym-based approach by incorporating privacy-preserving techniques to achieve CR alongside EP. These techniques are out of the scope of this SoK chapter and we do not elaborate on them here. These techniques enable voters to evade coercion by overwriting their votes. In [Iov+17], the authority responsible for generating pseudonyms is referred to as the *Registration Teller*. In [Que+20], the equivalent role is termed the *Certificate Authority*.

Privacy Guarantees: As long as the link between ID_i and vk_i is kept secret, EP holds in these three papers. In general, they all require trust in EA for EP and to some extent, for verifiability. The papers aiming for CR, [Iov+17; Que+20], need to trust EA for other security properties they achieve as well.

Limitations:

- Belenios:
  1. Trust Assumptions:
     - Belenios claims verifiability is ensured if either EA or PBB remains honest. However, subsequent research shows that PBB must be trusted for verifiability and, to some extent, privacy [HSB21; Bal+21a].
     - EA knows the mapping between voters and their pseudonyms, which means a corrupted EA can violate privacy by exploiting this knowledge.
  2. Verifiability Vulnerabilities: Belenios' reliance on pseudonyms introduces verifiability issues. Research paper [Bal+21a] by Baloglu et al. shows that corruption of either EA or PBB can lead to attacks compromising verifiability. Although partial mitigations were introduced later [Bal+21b], these issues are not fully resolved.
- CR papers:
  1. Trust Assumptions:
     - These protocols require a high level of trust in EA for all security properties, including privacy, coercion-resistance, and public verifiability.
     - EA knows the mapping between pseudonyms and voter identities as well as all voters' secret credentials. This knowledge allows a corrupted EA to overwrite any voter's choice by submitting a new ballot anonymously on their behalf, violating public verifiability. As explained for [Gro+18] also aiming for CR in the previous section, EA could impersonate all voters except one, revealing the remaining voter's choice, and breaking both privacy and coercion-resistance.
  2. Efficiency Concerns:
     - The protocol in [Iov+17] suffers from quadratic computational complexity during the tallying phase, inherited from the JCJ protocol [JCJ05]. This makes it impractical for large-scale elections.
     - In contrast, [Que+20] is designed to handle large-scale elections efficiently, addressing this limitation.

3.4.1.2 B-ANON-V

The protocols that fall under the B-ANON-V class include the following papers: [HB16; LH15; Kil+21; LH16; LHK16]. In this subclass, the voters themselves are responsible for generating their credentials. As with the subclass B-ANON-A, we differentiate between these works based on the approach that they take to ensure that only eligible voters can submit a ballot via anonymous channel (see Section 2.2.5 for the importance of voter eligibility). This is done using three different approaches in B-ANON-V: linkable ring signatures, and membership ZKP. Next, we explain each approach and the protocols that follow them.

1. Linkable Ring Signatures

In this sub-class, we have the VOTOR e-voting system introduced by Haines and Boyen [HB16]. It employs linkable ring signatures to achieve eligibility verifiability in addition to EP. Unlike traditional digital signature schemes including blind signatures (explained in Section 3.4.1.1), there is no single signer in ring signatures, but a "ring" of possible signers. A ring signature allows a signature to be attributed to a member of a ring without revealing which specific member signed it. This property ensures anonymity for the signer while maintaining verifiability that the signature belongs to a valid participant.

Ring signatures were first proposed in [RST01] by Rivest, Shamir, and Tauman in 2001. A significant enhancement of this concept is linkability, which enables the identification of signatures made by the same signer (i.e., linkability) without compromising the signer's anonymity. Specifically, one can confirm that a series of signatures belong to the same individual while remaining unaware of their identity.

Now we explain how VOTOR uses linkable ring signatures to achieve EP: Unlike B-ANON-A, here each voter V_i has to generate her signing key (sk_i, vk_i), keep the signing secret-key private, and use vk_i to update the joint ring verification key vk. Each voter then signs her vote and needs to anonymously publish it on PBB. To this end, VOTOR incorporates onion router TOR, which is a system that obscures the source and destination of the data by providing multiple layers of encryption. The validity of the signatures appearing on PBB could be verified using the joint verification key vk.

In VOTOR the authors use a version of ring signature, proposed by themselves in [BH18], which in addition to linkability, it offers *forward-security*. Linkability can track which ballots belong to the same voter, and forward-security allows voters to keep their verification key vk_i in each election and only update the secret signing key sk_i to maintain privacy. The last feature has the advantage that if vk_i is compromised in an election, it doesn't affect sk_i used in the previous elections.

Evaluation of Linkable Ring Signature Technique

Privacy Guarantees:

- Everlasting Privacy: As claimed in [HB16], VOTOR ensures EP by using TOR for anonymity and linkable ring signatures to unlink voter identities from their ballots.
- Public Verifiability: VOTOR provides verifiability through the joint ring verification key vk, ensuring that only eligible voters' ballots are counted.

Limitations: Efficiency: The forward-secure linkable ring signature used in VOTOR requires linear computation in the size of the electorate for each signature verification. Therefore, for large-scale elections, the total verification workload scales quadratically with the number of voters, making the system computationally infeasible.

2. Membership ZKP

In this subclass, four papers are included: [LH15; LH16; LHK16; Kil+21]. The base protocol is [LH15], which incorporates a membership ZKP to achieve public verifiability and EP. This work is improved in [LH16] to achieve the additional feature of receipt-freeness (RF), and further enhanced in [LHK16] to achieve coercion-resistance (CR). These improvements rely on the same membership ZKP and differ from the base protocol in the tally phase.

Additionally, the work in [Kil+21] provides an implementation of the protocol from [LH15]. A membership ZKP serves a purpose similar to the linkable ring signature we explained earlier, however, it uses a different technique: Membership ZKP allows a voter to prove knowledge of her secret key (secret credential) which is linked to a public verification key (public credential) that is published in a public list, without revealing which one.

In the base protocol [LH15], Locher and Haenni propose a version of membership NIZKP that allows a verifier to verify which proofs belong to the same sk_i. The e-voting system is similar to the one in linkable ring signature with the difference that here a NIZKP is being used: Voters have to create their own public/private credential pair as (vk_i, sk_i). Each V_i has to keep sk_i secret and append vk_i to the public list $\vec{vk}$. To cast a ballot, V_i uses sk_i to create a NIZKP π_i to show that she has the knowledge of a sk which belongs to a member of the list $\vec{vk}$. She then submits $b_i = (v_i, \pi_i)$ to PBB via an anonymous ballot submission channel. The tally is performed on b_is with correct π_i.

Evaluation of Membership ZKP

Privacy Guarantees: We observe that the base protocol of Locher and Haenni [LH15] achieves public verifiability and EP as they claim. In addition, to show it is suitable for large-scale elections, they implemented their e-voting system and provided benchmarks.
Limitations:

- The Base Protocol [LH15]: While the base protocol achieves EP and public verifiability, it is heavily reliant on secure anonymous submission channels. If these channels are compromised, the privacy guarantees are invalidated.

- The RF Version [LH16]: This version, to provide RF, allows voters to cast several ballots without any limitations. Therefore, it introduces a significant scalability issue. The tallying phase complexity grows linearly with the number of submitted ballots. If a corrupted voter submits a large number of ballots, the tallying process becomes inefficient, potentially stalling the election, while this voter remains anonymous.

- The CR Version [LHK16]: This version inherits the same scalability problem as the RF version but worsens it. Here, the tallying complexity grows quadratically with the number of ballots, making the protocol unsuitable for large-scale elections.

- Killer et al. [Kil+21]: This implementation does not explicitly incorporate anonymous channels, which undermines the theoretical privacy guarantees. As with the base protocol, the absence of secure anonymous submission channels compromises voter anonymity and EP.

3.4.2 B-ID

In the B-ID class, the ballots published publicly on PBB contain a piece of information (e.g., voter's ID, signature, or public-key) that could track it back to the voter. Therefore, achieving EP reduces to using unconditionally privacy-preserving techniques in the tally phase to ensure unlinkability.

Among the 25 papers we collected, 12 belong to the B-ID class: [Ge+19; Cra+96; Gra09; MN06; MN07; CPP13; PR19; GHS20; BDG13; Hai19; DGS12; Dem+13]. The B-ID class is further divided into two subclasses based on the approach that is used to tally the ballots. If the talliers use homomorphic vote aggregation, the protocol is classified as B-ID-HOM. Conversely, if mix-net is used to shuffle and re-randomize the ballots in the tally phase, it falls under B-ID-MIX. We refer the reader to Section 2.2 for details of the homomorphic and mixnet approach to tally ballots. This distinction is significant because the approaches employed in these two subclasses differ in a critical aspect, which justifies this classification. As with the B-ANON sub-class, we elaborate on this key difference in Section 3.5. In the following, we describe the protocols in each subclass in detail.

3.4.2.1 B-ID-HOM

The protocols categorized under the B-ID-HOM sub-class include the following papers:
[Cra+96; CPP13; DGS12; Ge+19; Gra09]. These protocols leverage homomorphic aggregation during the tally phase, enabling the computation of election results without directly accessing individual ballots, thereby preserving voter privacy. We differentiate between three approaches based on how trust is distributed among the talliers for privacy: no distribution, distributed trust, and secret-sharing. We do not discuss the papers in the first approach, [DGS12; Gra09] since in these protocols trust is not distributed and they could be considered as special cases of the later two categories where trust is distributed. In the following sections, we evaluate the protocols in these latter two categories, highlighting their advantages, limitations, and suitability for real-world elections.

1. Distributed Decryption

Among the mentioned works in B-ID-HOM, the homomorphic version of perfectly private audit trail (PPAT) by Cuvelier et al. [CPP13] distributes trust by incorporating a threshold encryption scheme. Cuvelier et al. propose a new cryptographic scheme named *commitment consistent encryption* (CCE). This scheme allows a party to derive a commitment based on an encryption that not only commits to the same message, but also is computationally hiding.

To take advantage of CCE to achieve EP, they incorporate a new entity in their voting system named secret bulletin board (SBB) which only the talliers have access to. Their e-voting protocol works as follows: The talliers $\mathsf{T}_1, \ldots, \mathsf{T}_k$ threshold share the joint encryption key (pk, sk) which results in the threshold pair of keys (pk^j, sk^j) per T_j. In the ballot submission phase, each V_i encrypts her vote as $e_i = \mathsf{Enc}(pk, v_i)$. V_i sends e_i to SBB. Using CCE, SBB derives a commitment $c_i = (prm, v_i, r_i)$ based on e_i which commits to the same message (vote v_i) and posts c_i on PBB since now this is perfectly hiding. In the tally phase, the tallier homomorphically aggregates the encryptions and commitment as $e = \sum_{i=1}^{n} e_i$ and $c = \sum_{i=1}^{n} c_i$ respectively. Then, each T_j uses sk_j to threshold decrypt the joint encryption e. This results in partial opening values for the joint commitment c: (v^j, r^j). They finally aggregate the joint opening values to achieve (v, r) and publish this as the opening value of c on PBB for public verifiability: Anyone can check $\mathsf{Verify}(c, v, r) == 1$.

Evaluation of Distributed Decryption

Privacy Guarantees: We observe that the homomorphic version of [CPP13], as claimed by the authors, provides the following two properties:

- Public Verifiability: This is achieved through the use of the CCE scheme and its homomorphic properties that validate the tally while keeping individual votes private.
- Practical Everlasting Privacy: This is realized using unconditionally hiding property of the commitments derived from the encrypted votes.
- Accountability: Unlike the secret sharing approach that we will explain next, PPAT provides mechanisms for resolving disputes during the submission phase, enabling accountability in case of irregularities.
- Efficient Voter Workload: Voter workload is independent of the number of talliers, making the protocol more scalable in theory compared to secret-sharing methods where voter workload increases linearly with the number of talliers.
- Instantiation: Cuvelier et al. propose an efficient instantiation of the CCE scheme using an extended version of ElGamal PKE [Gam83], where the derived commitment from this encryption is Pedersen (see Example 2.4).

Limitations:

- Negligible Advantage in Tallier Scaling: Although voter workload does not scale with the number of talliers, this advantage is practically insignificant since real-world elections typically involve only a small number of talliers.

We observe that [CPP13] successfully achieves the claimed security properties and provides an efficient implementation of its generic cryptographic scheme. Under realistic assumptions, we consider it a practical and reliable solution for ensuring EP in e-voting systems.

2. Secret-Sharing

The technique of secret-sharing to distribute trust among talliers was first proposed by Cramer et al. in 1996 [Cra+96]. In 2019, Ge et al. [Ge+19] introduced an e-voting system building upon Cramer et al.'s secret-sharing approach, addressing a specific issue in the original protocol, which we explain in detail below.

First, we provide a high-level overview of the e-voting system proposed by Cramer et al. [Cra+96]. Next, we identify the issue inherent in their approach and explain how Ge et al. resolved it in their work [Ge+19]. Additionally, we present an alternative solution to the same problem, proposed by Müller in our SoK paper [Hai+23], and compare it with Ge et al.'s solution.

High-Level Overview of the protocol in [Cra+96]:

For simplicity, we assume:

- Voters choose between two candidates, encoded as 0 and 1. The protocol can be extended to N candidates by running it in parallel N times.
- Full-threshold $(e.g., (k, k) - threshold)$ additive secret-sharing is used instead of arbitrary threshold $(e.g., (t, k) - threshold for t < k)$ secret-sharing.

Cryptographic Primitives Employed

1. *Commitment Scheme* ($\mathcal{C} = (\mathsf{KeyGen}_{\mathcal{C}}, \mathsf{Com}, \mathsf{Open})$)
   - *Properties:* Unconditionally hiding, computationally binding, and additively homomorphic.
2. *NIZKP of Knowledge* (Π)
   - *Relation:* $\mathcal{R} = \{((prm, c), (m, r)) \colon c = \mathsf{Com}(prm, m, r) \wedge m \in \{0, 1\}\}$.
   - *Purpose:* Proves that a commitment c binds to a vote m for one of the two possible candidates.

Protocol Phases

1. Setup Phase

- The election authority (EA):
  - Generates the election parameters prm.
  - Publishes prm on PBB.

2. Voting Phase

Each voter V_i performs the following steps:

- *Secret Sharing:*
  Secretly splits their vote v_i among the talliers $\mathsf{T}_1, \ldots, \mathsf{T}_k$ as $v_i = \sum_{j=1}^{k} v_i^j$.
- *Commitment Computation:*
  Computes commitments for each share as $c_i^j \leftarrow \mathsf{Com}(prm, v_i^j, r_i^j)$.
- *NIZKP Generation:*
  Constructs a NIZKP π_i for statement (prm, c_i) with witness (v_i, r_i), where: $c_i \leftarrow \sum_j c_i^j$ and $r_i \leftarrow \sum_j r_i^j$.
- *Publishing Ballot:*
  - Publishes the ballot on PBB: $b_i \leftarrow (i, (c_i^j)_j, \pi_i)$.
  - Privately sends opening values (v_i^j, r_i^j) to the respective talliers T_j.
- *Dispute Resolution:*
  - If a tallier T_j receives invalid opening values for c_i^j, it files a complaint on PBB.

3. Tallying Phase

- *Ballot Validation:* Discard any ballot b_i for which: The NIZKP π_i is invalid, and a complaint has been filed by a tallier.
- *Vote Aggregation:* Each tallier T_j aggregates its share of opening values from valid ballots: $v^j \leftarrow \sum_i v_i^j$ and $r^j \leftarrow \sum_i r_i^j$.
- *Publication and Verification:*
  - Talliers publish (v^j, r^j) and their correctness is verified using the Open algorithm: $\mathsf{Open}(prm, c^j, v^j, r^j) == 1$. Here, $c^j \leftarrow \sum_i c_i^j$.
- *Final Result:*
  - The final election result is computed as: $v \leftarrow \sum_j v^j$
  - The result is accepted if all previous checks pass successfully.

Evaluation of Secret-Sharing by Cramer et al.

The secret-sharing protocol by Cramer et al. [Cra+96], as described earlier, incorporates several cryptographic guarantees that contribute to its robustness. However, upon closer examination, the protocol lacks a critical property: public verifiability. Below, we elaborate on why this is the case, despite its strong foundational guarantees.

Existing Security Features:

1. Binding and Homomorphic Commitments:
   (a) The commitment scheme used in the protocol is both homomorphic and binding.
   (b) As a result, the final tally v is accepted (with overwhelming probability) only if $v = \sum_i v_i$, where v_i represents the vote embedded in V_i's aggregated commitments c_i.
   (c) This property ensures that talliers cannot alter the result during the tallying phase.
2. Soundness of NIZKP:
   - Each voter provides a non-interactive zero-knowledge proof (NIZKP) π to prove that their commitment corresponds to a valid vote (e.g., either 0 or 1).
   - This guarantees that voters can only commit to legitimate choices, preventing any invalid inputs from affecting the tally.

These two features, binding and soundness on the surface, seem to ensure the integrity of the protocol by preventing both voters and talliers from tampering with the election process. However, there is a gap in the public verifiability of the system, which lies in the lack of accountability during the handling of commitments by the talliers. To clarify, consider the following scenario: Consider a situation where tallier T_j claims it cannot process V_i's share of the commitment c_i^j because V_i allegedly failed to provide valid opening values (v_i^j, r_i^j) for c_i^j. Based on this claim, V_i's entire ballot is discarded. Without additional mechanisms, there is no way to determine whether T_j's claim is truthful. If T_j is honest, the claim may reflect a genuine issue with V_i's submission. However, if T_j is corrupted, it can falsely claim invalidity to exclude V_i's ballot from the tally.

Resolving the Public Verifiability Issue:
The public verifiability issue in the original secret-sharing protocol by Cramer et al. [Cra+96], as discussed earlier, can be resolved through two different approaches: By Ge et. [Ge+19] and our solution in [Hai+23]. Below, we describe these two solutions in detail and compare their strengths and weaknesses.

Solution 1: Receipts (Proposed by Ge et al. [Ge+19]) To address the verifiability gap, Ge et al. introduce a mechanism involving cryptographic receipts issued by talliers to voters. The process works as follows:

- *Voter-to-Tallier Communication:* Each voter V_i sends a tuple (c_i^j, v_i^j, r_i^j) to tallier T_j, where:
  - c_i^j is the share of the commitment.
  - (v_i^j, r_i^j) are the opening values corresponding to the share.

  The tallier verifies whether the opening values (v_i^j, r_i^j) are valid for the commitment c_i^j.
- *Receipt Issuance:* If the opening values are valid, the tallier T_j issues a cryptographic signature s_i^j on the commitment c_i^j as a receipt.
- *Publication on the Public Bulletin Board:* The voter publishes the commitments $(c_i^j)_j$ along with the received signatures $(s_i^j)_j$ on the PBB. These signatures serve as verifiable receipts proving that the talliers acknowledged the voter's submission.
- *Mitigation of Dishonest Talliers:* While a malicious tallier could refuse to issue a receipt, this issue can be mitigated by employing auditors. These entities periodically send test requests to talliers to ensure they respond correctly to valid submissions.

Analysis: Ge et al.'s modification effectively closes the above-mentioned verifiability gap under realistic assumptions. By requiring talliers to issue receipts, voters gain a mechanism to prove their ballots were submitted and acknowledged, enhancing transparency and trust.

Solution 2: Secret Bulletin Board An alternative solution involves introducing a secret bulletin board (SBB), as in PPAT approach, accessible only to talliers. The process is as follows:

- *Encrypted Submission:* Each voter V_i posts on SBB the following:
  - Commitments $(c_i^j)_j$ for their shares.
  - A NIZKP π_i proving the validity of the commitments.
  - Encrypted opening values $e_i^j \leftarrow \mathsf{Enc}(\mathsf{pk}_j, (v_i^j, r_i^j))$, using the respective tallier's public key pk_j.

- *Public Hash Publication:* The secret bulletin board hashes the encrypted values $(e_i^j)_j$ using a cryptographic hash function and posts the resulting hashes $(h_i^j)_j$ alongside the commitments and NIZKP on the public bulletin board (PBB). This allows voters to verify that their ballot has been published correctly.
- *Verifiable Decryption:* If a tallier T_j claims that the encrypted opening values (e_i^j) are invalid or missing, the tallier is required to publicly decrypt the ciphertext e_i^j using a verifiable decryption process. The hash h_i^j ensures that the decrypted values match the originally submitted ciphertext, preventing tampering.

Analysis: This approach removes the possibility of a tallier falsely claiming that a voter's opening values are invalid. By requiring verifiable decryption, any malicious behavior by talliers can be detected, thereby ensuring public verifiability and accountability.

You can see a comparison table of these two solutions in Table 3.1, and the comparison table of the ballot submission phase in all three works: [Cra+96; Ge+19; Hai+23] in Figure 3.1.

Criterion	**Solution 1: Receipts**	**Solution 2: Secret Bulletin Board**
Public Verifiability	Mitigates verifiability issues, but relies on external auditors to confirm valid responses. A dishonest tallier can refuse to provide receipts, though this can be mitigated with auditors.	Completely removes trust in talliers for verifiability, ensuring that only valid claims are allowed.
Accountability	Limited accountability; a dishonest tallier can refuse to issue receipts, but it is detectable through external means.	Full accountability; any malicious actions by a tallier can be detected, as they are forced to justify claims with verifiable decryption.
Everlasting Privacy	Maintains privacy under realistic assumptions by preventing any leakage during the voting phase.	Requires trusting all talliers, as they have access to encrypted opening values, which may pose a risk to privacy if compromised.
Practical Complexity	Simpler; only requires adding receipt issuance to the protocol. However, auditors may need to verify whether receipts were issued.	More complex; requires maintaining a secret bulletin board and implementing verifiable decryption for privacy protection.

TABLE 3.1: Comparison of Solutions for Public Verifiability in [Cra+96]

Conclusion: Both solutions address the verifiability gap in [Cra+96], but with different trade-offs:

Cramer *et al.*	**Ge *et al.***	**Alternative**
V_i: $(v_i^j)_j \leftarrow \mathsf{Share}(v_i)$, $c_i^j \leftarrow \mathsf{Com}(prm, v_i^j, r_i^j)$, $\pi_i \leftarrow \Pi((prm, c_i), (v_i, r_i))$		
Send (v_i^j, r_i^j) to T_j	Send (c_i^j, v_i^j, r_i^j) to T_j T_j: $s_i^j \leftarrow \mathsf{Sign}(sk_j, c_i^j)$	$e_i^j \leftarrow \mathsf{Enc}(pk_j, (v_i^j, r_i^j))$
$b_i \leftarrow (i, (c_i^j)_j, \pi_i)$	V_i: $\lightning$ if $\mathsf{Verify}(vk_j, c_i^j, s_i^j) = 0$ $b_i \leftarrow (i, (c_i^j)_j, \pi_i, (s_i^j)_j)$	$b_i \leftarrow (i, (c_i^j)_j, \pi_i, (e_i^j)_j)$
Send b_i to PBB		Send b_i to SBB SBB: $h_i^j \leftarrow H(e_i^j)$ $b'_i \leftarrow (i, (c_i^j)_j, \pi_i, (h_i^j)_j)$ Send b'_i to PBB $\mathsf{V}_i, \mathsf{T}_j$: $\lightning$ if $h_i^k \neq H(e_i^k)$ T_j: $(v_i^j, r_i^j) \leftarrow \mathsf{Dec}(sk_j, e_i^j)$
T_j: $\lightning$ if $\mathsf{Open}(prm, c_i^j, v_i^j, r_i^j) = 0$ or $\mathsf{Verify}((prm, c_i), \pi_i) = 0$		
	$\lightning$ if $\mathsf{Verify}(vk_k, c_i^k, s_i^k) = 0$	

FIGURE 3.1: The voting phase in solutions based on secret-sharing (see Section 3.4.2.1). Share denotes the share algorithm of a secret-sharing scheme, H a (cryptographic) hash function, $\lightning$ a complaint on PBB.

- Ge *et al.*'s receipt-based approach is simpler and maintains privacy more robustly. However, it relies on external auditing to prevent talliers from ignoring valid submissions.

- The secret bulletin board solution removes trust in talliers entirely for verifiability and offers better accountability. However, it compromises everlasting privacy by exposing opening values to talliers and introduces additional implementation complexity.

3.4.2.2 B-ID-MIX

The protocols that fall under the B-ID-MIX class include the following papers: [BDG13; CPP13; Dem+13; DGS12; GHS20; Hai19; MN06; MN07; PR19]. These protocols employ the mix-net approach during the tally phase, which involves shuffling and re-randomizing voters' identifiable ballots while maintaining the secrecy of the permutation and randomness. To ensure public verifiability, this process is typically accompanied by a proof of shuffle.

In comparison to the homomorphic approach, the mix-net method offers a distinct advantage: it is capable of handling more complex ballots, which may be necessary for multi-option elections, whereas homomorphic approaches are typically limited to simpler yes/no-type ballots.

The idea of using mix-net to achieve verifiable e-voting with EP originates from Moran and Naor's work [MN06]. In this work, Moran and Naor use a zero-knowledge proof of shuffle of unconditionally hiding commitments to the voter's vote.

We categorize the protocols in this class based on the distribution of trust for privacy into three approaches: limited distribution of trust for computational privacy, arbitrary distribution of trust for computational privacy, and arbitrary distribution of trust for computational privacy. In the following sections, we evaluate the protocols in these latter three categories, highlighting their advantages, limitations, and suitability for real-world elections.

1. Limited distribution of trust for computational privacy

The original work of Moran and Naor is indeed an on-site e-voting system that is designed to perform the tally phase in one step, meaning that it has a single trusted tallier. To mitigate trust on the tallier, Moran and Naor improve their work to a new verifiable on-site e-voting, but this time with two talliers T_0, and T_1. They name this improved version *Split-Ballot*.

On the high level, Split-Ballot works as follows: Each tallier T_j computes a random mask t_i^j for voter V_i, uses an unconditionally hiding commitment to commit to it as $c_i^j := \mathsf{Com}(prm, t_i^j)$, and publishes these commitments on PBB. In the ballot submission phase, each V_i receives her masks t_i^j from T_j. She then uses these two masks to perfectly hide her vote as $s_i := v_i - t_i^0 - t_i^1$. Then submits her ballot $b_i := (i, s_i)$ on PBB. Before explaining the tally phase, note that the commitment to V_i's masked vote is: $\mathsf{Com}(s_i) = \mathsf{Com}(v_i - t_i^0 - t_i^1)$. Using the homomorphic property of the commitment, we have $\mathsf{Com}(s_i) = \mathsf{Com}(v_i) - \mathsf{Com}(t_i^0) - \mathsf{Com}(t_i^1)$ which is equal to $\mathsf{Com}(s_i) = \mathsf{Com}(v_i) - c_i^0 - c_i^1$. Considering this, the tally phase proceeds as follows: The talliers publicly compute $d_i = \mathsf{Com}(s_i) + c_i^0 + c_i^1$. As we explained earlier, this is indeed equal to $\mathsf{Com}(v_i)$. Next, the tallier T_0 privately shuffles the commitment list $(d_i)_i$. Next, T_1 shuffles the output of T_0, which results in the shuffled commitment vector $(\tilde{d}_i)_i$. Using their joint knowledge of the opening values of $(c_i^0)_i$ and $(c_i^1)_i$ they open the shuffled vector $(\tilde{d}_i)_i$, publish the result which is the plain shuffled votes of voters, together with an unconditional ZKP of correct shuffle and opening for the sake of public verifiability.

Evaluation of Split-Ballot

Privacy Guarantees: The Split-Ballot protocol offers strong privacy guarantees that have been formally analyzed in their paper:

- *Practical everlasting privacy*: Moran and Naor formally proved in the Universal Composability (UC) framework that the protocol ensures everlasting privacy.
- *Computational privacy under a single honest tallier*: They also proved their protocol provides computational privacy as long as at least one of the two talliers remains honest.
- *Public verifiability*: They also proved that it achieves public verifiability mainly through the unconditional ZKP in the tally phase.
- *Formal verification*: A simplified version of Split-Ballot has been analyzed using ProVerif by Arapinis, Cortier, Kremer, and Ryan [Ara+13], further demonstrating its strong privacy guarantees.

Limitations: Despite its robust privacy guarantees, Split-Ballot suffers from two significant limitations:

- *Design constraint for two talliers*: The protocol is explicitly designed for elections with exactly two talliers. This restriction limits its scalability and adaptability to scenarios requiring a higher degree of distributed trust or involving more talliers.

- *Restricted to on-site voting*: The protocol is suitable only for on-site (booth) voting and cannot be applied to remote (Internet) voting scenarios. This limitation reduces its applicability in modern elections where remote participation is increasingly demanded.

2. Arbitrary distribution of trust for computational privacy

Among the mentioned works in B-ID-MIX, the mixnet version of PPAT by Cuvelier et al. [CPP13] distributes trust among an arbitrary number of tallier (not just 2 as in Split-Ballot) for computational privacy. We discussed the homomorphic version in Section 3.4.2.1. The mixnet version also incorporates their new cryptographic scheme named CCE, but with a different mechanism to be consistent with the mixing tally.

The high-level idea is to shuffle the encrypted votes on SBB and the commitments to the opening values on PBB in parallel, using the CCE scheme for connecting the public and secret board. More precisely, the e-voting based on the mixnet version of PPAT works as follows: The ballot submission phase is the same as in the homomorphic version. In the tally phase, a set of mix-servers with access to SBB are involved: $\mathsf{M}_1, \ldots, \mathsf{M}_N$. Server M_1 uses the original encryption vector $\vec{e}_0 := (e_i)_i$ as an input, permutes and re-randomizes it secretly, and publishes the result as $\vec{e}_1$ on SBB. This will be the input of M_2 and so on. To connect the two boards, talliers use the CCE scheme to derive a committed vector $\vec{c}_k$ for each shuffled encryption vector $\vec{e}_k$, and as before, publish the commitments on PBB. Finally, each tallier T_j uses her secret key sk_j to compute the joint opening $(\vec{v}, \vec{r})$ of the last shuffled commitment vector $\vec{c}_N$, which will be published on PBB.

Now any observer could confirm the correctness of the final election result by using the Open algorithm: $\mathsf{Open}(\vec{c}_l, \vec{v}, \vec{r}) \overset{?}{=} 1$. Regarding the proof of shuffle, Cuvelier et al. use a NIZKP denoted by π_k to show that each encrypted vector $\vec{e}_k$ is a shuffled output of the mix server M_{k-1} on the input $\vec{e}_{k-1}$. The advantage of π_k is that it is CCE-compatible, meaning that from π_k, M_k can derive another NIZKP π'_k which proves $\vec{c}_k$ is output of M_k on the input $\vec{c}_{k-1}$. The proof of correct shuffle for encryptions (π_k) is published on SBB, however, the proof of correct shuffle for the commitments is published publicly on PBB. The talliers first verify the correctness of all these NIZKPs and then compute the final result. We note that this check is necessary, as a corrupted mix-server could manipulate the votes in the re-encryption phase so that it violates the privacy of a target voter. This check prevents this threat.

Evaluation of Mixnet Version of PPAT

We conjecture that the mix version of [CPP13] achieves the security properties they claim.

Privacy Guarantees:

- *Practical Everlasting Privacy*: It guarantees practical EP through publishing unconditionally hiding commitments and ZKP's that do not leak any information on the voters' votes, even with identifiable ballots.

- *Computational Privacy* The protocol ensures computational privacy under the assumption that at least one mix server and a threshold of talliers are honest.

- *Public Verifiability*: This is guaranteed by enabling the verification of the correctness of shuffled commitments which are linked to the secret encryption of the votes.
- *Accountability* This is guaranteed through the soundness of the ZKPs each entity has to publish during the different phases of the election.
- *Performance Feasibility*: Gjøsteen, Haines, and Solberg [GHS20] demonstrate that the protocol can be instantiated efficiently, with a performance comparable to state-of-the-art NIZKP of shuffle techniques [TW10]. This efficiency allows it to handle complex ballots, making it suitable for real-world elections with diverse requirements.
- *Extensibility*: We studied the following two extensions of this protocol and confirm that they achieve the additional properties they claim:
  1. Pereira and Rønne [PR19] adapt the protocol for special voting methods like quadratic voting in [LW14], expanding its applicability.
  2. Haines [Hai19] introduces a cast-as-intended (see Section 2.2.4) mechanism to mitigate trust on voting devices, increasing voter confidence and security.

Limitation: We did not identify any significant limitations or flaws in this protocol. With the availability of efficient implementations and formal verifications, we consider it a robust and practical solution for achieving everlasting privacy under realistic assumptions. Additionally, its ability to handle complex ballot types further enhances its applicability to diverse election scenarios.

3. Arbitrary distribution of trust for everlasting privacy

In 2013, around the same time the PPAT protocol was introduced, Buchmann et al. [BDG13] proposed a similar approach to the mixnet version of PPAT but with a more ambitious objective: They aim to distribute trust not only for computational privacy as is done among the mix servers, but also for EP. The security of the protocol was formally proven using cryptographic methods in the PhD thesis of Demirel [Dem13], a co-author of the paper [BDG13]. Additionally, Arapinis et al. employed automated tools to provide symbolic proof of everlasting privacy for the protocol, further reinforcing its credibility. The versatility of Buchmann *et al.*'s protocol is underscored by its successful application in other e-voting systems. For instance:

1. *Mix-Net Version of Helios*: The protocol was employed in [DGS12] to provide unconditional privacy enhancements to the mix-net version of Helios [Adi08].
2. *Prêt à Voter*: In [Dem+13], the protocol was adapted to strengthen privacy guarantees in the mix-net version of *Prêt à Voter* [RS06b], a voting system with a focus on voter verifiability and auditability.

In what follows, we explain how does the e-voting protocol in [BDG13] work.

Details of the e-voting protocol in [BDG13]

Cryptographic Primitives

The protocol relies on the following cryptographic primitives.

- *Commitment Scheme* ($\mathcal{C} = (\mathsf{KeyGen}_{\mathcal{C}}, \mathsf{Com}, \mathsf{Open})$):
  - *Properties:* Unconditionally hiding, computationally binding, and homomorphic.
  - *Purpose:* Allows voters to commit to their votes while keeping them hidden until they are revealed.
- *Public-Key Encryption (PKE) Scheme* ($\mathcal{E} = (\mathsf{KeyGen}_{\mathcal{E}}, \mathsf{Enc}, \mathsf{Dec})$):
  - *Properties:* IND-CPA-secure and homomorphic.
  - *Purpose:* Encrypts vote-related information to protect it during transmission and mixing.
- *Non-Interactive Zero-Knowledge Proofs (NIZKPs):*
  - *Correct Re-encryption Proof:* Verifies that the set of output values is a valid shuffle and re-randomization of the input values.
  - *Consistency Proof:* Ensures that the same permutation and randomness are used across re-randomizations and re-encryptions.
  - The exact relation between these two NIZKPs are not clear in their paper.

Protocol Phases

1. Voting Phase

1. *Vote Commitment:* Each voter V_i commits to her vote v_i as:
$$c_i \leftarrow \mathsf{Com}(prm, v_i, r_i)$$
where prm is a public parameter, and r_i is random.
2. *Encryption of Opening Values:* The voter encrypts the vote and randomness separately under the talliers' public key pk:
$$e_i^0 \leftarrow \mathsf{Enc}(\mathsf{pk}, v_i), \quad e_i^1 \leftarrow \mathsf{Enc}(\mathsf{pk}, r_i).$$
3. *Proof Generation:* The voter creates a NIZKP π_i to prove consistency between the commitment c_i and the encrypted values (e_i^0, e_i^1).
4. *Ballot Submission:* The voter V_i submits her ballot as $b_i := (c_i, e_i^0, e_i^1, \pi_i)$ to the first mix server M_1.
5. *Public Bulletin Board:* M_1 publishes the commitments c_i of all ballots it received as a vector C_0, while keeping the corresponding ciphertexts E_0^0 and E_0^1 secret.

2. Mixing Phase

1. *Input:* The first mix server M_1 takes (C_0, E_0^0, E_0^1) from voters. Subsequent servers M_k take the outputs from the preceding server M_{k-1}.
2. *Re-randomization:* Each mix server M_k re-randomizes:
   - Commitments $c_{k-1,i} \in C_{k-1}$ using fresh randomness $\rho_{k,i}$ and updates associated encrypted randomness $e_{k-1,i}^1 \in E_{k-1}^1$:
$$\tilde{c}_{k,i} \leftarrow \mathsf{Com}(prm, v_i, r_i + \rho_{k,i}), \quad \tilde{e}_{k,i}^1 \leftarrow e_{k-1,i}^1 + \mathsf{Enc}(\mathsf{pk}, \rho_{k,i}).$$

- Encrypted votes $e^0_{k-1,i} \in E^0_{k-1}$ by applying homomorphic re-randomization.

3. *Shuffling:* The server M_k shuffles the re-randomized data into a new order using a random permutation:

$$(\tilde{C}_k, \tilde{E}^0_k, \tilde{E}^1_k) \xrightarrow{\text{shuffle}} (C_k, E^0_k, E^1_k).$$

4. *Proof Generation:* Each mix server generates:
   - A proof of "correct re-encryption" to verify the shuffling and re-randomization process.
   - A proof of "consistency" to ensure the same permutation and randomness were used.

5. *Output:* The mix server M_k sends (C_k, E^0_k, E^1_k) to M_{k+1} via a private channel with a proof of consistency. In case that $k = N$, M_k sends the outputs to the talliers.

3. Opening/Decryption Phase

1. *Decryption:* The talliers collectively decrypt the encrypted votes E^0_N and encrypted randomness E^1_N from the final mix server M_N using their secret key shares sk:

$$V^* \leftarrow \mathsf{Dec}(sk, E^0_N), \quad R^* \leftarrow \mathsf{Dec}(sk, E^1_N).$$

2. *Publishing Results:* The talliers post the decrypted votes V^* and randomness R^* on the bulletin board PBB.

4. Verification Phase

1. *Correctness of Decryption:* It is verified whether (V^*, R^*) is a valid opening for C_N.

2. *Validation of Proofs:* All NIZKPs of correct re-encryption and consistency, published by the mix servers, are checked.

Evaluation of the protocol in [BDG13]

Privacy Guarantees: The e-voting protocol described in Buchmann et al.'s conference paper [BDG13] and Demirel's PhD thesis [Dem13] claimed to achieve the following privacy guarantees:

- *Everlasting Privacy*: In their work, EP relies on cryptographic commitments that are unconditionally hiding and computationally binding, as well as the use of homomorphic encryption to protect sensitive data.

- *Computational Privacy*: Under the assumption that at least one mix server and a threshold of trustees remain honest, the protocol aimed to provide computational privacy against adversaries with bounded resources.

- *Public Verifiability*: The use of non-interactive zero-knowledge proofs (NIZKPs) in the protocol was intended to provide public verifiability, allowing any observer to confirm the correctness of the election process, including the proper mixing, re-encryption, and decryption of votes.

Despite these claims, our analysis in [Hai+23] reveals critical flaws that undermine these guarantees, which we describe next.

Limitations: The protocol suffers from significant design flaws that render it incapable of achieving its stated privacy guarantees.

1. *Imprecise Protocol Specification*: Their protocol descriptions lack clarity in several important aspects, making it difficult to understand how certain steps are executed securely. For instance, they did not clearly explain how M_{k+1} can verify M_k's proof of consistency without having access to M_k's input. These ambiguities leave gaps in the protocol's security reasoning and allow for potential exploitation.

2. *Vulnerability to Mix Server Corruption*: The protocol does not ensure that mix servers share a consistent view of the private trail of votes. This design flaw enables adversarial mix servers to compromise voter privacy. To clarify, consider the following scenario: If the first mix server M_1 is corrupted, it can replace all ciphertexts except for the one submitted by a specific voter V_i. The final tally will then consist only of M_1's fabricated votes and V_i's vote, enabling M_1 to learn how V_i voted. More sophisticated attacks can be carried out to break the privacy of multiple voters by exploiting the homomorphic properties of the encryption scheme.

3. *Lack of Trust Distribution for Privacy*:

    - Contrary to their claims, the attack that we explained in the second limitation shows that the protocol does not effectively distribute trust among mix servers for either computational or everlasting privacy. This critical shortcoming disproves the privacy theorems in both the conference paper and the PhD thesis.
    - We mentioned the symbolic verification performed by Arapinis et al. earlier. The reason that they managed to prove EP holds for this protocol is that in their analyses, they considered a single mix-server in the e-voting system, overlooking the vulnerabilities introduced by multiple mix servers.

4. *Exploitation of Homomorphic Encryption*: The homomorphic nature of the PKE scheme used in the protocol allows adversaries to construct manipulated ciphertexts. These manipulated ciphertexts can reveal partial or complete information about individual votes, especially when combined with malicious behavior from mix servers (as demonstrated in [Mül22] by Müller).

Considering the aforementioned flaws and limitations, we conclude that the e-voting protocol proposed in [BDG13] cannot be regarded as a viable solution for achieving everlasting privacy under realistic assumptions.

3.5 Results

In this section, we present a comprehensive summary of our findings, emphasizing the security guarantees and limitations of each of the 25 papers in our database, as explored in detail in the preceding chapters. We begin by analyzing the relationship between the two primary classes, B-ID and B-ANON, in Section 3.5.1. This is followed by an evaluation of these classes, where we identify their respective strengths and weaknesses. Finally, in Section 3.5.3, we summarize the results of our examination of the 25 protocols, providing answers to the research questions posed at the beginning of this chapter. We refer the reader to our summary Table 3.2 at the end of this chapter, which includes all these 25 papers together with their trust assumptions, limitations, and the security properties that they achieve.

3.5.1 Relation

Our study reveals a key relationship between the two classes, B-ANON and B-ID. These two classes differ fundamentally in how they achieve the core properties of *secure* e-voting: privacy and (public) verifiability. Specifically, the techniques employed to achieve everlasting privacy (EP) and the mechanisms for ensuring verifiability differ between the two.

Privacy

The distinction in achieving privacy lies in the specific methods used to ensure EP: In B-ANON, the ballots are anonymous and contain no information that could link them to the voter who cast each. This setup necessitates eligibility verifiability, ensuring that ballots originate from legitimate voters while preserving voter anonymity. Therefore, achieving EP in this class boils down to the method used to prove eligibility verifiability and unconditional anonymity of the ballot submission channel that the voters use to cast their votes. These two components, when combined, result in a secure voting system with EP.

In contrast, in B-ID, the ballots are identifiable and can be linked to their corresponding voter. The voters use a private channel to communicate with the EA or Ts. If this channel is not unconditionally private, a third party or adversary could break ballot privacy. At the same time, to achieve EP in this class, the privacy-preserving techniques that are used to tally the ballots need to be unconditionally privacy-preserving, meaning not to reveal the link between the voter and her vote by tallying the ballot using either a homomorphic tally approach or a mixnet tally approach.

Therefore, in B-ID, achieving EP breaks down to the unconditionally privacy-preserving technique used to tally ballots and unconditional privacy of the communication channel between voters and the internal parties. We also observe that the link between ID_i and b_i should break before the ballot submission in B-ANON since the info on PBB will not be safeguarded further, while in B-ID, this link is broken in the tally phase. Thus, the first difference is in the method they use to achieve EP, and also the election phase in which they break the individual links.

Verifiability

The next difference is in the technique used in each class to ensure verifiability. As we explained in the B-ANON Section 3.4.1 earlier, ensuring verifiability in this class breaks down to the method used to prove the eligibility of anonymous ballots. However, verifiability in B-ID follows from the correctness of the techniques used to tally the ballots.

Therefore, we conclude that these two classes are different in two main aspects: First, the methods they use to ensure EP (where they get their names from) and subsequently, the election phase they break the individual links. Second, the approach they take to ensure verifiability.

3.5.2 Basic Evaluation of B-ANON and B-ID

Based on our studies, and as we also mentioned while describing each protocol in each sub-class, we discovered that the general approach taken in B-ID to ensure EP is preferred to the one in B-ANON, or loosely speaking,

$$\text{B-ID} > \text{B-ANON}.$$

In what follows, we justify this statement.

As we discussed in the Relation Section 3.5.1 and also saw earlier in the protocol descriptions, the protocols in B-ANON require an unconditionally anonymous ballot submission channel, while the protocols B-ID need an unconditionally private but not necessarily anonymous channels in the system. There is a significant gap between the two, as we explain next. In fact, there are two significant differences between the two: adversarial power and mitigation techniques.

- **Adversarial Power**: As we explained in the introduction of this Chapter 3.1 and also Introduction of Chapter 1, perfect secrecy is only achievable in the private-key (symmetric) setting, and not in the public-key setting. However, the symmetric setting is impractical in the context of electronic (internet-based) voting. This simply means that both these unconditionally private and unconditionally anonymous channels cannot be constructed for use in e-voting systems. Instead, the unconditional assumption of these channels follows from the adversarial power assumptions: he cannot break privacy in the first and anonymity in the second one.

  Now, the question is which assumption is stronger and which one is more realistic to hold? To answer this question, consider the following fact. The widely recognized truth is that there are multiple organizations in the world that monitor the internet and store the traffic, but since storing all the communication data is infeasible in terms of storage and costs, they store the relevant meta-data. If the meta-data related to today's election is accessible to the future adversary, this could instantly break the unconditional anonymity assumption of the channel; however, if just meta-data is breached, the privacy of the channel still holds. Therefore, it is more realistic to make the assumption that the adversary cannot break privacy of the channel, than its anonymity.

- **Mitigation Techniques**: We mentioned that unconditionally private and unconditionally anonymous channels could not be constructed, however, there are ways to realize each one in practice: For private channels, we can use post-quantum TLS which is easily deployable in real-world elections, for example the one in [SSW20] without the voters needing to install anything or put any effort to use it. For anonymous communication channels, it is not as widely used as TLS and we cannot assume that the majority of voters can have access to it. In other words, the joint anonymity of the voters who utilize it, is not large enough to protect the anonymity of the rest. Our anonymous reviewer for the PETS conference suggested that this could be mitigated as follows: the meta-data is not enough to link individual votes to the voters if all the ballots are published together at the end of the submission phase. However, this violates the vote-and-go property since the voters need to stay after they cast their ballots for the sake of verifiability.

These two reasons together justify our statement about these two classes: B-ID $>$ B-ANON.

3.5.3 Answers to the research questions

We raised four research questions, for which we provide answers here based on our study. Before that, we would like to summarize our findings per sub-class that we considered in each of the B-ANON and B-ID classes.

Summary of Our Findings

In this part, we summarize our findings in each sub-class.

- **B-ANON-A**: We categorized the protocols in this subclass into three approaches: no identities, blind signatures, and pseudonyms. Our study shows that all the protocols in this sub-class put much trust in the EA who is responsible for generating voters' credentials. However, if the EA is corrupted, among all, it has the lowest impact on Belenios [CGG19]. It is an open problem to clarify to which degree EA needs to be trusted for EP, or remove the trust on EA completely. In total, we consider Belenios as a reasonable solution in B-ANON class, under the assumption of unconditionally anonymous communication channel.
- **B-ANON-V**: We categorized the protocols in this sub-class into two approaches: linkable ring signature, and membership ZKP. We identified the [LH15] protocol as a practically efficient solution to EP in B-ANON under the assumption of unconditionally anonymous communication channel. As we described in Section 3.4.1.2, all the other protocols in this subclass are inefficient for large-scale elections [HB16] or not robust [LH16; LHK16].
- **B-ID-HOM**: For secure e-voting with everlasting privacy and simple ballot types, secret-sharing and distributed decryption both serve as practical solutions. Secret-sharing, however, has distinct advantages: it supports a broader range of cryptographic primitives by using CCE, potentially enabling future enhancements such as receipt-freeness (see Section 3.6 for an update state-of-the-art), and it may provide stronger privacy guarantees if trustees' data is compromised by future adversaries.

  The primary open challenge remains the formal verification of these approaches to substantiate their security claims and address existing conjectures.
- **B-ID-MIX**: Among the protocols in this sub-class, the mixing variant of PPAT [CPP13] is unique in its ability to distribute computational privacy across any number of talliers truly. Other methods, by comparison, require either full trust in all talliers or allow for minimal exceptions. A key unresolved issue here is conducting a rigorous formal analysis to confirm the security properties of the PPAT protocol.

To conclude this section, we share our key-insights here which combined with the above-mentioned discussions, answers all the questions raised in the beginning of this chapter.

Solved Problems

Summarizing the above-mentioned findings results into the following conclusion: Both classes B-ID and B-ANON contain reasonable solutions under their respective class assumptions, mentioned earlier, to design a secure e-voting that achieves EP.

For simple ballot types, there are only solutions in the B-ID class: the approach based on Cramer et al. secret-sharing [Cra+96], and the homomorphic version of PPAT in [CPP13]. The first one offers EP toward the public and a threshold of talliers, while the next one offers EP only toward the public (practical EP).

For complex ballot types, there are solutions in both classes: In B-ID, mixnet version of PPAT in [CPP13], and in B-ANON, Belenios [CGG19] and [LH15], where both offer practical EP.

All the above-mentioned approaches are efficient for large-scale elections, and particularly, Belenios has been deployed and widely used in numerous elections.

Open Problems

1. **Formal Protocol Analysis**: The promising approaches that we mentioned above, namely [Cra+96; CPP13; CGG19], need to be formally analyzed on the protocol level than just the cryptographic level. Particularly, formal analysis of EP in Belenios needs to be done.

2. **Deployable E-Voting System**: Among the above-mentioned promising protocols, only Belenios is B-ANON has been deployed. It is an open problem to develop full-fledged deployable versions of the other promising approaches in the preferred class, B-ID.

3. **Weaker Trust for Arbitrary Ballot Types**: The three promising approaches that can handle complex ballots, namely [CPP13; CGG19; LH15] need full trust in the EA or all talliers to achieve EP. It is an open problem to mitigate this trust.

4. **Receipt-Freeness**: None of the promising approaches achieve RF in addition to EP. It is an open problem to design a secure e-voting system with both these properties (see Section 3.6 for an update on this previously open problem)

3.6 Post-Publication Updates

As previously mentioned in this chapter, we collected 25 papers while conducting research on secure electronic voting systems that offer everlasting privacy (EP). Following the submission of our paper in 2022 to the Privacy Enhancing Technologies Symposium (PETS) and its subsequent acceptance and publication in 2023, several new e-voting systems with EP were introduced in the literature. Leaving out the decentralized voting systems using blockchain technology and MPC ones, there are six new e-voting systems with EP in the literature: [SGS23; Mos+24; Poi24; MPP24; Cor+24; DPP24]. Here, we briefly explain to which class these six protocols belong and the additional features they provide beyond EP:

- [SGS23]: **Receipt-Free Electronic Voting from zk-SNARK**
  Following the footsteps of Locher and Haenni in [LH16], Sheikhi et al. design an e-voting system that provides the same security features, namely verifiability, and receipt-freeness in addition to EP. However, as we discussed before in this chapter, the e-voting protocol in [LH16] is not robust. Sheikhi et al.

solve this issue by introducing a new cryptographic primitive while providing a pen-and-paper proof that it still preserves those properties. They still use an anonymous ballot submission channel, with the difference that now the voter's pseudonym is included in the ballot, while a commitment to this pseudonym is published in the setup phase next to the voter's ID. As a result, their e-voting system belongs to the B-ID class.

- [Mos+24]: **Direct and Transparent Voter Verification with Everlasting Receipt-Freeness**: In this work, which is the subject of Chapter 4, Mosaheb et al. design an e-voting system which adds everlasting privacy feature to Hyperion, explained in Appendix 6.2. They also introduce the new feature of everlasting receipt-freeness. They use cryptographic tracking terms inherited from Hyperion and redesign them to be compatible with the mixnet version of PPAT to achieve EP while preserving both individual and universal verifiability and achieving the new feature of everlasting RF. Therefore, everlasting Hyperion belongs to the B-ID class.

- [Poi24]:**Efficient Universally-Verifiable Electronic Voting with Everlasting Privacy**
  Pointcheval in [Poi24] uses the best identifiable approach in the B-ID class of protocols, i.e., the homomorphic version of PPAT, while he extends this approach to accept complex ballots too, unlike before. Pointcheval makes the proof of consistency and validity in PPAT more efficient by using a linearly homomorphic signature with randomization tags, namely an FHS signature which allows to provide strong receipt-freeness too. In summary, he claims his approach offers universal verifiability, and strong recept-freeness in addition to EP.

- [MPP24]: **DeVoS: Deniable Yet Verifiable Vote Updating**
  Müller et al. design an electronic voting system named DeVoS which aims to mitigate coercion by introducing a new party named "posting trustee PT" which is responsible for updating the vote encryption vectors by re-randomizing the encryption of votes (in case of mixnet tally) or using additively homomorphic feature of the PKE to update the vector of encrypted votes which looks like a dummy vote making it difficult to spot who updated the vote or not. This basic version of DeVoS is then combined with the PPAT approach (compatible with both versions) to achieve EP. Therefore, DeVoS belongs to the B-ID class. Müller et al. claim DeVoS provides public verifiability, coercion mitigation (deniable vote updating approach), and strong participation privacy in addition to EP.

- [Cor+24]: **Election Eligibility with OpenID: Turning Authentication into Transferable Proof of Eligibility**
  Cortier et al. turn the pure OpenID Connect into an OpenID-Eligibility protocol, which is a privacy-preserving proof of eligibility (a signature), in order to design an e-voting system that provides eligibility, individual and universal verifiability in addition to EP. This helps PBB to prove eligibility and prevent a malicious PBB from ballot stuffing. Using an OpenID-EligibilityZK-ID which is a ZKP of eligibility provided by the PBB, the signature keeps the identity of the voters secret while proving their eligibility. This technique could potentially help to achieve EP based on it applies to which underlying e-voting system. Cortier et al. apply this technique to Helios-like voting protocols (also

Belenios) under standard trust assumptions. Therefore, their e-voting protocol still classifies in B-ANON, while they achieve an extra feature here: voter eligibility.

- [DPP24]: **Encryption Mechanisms for Receipt-Free and Perfectly Private Verifiable Elections**
Doan et al. designed the first e-voting system with everlasting privacy that is universally verifiable and provides receipt-freeness too. They build their design based on the PPAT and use their new traceable receipt-free encryption (TREnc) which is compatible with the PPAT approach, to add the RF feature to the system. The trace still doesn't let the voters prove to a coercer or vote buyer how they voted. As they designed two encryption schemes where one supports homomorphic tally and the other supports mixnet tally, the e-voting systems combined with their mechanism (e.g., Helios [Adi08] in the first case, and Verificatum [Wik11] in the second case) belong to the B-ID class.

Sub-classes	Protocols	Distinguishing feature ($*$) or used crypto primitive (†)	Public Verifiability	Everlasting Privacy	Additional Properties	Practical Limitations
B-ANON-A (Sec. 3.4.1.1)	[Ara+13]	$*$ No identities	$\mathcal{T}_{\mathsf{EA}}$	$\mathcal{T}_{\mathsf{EA}}$		
	[FOO92]	† Blind signatures	$\mathcal{T}_{\mathsf{EA}}$	+		$\mathcal{AS}$
	[Kai+21]		$\mathcal{DT}_{\mathsf{EA}}$	+		Flawed crypto, $\mathcal{AS}$
	[Gra10]		$\mathcal{T}_{\mathsf{EA}}$	+		All voters online, not robust
	[Gro+18]		$\mathcal{T}_{\mathsf{EA}}$!	$\mathcal{T}_{\mathsf{EA}}$!	CR($\mathcal{T}_{\mathsf{EA}}$)	Small-scale elections, $\mathcal{AS}$
	[CGG19]	$*$ Pseudonyms	$\mathcal{T}_{\mathsf{EA}}$	$\mathcal{DT}_{\mathsf{EA}}$		
	[Iov+17]		$\mathcal{T}_{\mathsf{EA}}$	$\mathcal{T}_{\mathsf{EA}}$	CR($\mathcal{T}_{\mathsf{EA}}$)	Small-scale elections
	[Que+20]		$\mathcal{T}_{\mathsf{EA}}$	$\mathcal{T}_{\mathsf{EA}}$!	CR($\mathcal{T}_{\mathsf{EA}}$)!	
B-ANON-V (Sec. 3.4.1.2)	[HB16]	† Linkable ring signatures	+	+		Small-scale elections
	[LH15; Kil+21; LH16; LHK16]	† NIZKP	+	+	RF(+) [LH16], CR(+) [LHK16]	Not robust [LH16; LHK16]
B-ID-HOM (Sec. 3.4.2.1)	[DGS12; Gra09]	$*$ Single tallier	+	$\mathcal{T}$		
	[CPP13]	† Threshold decryption	+	$\mathcal{T}$	ACC(+)	
	[Cra+96; Ge+19]	† Secret-sharing	$\mathcal{T}_{\mathsf{T}}$ [Cra+96], + [Ge+19]	$\mathcal{DT}$		
B-ID-MIX (Sec. 3.4.2.2)	[MN06; MN07]	$*$ Limited $\mathcal{DT}$ for Comp. P	+	$\mathcal{T}$		On-site voting
	[CPP13; GHS20; PR19; Hai19]	$*$ Arbitrary $\mathcal{DT}$ for Comp. P	+	$\mathcal{T}$	ACC(+)	
	[BDG13; Dem13; DGS12; Dem+13]	$*$ Arbitrary $\mathcal{DT}$ for Everl. P	+	$\mathcal{T}$!		

TABLE 3.2: Overview on the classification of e-voting protocols with everlasting privacy. Security properties not relying on any trust assumption are marked $+$. Revised trust assumptions are marked !. Trust assumptions are denoted as $\mathcal{T}$ for full trust, $\mathcal{DT}$ for distributed trust; trust to specific parties is denoted in subscript. Additional security properties are marked as CR for coercion-resistance, RF for receipt-freeness, ACC for accountability, followed by trust assumptions in parenthesis. $\mathcal{AS}$ denotes necessity of anonymous submission channel for privacy.

Chapter 4

Direct and Transparent Voter Verification with Everlasting Receipt-Freeness

4.1 Introduction

As discussed in Section 2.2, achieving individual security properties in electronic voting is relatively straightforward. However, one of the main challenges lies in simultaneously ensuring two or more security properties, such as verifiability and coercion-resistance. As explained in Section 2.2, verifiability ensures that voters can confirm their votes have been correctly included in the final tally, while coercion-resistance prevents a voter from being forced to reveal how they voted or from complying with a coercer's demands. Successfully balancing these two objectives is essential for building trust in e-voting systems, however, achieving both in tandem presents significant challenges.

The primary reason that balancing verifiability and coercion-resistance is difficult lies in the conflict of transparency and secrecy inherent in each goal. Verifiability requires a certain level of transparency in the voting process to allow voters and observers to verify that each vote has been correctly counted. This transparency often involves giving voters a receipt or access to an audit trail that they can use to check whether their vote was recorded as cast. However, this same transparency can be exploited by a coercer. If a voter can prove how they voted, a coercer can force them to provide such proof, undermining the voter's ability to cast their vote freely.

Conversely, coercion-resistance depends on preserving the secrecy of the vote, ensuring that no one, not even the voter, can convincingly prove how they voted. This allows the voter to deny their real vote in the face of coercion. However, a high level of secrecy can make it difficult to provide verifiable evidence that all votes were counted correctly, thereby undermining confidence in the election results. Therefore, the challenge lies in designing a system that provides sufficient transparency to ensure verifiability, while still protecting voters from being able to prove their vote to a coercer.

The principal aim of E2E verifiability, as discussed in Section 2.2, is to leverage modern cryptography to allow a voter to confirm that her vote has been accurately counted in the final tally, while ensuring that no third party or coercer, regardless of her cooperation, can determine how she voted. This ensures the integrity of the election results, while also protecting the voter's privacy in the presence of a coercer.

On the other hand, in many electoral scenarios, as outlined in Chapter 3, voter privacy needs to be safeguarded not just during the election process, but also in the future. This is especially important given the rapid advances in computational power and the potential emergence of quantum computing. These developments could render current cryptographic techniques vulnerable, allowing adversaries to break traditional cryptographic schemes that are considered secure today, mainly the ones whose security breaks down to the hardness of factoring problems and discrete logarithm. Therefore, it is crucial to protect voter privacy against future adversaries with more advanced computational resources as well. The concept of everlasting privacy was introduced by Moran and Naor [MN06] to address this concern. Everlasting privacy guarantees that even computationally unbounded adversaries, who have unlimited resources in the future, will not be able to compromise the privacy of voters. This concept envisions a scenario where, even if an adversary can break traditional cryptographic schemes, they will still be unable to learn how a voter voted.

As elaborated in our systematization of knowledge (SoK) on everlasting privacy in Chapter 3, a more practical adaptation of this idea, termed practical everlasting privacy, was later introduced in [Ara+13]. This concept recognizes that future adversaries may not have access to all information used in the election, but only to the publicly available data, such as that posted on the bulletin board for tally verification. Hence, practical everlasting privacy limits the adversary's ability to compromise voter privacy by restricting their access to this publicly available information, thus offering a more feasible version of everlasting privacy in real-world scenarios.

In this chapter, based on our paper [Mos+24], we propose a protocol that achieves the three above-mentioned key goals simultaneously: E2E verifiability, everlasting privacy, and coercion mitigation against a computationally unbounded coercer. In addition, we propose a new security property, everlasting receipt-freeness, and prove that our protocol achieves it. Our protocol builds on Hyperion [Dam+24], detailed in Section 2.2 of this thesis as well, which is an electronic voting system designed to provide transparent verifiability and strong coercion-resistance against a limited coercer. However, Hyperion does not offer everlasting privacy, leaving it vulnerable to future computational advancements that could threaten voter privacy.

To enhance the security properties of Hyperion, we extend its capabilities to provide everlasting privacy while also increasing the coercer's potential power. In particular, we assume that the coercer can interact with the voter at any point, including before the casting phase, allowing for a broader range of coercion scenarios. By considering coercion before the voting process begins, our protocol ensures higher levels of coercion-resistance and robustness in adversarial environments.

In summary, our contribution is the design of a protocol that not only offers transparent verifiability but also ensures everlasting privacy against future adversaries and introduces the new feature of everlasting receipt-freeness in the literature. By addressing the limitations of existing systems, such as Hyperion, our work moves towards more secure and future-proof electronic voting systems.

Structure of the Chapter. This chapter is divided into six sections. We start by the literature review in Section 4.2. In Section 4.3, we continue by introducing the preliminaries, explaining the notations, identifying the parties involved in our e-voting system, and describing the game-based security proofs preliminaries. Section 4.4, outlines the cryptographic primitives that we used, followed by a detailed explanation of the protocol in section 4.5. Section 4.6 focuses on the voter's perspective within the proposed system. Finally, in Section 4.7, we present the game-based proofs of the security properties.

4.2 Literature Review

As mentioned in Hyperion [Dam+24], most of the verifiable voting systems that aim to achieve end-to-end verifiability (see Section 2.2.5) require the voter to confirm the presence of their encrypted/committed vote on PBB. This provides recorded-as-cast verifiability, however, the voters still need to confirm cast-as-intended, meaning that they believe their masked vote is created correctly, which is complicated for an ordinary voter in terms of usability. This leads to the importance of designing a system that is both direct and transparent, in the sense that voters can verify their vote in plaintext without any difficulty, but the system still needs to provide the necessary security guarantees, mainly ballot privacy (no link between the plaintext votes and the voter). Tracker-based voting systems solve this problem by offering verification in plaintext, but then the problem of coercion arises here. It is important when the voter receives her tracker, since it affects coercion. Selene [RRI16] delays the tracker return to the voters to the verification phase.

In our paper [Hai+23], which is the main subject of Chapter 3 and is discussed there in detail, we had an extensive survey on e-voting systems with everlasting privacy (EP). At the time of conducting research, we identified that designing an electronic voting system that satisfies both EP and receipt-freeness without the use of anonymous submission channels is an open problem. Later, a few works offering solutions to this problem were proposed in the literature. As we mentioned in Section 3.6, among these post-publication e-voting protocols offering EP, the following ones offer RF too: [SGS23], [Poi24], and [DPP24]. We mentioned in Chapter 3 that the approaches achieving EP based on privacy-preserving techniques (B-ID) are preferred to the ones based on anonymous channels (B-ANON). Among the mentioned papers, [Poi24] and [DPP24] are in B-ID. In [Poi24], individual verifiability is achieved through checking the pair of voter identity and Pedersen commitment (see

Example 2.4) (ID_i, c_i). In [DPP24], to offer RF on top of EP, they replace ElGamal encryption in CCE and replace it by a traceable receipt-free encryption (TREnc) scheme that is compatible with PPAT. This encryption provides a trace t, independent of the voter's vote, which enables the voters to find their vote on PBB. The trace t is a cryptographic verification key. In this approach, the voters receive the trace t (required for individual verifiability) upon creating their ballot, hence it offers no protection against a coercer.

In our work [Mos+24], which we explain in detail in this chapter, we combine the PPAT approach with Hyperion. This results in an e-voting system which is direct and transparent, offers both individual and universal verifiability, everlasting privacy, everlasting receipt-freeness and coercion mitigation.

4.3 Preliminaries

This section presents two sets of notations: one for the cryptographic primitives explained in Section 4.4 and another for the game-based proofs detailed in Section 4.7. Following this, we provide an overview of the parties involved in our proposed e-voting scheme, along with an explanation of the algorithms and oracles used in Section 4.4.

4.3.1 Notation

The first set of notations used in Section 4.4 are as follows: vectors are represented by bold letters, e.g., $\mathbf{e}$ and $\mathbf{E}$, while algorithms and the involved parties are denoted in sans-serif, for instance, $\mathsf{Shuffle}$ and Adm. The blackboard bold style $\mathbb{G}$ represents a cyclic group of prime order q, with specific groups labeled as $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. The corresponding group generators are denoted as $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$. Public and private keys are indicated as pk and sk, possibly indexed to specify the key owner. The set of all permutations of size n is noted as $\mathcal{S}_n$, with individual permutations represented by π and their inverses by π^{-1}. We use Π to denote a zero-knowledge proof, P for a Commitment Consistent proof (refer to Section 4.4), and σ to indicate the voter's signature. Last but not least, we express the security parameter using λ.

The second set of notations used in Section 4.7 are as follows: we indicate the uniformly random assignment of a value a to a variable x with $x \leftarrow_{\$} a$. We use $\top$ to indicate valid (accept) and $\perp$ to indicate invalid (reject) output when we run an algorithm. Adversary is denoted by $\mathcal{A}$ and voting system is denoted by $\mathcal{V}$. β is a binary value in the game, where 0 represents the real and 1 represents the simulated world. An oracle is shown by $\mathcal{O}$ and there are six oracles, used throughout all the games, name $\mathcal{O}\mathsf{board}$, $\mathcal{O}\mathsf{cast}$, $\mathcal{O}\mathsf{voteLR}$, $\mathcal{O}\mathsf{tally}$, $\mathcal{O}\mathsf{verify}$ and $\mathcal{O}\mathsf{getReceipt}$. We define each oracle later in the corresponding games in Section 4.7. The identities of voters V are divided into honest (H) and dishonest (D) with public and secret credentials shown by PU and U respectively. The honest voters who have to verify their votes are shown by $\mathsf{H}_{\mathsf{check}}$, the ones who verify in practice by $\mathsf{Checked}$ and the ones who successfully verify by Happy. We show the stored data for verification by state.

4.3.2 Parties

The e-voting protocol is executed by the following parties.

Voters (V): n voters $\mathsf{V}_1, \mathsf{V}_2, \ldots, \mathsf{V}_n$ with identities $\mathsf{ID}_1, \mathsf{ID}_2, \ldots, \mathsf{ID}_n$.

Election Admin (Adm): responsible for checking the validity of submitted ciphertexts by V, and relaying information between V and SBB.

Election Authority (EA): responsible for generating the election parameters.

Talliers (T): k talliers $\mathsf{T}_1, \mathsf{T}_2, \dots, \mathsf{T}_k$ who threshold share the secret key and cooperate to decrypt the ciphertexts.

Mix-servers (M): N mix servers $\mathsf{M}_1, \mathsf{M}_2, \dots, \mathsf{M}_N$ responsible for the mixing part of the tally phase.

Bulletin Board (PBB and SBB): similar to [CPP13], we assume two secure broadcast channels: PBB is the public board for all participants and SBB is the secret board shared with the talliers T and mix-servers M. All designated participants share the same view of each board.

4.3.3 Algorithms

$\mathsf{Publish}$: gives the data of the secret and public bulletin boards to $\mathcal{A}$.

Sim (Simulator): an efficient algorithm that creates a valid proof for a statement $y \in Y$ without access to the (secret) witness $x \in X$.

$\mathsf{VerifyTally}$: verifies the pair (res, Π) which is the election result and tally (including all the outputs and proofs) respectively.

$\mathsf{ValidBoard}$: using this algorithm, internal parties can verify the private board and voters or any third party can verify the public board (shown by BB in the game).

$\mathsf{Recover}$: this algorithm using the set of honest ballots and the board that is going to be tallied, can detect how $\mathcal{A}$ has modified the ballots of voters H.

Setup: the election setup with the security parameter λ outputs a pair of public and secret data as (pd, sd). We sometimes refer to pd as prm.

$\mathsf{Register}$: upon registration of V with identity $id \in \{ID_1, \dots, ID_n\}$, she receives her public and private credentials as (pc, sc). In the protocol description in Section 4.5, we refer to this credential as (pk, x). This pair is indeed the trapdoor (or Hyperion) keys of the voter.

4.4 Cryptographic Primitives and Assumptions

In this section, we provide a detailed explanation of the cryptographic primitives utilized in our proposed e-voting scheme, as outlined in Section 4.5.

Digital Signature

We consider a public key signature scheme with existential unforgeability under chosen message attacks (EUF-CMA) [GMR88]. This property ensures that an adversary cannot generate a signature for any message that has not already been signed by the rightful owner of the secret key. As described in Chapter 2, a digital signature scheme is represented by a set of three algorithms, denoted by the tuple $\mathcal{S} = (\mathsf{KG_s}, \mathsf{Sign}, \mathsf{Verify_s})$. We refer the reader to Chapter 2 for further details on these algorithms.

Non-interactive Zero Knowledge Proofs of Knowledge (NIZKPoK)

To prove knowledge of a secret s, we use the Schnorr proof [Sch91] and combine it with (strong) Fiat-Shamir transformation [BPW12] to achieve a non-interactive proof system. We then use the notation NIZKPoK(s) for the combination. The voters in our scheme use this proof as Π to show knowledge of a secret x to the Adm. The proof Π is non-malleable and bound to the identity of the voter.

Elliptic Curve

Similar to [CPP13], a bilinear map (see Section 2.3) in SXDH[1] setting is denoted by $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ where there are no efficiently computable homomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$. In this setting, the two following problems are computationally intractable in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$. The security of our protocol relies on this assumption.

- Decisional Diffie-Hellman (DDH): given g^{x_1} and g^{x_2} for uniformly independently chosen $x_1, x_2 \in \mathbb{Z}_q$, $g^{x_1 x_2}$ is indistinguishable from a random value in $\mathbb{G}$.
- Computational 1-Diffie-Hellman Inversion (1-DHI): given $g^x \in \mathbb{G}$ with $x \in \mathbb{Z}_q$, it is intractable to compute $g^{1/x}$.

Threshold ElGamal Encryption (TEG)

We use the basic IND-CPA threshold ElGamal cryptosystem in [CGS97; FP01] to encrypt a message m in group $\mathbb{G}_1$ with generator g. Let sk be the encryption secret key and $pk := g^{sk}$ the public key. Using the $(t,k)-$threshold Shamir secret sharing (SSS) scheme defined in 2.21, we split sk into k shares: $\mathsf{SSS}(sk) = (sk_i)_{i=1}^{k}$, distributing them among the shareholders T_i. Each T_i then computes their verification key $vk_i := g^{sk_i}$. The Enc and Dec algorithms work as follows:

- $\mathsf{Enc}_{pk}(m; r) = (m \cdot pk^r, g^r)$, with $r \in \mathbb{Z}_q$.
- To decrypt a ciphertext $(a, b) \in \mathbb{G}_1^2$, each T_i takes the following steps:
  - Compute $b_i := b^{sk_i}$.
  - Use vk_i to prove $\log_g^{vk_i} = \log_b^{b_i}$.
  - Without loss of generality, assume the log equality holds for any index i in $S_t := \{1, \ldots, t\}$.
  - Compute Lagrange coefficients in Lagrange interpolation $sk = \sum_{i=1}^{t} sk_i \cdot \lambda_{0,i}^{S_t}$ as $\lambda_{0,i}^{S_t} = \prod_{j \in S_t \setminus \{i\}} \frac{j}{j-i}$.
  - Compute $\prod_{i \in S_t} b_i^{\lambda_{0,i}^{S_t}} = \prod_{i \in S_t} b^{sk_i \cdot \lambda_{0,i}^{S_t}} = b^{sk}$.
  - Derive $\mathsf{Dec}_{sk}(a, b) = a / b^{sk} \bmod q$.

[1]Symmetric External Diffie–Hellman (SXDH) assumption: when DDH assumption, explained in the text, holds in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

NM-CPA (Non-Malleable Chosen Plaintext Attack) Security

As originally defined by Dolev–Dwork–Naor [DDN91] and widely adopted in the literature, an encryption scheme is considered non-malleable under CPA attack model if the following holds: After interacting with the oracles defined by CPA, when the adversary is given a challenge ciphertext c corresponding to a message m of their own choice, they are unable to produce — despite any further interactions with the oracles — a ciphertext c' whose plaintext m' has a nontrivial binary relation (i.e. $\sim$) with m (the trivial relation $m \sim m$ is excluded from consideration). For a formal definition of this security notion and its relation to other security notions, we refer the interested reader to [DDN91; Bel+98].

Commitment Consistent Encryption (CCE)

We use the Perfectly Private Audit Trail for Complex ballots (PPATC), which is the mix-net version of CCE proposed in [CPP13]. Let $m \in \mathbb{G}_1$ be a message and pk_T be the talliers' joint public key. Given a ciphertext $e = \mathsf{Enc}_{pk_T}(m)$, the goal is to derive a commitment $c = \mathsf{Com}(m; r)$ with a randomness $r \in \mathbb{Z}_q$ to the same encrypted message m. As explained in [CPP13], for simplicity, the opening value of the commitment is considered to be r instead of (m, r). To clarify, consider the talliers hold the following keys: $sk_T = (x_1, x_2) \in \mathbb{Z}_q^2$ and $pk_T = (g_1 := g^{x_1}, g_2 := g^{x_2}) \in \mathbb{G}_1^2$. The public set of parameters is $prm := \{q, \mathbb{G}_1, \mathbb{G}_2, g, h, h_1\}$ where g and h are the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively and $h_1 \in \mathbb{G}_2$. Let $\mathbf{r} = (r, r_1, r_2) \in_R \mathbb{Z}_q^3$ be the randomness. The CCE ciphertext, shown by $\mathsf{CCE}(prm, m; \mathbf{r})$ is a tuple $(\mathbf{e}, \mathbf{c}) = (e_1, e_2, e_3, c_1, c_2)$ where $e_1 := g^{r_1}$, $e_2 := g^{r_2}$, $e_3 := g_1^r g_2^{r_2}$ and $c_1 := h^r h_1^{r_1}$, $c_2 := m g_1^{r_1}$. In fact, $\mathbf{e}$ is the TEG encryption of the opening value $\mathbf{r}$, and $\mathbf{c}$ is the desired commitment to the message m. Throughout this chapter, for the sake of simplicity, we sometimes use $\mathbf{cce}$ instead of $(\mathbf{e}, \mathbf{c})$.

In our protocol, we employ three additional algorithms from [CPP13] for the CCE ciphertext: Dec_c, Open, and Verify_c. The subscript c signifies that these algorithms belong to a commitment scheme. They work as follows.

- $\mathsf{Dec}_c(sk_T, \mathbf{cce}) := c_2 / {e_1}^{x_1}$,
- $\mathsf{Open}(sk_T, \mathbf{cce}) := e_3 / e_2^{x_2}$,
- $\mathsf{Verify}_c(pk_T, c_1, c_2, m, o) := 1$ if $e(g_1, c_1) = e(o, h) e(c_2/m, h_1)$ and 0 otherwise, with o being the opening value of $\mathbf{cce}$.

According to [CPP13], the PPATC scheme is NM-CPA secure in the random oracle model in the SXDH setting.

Sigma Protocol

We use the validity proof of CCE in [CPP13] but in an interactive manner, where the verifier generates the challenge ch randomly. The reason is that later in our e-voting scheme, we require the verifier (Adm) to check the correctness of the submitted $\mathbf{cce}$ ciphertext by the prover (V) and publish it on BB. Here is the interactive version: let the $\mathbf{cce}$ ciphertext be $(\mathbf{e}, \mathbf{c}) = (e_1, e_2, e_3, c_1, c_2)$. We build a Sigma ($\Sigma$) protocol with the transcript $(\mathbf{a}, \mathrm{ch}, \mathbf{z})$ for the relation

$$\mathcal{R} = \{((prm, (\mathbf{e}, \mathbf{c})), (m, \mathbf{r})) : \mathsf{CCE}(prm, m; \mathbf{r}) = (\mathbf{e}, \mathbf{c})\}$$

as follows: the prover generates random values $s, s_1, s_2 \in_R \mathbb{Z}_q$ and computes

$$(e_1', e_2', e_3', c_1') := (g^{s_1}, g^{s_2}, g_1^s g_2^{s_2}, h^s h_1^{s_1}).$$

Then, she sends (e_1', e_2', e_3', c_1') as $\mathbf{a}$ to the verifier. The verifier generates and sends a uniformly random challenge ch to the prover. The prover computes the response $\mathbf{z} = (z, z_1, z_2)$ with $z := s + ch \cdot r,\ z_1 := s_1 + ch \cdot r_1,\ z_2 := s_2 + ch \cdot r_2$ and sends $\mathbf{z}$ to the verifier. The verifier computes $e_1'' := \frac{g^{z_1}}{{e_1}^{ch}},\ e_2'' := \frac{g^{z_2}}{{e_2}^{ch}},\ e_3'' := \frac{{g_1}^z {g_2}^{z_2}}{{e_3}^{ch}},\ c_1'' := \frac{h^z {h_1}^{z_1}}{{c_1}^{ch}}$. If $e_1'' = e_1',\ e_2'' = e_2',\ e_3'' = e_3'$ and $c_1'' = c_1'$ all hold, she returns 1; otherwise, 0.

In [CPP13], this interactive proof is transformed into a non-interactive proof using the Fiat-Shamir transformation. In our case, it is important that the proof is fresh, especially we don't want a coercer to forward a ciphertext with an unknown vote to the voter and instruct her to submit this. To avoid this, we can use the non-interactive proof mentioned above which will be extractable in the plaintext message. Alternatively, one can make a two-move protocol, where the authorities first send a challenge to the voter, and the voter includes this challenge in the hash used for the Fiat-Shamir transformation.

Re-Randomization (ReRand)

We use this algorithm to re-randomize the three following pre-defined primitives: a CCE ciphertext, a TEG ciphertext and a public key of the form g^x for a secret x. This algorithm operates slightly differently for each. First, for the CCE ciphertext $\mathbf{cce} = \mathsf{CCE}(prm, m; \mathbf{r})$ with $\mathbf{r} \in_R \mathbb{Z}_q^3$, we multiply the ciphertext by encryption of the unity element $1 \in \mathbb{G}_1$ with a randomness. Second, for the TEG ciphertext $\mathbf{E} = \mathsf{Enc}_{pk}(m; r) = (m \cdot pk^r, g^r)$ with $r \in \mathbb{Z}_q$, we first exponentiate the ciphertext by a random value and then re-randomize the output by a second random value. Lastly, for the public key g^x, we exponentiate it by a random number. More precisely, it works as follows.

- $\mathsf{ReRand}(\mathbf{cce}, \mathbf{r}') := \mathbf{cce} \cdot \mathsf{CCE}(prm, 1; \mathbf{r}') = \mathsf{CCE}(prm, m; \mathbf{r}'')$ with $\mathbf{r}' \in_R \mathbb{Z}_q^3$ and $\mathbf{r}'' = \mathbf{r} + \mathbf{r}'$.
- $\mathsf{ReRand}(\mathbf{E}, s, r') := \mathsf{ReRand}(\mathsf{Exp}(\mathbf{E}, s), r') = \mathsf{Enc}_{pk}(m^s; rs + r')$ with $\mathsf{Exp}(\mathbf{E}, s) := ((m \cdot pk^r)^s, (g^r)^s)$ and $s, r' \in \mathbb{Z}_q$.
- $\mathsf{ReRand}(pk, s) := (g^x)^s$ for $s \in \mathbb{Z}_q$.

For the sake of brevity, instead of $\{\mathsf{ReRand}(\mathbf{cce}, \mathbf{r}'), \mathsf{ReRand}(\mathbf{E}, s, r'), \mathsf{ReRand}(pk, s)\}$ we write $\mathsf{ReRand}\{(\mathbf{cce}, \mathbf{r}'), (\mathbf{E}, s, r'), (pk, s)\}$. Moreover, we omit the randomness when it is not needed for further computations.

Shuffle

In order to shuffle, meaning to permute and re-randomize the CCE ciphertexts in parallel with the ElGamal encryption $\mathbf{E}$ of a verification term, and the voters' public keys of the form g^x for a secret x, each mix server M_j for $j \in \{1, \cdots, N\}$, uses the $\mathsf{Shuffle}$ algorithm that we explain next. Assume we have a list of size n as $\{(\mathbf{e}_i, \mathbf{c}_i), \mathbf{E}_i, pk_i\}_{i \in \{1, \ldots, n\}}$ before mixing, where

$$(\mathbf{e}_i, \mathbf{c}_i) := \mathbf{cce}(prm, m_i; \mathbf{r}_i),\ \mathbf{E}_i := \mathsf{Enc}_{pk_T}(g; 0) = (g, 1),\ \text{and } pk_i := g^{x_i}.$$

The terms $\mathbf{E}_i$ are the same for each V_i in the beginning, but will change later in the mixing procedure. In the jth stage of mixing, the mix server M_j generates a random permutation $\pi^j \in \mathcal{S}_n$, and for each voter V_i generates a random vector $\mathbf{r}_i^j \in \mathbb{Z}_q^3$, and two random values $r_i^j, s_i^j \in \mathbb{Z}_q$. Assume $\{(\mathbf{e}_i^{j-1}, \mathbf{c}_i^{j-1}), \mathbf{E}_i^{j-1}, pk_i^{j-1}\}_i$ is the output of M_{j-1}.[2] Then, M_j runs the $\mathsf{Shuffle}$ algorithm as follows.

$$\mathsf{Shuffle}\left(\{(\mathbf{e}_i^{j-1}, \mathbf{c}_i^{j-1}), \mathbf{E}_i^{j-1}, pk_i^{j-1}\}_i, \{\mathbf{r}_i^j, s_i^j, r_i^j\}_i, \pi^j\right) := \\ \{\mathsf{ReRand}\{((\mathbf{e}_{\pi^j(i)}^{j-1}, \mathbf{c}_{\pi^j(i)}^{j-1}), \mathbf{r}_{\pi^j(i)}^j), (\mathbf{E}_{\pi^j(i)}^{j-1}, s_{\pi^j(i)}^j, r_{\pi^j(i)}^j), (pk_{\pi^j(i)}^{j-1}, s_{\pi^j(i)}^j)\}\}_i$$

This will be the output of M_j that we show by $\{(\mathbf{e}_i^j, \mathbf{c}_i^j), \mathbf{E}_i^j, pk_i^j\}_i$. Let's define $\mathbf{cce}^j := (\mathbf{e}^j, \mathbf{c}^j)$. Similar to [CPP13], M_j computes two commitment consistent proofs of shuffle with respect to π^j: $\mathrm{P}_{\mathbf{cce}}^j$ and $\mathrm{P}_{\mathbf{c}}^j$. The first proof demonstrates that $\mathbf{cce}^j$ is a shuffle of $\mathbf{cce}^{j-1}$ and the second proof demonstrates that $\mathbf{c}^j$ is a shuffle of $\mathbf{c}^{j-1}$. According to [CPP13], the proof $\mathrm{P}_{\mathbf{cce}}^j$ is CCE-compatible, meaning that from this proof, one can derive the second proof $\mathrm{P}_{\mathbf{c}}^j$. In our case, M_j extends these proofs to also show that first, $\mathbf{E}^j$ and pk^j are shuffled in parallel with the same permutation π^j, second $\mathbf{E}^j$ is exponentiated with the same randomness s^j as in pk^j[3], and lastly, proof of knowledge of s^j s.t. $\mathsf{ReRand}(pk, s^j) = pk^j$. We show these extended proofs by $\mathrm{P}_{\mathbf{cce},\mathbf{E},pk}^j$ and $\mathrm{P}_{\mathbf{c},\mathbf{E},pk}^j$.

In [GHS20], Gjøsteen et al. present an efficient mixnet for the PPATC scheme including a machine-verified proof of the protocol. Our proposed mixnet is a straightforward extension, in particular each mixnet M_j commits to their permutation π^j, which can be reused for the extra terms being mixed in parallel. Note that we deliberately kept the two terms g^{x_i} of the voter V_i and the encryption $\mathbf{E}_i$ in the same group.

4.5 Everlasting Hyperion E-Voting Scheme

In the everlasting Hyperion scheme, two critical pieces of information must be safeguarded from future adversaries: the encryption of votes, and the tracking terms. To protect vote encryption, we employ the PPAT method detailed in Section 4.3 where V_i's vote is securely embedded within the CCE ciphertext $\mathbf{cce}_i$. For the tracking terms represented by $\mathbf{E}_i$, we treat them as perfectly hiding commitments[4] and rerandomize them in a perfectly private way. This approach effectively preserves both individual and universal verifiability, maintaining transparency without compromising security.

Building on the security measures described, this section breaks down the stages of our e-voting protocol, detailing each phase: setup, submission, tally, and verification. At each stage, specific parties engage with carefully chosen cryptographic techniques to ensure secure and transparent processing of votes. The roles of these parties and the cryptographic methods employed are further explained in Sections 4.3 and 4.4, respectively.

[2]The first mix server M_1 has to operate on the mentioned original values before mixing.
[3]One can use a simple proof of discrete log equality.
[4]The exponents are chosen uniformly randomly and are therefore indistinguishable.

4.5.1 Setup phase

During this phase, EA generates the election parameters prm, determining the voting method, defining the set of candidates, and providing any other necessary information. This setup also includes encryption of the public value g under talliers public key pk_T as $E_g := \mathsf{Enc}_{pk_T}(g;0) = (g,1)$. As explained in Section 4.4, this encrypted value, initially the same for each voter V_i, serves as the voter's tracking term but will change as the process progresses. Therefore, we refer to it as E_i from the beginning. EA creates a vector $\mathbf{E} = \{E_i\}_i$ and sends $\mathbf{E}$ to SBB. We also assume that each voter V_i has a signing key pair (vk_i, sk_i).

4.5.2 Submission phase

This phase consists of two parts. In the first part, each voter V_i prepares their ballot and sends it privately to Adm. In the second part, Adm is responsible for checking the validity of the submitted CCE by V_i. We explain these two sub-phases as follows.

Ballot Preparation

1. Similar to Hyperion [Dam+24], V_i generates an ephemeral trapdoor key $x_i \in \mathbb{Z}_q$ and computes the public trapdoor key $pk_i := g^{x_i}$ using her device. Next, V_i signs pk_i as $\sigma_{i1} := \mathsf{Sign}_{sk_i}(pk_i)$ and computes Π_i :=NIZKPoK(x_i) as the proof of knowledge.

2. V_i commits to her vote v_i using the CCE scheme explained in the Section 2.3 by computing the ciphertext $\mathbf{cce}_i := (\mathbf{e}_i, \mathbf{c}_i) = \mathsf{CCE}(prm, v_i; \mathbf{r}_i)$ with $\mathbf{r}_i \in_R \mathbb{Z}_q^3$. Then, she signs it as $\sigma_{i2} := \mathsf{Sign}_{sk_i}(\mathbf{cce}_i)$.

Now, the voter V_i sends $\{\mathsf{ID}_i,\ pk_i, \sigma_{i1},\ \Pi_i,\ \mathbf{cce}_i,\ \sigma_{i2}\}$ privately to Adm.

Validity Check

The voter V_i as the prover and the Adm as the verifier run the Sigma protocol Σ_i for Adm to check whether $\mathbf{cce}_i$ is a valid CCE or not. If it is, then Adm proceeds by re-randomizing it as $\mathbf{cce}'_i := (\mathbf{e}'_i, \mathbf{c}'_i) = \mathsf{ReRand}(\mathbf{e}_i, \mathbf{c}_i)$. Finally, Adm sends $\mathbf{cce}_i$ and $\mathbf{cce}'_i$ together with a proof of correct re-randomization (Π^*_i) to SBB and publishes $\{\mathsf{ID}_i, pk_i, \sigma_{i1}, \mathbf{c}'_i\}$ on PBB.

4.5.3 Tally phase

This phase consists of three parts. First, in the *mixing* part, each mix server sequentially runs the Shuffle algorithm, publishing the commitments on PBB and the encryption of the opening values on SBB. Next, in the *reverse mixing* part, the mix-servers run the mix-net backwards just on the term E^N - the output of the last mix server M_N - to retrieve the correct tracking term for the voter V_i. Lastly, in the *decryption of opening and verification terms* part, the talliers decrypt the output of the reverse mixing, denoted as α_i, and send it to Adm, who then forwards this α_i term back to V_i. More precisely, the tally phase works as follows.

Mixing

First, M_1 generates a random permutation $\pi^1 \in \mathcal{S}_n$, and for each V_i generates a random vector $\mathbf{r}_i^1 \in \mathbb{Z}_q^3$, and two random values $r_i^1, s_i^1 \in \mathbb{Z}_q$. Then, it runs the shuffle algorithm as $\mathsf{Shuffle}\big(\{(\mathbf{e}'_i, \mathbf{c}'_i), E_i, pk_i\}_i, \{\mathbf{r}^1_i, s^1_i, r^1_i\}, \pi^1\big)$. Let $\{(\mathbf{e}^1_i, \mathbf{c}^1_i), E^1_i, pk^1_i\}_i$ be the output of M_1. As explained in Section 4.4, M_1 generates the proofs $\mathrm{P}^1_{\mathbf{cce},E,pk}$ and $\mathrm{P}^1_{\mathbf{c},E,pk}$. Then, she sends $\{\mathbf{c}^1_i, pk^1_i\}_i$ and $\mathrm{P}^1_{\mathbf{c},E,pk}$ to PBB and sends $\{\mathbf{e}^1_i, E^1_i\}_i$ and $\mathrm{P}^1_{\mathbf{cce},E,pk}$ to SBB. The output of M_1 is the input of the next mix server M_2 and so on. Accordingly, the output of the last mix server M_N is $\{(\mathbf{e}^N_i, \mathbf{c}^N_i), E^N_i, pk^N_i\}_i$. Similar to M_1, M_N sends $\{\mathbf{c}^N_i, pk^N_i\}_i$ and $\mathrm{P}^N_{\mathbf{c},E,pk}$ to PBB and sends $\{\mathbf{e}^N_i, E^N_i\}_i$ and $\mathrm{P}^N_{\mathbf{cce},E,pk}$ to SBB.

Reverse mixing

Define $s^{total} := s^1_{\pi^1\pi^2\cdots\pi^N(i)} s^2_{\pi^2\cdots\pi^N(i)} \cdots s^N_{\pi^N(i)}$.[5] A simple calculation shows that the output E^N_i by M_N is indeed $\mathsf{Enc}_{pk_T}(g^{s^{total}})$. However, the value $g^{s^{total}}$ does not belong to V_i, but to $\mathsf{V}_{\pi^1\pi^2\cdots\pi^N(i)}$. Hence, the mix servers have to run the mixnet backward just on the middle term E^N: M_N uses $(\pi^N)^{-1}$ to shuffle and a fresh randomness $r'^N_i \in \mathbb{Z}_q$ to re-randomize E^N_i (without exponentiation). M_N sends the result E'^N_i with $\Pi^N :=$NIZKPoK$((\pi^N)^{-1}, r'^N_i)$ to SBB. M_{N-1} proceeds with E'^N_i with the inverse permutation $(\pi^{N-1})^{-1}$. Ultimately, E'^1_i which is the final value and the output of M_1 will be posted on SBB. This is indeed the encrypted tracking term that belongs to V_i.

Decryption of Opening and verification terms

1. Talliers decrypt each ciphertext $\mathbf{cce}^N_i = (\mathbf{e}^N_i, \mathbf{c}^N_i)$ as $vote_i := \mathsf{Dec}_{sk_T}(\mathbf{cce}^N_i)$ using sk_T. Then, they run the Open algorithm to obtain $o_i := \mathsf{Open}(sk_T, \mathbf{cce}^N_i)$. They publish $vote_i, o_i$ on PBB together with a proof of correct decryption Π_T.

2. Talliers use the TEG decryption algorithm to jointly decrypt the TEG encryption E'^1_i and send the result, named α_i, to Adm with a proof of correct decryption Π^*_T. Then, Adm sends the α_i-term to the voter V_i.

4.5.4 Verification phase

The voter V_i raises α_i to the power of her trapdoor secret key x_i. Then, she can verify her vote by finding the row which contains $\alpha_i^{x_i}$ as the shuffled public key, i.e. pk^N_i.

In the event of coercion, if the coercer instructs V_i to vote for candidate v_k, she can find a row on PBB that contains the pair $(vote_k, pk^N_k)$ with $vote_k = v_k$. Then she claims that her alpha-term is $\bar{\alpha} := (pk^N_k)^{-x_i}$.

This concludes our everlasting Hyperion scheme. We would like to emphasize that the backward mixing is necessary as the talliers T and the Adm do not know any of the random permutations used by each mix server, so they cannot retrieve the correct voter index. Moreover, the coercer cannot detect that the coerced voter has faked the alpha-term unless he is colluding with Adm or he has access to the private channel used by Adm for notifying the voter.

4.6 Voter Experience

Understanding an e-voting protocol from the voter's perspective is crucial for ensuring clarity, usability, and security. By examining how the everlasting Hyperion

[5]The index $\pi^1\pi^2\cdots\pi^N(i)$ is composition of the permutations in group $\mathcal{S}_n$ with the same order: $\pi^1\left(\pi^2\left(\cdots\left(\pi^N(i)\right)\right)\right)$.

scheme performs for each voter, we gain insight into how each step supports privacy and verifiability — both fundamental to voter trust. This perspective allows us to assess how easily voters can navigate the system, protect their anonymity, and confirm that their votes are securely cast and counted, even under coercion threats.

In this section, we present an overview of the everlasting Hyperion scheme, as detailed in Section 4.5, from the perspective of an individual voter, V_i. This experience covers two scenarios: first, the base protocol without any coercion threats, and second, the protocol's adaptations when coercion is present. In both scenarios, we assume that V_i possesses a unique signing key pair (vk_i, sk_i), and upon joining the voting process, her device generates an ephemeral trapdoor key pair (pk_i, x_i).

4.6.1 The Base Protocol

The experience of V_i begins as follows.

- V_i picks her choice v_i, and her device generates the CCE ciphertext on v_i (i.e. commits to v_i) and signs it.
- V_i's device signs pk_i and sends it to Adm along with the signed ciphertext on the vote and ID_i.

Then, V_i's commitment is re-randomized and published on PBB along with her trapdoor public key and ID_i. Subsequently, information on the PBB is permuted and re-randomized with each step being displayed on the PBB. Finally, the voter is invited to check PBB.

- After a certain period, V_i receives a notification term denoted α_i privately from Adm. She inputs α_i to her device to extract her unique tracking number. She then uses this tracking number to verify that her vote appears correctly in the tally on the PBB.

4.6.2 The Protocol in the Event of Coercion

In the event of coercion, V_i needs to take additional steps to evade the coercer's demand. The vote-casting steps are identical to the base protocol.

- Once the tally is published, V_i checks PBB and finds an alternative tracking number appearing against the coercer's required vote and saves that in her device.
- Then, V_i's device computes a fake α_i term such that when raised to the power of x_i, results in the alternative tracking term. This step requires knowledge of x_i.
- After a certain period, the voter receives a notification containing the real α term. She inputs this into her device to extract her unique tracking term, and uses this tracking number to verify her vote on PBB.
- If the coercer requests the tracking or the α term, she can provide the fake one that was computed by her device earlier.

4.7 Analysis of the Protocol and the Game-Based Security Proofs

As discussed in Chapter 3, one significant advantage of using mix-net tallying over homomorphic tallying is that voters do not need to provide proofs demonstrating that their votes fall within the valid set of candidates. Instead, they only need to prove that they have knowledge of the vote they submitted.

In our e-voting system, which is also based on mix-net tally, for the sake of coercion-resistancy, voters are not permitted to submit their ballots directly on PBB; this responsibility is delegated to a separate party, Adm. Now, in the event of a dispute, there may be ambiguity in the submission phase over whether an invalid vote was submitted by a voter V_i or if Adm erroneously or purposefully disregarded a valid vote. To address this issue, our e-voting protocol incorporates a step where V_i engages in an efficient interactive Σ protocol with Adm. This ensures that the voter can prove her submitted vote v_i is valid, i.e., $v_i \in \mathbf{G}_1$. This added measure provides accountability in the submission phase while maintaining efficiency.

As we mentioned in Section 2.2.3, one way to prove ballot privacy holds in an e-voting system is using the game-based security proofs. Bernhard et al. [Ber+15] introduce a new game-based definition of privacy, called BPRIV, based on their extensive research in [Ber+15]. This is the basis of the privacy definition that we use later in our proofs. Game-based security proofs provide a structured framework for evaluating the security of cryptographic protocols, including e-voting systems. These proofs are formulated as adversarial games where the protocol's robustness is challenged by (hypothetical) adversaries with specific capabilities. This methodology allows us to formalize and prove security properties in a controlled and rigorous manner. For each one of the security properties that we aim to prove here, we consider different assumptions on the adversarial power.

In what follows, we provide a computational proof of verifiability, followed by game-based proofs for other security properties, including ballot privacy with definition du–mb–BPRIV, everlasting privacy with definition Everlasting–BPRIV, and everlasting receipt-freeness with definition Everlasting–RF. For the notations and definition of algorithms, refer to Section 4.3.

4.7.1 Verifiability

The verifiability proof of everlasting Hyperion is similar to the verifiability proof of Hyperion in Theorem 6.3 in [Dam+24]. More particularly, each voter V_i uses two separate devices for first casting her vote in the submission phase (includes V_i's signing key sk_i), and later, verifying their vote in the verification phase (includes the Hyperion term α_i). The assumption is that at most one of these devices is corrupted.

In [Dam+24], they define a verifiability experiment against a malicious board and malware, named Verif–MBM that captures the desired verifiability. Verifiability against a malicious board requires the voters to check that a valid vote with their ID exists on the PBB and if this check fails, the adversary loses the game. In addition, in Verif–MBM, the election setup is created honestly (i.e., not by $\mathcal{A}$). The adversary wins the game if he outputs a valid BB and tally, and succeeds in one of the following three cases, captured in the verifiability game:

1. Change the vote of a verifying voter with at least one honest device

2. Ballot stuffing or change (excluded deleting) a cast vote of the voters with an honest vote-casting device.

3. Cast more than one vote per voter for the other eligible voters.

The advantage of $\mathcal{A}$ in the game is defined as $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{Verif-MBM}}$. Now, we can prove the following theorem in a similar way.

Theorem 4.1. *The advantage of a PPT adversary $\mathcal{A}$ in the game* Verif–MBM *for the everlasting Hyperion voting scheme (detailed in Section 4.5), under the EUF-CMA assumption for signatures, soundness of the mix-net, correctness of the encryption, simulation-soundness extractability for the proofs of knowledge, and under the 1-DHI assumption, is negligible.*

Proof. Here, we provide a high-level idea of the proof of verifiability.

First, due to the soundness property of the NIZK proofs of the mix-net explained in Section 4.4 and the DDH assumption of the binding property of the commitment openings, if the ballots that are cast honestly are added to BB, they will validate.

For honest voters with honest vote-casting devices, the EUF-CMA signatures of the voters on their public keys ensure that each voter can only cast their vote once and that their vote is securely tied to their identity. This prevents ballot stuffing, where $\mathcal{A}$ might try to submit multiple votes on their behalf. The vote of these voters remains unaltered. If these voters do not verify their vote, $\mathcal{A}$ can delete it in the game.

Second, for the voters with malicious vote-casting devices who do not verify their votes, there is no guarantee.

Next, for voters with corrupted verification devices, who do verify their ballot, there is a ballot with their ID on PBB, but there is no guarantee which vote it contains.

Finally, the voters with honest verification devices who check their votes can also verify it successfully. This is grounded in the 1-DHI assumption (see Section 4.4), which ensures the extractability of the exponents x_i in the submitted public key and of s_i in the final mixed term $\mathsf{pk}_i^{s_i} = g^{x_i s_i}$ alongside the plaintext vote $vote_i$. More precisely, assume by contradiction that there exists $x_j,\ s_j$ for some index $j \neq i$ such that $(\alpha_i)^{x_i} = \mathsf{pk}_j^{s_j} = g^{x_j s_j}$. This results in an improbable relationship between the exponents: $\alpha_i = (g^{x_j s_j})^{1/x_i}$ which means $g^{1/x_i} = \alpha_i^{1/(s_j x_j)}$. This is a violation of the 1-DHI assumption. In fact, this computational assumption gives a negligible advantage to the adversary to win the game. □

4.7.2 Ballot Privacy

In some electronic voting schemes such as Helios [Adi08] and its variants, verification phase happens before the tally phase, however, in tracker-based schemes such as Selene, Hyperion, and everlasting Hyperion which aim to mitigate coercion, verification must happen after the tally phase, when the voters are informed of their trackers or tracking terms. Hence, we require to use a security notion in the literature that captures this scenario called delayed verification. We use the notion *delay-use malicious-ballotbox ballot privacy* (du–mb–BPRIV) in [Dra+22] which is based on the former ballot privacy notion mb–BPRIV in [CLW20].

In both these definitions, privacy is secured against a malicious board (mb), i.e., ballot privacy is proved in a strong adversary model, where he can control both the submitted votes and the tally procedure. Similar to [Dra+22], we consider an experiment $\mathsf{Exp}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}^{\mathsf{du-mb-BPRIV},\mathsf{Recover},\beta}(\lambda)$ depicted in Figure 4.1 where the oracles that $\mathcal{A}$ has access to are defined too. In this experiment, $\mathcal{A}$ wins if he can guess β correctly, i.e., he is in the real or the simulated world.

As mentioned in [Ber+15], when the tally includes more information than the final election result, i.e. NIZK proofs and other auxiliary information, we need to use a simulator Sim to ensure these information also don't reveal any information to $\mathcal{A}$. Indeed, for the tally output (res, Π), the simulator fakes Π with respect to BB_1.

Let's first start with the adversarial power in the game. $\mathcal{A}$ has access to 4 different oracles: $\mathcal{O}\mathsf{board}, \mathcal{O}\mathsf{voteLR}, \mathcal{O}\mathsf{tally}$ and $\mathcal{O}\mathsf{verify}$. Let BB_β be the bulletin board corresponding to the world β, from which only one is visible to $\mathcal{A}$. The adversary by calling oracle $\mathcal{O}\mathsf{board}$, gets $\mathsf{Publish}(\mathsf{BB}_\beta)$ as the output, which gives him access to the input of BB_β (public and private boards), especially ballots of H. Therefore, $\mathcal{A}$ can control both these boards referred to as BB in the game. However, we assume that the verification of these boards using the algorithm ValidBoard is honest. The secret board is verified by internal parties (authorities) and the public board by the voters or any third party.

The oracle $\mathcal{O}\mathsf{voteLR}$ on the input id, v_0, v_1, outputs two ballots for the corresponding voter using the Vote algorithm: b_0 that is published on BB_0 and b_1 that is published on BB_1. The variable state has to split into a *pre* and *post* parts, which corresponds to the verification info generated before tally or afterwards (including the tally itself). Since in our everlasting Hyperion scheme, the state updates after the tally with a verification term g^{r_i}. The verification always happen on β_0; that's why we have the post-state $\mathsf{state}_{post,0}$ appended to $\mathsf{V}[id]$.

The next oracle $\mathcal{O}\mathsf{tally}$ operates differently for the real and the fake world: for $\beta = 0$, it outputs the correct tally of BB as usual. However, for $\beta = 1$, it first calls the algorithm Recover to see how $\mathcal{A}$ tampered with ballots on BB_1 and it makes a new board named BB'. Indeed, by Recover, we look for the unaltered ballots output by Vote on BB_1 and change them to Vote output on BB_0. The tally is computed on this board as (res, Π). Then, the proof Π is faked using the simulation algorithm on the tally result as Π', and the final output is (res, Π').

Lastly, the oracle $\mathcal{O}\mathsf{verify}$, among the voters who have to verify their votes, returns the ones who successfully verify.

Now, we explain the experiment $\mathsf{Exp}^{\mathsf{du\text{-}mb\text{-}BPRIV,Recover},\beta}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}(\lambda)$. The voting system $\mathcal{V}$ starts with generating empty vectors and setting up the election with the security parameter λ. Then, the voters are registered one by one and the corrupted voters (controlled by $\mathcal{A}$) are added to the list CU (lines 4 to 7). Based on the input data that $\mathcal{A}$ has and using the oracles $\mathcal{O}\mathsf{board}$ and $\mathcal{O}\mathsf{voteLR}$, he creates a board BB. If BB is invalid, then we return a random bit d. Otherwise, $\mathcal{A}$ makes an initial guess d^* (line 12). In line 13, $\mathcal{A}$ tallies the board and outputs the result pair (res^*, Π^*). If this result fails to verify, we output a random bit. Otherwise, we let $\mathcal{A}$ to make another guess d using the $\mathcal{O}\mathsf{verify}$. Now, if all the voters that we expect to verify, did not verify, i.e. $\mathsf{H}_{\mathsf{check}} \not\subseteq \mathsf{Checked}$, we return a random bit. If not all the voters that have to verify their vote, successfully verify, i.e., $\mathsf{H}_{\mathsf{check}} \not\subseteq \mathsf{Happy}$, we return the initial adversary guess d^*, otherwise, we return the second adversary guess d in line 16 as the output of the game.

Definition 4.1. *The voting system $\mathcal{V}$ satisfies* du–mb–BPRIV *with respect to* Recover *if there exists an efficient simulator* Sim, *such that for any efficient adversary $\mathcal{A}$, the advantage*

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{du\text{-}mb\text{-}BPRIV,Recover},0}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{du\text{-}mb\text{-}BPRIV,Recover},1}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}(\lambda) = 1\right] \right|$$

is negligible.

Theorem 4.2. *Everlasting Hyperion scheme, detailed in Section 4.5, satisfies* du–mb–BPRIV *definition assuming that the proofs are zero-knowledge, correct and sound and the SXDH assumption holds.*

Proof. We prove the theorem using a sequence of seven game hops starting from the experiment $\mathsf{Exp}^{\mathsf{du\text{–}mb\text{–}BPRIV},\mathsf{Recover},0}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}(\lambda)$. Before explaining each game hop, note that due to the perfect correctness of the ElGamal encryption scheme and the ZK proofs, $\mathcal{A}$ can always create a verifying tally. Therefore, for the adversary not to lose the game instantly, without loss of generality, we can assume that board BB generated by $\mathcal{A}$ in line 8 and the tally in line 13, are valid and verifiable in lines 10 and 14, respectively.

Now, we start with the everlasting Hyperion scheme and progressively modify the voting process at each hop, discussing how the proof should be adjusted.

In the first hop, we simulate all the ZK proofs. This is possible due to the zero-knowledge property of the proofs in our scheme.

In the second hop, we want the ballots produced by $\mathcal{A}$ to be processed normally during the tally phase. To achieve this, for ballots that are not being altered by $\mathcal{O}\mathsf{voteLR}$, we discontinue the use of the Dec and Open algorithms on *cce* ciphertext during the tally phase. Instead, we use the vote input given to the voting oracle as $vote_i$, along with the opening value o_i provided by the oracle, corresponding to the voter's vote and the opening value of the commitment to that vote. Additionally, for the set of ballots that are output of $\mathcal{O}\mathsf{voteLR}$, due to simulated proof of correct decryption (from hop 1), the soundness of the proof of shuffle for the PPATC approach, the perfect correctness of ElGamal encryption, and the perfect correctness of the openings of CCE commitments, the set of votes obtained from decryption of the ballots output by the vote oracle matches the set of input votes given to the voting oracle. Therefore, this game hop does not change the advantage.

In the third hop, we aim to preserve the set of voters who correctly verify their votes. Note that, as explained earlier, the verification is always performed on BB_0 and what we do next does not leak any information on β to $\mathcal{A}$. As we explained in Section 4.3, the $\mathsf{Shuffle}$ algorithm has three sets of input per voter V_i: $((\mathbf{e}_i, \mathbf{c}_i), \mathbf{E}_i, pk_i)$. The commitment terms $\mathbf{c}_i$ and pk_i are being re-randomized and are thus perfectly hiding, so we leave them as they are. Under the DDH assumption, we change the random elements in e_i and $\mathbf{E}_i$ per voter and per mix server (for the output elements of the vote oracle) to make them indistinguishable by $\mathcal{A}$. This requires a maximum $2n \times N$ uses of the DDH assumption, which makes the advantage negligible.

In the fourth hop, we aim to keep the tally result consistent with $\beta = 0$ since as we mentioned before, the verification is done according to BB_0. Therefore, we change the PPATC ballots from BB_0 and replace them with the ones from BB_1. This game hop doesn't change the advantage too, since now the adversary can track the ballots of corrupted voters using the Hyperion scheme without gaining any knowledge about β, due to NM-CPA security of PPATC scheme.

In the fifth and the sixth game hops, we first restore the mixnet using the DDH assumption and next, decrypt the ballots output by the vote oracle.

Finally, in the last game hop, we move back from all the simulated proofs to the real proofs. Now we have reached $\mathsf{Exp}^{\mathsf{du\text{–}mb\text{–}BPRIV},\mathsf{Recover},0}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}(\lambda)$ as desired. □

```
$\mathsf{Exp}^{\mathsf{du\text{-}mb\text{-}BPRIV},\mathsf{Recover},\beta}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}(\lambda)$
$\mathsf{Checked}, \mathsf{Happy} \leftarrow \emptyset$
$\mathsf{V}, \mathsf{PU}, \mathsf{U}, \mathsf{CU} \leftarrow \mathsf{empty}$
$(\mathsf{pd}, \mathsf{sd}) \leftarrow \mathsf{Setup}(\breve{\ })$
for $id$ in $\mathcal{I}$ do
    $(pc, sc) \leftarrow \mathsf{Register}(id)$,
    $\mathsf{U}[id] \leftarrow sc, \mathsf{PU}[id] \leftarrow pc$
    if $id \in \mathsf{D}$ then $\mathsf{CU}[id] \leftarrow \mathsf{U}[id]$
$\mathsf{BB} \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{voteLR},\mathcal{O}\mathsf{board}}(\mathsf{pd}, \mathsf{CU}, \mathsf{PU}, \mathsf{H}_{\mathsf{check}})$
if $\mathsf{H}_{\mathsf{check}} \not\subseteq \mathsf{V}$ then $d$; return $d$
if $\mathsf{ValidBoard}(\mathsf{BB}, \mathsf{pk}) = \perp$ then
    $d$; return $d$
$d^* \leftarrow \mathcal{A}()$;
$(r^*, \Pi^*) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{tally}_{\mathsf{BB},\mathsf{BB}_0,\mathsf{BB}_1},\mathcal{O}\mathsf{board}}()$
if $\mathsf{VerifyTally}((\mathsf{pd}, \mathsf{PBB}, r^*), \Pi^*) = \perp$ then
    $d$; return $d$
$d \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{verify}}()$
if $\mathsf{H}_{\mathsf{check}} \not\subseteq \mathsf{Checked}$ then
    $d$; return $d$
if $\mathsf{H}_{\mathsf{check}} \not\subseteq \mathsf{Happy}$ then return $d^*$
return $d$
```

```
$\mathcal{O}\mathsf{board}$
return $\mathsf{Publish}(\mathsf{BB}_\beta)$
```

```
$\mathcal{O}\mathsf{voteLR}(id, v_0, v_1)$
if $id \in \mathsf{H}$ then
    $(pc, b_0, \mathsf{state}_{\mathsf{pre},0}, \mathsf{state}_{\mathsf{post},0}) \leftarrow \mathsf{Vote}(\mathsf{pd}, pc, v_0)$
    $(pc, b_1, \mathsf{state}_{\mathsf{pre},1}, \mathsf{state}_{\mathsf{post},1}) \leftarrow \mathsf{Vote}(\mathsf{pd}, pc, v_1)$
    $\mathsf{V}[id] \leftarrow \mathsf{V}[id] \| (\mathsf{state}_{\mathsf{pre},\beta}, \mathsf{state}_{\mathsf{post},0}, v_0)$
    $\mathsf{BB}_0 \leftarrow \mathsf{BB}_0 \| (id, (pc, b_0))$
    $\mathsf{BB}_1 \leftarrow \mathsf{BB}_1 \| (id, (pc, b_1))$
```

```
$\mathcal{O}\mathsf{tally}_{\mathsf{BB},\mathsf{BB}_0,\mathsf{BB}_1}$ for $\beta = 0$
$(r, \Pi) \leftarrow \mathsf{Tally}(\mathsf{BB}, \mathsf{pd}, \mathsf{sd})$
return $(r, \Pi)$
```

```
$\mathcal{O}\mathsf{tally}_{\mathsf{BB},\mathsf{BB}_0,\mathsf{BB}_1}$ for $\beta = 1$
$\mathsf{BB}' \leftarrow \mathsf{Recover}(\mathsf{BB}, \mathsf{BB}_0, \mathsf{BB}_1)$
$(r, \Pi) \leftarrow \mathsf{Tally}(\mathsf{BB}', \mathsf{pd}, \mathsf{sd})$
$\Pi' \leftarrow \mathsf{Sim}(\mathsf{pd}, \mathsf{Publish}(\mathsf{BB}), r)$
return $(r, \Pi')$
```

```
$\mathcal{O}\mathsf{verify}$ for $id \in \mathsf{H}_{\mathsf{check}}$
$\mathsf{Checked} \leftarrow \mathsf{Checked} \cup \{id\}$
if $\mathsf{VerifyVote}(id, \mathsf{V}[id], \mathsf{BB}, \mathsf{PU}[id], \mathsf{U}[id]) = \top$ then
    $\mathsf{Happy} \leftarrow \mathsf{Happy} \cup \{id\}$
return $\mathsf{Happy}$
```

FIGURE 4.1: du–mb–BPRIV ballot privacy against a dishonest ballot box from [Dra+22].

4.7.3 Everlasting Privacy

In the context of everlasting privacy, we do not consider a malicious board (mb) scenario. Instead, we assume that the bulletin board is trusted and the ballots deliver perfectly secret, which is an updated version of BPRIV in [Ber+15]. The main justification for this assumption is that it is infeasible to achieve privacy against a computationally unbounded adversary that is active during the vote-casting phase and has access to the secret bulletin board SBB. Such an adversary is capable of breaking the computational soundness of the underlying ZK proofs, as well as any other cryptographic primitives based on computational assumptions, which are also insecure against quantum attacks.

In our everlasting privacy definition Everlasting–BPRIV depicted in Figure 4.2 as experiment $\mathsf{Exp}^{\mathsf{Everlasting\text{-}BPRIV},\beta}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}(\lambda)$, we consider an adversary that possesses the credentials of certain (corrupted) voters, allowing it to control them and track their ballots using the Hyperion mechanism. This capability is captured by $\mathcal{O}$cast for dishonest voters in the set D. In contrast to the previous definition du–mb–BPRIV, the adversary in Everlasting–BPRIV only has access to the public bulletin board PBB, which contains the publicly available data of the ballots. This also includes the proof Π as the output of the algorithm Tally in $\mathcal{O}$tally, which now only contains the public proofs. The definition Everlasting–BPRIV is as follows.

Definition 4.2. *The voting system $\mathcal{V}$ satisfies* Everlasting–BPRIV *if there exists an efficient simulator* Sim, *such that for any unbounded adversary $\mathcal{A}$, the advantage*

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{Everlasting\text{-}BPRIV},0}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{Everlasting\text{-}BPRIV},1}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}(\lambda) = 1\right] \right|$$

is zero (or bounded by some small probability).

```
$\mathsf{Exp}^{\mathsf{Everlasting\text{–}BPRIV},\beta}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}(\lambda)$
1 : $\mathsf{V}, \mathsf{PU}, \mathsf{U}, \mathsf{CU} \leftarrow \mathsf{empty}$
2 : $(\mathsf{pd}, \mathsf{sd}) \leftarrow \mathsf{Setup}(\breve{})$
3 : for $id$ in $\mathcal{I}$ do
4 :     $(pc, sc) \leftarrow \mathsf{Register}(id)$,
5 :     $\mathsf{U}[id] \leftarrow sc, \mathsf{PU}[id] \leftarrow pc$
6 :     if $id \in \mathsf{D}$ then $\mathsf{CU}[id] \leftarrow \mathsf{U}[id]$
7 : $d \leftarrow \mathcal{A}^{\mathcal{O}\cdot}(\mathsf{pd}, \mathsf{CU}, \mathsf{PU})$
8 : return $d$

$\mathcal{O}\mathsf{board}$
1 : return $\mathsf{Public}(\mathsf{BB}_\beta)$

$\mathcal{O}\mathsf{cast}(id, b)$
1 : if $id \in \mathsf{D}$ then
2 : if $\mathsf{Valid}(b, \mathsf{BB}_\beta) = \perp$ return $\perp$
3 : else
4 :     $\mathsf{BB}_0 \leftarrow \mathsf{BB}_0 \| (id, (pc, b))$
5 :     $\mathsf{BB}_1 \leftarrow \mathsf{BB}_1 \| (id, (pc, b))$

$\mathcal{O}\mathsf{voteLR}(id, v_0, v_1)$
1 : if $id \in \mathsf{H}$ then
2 :     $(pc, b_0, \mathsf{state}_{\mathsf{pre},0}, \mathsf{state}_{\mathsf{post},0}) \leftarrow \mathsf{Vote}(\mathsf{pd}, pc, v_0)$
3 :     $(pc, b_1, \mathsf{state}_{\mathsf{pre},1}, \mathsf{state}_{\mathsf{post},1}) \leftarrow \mathsf{Vote}(\mathsf{pd}, pc, v_1)$
4 : if $\mathsf{Valid}(b_\beta, \mathsf{BB}_\beta) = \perp$ return $\perp$
5 : else
6 :     $\mathsf{V}[id] \leftarrow \mathsf{V}[id] \| (\mathsf{state}_{\mathsf{pre},\beta}, \mathsf{state}_{\mathsf{post},0}, v_0)$
7 :     $\mathsf{BB}_0 \leftarrow \mathsf{BB}_0 \| (id, (pc, b_0))$
8 :     $\mathsf{BB}_1 \leftarrow \mathsf{BB}_1 \| (id, (pc, b_1))$

$\mathcal{O}\mathsf{tally}_{\mathsf{BB}_0}$ for $\beta = 0$
1 : $(r, \Pi) \leftarrow \mathsf{Tally}(\mathsf{BB}_0, \mathsf{pd}, \mathsf{sd})$
2 : return $(r, \Pi)$

$\mathcal{O}\mathsf{tally}_{\mathsf{BB}_0,\mathsf{BB}_1}$ for $\beta = 1$
1 : $(r, \Pi) \leftarrow \mathsf{Tally}(\mathsf{BB}_0, \mathsf{pd}, \mathsf{sd})$
2 : $\Pi' \leftarrow \mathsf{Sim}(\mathsf{pd}, \mathsf{Public}(\mathsf{BB}_1), r)$
3 : return $(r, \Pi')$
```

FIGURE 4.2: Everlasting–BPRIV: Everlasting ballot privacy.

Based on Definition 4.2, we can prove the following theorem.

Theorem 4.3. *Everlasting Hyperion scheme, detailed in Section 4.5, satisfies* Everlasting–BPRIV *definition assuming the public proofs have perfect zero-knowledge.*

Proof. As here we assumed the BB is honest and the tally is computed honestly, the proof is shorter than Theorem 4.2. As before, we prove the theorem using a sequence of game hops starting from $\mathsf{Exp}^{\mathsf{Everlasting\text{–}BPRIV},0}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}(\lambda)$. In the first game hop, we simulated all the ZK proofs on PBB. Since we assumed these proofs are perfect, this game hop doesn't change the advantage for $\mathcal{A}$. In the second hop, we change the uniformly random re-randomization of the commitments $\mathbf{c}_i$ and pk_i, which are published on PBB, with random values for the voters V_i that correctly verified their votes. This doesn't change the adversary's advantage and keeps the votes information-theoretically hidden. In the third hop, we replace the commitments $\mathbf{c}_i$ with the ones on the board BB_1 output by the vote oracle, while keeping the other commitments pk_i and the tally compatible with $\beta = 0$. This doesn't change the advantage too, since the commitments are perfectly hiding. Note that the ballots submitted by the $\mathcal{O}\mathsf{cast}$ for corrupted voters can be decrypted by the unbounded adversary $\mathcal{A}$ and hence don't influence the advantage. Now we have reached $\mathsf{Exp}^{\mathsf{Everlasting\text{–}BPRIV},1}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}(\lambda)$ as desired. □

4.7.4 Everlasting Receipt-Freeness

In the context of everlasting receipt-freeness, for the same reason as in everlasting privacy, we assume honest board and tally computations, and indeed, another updated version of BPRIV. Our everlasting receipt-freeness definition Everlasting–RF depicted in Figure 4.3 as experiment $\mathsf{Exp}^{\mathsf{Everlasting\text{–}RF},\beta}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}(\lambda)$, is similar to the experiment of Everlasting–BPRIV with some slight differences: first, we have an additional oracle $\mathcal{O}$getReceipt which fakes a receipt capture the coercion mitigation mechanism in our everlasting Hyperion scheme.

Moreover, $\mathcal{A}$ needs to additionally pick two honest voters with identities id_1 and id_2 where $id_1 \neq id_2$ and two votes v_a and v_b (line 7). Then, the voter with id_1 votes for v_a in the real world and for v_b in the fake world (line 8). The voter with id_2 votes oppositely: for v_b in the real world and for v_a in the fake world (line 9). This type of voting prevents $\mathcal{A}$ to win the game instantly since in each world, these two voters are voting differently, he can guess β based on the public result. $\mathcal{A}$ can also ask for a receipt from voter id_1 using the $\mathcal{O}$getReceipt with the instruction to vote for v_b (line 10).

As defined in Figure 4.3, for the real world, we assume $\mathcal{A}$ gets the secret credential and the verification term of voter id_1 as $(\mathsf{U}(id_1), \mathsf{V}[id_1])$ respectively. The term $\mathsf{V}[id_1]$ is indeed the Hyperion term or α term in our scheme for voter id_1. We also assume he has access to all the random coins and information available to the voter. In contrast, for the fake world $\beta = 1$, we manipulate or Fake the alpha term for the adversary's choice v_b.

Definition 4.3. *The voting system* $\mathcal{V}$ *satisfies* Everlasting–RF *if there exists an efficient simulator* Sim *and faking algorithm* Fake*, such that for any unbounded adversary* $\mathcal{A}$*, the advantage*

$$\left| \Pr\left[\mathsf{Exp}^{\mathsf{Everlasting\text{–}RF},0}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{Everlasting\text{–}RF},1}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}(\lambda) = 1\right] \right|$$

is zero (or bounded by some small probability).

This definition is weaker than the receipt-freeness definition in the BeleniosRF scheme [CFG15], where, unlike in our case, $\mathcal{A}$ is allowed to submit a ballot on behalf of the voter. If we were to permit $\mathcal{A}$ to cast a ballot on behalf of the voter, considering the validity proof of the CCE ciphertext as σ during the vote-casting phase, it could potentially introduce new attacks in our scheme. However, we believe that we can adapt BeleniosRF to our setup, including the proof of plaintext knowledge during the vote submission phase, to fulfill their definition as a future work.

Theorem 4.4. *Everlasting Hyperion scheme, detailed in Section 4.5, satisfies* Everlasting–RF *definition assuming the public proofs have perfect zero-knowledge.*

Proof. This proof is similar to the proof of Theorem 4.3 for Everlasting–BPRIV in Definition 4.2. The only difference here is that the voter id_1 wants to vote for v_a while the $\mathcal{A}$'s instruction is to vote for v_b. Now, if she picks the shuffled public key commitment term next to a vote v_b, which belongs to a dishonest voter, the adversary will notice she is faking her alpha term and he wins the game. Therefore, the probability that $\mathcal{A}$ wins the game depends on the number of dishonest voters who voted to v_b. This means that the advantage of the adversary is bounded by $\frac{|D|}{n}$, where $|D|$ is the number of dishonest voters and n is the total number of voters. □

```
$\mathsf{Exp}^{\mathsf{Everlasting\text{–}RF},\beta}_{\mathcal{A},\mathcal{V},\mathsf{Sim}}(\lambda)$
1 :  $\mathsf{V},\mathsf{PU},\mathsf{U},\mathsf{CU} \leftarrow$ empty
2 :  $(\mathsf{pd},\mathsf{sd}) \leftarrow \mathsf{Setup}(\breve{\ })$
3 :  for $id$ in $\mathcal{I}$ do
4 :      $(pc, sc) \leftarrow \mathsf{Register}(id)$,
5 :      $\mathsf{U}[id] \leftarrow sc, \mathsf{PU}[id] \leftarrow pc$
6 :      if $id \in \mathsf{D}$ then $\mathsf{CU}[id] \leftarrow \mathsf{U}[id]$
7 :  $id_1 \neq id_2, v_a, v_b \leftarrow \mathcal{A}_1(\mathsf{pd},\mathsf{CU},\mathsf{PU})$
8 :  Call $\mathcal{O}\mathsf{voteLR}(id_1, v_a, v_b)$
9 :  Call $\mathcal{O}\mathsf{voteLR}(id_2, v_b, v_a)$
10 : $d \leftarrow \mathcal{A}_2^{\mathcal{O}}(\mathsf{pd},\mathsf{CU},\mathsf{PU},\mathsf{Receipt})$
11 : return $d$
```

```
$\mathcal{O}\mathsf{board}$
1 :  return $\mathsf{Public}(\mathsf{BB}_\beta)$
```

```
$\mathcal{O}\mathsf{cast}(id, b)$
1 :  if $id \in \mathsf{D}$ then
2 :  if $\mathsf{Valid}(b, \mathsf{BB}_\beta) = \perp$ return $\perp$
3 :  else
4 :      $\mathsf{BB}_0 \leftarrow \mathsf{BB}_0 \| (id, (pc, b))$
5 :      $\mathsf{BB}_1 \leftarrow \mathsf{BB}_1 \| (id, (pc, b))$
```

```
$\mathcal{O}\mathsf{voteLR}(id, v_0, v_1)$
1 :  if $id \in \mathsf{H}$ then
2 :      $(pc, b_0, \mathsf{state}_{\mathsf{pre},0}, \mathsf{state}_{\mathsf{post},0}) \leftarrow \mathsf{Vote}(\mathsf{pd}, pc, v_0)$
3 :      $(pc, b_1, \mathsf{state}_{\mathsf{pre},1}, \mathsf{state}_{\mathsf{post},1}) \leftarrow \mathsf{Vote}(\mathsf{pd}, pc, v_1)$
4 :  if $\mathsf{Valid}(b_\beta, \mathsf{BB}_\beta) = \perp$ return $\perp$
5 :  else
6 :      $\mathsf{V}[id] \leftarrow \mathsf{V}[id] \| (\mathsf{state}_{\mathsf{pre},\beta}, \mathsf{state}_{\mathsf{post},0}, v_0)$
7 :      $\mathsf{BB}_0 \leftarrow \mathsf{BB}_0 \| (id, (pc, b_0))$
8 :      $\mathsf{BB}_1 \leftarrow \mathsf{BB}_1 \| (id, (pc, b_1))$
```

```
$\mathcal{O}\mathsf{tally}_{\mathsf{BB}_0}$ for $\beta = 0$
1 :  $(r, \Pi) \leftarrow \mathsf{Tally}(\mathsf{BB}_0, \mathsf{pd}, \mathsf{sd})$
2 :  return $(r, \Pi)$
```

```
$\mathcal{O}\mathsf{tally}_{\mathsf{BB}_0,\mathsf{BB}_1}$ for $\beta = 1$
1 :  $(r, \Pi') \leftarrow \mathsf{Tally}(\mathsf{BB}_0, \mathsf{pd}, \mathsf{sd})$
2 :  $\Pi \leftarrow \mathsf{Sim}(\mathsf{pd}, \mathsf{Public}(\mathsf{BB}_1), r)$
3 :  return $(r, \Pi)$
```

```
$\mathcal{O}\mathsf{getReceipt}$
1 :  if $\beta = 0$ then
2 :      $\mathsf{Receipt} = (\mathsf{U}(id_1), \mathsf{V}[id_1])$
3 :  if $\beta = 1$ then
4 :      $\mathsf{Receipt} = \mathsf{Fake}((\mathsf{U}(id_1), \mathsf{V}[id_1]), v_b, \mathsf{BB}_\beta, (r, \Pi))$
5 :  return Receipt
```

FIGURE 4.3: Everlasting–RF: Everlasting Receipt-Freeness.

Chapter 5

Zero-Knowledge Proofs from Learning Parity with Noise: Optimization, Verification and Application

5.1 Introduction

In 1989, when the concept of Zero-Knowledge Proofs (ZKPs), explained in Chapter 2, was introduced, it was a theoretical concept that had two major use cases: prevent identity theft, and in general, a tool to enforce honest behavior in protocols [Gol23]. Over the years, ZKP protocols have transitioned from these early, theoretical constructs into more efficient and practical implementations both within and beyond blockchain technology.

In blockchain, they enhance privacy by enabling confidential transactions in zero-knowledge, as implemented in privacy-focused cryptocurrencies like Zcash, in addition to protecting user identities. They optimize storage by verifying the data without storing or transmitting the entire dataset on-chain. Users can authenticate by proving their credentials in ZK, which supports decentralized identity management.

Beyond blockchain, ZKPs are used in various applications, including secure e-voting, authentication, privacy-preserving machine learning, cloud storage and computation, financial systems, and healthcare, just to name a few, allowing secure data sharing and computations without revealing sensitive information.

In the context of e-voting, which is the central focus of this thesis, ZKPs serve as a crucial building block with multiple transformative use cases. For instance, ZKPs can facilitate the design of coercion-resistant systems by enabling voters to cast valid votes without revealing their choices to potential vote buyers, as discussed in Chapter 4. They also ensure the integrity of the final tally without compromising the secrecy of the election process.

In systems employing mix-net tallies, ZKPs enable verification of the correctness of the mix-net process without disclosing the underlying secret permutations or randomizations, thereby preserving the anonymity of individual votes. Similarly, in homomorphic tally systems, ZKPs allow tallying authorities to prove the correct decryption of the aggregation of votes while maintaining the confidentiality of each voter's choice. Lastly, ZKP can be used to authenticate voters securely, verifying their eligibility without compromising their private credentials or revealing their vote. These are indeed a few examples of the importance and use-cases of zero-knowledge proofs.

Zero-knowledge proofs differ from one another and can vary significantly based on their specific features, as listed below.

1. Type of the statement:
   - *specific*: Some ZKPs are designed to prove specific properties on encrypted data such as plaintext equality, encrypted bits, encrypted DH-triples, or polynomial relations among encrypted values [BS20].
   - *arbitrary*: Others are designed to prove any arbitrary relation, often using a universal circuit, among encrypted ciphertexts or committed values.
2. Cryptographic primitives: Each ZKP system can support different cryptographic objects, such as public-key encryptions, commitments, signatures, etc.
3. Security: The underlying security assumptions can differ ranging from classical (e.g., discrete logarithm, integer factorization, etc.) to post-quantum (e.g., code-based, lattice-based,
etc.).

4. Efficiency: They vary depending on how fast the proof can be generated and verified. This includes the computational workload for both the prover and verifier, as well as the size of the communication required between them.

In this chapter, our focus is on ZKP whose security is based on the extremely conservative assumption named Learning Parity with Noise (LPN), explained in detail in Section 2.4.2. Considering the definition of the LWE problem provided in the same section, one could see that the LPN problem is a specific instance of the LWE problem. Specifically, this occurs when the modulus q is set to 2 and the error distribution is Bernoulli rather than Gaussian. To date, no significant speedups have been achieved in improving LPN's cryptanalysis under neither classical nor quantum models, making it a good candidate for PQC-based protocols.

We choose to study one of the most prominent ZKPs together with the proposed commitment scheme by Jain et al. in [Jai+12], whose security is based on the LPN assumption. Concerning the type of the statement, this ZKP can be used to prove arbitrary binary statements among committed bit vectors, and concerning efficiency, the binary modulus[1] makes LPN-based systems more efficient compared to LWE, since the multiplication and addition operations in $\mathbb{Z}_2$ can be implemented with binary gates such as AND and XOR. Therefore, this also makes LPN-based circuits low-depth and a reasonable candidate for applications with hardware constraints that require fast, low-level implementations. As an example, in [Pra+20] the authors use Jain et al.'s ZKP to implement a Physical Unclonable Function (PUF) on a microcontroller. Despite the advancements made, the potential and understanding of the original ZKPs proposed by Jain et al., as well as the subsequent work building upon them, remain limited in several critical aspects:

1. Slow Verifier: While the prover in these ZKPs is computationally efficient, the verifier's computational overhead is prohibitively high, making the protocol impractical for real-world applications. For instance, Bellini et al. in [Bel+20] present performance benchmarks (see Table 5 in their paper) that highlight this inefficiency. Specifically, in the underlying Σ-protocol of the basic ZKP, which is used to prove the well-formedness of a single commitment, the prover requires less than 2 milliseconds to generate the proof. In contrast, the verifier takes an average of over 116 milliseconds to complete the verification, a disparity of several orders of magnitude.

2. Security Understanding: The security guarantees of these ZKPs require further analysis and formalization. While Jain et al. provided pen-and-paper security proofs for their commitment scheme and the basic version of their ZKP, these proofs lack formal verification. Furthermore, the inherent complexity of these ZKPs makes it challenging to fully understand their security properties and to implement them correctly, which is crucial to ensuring practical security.

3. Limited Application in the Literature: To date, the primary application focus of these ZKPs has been within the Internet of Things (IoT). However, there has been little exploration of their potential for broader privacy-preserving applications. Extending their applicability to other domains could significantly enhance their utility and relevance in cryptographic research.

Addressing these limitations could pave the way for further optimizations and novel use cases, making these protocols more practical, secure, and versatile in privacy-preserving technologies. To this end, we make the following contributions.

[1]As explained in Section 2.4.2, the operations are in the field $\mathbb{F}_2$.

1. Optimization: We modify the original ZKP proposed by Jain et al. to make the computational complexity of the verifier as light as the prover. While our modification fully preserves the security of the original ZKP, our benchmarks demonstrate that our modifications achieve efficiency improvements by several orders of magnitude.

2. Security Review: We analyzed the only open-source implementations of the original ZKP in [Bel+20] and discovered inconsistencies in the code. Our findings reveal that a malicious prover could exploit these flaws to deceive an honest verifier into accepting a false statement as valid.

3. Formal Verification: For the first time, we use EasyCrypt to formally prove the security of the full modified ZKP and the commitment scheme. As this part is not a contribution of the author of this thesis, we direct interested readers to our full paper [Hai+25] for further details.

4. New Application: We demonstrate the construction of an efficient ZKP of shuffle with security grounded in the LPN assumption. Such cryptographic protocols serve as fundamental building blocks for verifiable mixing networks [HM20], which enable the anonymization of sensitive data in a verifiable manner. As mentioned in Section 2.4.2, despite the significant theoretical potential of CBC, its practical applications, especially those built on the LPN problem, have remained limited. This is primarily due to challenges such as large key sizes, which hinder efficiency, and the absence of structured frameworks (i.e., utilizing specific code to generate the keys) that can systematically enhance security. To demonstrate the broader capabilities of CBC despite the mentioned limitations, we design the first code-based verifiable e-voting protocol as a concrete example, showcasing its suitability for privacy-preserving systems.

Structure of the Chapter. This chapter is divided into seven sections. Section 5.2 provides a literature review to establish the background for our work. Section 5.3 introduces the preliminaries, including notation and the roles of the parties in our electronic voting system. In Section 5.4, we describe the cryptographic primitives used and the underlying hardness assumptions. Section 5.5 delves into the original LPN-based commitment scheme and Σ-protocols proposed by Jain et al. [Jai+12], discusses their limitations, and presents an optimized version of the protocol. Section 5.6 reviews the security of the sole open-source implementation of the original protocol. In Section 5.7, we introduce the first code-based proof of shuffle, which serves as a building block for our PQ E-voting system, which is the first verifiable code-based e-voting system, detailed in the final Section 5.8.

5.2 Literature Review

In this section, we delve into the related work in the literature, focusing on topics relevant to our research. These include code-based ZKPs, PQ ZKP of shuffle, and PQ verifiable e-voting. For the formal verification of CBC we refer the interested reader to our paper [Hai+25].

5.2.1 Code-Based Zero-Knowledge Proofs

We provided some details on the origin of CBC and examples of hard problems and cryptographic schemes based on them, however, we did not specifically discuss code-based ZKPs. This concept originated with Stern's pioneering work [Ste93]. As we explained in the introduction of Section 5.1, they are broadly categorized into two types: those designed to prove specific statements and those capable of proving arbitrary statements.

The first category, exemplified by works such as [Bet+22; HMT13; Bid+22; GPS22; Mel+23], focuses on proving knowledge of a committed or encrypted value. These proofs are particularly effective for constructing efficient code-based digital signatures.

The second category, which addresses arbitrary statements, is far less explored, with only a few protocols available in the literature, including Jain et al. [Jai+12], studied in this chapter, and [Bel+20; DMV23]. As explained in Section 2.4.2, there are different metrics to measure the distance of a code. A key distinction between [Jai+12] and the latter two protocols lies in the used metric: Jain et al. employ the Hamming metric, while [Bel+20] and [DMV23] adopt the rank metric. While rank metric-based approaches often result in more efficient constructions, they are controversial because several rank-based cryptographic schemes have been broken. The main reason is that rank-based cryptographic schemes usually rely on codes with specific algebraic structures, which makes them highly susceptible to algebraic attacks to reconstruct the private key or break the scheme. For more details, refer to [Bar+23] and the references therein.

Consequently, to the best of our knowledge, the ZKP proposed by Jain et al. remains the only ZKP capable of proving arbitrary statements among committed values while relying on the well-established hardness of decoding random linear codes in the Hamming metric. This is an assumption that has not been disputed to date.

5.2.2 Post-Quantum Zero-Knowledge Proofs of Shuffle

Several post-quantum zero-knowledge proofs of shuffle have been introduced in the academic literature, including works such as [Ara+23; Ara+21; HSS23; HMS21; CMM19; CMM17]. These protocols typically derive their security from the Ring Learning With Errors (RLWE) problem, a computational assumption favored in LBC (lattice-based cryptography) for its potential to yield more efficient zero-knowledge proofs. The efficiency comes from the fact in RLWE, the ring structure is used to replace the general vectors and matrices of the standard LWE problem with polynomials or elements in a quotient ring of polynomials. In other words, instead of working with large vectors, operations are performed on ring elements, leading to better performance in both computation and storage.

Nevertheless, the reliance on RLWE introduces trade-offs, as its security hinges on the structure of specialized lattices, potentially providing weaker assurances compared to the general LWE assumption. On the contrary, the security of our proposed proof of shuffle is fundamentally different. It relies on the hardness of decoding random linear codes, which as mentioned in Section 2.4.2 is a problem regarded as a more conservative and broadly accepted assumption in post-quantum cryptography. This approach not only strengthens the security guarantees but also avoids the structural dependencies inherent to RLWE, offering a robust alternative for practical applications.

5.2.3 Post-Quantum Verifiable E-Voting

As discussed in Chapter 2, there are two primary methods for tallying ballots and designing verifiable e-voting systems: the homomorphic approach and the mixnet approach. Each offers distinct mechanisms and is suited to different voting scenarios.

1. Homomorphic Tallying: In the homomorphic approach, the encryption or commitment scheme used to secure voters' ballots must be additively homomorphic. This property enables the aggregation of encrypted or committed ballots directly, followed by the decryption or opening of the aggregated result in a verifiable manner. A key advantage of this method is that the aggregation can be verified by any observer without requiring additional proofs. However, voters must prove the well-formedness of their encrypted ballots, which can incur significant computational and memory overhead. Consequently, this method is most effective for elections with simple ballot types, such as binary (yes/no) choices. Among existing PQ e-voting solutions, the lattice-based protocols Evolve [Pin+17] and Epoque [BHM21] employ homomorphic tallying.

2. Verifiable Mixing: The mixnet approach processes encrypted/committed ballots through a sequence of mixing servers. These servers re-randomize and shuffle the encrypted ballots, resulting in effectively anonymizing the votes. Unlike the homomorphic approach, voters are not required to prove the well-formedness of their ballots, as the ballots that are not well-formed will be removed later. This allows the mixnet method to support arbitrarily complex ballot types. To ensure the integrity of the process, several techniques have been developed to make mixing networks verifiable [HM20]. PQ solutions for verifiable mixing have also primarily relied on lattice-based ZKPs of shuffle. These include methods that are grounded in the RLWE assumption. However, an alternative to these ZKPs is the trip-wire technique explained in [BHM20], which offers a distinct advantage: it can be efficiently realized using primitives based on the more conservative LWE problem. This avoids the need for a lattice-based ZKP of correct decryption [Gjø+22; Sil22], reducing complexity. That said, the trip-wire technique requires the presence of an active verifier, whereas ZKPs of shuffle can be passively verified, offering more flexibility in certain applications.

5.3 Preliminaries

In this section, we introduce the notation specific to this chapter, complementing the general notation outlined in Section 2.1 of the Preliminaries chapter. For consistency, we also adopt the notation used in the work of Jain et al. [Jai+12]. Following this, we provide an overview of the parties involved in our proposed post-quantum e-voting scheme.

5.3.1 Notation

In this chapter, we reserve the bold uppercase letter $\mathbf{A}$ to represent a matrix. We define the binary sets $\mathcal{I} = \{0,1\}$ and $\mathcal{I}^k = \{0,1\}^k$. The Hamming weight of a vector $\mathbf{v} \in \mathcal{I}^k$ is $\|\mathbf{v}\|_1 = \sum_{i=1}^{k} \mathbf{v}[i]$. We define the set of all k-bit vectors of weight exactly w as $\mathcal{I}_w^k = \{\mathbf{v} \in \mathcal{I}^k \colon \|\mathbf{v}\|_1 = w\}$. We denote the vector of all zeros by $\mathbf{0}^k$ and the vector of all ones by $\mathbf{1}^k$. The set of all permutations of size n is noted as

$\mathcal{S}_n$, with individual permutations represented by π and their inverses by π^{-1}. The permutation of a vector $\mathbf{v} \in \mathcal{I}^k$ with the permutation $\pi \in \mathcal{S}_k$ is denoted by $\pi(\mathbf{v})[\cdot] = \mathbf{v}[\pi(\cdot)]$. We denote the concatenation of $\mathbf{u}$ and $\mathbf{v}$ by $\mathbf{u}\|\mathbf{v}$. Instead of writing mod2, we use the mathematical operation XOR as $\oplus$. For the sake of notation consistency in this chapter with [Jai+12], we use $\ell \in \mathbb{N}$ to denote the main security parameter.

5.3.2 Parties

Our e-voting protocol is executed by the following parties.

Voters (V)**:** n voters $\mathsf{V}_1, \mathsf{V}_2, \ldots, \mathsf{V}_n$ with identities $\mathsf{ID}_1, \mathsf{ID}_2, \ldots, \mathsf{ID}_n$.

Election Authority (EA)**:** Responsible for generating the election parameters.

Tallier (T)**:** A single tallier T who is trusted for privacy and is responsible for the tally phase.

Bulletin Board (PBB)**:** We assume that all information from different parties such as V_is, EA, and T on PBB are authenticated by using a code-based digital signature.

5.4 Cryptographic Primitives and Assumptions

In this section, we provide a detailed explanation of the cryptographic primitives utilized in our proposed e-voting scheme and our verifiable code-based shuffle, as outlined in the Sections 5.8 and 5.7.

Commitment Scheme (CS)

We explained a CS in detail in Section 2.3.3. The commitment scheme that we study in this chapter satisfies the following three properties.

- Correctness: This property ensures that the CS behaves as expected, meaning that if the parameters prm and the commitment c are generated correctly, with overwhelming probability the Verify algorithm accepts. That is, for the security parameter λ we have:

$$Pr[\mathsf{Verify}(prm, c, m, d) = 1; prm \overset{R}{\leftarrow} \mathsf{KeyGen}(1^\lambda),$$
$$m \in \mathcal{M}, (c, d) \overset{R}{\leftarrow} \mathsf{Com}(prm, m)] = 1$$

- Perfectly Binding: This property ensures that the committer cannot produce two different valid messages for the same commitment, even with unlimited computational power. In other words: For all $prm \in \mathsf{KeyGen}$, $m, m' \in \mathcal{M}$ and all c, d, d', if $\mathsf{Verify}(prm, c, m, d) = 1$ and $\mathsf{Verify}(prm, c, m', d') = 1$, then $m = m'$.

- Computationally Hiding: This property ensures that no efficient (polynomial-time) adversary can determine the committed message with high probability. In other words, with overwhelming probability over the choice of $prm \overset{R}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$, for every $m, m' \in \mathcal{M}$ and

$$(c, d) \overset{R}{\leftarrow} \mathsf{Com}(prm, m),\ (c', d') \overset{R}{\leftarrow} \mathsf{Com}(prm, m'),$$

the distributions of c and c' are computationally indistinguishable.

Learning Parity with Noise (LPN)

In Section 2.4.2, we briefly introduced the search LPN problem. In this chapter, we highlighted that it can be regarded as a special case of the Learning with Errors (LWE) problem. Here, we formally define the decisional LPN problem and introduce a specific variant of it, referred to as the exact LPN (xLPN) problem, which was first defined in [Jai+12].

Definition 5.1 (Learning Parity with Noise (LPN))**.** *For $k, \ell \in \mathbb{N}$, let χ be an error distribution over $\mathcal{I}^k$. The* decisional (χ, ℓ, k)-LPN problem *is (s, ϵ)-hard if for every distinguisher* D *of size s:*

$$\left| \Pr_{\mathbf{A},\mathbf{x},\mathbf{e}} [\mathsf{D}(\mathbf{A}, \mathbf{A} \cdot \mathbf{x} \oplus \mathbf{e}) = 1] - \Pr_{\mathbf{A},\mathbf{r}} [\mathsf{D}(\mathbf{A}, \mathbf{r}) = 1] \right| \leq \epsilon,$$

where $\mathbf{A} \overset{R}{\leftarrow} \mathcal{I}^{k\times\ell}$, $\mathbf{e} \overset{R}{\leftarrow} \chi$, $\mathbf{r} \overset{R}{\leftarrow} \mathcal{I}^k$, *and* $\mathbf{x} \overset{R}{\leftarrow} \mathcal{I}^\ell$.

We denote by LPN_τ, where $\tau \in (0, \frac{1}{2})$, the variant of LPN problem where the noise distribution χ is Bernoulli. Specifically, each bit $\mathbf{e}[i]$ is distributed independently and identically with $\Pr[\mathbf{e}[i] = 1] = \tau$.

In [Jai+12], Jain et al. proposed a novel variant of the LPN problem, which serves as the foundation for the security of their commitment scheme and zero-knowledge proof. In this exact LPN problem with parameter τ, noted as xLPN_τ, the error vector is constrained to have a fixed Hamming weight of $\lceil k\tau \rfloor$, rather than merely having an expected weight of $k\tau$, as in LPN_τ.

Definition 5.2 (Exact Learning Parity with Noise (xLPN))**.** *Let $k, \ell \in \mathbb{N}$ and let $0 < \tau < \frac{1}{2}$ be the noise parameter. The* decisional exact (τ, ℓ, k)-LPN problem *is (s, ϵ)-hard if for every distinguisher* D *of size s:*

$$\left| \Pr_{\mathbf{A},\mathbf{x},\mathbf{e}} [\mathsf{D}(\mathbf{A}, \mathbf{A} \cdot \mathbf{x} \oplus \mathbf{e}) = 1] - \Pr_{\mathbf{A},\mathbf{r}} [\mathsf{D}(\mathbf{A}, \mathbf{r}) = 1] \right| \leq \epsilon,$$

where $\mathbf{A} \overset{R}{\leftarrow} \mathcal{I}^{k\times\ell}$, $\mathbf{e} \overset{R}{\leftarrow} \mathcal{I}^k_{\lceil k\tau \rfloor}$, $\mathbf{r} \overset{R}{\leftarrow} \mathcal{I}^k$, *and* $\mathbf{x} \overset{R}{\leftarrow} \mathcal{I}^\ell$ *is fixed and secret.*

Jain et al. established a polynomial-time reduction between the Decisional xLPN_τ problem and the Search LPN_τ problem, effectively linking the security of xLPN_τ to the difficulty of solving LPN_τ. The hardness of LPN_τ assumption itself is rooted in the decoding random linear codes problem, as demonstrated in [BKW03] and discussed in this thesis in Section 2.4.2. We will choose the parameters of the commitment scheme and ZKP such that these reductions hold true.

Sigma($\Sigma-$) Protocols

We explained ZKPs and their history in detail in Section 2.3.1. However, we did not explain the Sigma protocols, which are an important part of this chapter. A Sigma protocol is a special type of three-move interactive proof system that is designed for a prover P to convince a verifier V that they know a private witness w for a public statement x in a relation $\mathcal{R}$. The three-move or transcript of the $\Sigma-$protocol is considered as a triple (a, ch, z) for a commitment a, a challenge ch, and a response z. More formally, we have the following definition.

Definition 5.3 (Σ-protocol)**.** *Let (P, V) be a pair of connected PPT Turing machines and let $\mathcal{R}$ be a binary relation. Then (P, V) is* Σ-protocol for relation $\mathcal{R}$ *with challenge set $\mathcal{E}$ if the following conditions are satisfied:*

1. Form*: The protocol is of the following form:*

   (a) P *creates a commitment a and sends it to* V*.*

   (b) V *draws a challenge $ch \xleftarrow{R} \mathcal{E}$ and sends it to* P*.*

   (c) P *sends a response z to* V*.*

   (d) V *returns 0 or 1.*

   A protocol transcript (a, ch, z) is called accepting *if* V *returns 1.*

2. Completeness*: For all $(x, w) \in \mathcal{R}$, the verifier in $\langle \mathsf{P}(x, w), \mathsf{V}(x)\rangle$ returns 1.*

3. Special soundness*: There exists a ppt algorithm* E *(the* knowledge extractor*) that takes accepting transcripts $\{(a, ch, z_{ch}) \colon ch \in \mathcal{E}\}$ with the same commitment a as input, and outputs w' such that $(x, w') \in \mathcal{R}$.*

4. Special honest-verifier zero-knowledge (SHVZK)*: There exists a ppt algorithm* S *(the* simulator*) that takes x and $ch \in \mathcal{E}$ as inputs, and outputs a triple (a, ch, z) whose distribution is (computationally) indistinguishable from accepting protocol transcripts generated by real protocol runs.*

The Σ-protocols that we study in this chapter can be turned into non-interactive ZKPs with the Fiat-Shamir (FS) heuristic or into interactive ZKPs with standard techniques, such as [Dam+95].

5.5 Jain et al. Original Scheme and Our Optimization

In this section, we explain the original cryptographic primitives proposed by Jain et al. in [Jai+12] including the xLPN-based commitment and the LPN-based Sigma protocols.

5.5.1 Commitment Scheme

The utilized commitment scheme is a perfectly binding LPN-based string commitment. Let the public-key $\mathbf{A} \in \mathcal{I}^{k\times(\ell+v)}$ be a random binary matrix, $\mathbf{r} \in \mathcal{I}^{\ell}$ be a uniformly random vector, and $\mathbf{e} \in \mathcal{I}^{k}$ be a uniformly random error vector with a fixed low weight. We then have the following definition.

Definition 5.4 (xLPN commitment scheme [Jai+12])**.** *Let $\ell \in \mathbb{N}$ be the main security parameter, $0 < \tau < 0.25$ be the noise parameter, $v \in \mathbb{N}$ be the message length, $k \in \Theta(\ell + v)$, and $w = \lceil \tau k \rceil$ be the weight. We assume these parameters as implicit input to all algorithms.*

- KeyGen*:*

  1. *Sample $\mathbf{A}' \xleftarrow{R} \mathcal{I}^{k\times\ell}$ and $\mathbf{A}'' \xleftarrow{R} \mathcal{I}^{k\times v}$.*
  2. *Return public key $\mathbf{A} = \mathbf{A}' \| \mathbf{A}'' \in \mathcal{I}^{k\times(\ell+v)}$.*

- $\mathsf{Com}(\mathbf{A}, \mathbf{m})$*:*

  1. *Sample $\mathbf{r} \xleftarrow{R} \mathcal{I}^{\ell}$ and $\mathbf{e} \xleftarrow{R} \mathcal{I}^{k}_{w}$.*
  2. *Set $\mathbf{c} = \mathbf{A} \cdot (\mathbf{r}\|\mathbf{m}) \oplus \mathbf{e}$.*
  3. *Return commitment/opening pair $(\mathbf{c}, (\mathbf{r}, \mathbf{e}))$.*

- $\mathsf{Verify}(\mathbf{A}, \mathbf{c}, \mathbf{m}', \mathbf{r}')$:

    1. *Compute* $\mathbf{e}' = \mathbf{A} \cdot (\mathbf{r}' \| \mathbf{m}') \oplus \mathbf{c}$.
    2. *Return 1 if* $\|\mathbf{e}'\|_1 = w$ *and 0 otherwise.*

Jain et al. provided a pen-and-paper proof that their CS in Definition 5.4 achieves perfect binding and computational hiding, assuming the parameters are chosen carefully to satisfy the following two key conditions.

- Perfect Binding: In this CS, perfect binding relies on a property of the public key matrix $\mathbf{A}$. Specifically, the matrix $\mathbf{A}$ must have a minimum Hamming distance larger than $2w$, where $w = [\tau k]$ is the weight of the error vector $\mathbf{e}$. This ensures that no two distinct error vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ can satisfy the equation $\mathbf{c} = \mathbf{A} \cdot (\mathbf{r} \| \mathbf{m}) \oplus \mathbf{e}$ for the same commitment $\mathbf{c}$. If this distance condition holds with overwhelming probability, the scheme is perfectly binding, preventing the committer from generating multiple valid openings.

- Computational Hiding: In this CS, hiding is based on the hardness of the xLPN_τ problem. The problem involves distinguishing between noisy linear combinations of rows of $\mathbf{A}$ and uniform random values. If xLPN_τ is hard, the commitment $\mathbf{c} = \mathbf{A} \cdot (\mathbf{r} \| \mathbf{m}) \oplus \mathbf{e}$ is indistinguishable from random, effectively hiding the message $\mathbf{m}$.

In summary, the security of the commitment scheme is ensured by the interplay of two factors: the hardness of xLPN_τ, achieved through careful selection of parameters τ and k, and the structural property of $\mathbf{A}$, which requires a minimum distance greater than $2w$.

5.5.2 $\Sigma-$Protocols

Jain et al. proposed a framework of LPN-based Σ-protocols tailored to proving specific relations among cryptographic commitments. These protocols are modular[2] and can be combined to construct a general-purpose Σ-protocol capable of proving arbitrary relations. Specifically, they introduced three distinct Σ-protocols:

- Well-formedness of a Commitment: This protocol proves that a single commitment $\mathbf{c}$ is correctly formed, meaning it satisfies the expected constraints with respect to a secret witness $(\mathbf{e}, \mathbf{s})$ where $\mathbf{s}$ is the concatenation $(\mathbf{r} \| \mathbf{m})$.

- Linear Relations Between Commitments: This protocol demonstrates that commitments satisfy linear relations, such as $\mathbf{c}_1 \oplus \mathbf{c}_2 = \mathbf{c}_3$, without revealing the committed values.

- Multiplicative Relations Between Commitments: This protocol proves multiplicative relations between commitments, such as $\mathbf{c}_1 \circ \mathbf{c}_2 = \mathbf{c}_3$.

The linear and multiplicative protocols are constructed using multiple instances of the well-formedness protocol, combined with additional mechanisms for assembling these instances. Consequently, the well-formedness protocol acts as the foundational building block for the entire framework. The general-purpose protocol enables proving that committed values $\mathbf{m}_0$ and $\mathbf{m}_1$ satisfy $\mathbf{m}_0 = C(\mathbf{m}_1)$ for an arbitrary

[2]Modularity means that these protocols act as "plug-and-play" components, meaning that they can be used as individual building blocks to construct more complex protocols.

circuit C, by committing to intermediate values $x_1, \ldots, x_d$ representing the outputs of each layer of C, and verifying the correctness of each layer using $\Sigma-$protocols for linear and bitwise (multiplication) operations.

While in our paper [Hai+25] we include the formal verification of all three Sigma protocols, in this section, we focus exclusively on the first $\Sigma-$protocol: the proof of well-formedness. This proof ensures the integrity of commitments before they are used in more complex relations. The full details of the protocol are illustrated in Figure 5.1. Briefly, the statement or the joint input to P and V is the pair $(\mathbf{A}, \mathbf{c})$ and P's witness or secret information is the pair $(\mathbf{e}, \mathbf{s})$. P wants to convince V that she knows $(\mathbf{e}, \mathbf{s})$ satisfying $\mathbf{c} = \mathbf{A} \cdot \mathbf{s} \oplus \mathbf{e}$ with $\|\mathbf{e}\|_1 = w$ without revealing the witness.

Next, we delve deeper into the main concepts and the protocol flow depicted in Figure 5.1, followed by a discussion of its security properties.

- **Core Concept of the Protocol**: The core concept behind this $\Sigma-$protocol can be summarized as follows: In the initial step, P creates a mask $\mathbf{t} = \mathbf{A} \cdot \mathbf{v} \oplus \mathbf{f}$, where $\mathbf{v}$ is a secret vector, and $\mathbf{f}$ is a random vector. The protocol proceeds based on the V's challenge, where the prover either reveals the mask or the masked commitment. To ensure that revealing the masked commitment does not leak any information about the original commitment $\mathbf{c}$, especially about the error vector $\mathbf{e}$, Jain et al. use a clever technique. First, the vector $\mathbf{f}$ in the mask is drawn uniformly at random from $\mathcal{I}^k$ (not from $\mathcal{I}^k_w$ as in the xLPN$_\tau$ commitment). Then, the prover applies an additional layer of masking by shuffling $\mathbf{f} \oplus \mathbf{e}$ with a random permutation π.

- **Protocol Flow**: The flow of the Σ-protocol is structured as follows: Initially, P creates two random elements as masks: a permutation π and a vector $\mathbf{t}_0 = \mathbf{A} \cdot \mathbf{v} \oplus \mathbf{f}$, with both $\mathbf{v}$ and $\mathbf{f}$ chosen uniformly at random. Then, P generates three commitments using *any* perfectly binding CS (denoted by $(\mathsf{Com}', \mathsf{Verify}')$):

  - c_0 commits to the pair of masks $(\pi, \mathbf{t}_0)$,
  - c_1 commits to the permutation of $\mathbf{f}$ with π,
  - c_2 commits to the permutation of $\mathbf{f} \oplus \mathbf{e}$ with π.

  The prover P sends the commitments $a = (c_0, c_1, c_2)$ to the verifier V, who responds with a random challenge $e \in \{0, 1, 2\}$. Depending on the challenge e, the prover reveals specific combinations of the commitments:

  - For $e = 0$, the prover opens c_0 and c_1,
  - For $e = 1$, the prover opens c_0 and c_2,
  - For $e = 2$, the prover opens c_1 and c_2.

  The verifier V then checks the correctness of the openings and verifies the following conditions depending on the challenge:

  - If $e = 0$, the verifier checks if $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_1)$ is in the image of $\mathbf{A}$ and ensures π is a valid permutation.
  - If $e = 1$, the verifier checks if $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_2) \oplus \mathbf{c}$ lies in the image of $\mathbf{A}$.
  - If $e = 2$, the verifier checks whether the Hamming weight $\|\mathbf{t}_1 \oplus \mathbf{t}_2\|_1 = w$.

- **Security of the $\Sigma-$Protocol**: Jain et al. presented a pen-and-paper proof to show that this protocol satisfies the requirements of a Σ-protocol described in Definition 5.3, and we have formally verified the theorem in EasyCrypt (refer

Original Σ-protocol of well-formedness

Prover $\mathsf{P}((\mathbf{A},\mathbf{c}),(\mathbf{e},\mathbf{s}))$ — Verifier $\mathsf{V}(\mathbf{A},\mathbf{c})$

```
Prover:
  $\pi \overset{R}{\leftarrow} \mathcal{S}_k, \mathbf{v} \overset{R}{\leftarrow} \mathcal{I}^{\ell+v}, \mathbf{f} \overset{R}{\leftarrow} \mathcal{I}^k$
  $(c_0,d_0) \overset{R}{\leftarrow} \mathsf{Com}'(\pi, \mathbf{t}_0 = \mathbf{A}\cdot\mathbf{v} \oplus \mathbf{f})$
  $(c_1,d_1) \overset{R}{\leftarrow} \mathsf{Com}'(\mathbf{t}_1 = \pi(\mathbf{f}))$
  $(c_2,d_2) \overset{R}{\leftarrow} \mathsf{Com}'(\mathbf{t}_2 = \pi(\mathbf{f}\oplus\mathbf{e}))$
                $\xrightarrow{a = (c_0,c_1,c_2)}$
Verifier:
  $e \overset{R}{\leftarrow} \{0,1,2\}$
                $\xleftarrow{e}$
Prover:
  if $e = 0$ then
    $z \leftarrow (\pi, \mathbf{t}_0, \mathbf{t}_1, d_0, d_1)$
  elseif $e = 1$ then
    $z \leftarrow (\pi, \mathbf{t}_0, \mathbf{t}_2, d_0, d_2)$
  elseif $e = 2$ then
    $z \leftarrow (\mathbf{t}_1, \mathbf{t}_2, d_1, d_2)$
                $\xrightarrow{z}$
Verifier:
  if $e = 0$ then
    $\mathsf{Verify}'(\mathbf{t}_0, c_0, d_0) \overset{?}{=} 1, \mathsf{Verify}'(\mathbf{t}_1, c_1, d_1) \overset{?}{=} 1$
    $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_1) \overset{?}{\in} \mathrm{img}(\mathbf{A}), \pi \overset{?}{\in} \mathcal{S}_k$
  if $e = 1$ then
    $\mathsf{Verify}'(\mathbf{t}_0, c_0, d_0) \overset{?}{=} 1, \mathsf{Verify}'(\mathbf{t}_2, c_2, d_2) \overset{?}{=} 1$
    $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_2) \oplus \mathbf{c} \overset{?}{\in} \mathrm{img}(\mathbf{A})$
  if $e = 2$ then
    $\mathsf{Verify}'(\mathbf{t}_1, c_1, d_1) \overset{?}{=} 1, \mathsf{Verify}'(\mathbf{t}_2, c_2, d_2) \overset{?}{=} 1$
    $\|\mathbf{t}_1 \oplus \mathbf{t}_2\|_1 \overset{?}{=} w$
```

FIGURE 5.1: Original Σ-protocol for $\mathcal{R} = \{(\mathbf{A},\mathbf{c}),(\mathbf{e},\mathbf{s}) \colon \mathbf{c} = \mathbf{A}\cdot\mathbf{s}\oplus\mathbf{e} \wedge \|\mathbf{e}\|_1 = w\}$ (well-formedness of a commitment).

to our paper [Hai+25]). Here, we provide a high-level explanation of why the security of the protocol holds.

– Special Honest-Verifier Zero-Knowledge: The protocol is special honest-verifier zero-knowledge (HVZK) because of the following properties for each challenge e:

  * For $e = 0$, the verifier's output contains no information about the original commitment $\mathbf{c}$.
  * For $e = 1$, the secret vector $\mathbf{v}$ perfectly blinds $\mathbf{f}$, which in turn perfectly blinds $\mathbf{e}$.
  * For $e = 2$, the secret permutation π perfectly blinds both $\mathbf{f}$ and $\mathbf{f} \oplus \mathbf{e}$.

- Special Soundness: The protocol ensures special soundness by guaranteeing that, if all checks for the possible challenges e succeed, the validity of these checks implies the existence of a valid solution $(\mathbf{s}', \mathbf{e}')$ such that $\mathbf{A} \cdot \mathbf{s}' \oplus \mathbf{e}'$ equals the commitment, and the error vector $\mathbf{e}'$ has Hamming weight w. This guarantees that, if the protocol is sound, a dishonest prover cannot generate a valid proof without possessing the correct secret values $\mathbf{s}'$ and $\mathbf{e}'$.

5.5.3 Optimization: Improving Verifier Efficiency

In this section, we demonstrate how the computational efficiency of the verifier algorithm in the original Σ-protocols (as detailed in Section 5.5.2) can be improved significantly, by several orders of magnitude, in a straightforward manner without compromising security. Our approach applies to all Σ-protocols in this family, covering Σ−protocol for linear and multiplicative relations too, but we focus here on the basic protocol for proving knowledge of a valid opening. We begin by outlining the problem, followed by presenting our solution, and conclude with a security proof for the proposed approach.

- Problem Description: In the original Σ-protocol in Figure 5.1, the verifier V performs the following computational checks:

  - For $e = 0$: $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_1) \overset{?}{\in} \text{img}(\mathbf{A})$.
  - For $e = 1$: $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_2) \oplus \mathbf{c} \overset{?}{\in} \text{img}(\mathbf{A})$.

  While numerical linear algebra provides various algorithms to verify whether a vector belongs to the image of a linear function, defined by $\mathbf{A}$ here, the most computationally efficient exact (and not approximate) algorithms we know have a computational complexity of $\mathcal{O}(n^3)$, where n is the size of the vector. This inefficiency is evident in benchmarks reported by [Bel+20], which implemented the original Σ-protocol [Jai+12]. They observed that each such check took more than 170 ms, whereas other operations (e.g., random sampling, computing/verifying commitments, and checking Hamming weights) required less than 0.5 ms. Since the Σ-protocol needs to be repeated multiple times to minimize knowledge error, this computational bottleneck creates a significant overhead, rendering the original protocol impractical for many applications.

- Our Solution: Our key observation is that the prover's commitments provide enough information for the verifier to compute solutions to the equations efficiently. Specifically:

  - For $e = 0$: The prover can provide $\mathbf{v}$ such that $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_1) = \mathbf{A} \cdot \mathbf{v}$.
  - For $e = 1$: The prover can provide $\mathbf{v}' = \mathbf{v} \oplus \mathbf{s}$ such that $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_2) \oplus \mathbf{c} = \mathbf{A} \cdot \mathbf{v}'$.

  With this modification, the verifier can check:

  - $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_1) \overset{?}{=} \mathbf{A} \cdot \mathbf{v}$ for $e = 0$.
  - $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_2) \oplus \mathbf{c} \overset{?}{=} \mathbf{A} \cdot \mathbf{v}'$ for $e = 1$.

This modification basically gives the answer to the linear system of equations for challenges $e = 0, 1$, while these answers $\mathbf{v}$ and $\mathbf{v}'$ contain no secret information and still preserve the protocol's security. This dramatically reduces the computational cost of V, as matrix-vector multiplications are significantly faster than the original image membership checks. The protocol flow for the our optimized version is depicted in Figure 5.2.

Optimized Σ-protocol of well-formedness

Prover $\mathsf{P}((\mathbf{A},\mathbf{c}),(\mathbf{e},\mathbf{s}))$		Verifier $\mathsf{V}(\mathbf{A},\mathbf{y})$
$\pi \overset{R}{\leftarrow} \mathcal{S}_k, \mathbf{v} \overset{R}{\leftarrow} \mathcal{I}^{\ell+v}, \mathbf{f} \overset{R}{\leftarrow} \mathcal{I}^k$		
$(c_0, d_0) \overset{R}{\leftarrow} \mathsf{Com}'(\pi, \mathbf{t}_0 = \mathbf{A}\cdot\mathbf{v} \oplus \mathbf{f})$		
$(c_1, d_1) \overset{R}{\leftarrow} \mathsf{Com}'(t_1 = \pi(\mathbf{f}))$		
$(c_2, d_2) \overset{R}{\leftarrow} \mathsf{Com}'(t_2 = \pi(\mathbf{f} \oplus \mathbf{e}))$		
	$\xrightarrow{a = (c_0, c_1, c_2)}$	
		$e \overset{R}{\leftarrow} \{0,1,2\}$
	$\xleftarrow{e}$	
if $e = 0$ **then**		
$z \leftarrow (\pi, \mathbf{t}_0, \mathbf{t}_1, d_0, d_1, \boxed{\mathbf{v}})$		
elseif $e = 1$ **then**		
$z \leftarrow (\pi, \mathbf{t}_0, \mathbf{t}_2, d_0, d_2, \boxed{\mathbf{v}' = \mathbf{v} \oplus \mathbf{s}})$		
elseif $e = 2$ **then**		
$z \leftarrow (\mathbf{t}_1, \mathbf{t}_2, d_1, d_2)$		
	$\xrightarrow{z}$	
		if $e = 0$ **then**
		$\mathsf{Verify}'(\mathbf{t}_0, c_0, d_0) \overset{?}{=} 1, \mathsf{Verify}'(\mathbf{t}_1, c_1, d_1) \overset{?}{=} 1$
		$\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_1) \boxed{\overset{?}{=} \mathbf{A}\cdot\mathbf{v}}, \pi \overset{?}{\in} \mathcal{S}_k$
		if $e = 1$ **then**
		$\mathsf{Verify}'(\mathbf{t}_0, c_0, d_0) \overset{?}{=} 1, \mathsf{Verify}'(\mathbf{t}_2, c_2, d_2) \overset{?}{=} 1$
		$\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_2) \oplus \mathbf{c} \boxed{\overset{?}{=} \mathbf{A}\cdot\mathbf{v}'}$
		if $e = 2$ **then**
		$\mathsf{Verify}'(\mathbf{t}_1, c_1, d_1) \overset{?}{=} 1, \mathsf{Verify}'(\mathbf{t}_2, c_2, d_2) \overset{?}{=} 1$
		$\lVert \mathbf{t}_1 \oplus \mathbf{t}_2 \rVert_1 \overset{?}{=} w$

FIGURE 5.2: Optimized Σ-protocol for $\mathcal{R} = \{(\mathbf{A},\mathbf{c}),(\mathbf{e},\mathbf{s}) \colon \mathbf{c} = \mathbf{A}\cdot\mathbf{s} \oplus \mathbf{e} \wedge \lVert\mathbf{e}\rVert_1 = w\}$ (well-formedness of a commitment)

- Implementation and Results: We incorporated this modification into the implementation by Bellini et al.,[3] which is the only open-source implementation of this Σ−protocol. For our experiments, we used the parameters $\ell = 128$,

[3]See https://github.com/Crypto-TII/2020-CANS-rank_commitments (accessed 26.04.2024).

$k = 1320$, $v = 2640$, and $w = 284$, which ensure that the underlying decoding problem has a classical and quantum computational hardness of at least 2^{128} operations, as detailed in the work by Bellini et al. in section 5.1. The tests were conducted on a standard laptop to assess the practical efficiency of the protocol after applying the optimization.

Our findings demonstrate a significant improvement. Specifically, our experiment showed that these checks became over 7137 times faster on average, reducing the run-time from over 170 ms to less than 0.025 ms. This improvement makes Jain et al.'s Σ-protocol as fast as Bellini et al.'s rank-based ZKP, despite the former relying on a more conservative hardness assumption. In addition, it is worth mentioning that the transcript size of the Σ-protocol with the parameters specified earlier and the commitment scheme Com' in Figure 5.1 instantiated using Jain et al.'s CS specified in Section 5.4 is bounded by $7\frac{2}{3} \cdot k = 1.265$ Kilobytes.

5.5.3.1 Security of Our Optimized Protocol

We provide two proofs to show that our optimized protocol remains a Σ-protocol for the original relation

$$\mathcal{R} = \{((\mathbf{A}, \mathbf{c}), (\mathbf{s}, \mathbf{e})) \colon \mathbf{c} = \mathbf{A} \cdot \mathbf{s} \oplus \mathbf{e} \wedge \|\mathbf{e}\|_1 = w\}.$$

The first proof is a pen-and-paper proof presented here, and the second proof is formally verified in EasyCrypt (see our paper [Hai+25]). For the first proof, we show that our protocol provides three properties: correctness, special soundness, and special honest verifier zero-knowledge (HVZK).

Correctness

If P and V are honest, the transcript is always accepted:

- For $e = 0$: $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_1) = (\mathbf{A} \cdot \mathbf{v} \oplus \mathbf{f}) \oplus \mathbf{f} = \mathbf{A} \cdot \mathbf{v}$.
- For $e = 1$: $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_2) \oplus \mathbf{c} = \mathbf{A}(\mathbf{v} \oplus \mathbf{s})$.
- For $e = 2$: $\|\mathbf{t}_1 \oplus \mathbf{t}_2\|_1 = \|\mathbf{f} \oplus (\mathbf{f} \oplus \mathbf{e})\|_1 = \|\mathbf{e}\|_1 = w$.

Special Soundness

We prove that it is possible to efficiently extract a valid witness $(\mathbf{s}, \mathbf{e})$ for the statement $(\mathbf{A}, \mathbf{c})$ using three accepting transcripts corresponding to challenges $e = 0$, $e = 1$, and $e = 2$. The key observation here is that the acceptance of these transcripts directly relies on the binding property of the commitment scheme $(\mathsf{Com}', \mathsf{Verify}')$. Specifically, the acceptance of the transcript for $e = 0$ ensures that the relation $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_1) = \mathbf{A} \cdot \mathbf{v}$ holds, where $\mathbf{v}$ is a vector derived during the protocol. Similarly, the acceptance of the transcript for $e = 1$ implies the relation $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_2) \oplus \mathbf{c} = \mathbf{A}(\mathbf{v} \oplus \mathbf{s})$, where $\mathbf{s}$ is the secret we aim to extract. By combining these two equations, we can isolate and compute the first component of the witness, $\mathbf{s}$. Once $\mathbf{s}$ is determined, we use the linearity of the relation $\mathbf{c} = \mathbf{A} \cdot \mathbf{s} \oplus \mathbf{e}$ to extract the second component, $\mathbf{e}$, by subtracting $\mathbf{A} \cdot \mathbf{s}$ from $\mathbf{c}$.

Special Honest-Verifier Zero-Knowledge

For each challenge $e \in \{0, 1, 2\}$, we describe how the real protocol can be adapted in such a way that the challenge e is known, but the witness remains unknown. The modified protocol will produce a family of transcripts that is indistinguishable from the transcripts generated by the real protocol. Specifically, the structure of the simulator follows the same steps as the original protocol, but with the following changes tailored to each possible challenge:

- $e = 0$: In this case, we simulate the response by generating a fresh commitment pair (c_2, d_2) drawn randomly from the commitment space, i.e., $(c_2, d_2) \overset{R}{\leftarrow} \mathsf{Com}'(0)$.
- $e = 1$: Here, we begin by sampling random values for $\mathbf{t}_2$ and $\mathbf{v}'$, with $\mathbf{t}_2$ being sampled from the space $\mathcal{I}^k$ and $\mathbf{v}'$ from $\mathcal{I}^{\ell+v}$. Additionally, we select a random permutation π from $\mathcal{S}_k$. We then compute $\mathbf{t}_0$, which is set to $\mathbf{t}_0 \leftarrow \mathbf{A} \cdot \mathbf{v}' \oplus \mathbf{c} \oplus \pi^{-1}(\mathbf{t}_2)$. Finally, a random commitment pair (c_1, d_1) is generated as $(c_1, d_1) \overset{R}{\leftarrow} \mathsf{Com}'(0)$.
- $e = 2$: For this challenge, we sample two vectors, $\mathbf{t}_1$ from $\mathcal{I}^k$ and $\mathbf{t}_2$ from $\mathcal{I}^k_w$. These vectors are drawn randomly, ensuring that they align with the expected distribution. Then, we create a commitment (c_0, d_0) by sampling from the commitment space $(c_0, d_0) \overset{R}{\leftarrow} \mathsf{Com}'(0)$.

The key observation here is that, by using the hiding property of the commitment scheme $(\mathsf{Com}', \mathsf{Verify}')$, the transcripts generated by this modified protocol will be computationally indistinguishable from those generated by the real protocol. This indistinguishability ensures that an adversary cannot differentiate between a real proof and a simulated one, even though the simulator knows the challenge e but not the witness.

5.6 Security Review of an Implementation

In our analysis, we discovered a significant flaw in the only publicly available implementation of the ZKP proposed by [Jai+12]. Specifically, the implementation by Bellini et al. [Bel+20], as we also reflect in Figure 5.3, fails to satisfy the soundness property, which is a critical requirement for the protocol's security. As a result, a malicious prover can manipulate the protocol to produce convincing yet invalid proofs, thereby undermining the verifier's ability to detect dishonest behavior. Notably, the same soundness vulnerability is present in the implementation of the rank-metric ZKP proposed in the same work [Bel+20]. This issue relates to the handling of the permutation π, an essential component of the commitment scheme in the first round of the Sigma protocol. Next, we explain this flaw.

Flaw: Failure to Commit to the Permutation π.

The root of this issue lies in how the first commitment c_0 is computed in the protocol. Instead of following the specification provided in [Jai+12], where c_0 should commit to both the permutation π and the vector $\mathbf{t}_0$, i.e., $c_0 = \mathsf{Com}(\pi, \mathbf{t}_0)$, the implementation commits only to $\mathbf{t}_0$, i.e., $c_0 = \mathsf{Com}(\mathbf{t}_0 = \mathbf{A} \cdot \mathbf{v} \oplus \mathbf{f})$. This omission of π allows a dishonest prover to bypass the protocol's soundness checks. Consequently, Flawed Σ-protocol of well-formedness (as implemented in [Bel+20])

Prover $\mathsf{P}((\mathbf{A},\mathbf{c}),(\mathbf{e},\mathbf{s}))$ | Verifier $\mathsf{V}(\mathbf{A},\mathbf{c})$

```
$\pi \xleftarrow{R} \mathcal{S}_k, \mathbf{v} \xleftarrow{R} \mathcal{I}^{\ell+v}, \mathbf{f} \xleftarrow{R} \mathcal{I}^k$
$(c_0, d_0) \xleftarrow{R} \mathsf{Com}'(\cancel{\pi}, \mathbf{t}_0 = \mathbf{A}\cdot\mathbf{v} \oplus \mathbf{f})$
$(c_1, d_1) \xleftarrow{R} \mathsf{Com}'(t_1 = \pi(\mathbf{f}))$
$(c_2, d_2) \xleftarrow{R} \mathsf{Com}'(t_2 = \pi(\mathbf{f} \oplus \mathbf{e}))$
                    $\xrightarrow{a = (c_0, c_1, c_2)}$
                                                    $e \xleftarrow{R} \{0,1,2\}$
                    $\xleftarrow{e}$
if $e = 0$ then
    $z \leftarrow (\pi, \mathbf{t}_0, \mathbf{t}_1, d_0, d_1)$
elseif $e = 1$ then
    $z \leftarrow (\pi, \mathbf{t}_0, \mathbf{t}_2, d_0, d_2)$
elseif $e = 2$ then
    $z \leftarrow (\mathbf{t}_1, \mathbf{t}_2, d_1, d_2)$
                    $\xrightarrow{z}$
                                                    if $e = 0$ then
                                                        $\mathsf{Verify}'(\mathbf{t}_0, c_0, d_0) \overset{?}{=} 1, \mathsf{Verify}'(\mathbf{t}_1, c_1, d_1) \overset{?}{=} 1$
                                                        $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_1) \overset{?}{\in} \mathrm{img}(\mathbf{A}), \cancel{\pi \overset{?}{\in} \mathcal{S}_k}$
                                                    if $e = 1$ then
                                                        $\mathsf{Verify}'(\mathbf{t}_0, c_0, d_0) \overset{?}{=} 1, \mathsf{Verify}'(\mathbf{t}_2, c_2, d_2) \overset{?}{=} 1$
                                                        $\mathbf{t}_0 \oplus \pi^{-1}(\mathbf{t}_2) \oplus \mathbf{c} \overset{?}{\in} \mathrm{img}(\mathbf{A})$
                                                    if $e = 2$ then
                                                        $\mathsf{Verify}'(\mathbf{t}_1, c_1, d_1) \overset{?}{=} 1, \mathsf{Verify}'(\mathbf{t}_2, c_2, d_2) \overset{?}{=} 1$
                                                        $\|\mathbf{t}_1 \oplus \mathbf{t}_2\|_1 \overset{?}{=} w$
```

FIGURE 5.3: Implemented Σ-protocol for $\mathcal{R} = \{(\mathbf{A},\mathbf{c}),(\mathbf{e},\mathbf{s})\colon \mathbf{c} = \mathbf{A}\cdot\mathbf{s} \oplus \mathbf{e} \wedge \|\mathbf{e}\|_1 = w\}$ (well-formedness of a commitment) by [Bel+20].

it becomes possible to convince an honest verifier that a vector $\mathbf{c} = \mathbf{A}\cdot\mathbf{s} \oplus \mathbf{e}$ is valid, even when $\|\mathbf{e}\|_1 \neq w$, which should be prohibited in a secure $\Sigma-$protocol.

To illustrate the exploit, consider the following example with parameters $k = 3$ and $w = 1$, where:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{c} = \mathbf{A}\cdot\mathbf{s}.$$

Although $(\mathbf{A},\mathbf{c})$ is not a valid statement (as $\mathbf{e} = \mathbf{0}^3$ and $\|\mathbf{e}\|_1 = 0 \neq w$), a malicious prover can deceive the verifier as follows:

1. The prover selects $\mathbf{t}_0 = \mathbf{0}^3$, $\mathbf{t}_1 = (1,0,1)^T$, and $\mathbf{t}_2 = (0,0,1)^T$.

2. Depending on the challenge e, the prover responds with:
   - $e = 0$: The prover sends $\mathbf{t}_0$, $\mathbf{t}_1$, and $\pi_0 = (12)(3)$.
   - $e = 1$: The prover sends $\mathbf{t}_0$, $\mathbf{t}_2$, and $\pi_1 = (13)(2)$.
   - $e = 2$: The prover sends $\mathbf{t}_1$ and $\mathbf{t}_2$.

Despite the invalidity of $\mathbf{c}$, the verifier in the faulty implementation accepts the transcript because all checks appear to hold.[4]

- For $e = 0$, the verifier confirms that $\mathbf{t}_0 \oplus \pi_0^{-1}(\mathbf{t}_1) = (0, 1, 1)^T \in \operatorname{img}(\mathbf{A})$.
- For $e = 1$, the verifier verifies that $\mathbf{t}_0 \oplus \pi_1^{-1}(\mathbf{t}_2) \oplus \mathbf{c} = (1, 0, 0)^T \oplus \mathbf{A} \cdot \mathbf{s} \in \operatorname{img}(\mathbf{A})$.
- For $e = 2$, the verifier checks that $\|\mathbf{t}_1 \oplus \mathbf{t}_2\|_1 = \|(1, 0, 0)^T\|_1 = 1 = w$.

This example demonstrates how the verifier is misled due to the exclusion of π from the first commitment c_0. As a result, the prover exploits this loophole to fabricate valid-looking proofs for invalid statements. This finding underscores the importance of adhering to the original protocol specifications to preserve security guarantees.

5.7 Code-Based Proof of Shuffle

In this section, we present the first proof of shuffle whose security is based on the hardness of decoding random linear codes, a significant step toward cryptographic proof systems resilient to quantum attacks. The proof of shuffle that we present in this section builds upon the classic *cut-and-choose* paradigm introduced by Sako and Kilian [SK95], initially instantiated with ElGamal encryption (see Section 2.3.2) and corresponding ZKPs based on the discrete logarithm problem. Their method has been employed in systems like Helios e-voting [Adi08], which is one of the most widely recognized and extensively studied e-voting systems.

Building on this well-established framework, we present an instantiation of the proof of shuffle using our optimized LPN-based Σ-protocols from Section 5.5.3, yielding a proof of shuffle under code-based cryptography assumptions. We begin by presenting a high-level overview of the cut-and-choose paradigm to establish the foundational concept. Following this, we delve into the specifics of our proof of shuffle, providing a detailed explanation of its design and the subroutine it employs. The discussion then transitions to the protocol flow, outlining the step-by-step execution. Finally, we conclude with an analysis of the security and efficiency of our proof of shuffle.

5.7.1 Cut-and-Choose Framework

The cut-and-choose idea is a robust cryptographic technique for zero-knowledge proofs, enabling a prover to convince a verifier that some operation, such as a shuffle or a transformation, has been performed correctly or prove a relation among hidden data (i.e., encrypted or committed), without revealing sensitive data. The core idea is that V presents P with a random challenge, and P must respond appropriately. The challenge effectively "cuts" the prover's commitments into different categories

[4] Multiplying the matrix $\mathbf{A}$ with an arbitrary vector $(a, b, c)^T$ shows that the vectors in the img$\mathbf{A}$ are of the form (a, b, b) for $a, b \in \mathcal{I}$.

(e.g., revealing certain intermediate steps or relationships) that V can check. V's challenge forces P to "choose" a response that aligns with the stated claims. If P is honest, she can always satisfy the challenges. However, if P is dishonest, she cannot simultaneously satisfy both challenges without revealing inconsistencies with high probability, when the protocol is repeated multiple times.

To understand the cut-and-choose method in action, consider the case of proof of shuffle. P wants to demonstrate that a shuffled vector of commitments corresponds to the original vector after applying some secret permutation. However, P does not want to reveal the permutation or the plaintext data.

- Step 1: Commitment to Intermediate State
  P creates an intermediate state by applying a random permutation to the same plaintext messages and committing to the result.
- Step 2: Verifier's Challenge
  V issues one of the two following challenges:
  - $ch = 0$: Prove that the intermediate commitments correspond to the plaintext data (i.e., reveal and verify the random permutation).
  - $ch = 1$: Prove that the output commitments are related to the intermediate commitments by the secret permutation (i.e., reveal the composite permutation combining the secret and random permutations).
- Step 3: Prover's Response
  P responds based on V's challenge, revealing the required information (e.g., the intermediate commitments for $ch = 0$ or the composite permutation for $ch = 1$)
- Step 4: Repetition
  The protocol is repeated multiple times to ensure soundness.

Notation Warning. In what follows, we generalize our notation for commitments to vectors, applying the commitment scheme to individual vector entries. This allows us to operate on commitments collectively while maintaining their cryptographic properties.

5.7.2 Main Idea

Let σ be a secret permutation. Consider a vector of commitments $\vec{c}$ and a vector of plaintext messages $\vec{m}$, where $\vec{c} \in \mathsf{Com}(\sigma(\vec{m}))$. The proof begins with the prover generating an intermediate commitment vector $\vec{\gamma} \overset{R}{\leftarrow} \mathsf{Com}(\pi(\vec{m}))$, which corresponds to the same plaintext messages $\vec{m}$ but shuffled using an independent random permutation π.

Depending on the random challenge of the verifier, P performs one of the following tasks:

1. $ch = 0$: Opens $\vec{\gamma}$ to prove it is a commitment to $\vec{m}$.
2. $ch = 1$: Reveals the composite permutation $\sigma' := \sigma \circ \pi^{-1}$ and proves that the vectors $\vec{c}$ and $\sigma'(\vec{\gamma})$ commit to the same plaintext messages.

To perform the second task, P and V run a Σ-protocol to prove the equality of the committed messages. This Σ-protocol is derived from our optimized Σ-protocol for the linear relations described in Sections 5.5.2 and 5.5.3. As explained earlier, repeated execution and the Fiat-Shamir transformation ensure soundness in the non-interactive setting.

5.7.3 Subroutine: Plaintext Equality

A crucial component of our proof of shuffle is the Σ-protocol for plaintext equality. We suggest that we use a special case of our optimized version of the Σ-protocol for linear relations to realize this proof system. More precisely, the Σ-protocol for linear relation works as follows: It proves that three commitments $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3$ satisfy:

$$\mathbf{c}_i = \mathsf{Com}(\mathbf{m}_i) \quad \text{for all } i \in \{1, 2, 3\},$$

and that:

$$\mathbf{X}_1 \cdot \mathbf{m}_1 \oplus \mathbf{X}_2 \cdot \mathbf{m}_2 = \mathbf{m}_3,$$

for known matrices $\mathbf{X}_1, \mathbf{X}_2 \in \mathcal{I}^{v \times v}$. For the special case of $\mathbf{X}_1 = \mathbf{1}$ and $\mathbf{X}_2 = \mathbf{0}$, we derive a Σ-protocol for plaintext equality. The protocol is efficiently generalized to vector commitments by running it in parallel for all entries and merging the outputs. The full description of this protocol is depicted in Figure 5.4. This protocol involves executing the Σ-protocol for well-formedness (as illustrated in Figure 5.2) two times, along with an additional check, $\mathbf{x}_1 \overset{?}{=} \mathbf{x}_2$ for $e = 0$ or $\mathbf{x}_1' \overset{?}{=} \mathbf{x}_2'$ for $e = 1$, that incurs a computationally insignificant overhead.

5.7.4 Protocol Flow

Now we have all the necessary ingredients to explain our proof of shuffle. The protocol flow is shown in detail in Figure 5.5. We proceed step by step as explained in the Main Idea Section 5.7.2.

1. **Initial Setup:** P takes as input a pair of statement/witness as $((\vec{c}, \vec{m}), (\sigma, \vec{d}))$ that respects the following shuffle relation:

$$\mathcal{R} = \{((\vec{c}, \vec{m}), (\sigma, \vec{d})) \colon \mathsf{Verify}'(\sigma(\vec{m}), \vec{c}, \vec{d}) = 1\},$$

   and V takes as input the statement $(\vec{c}, \vec{m})$.

2. **Step 1 - Prover Chooses a Random Permutation:** P chooses a random permutation $\pi \in \mathcal{S}_k$ and then commits to the permuted vector $\pi(\vec{m})$ by generating a commitment/opening pair $(\vec{\gamma}, \vec{\delta})$ as follows:

$$(\vec{\gamma}, \vec{\delta}) \overset{R}{\leftarrow} \mathsf{Com}(\pi(\vec{m})).$$

3. **Step 2 - Verifier's Challenge:** V receives the commitment vector $\vec{\gamma}$ and returns a random bit $e \overset{R}{\leftarrow} \{0, 1\}$ as a challenge.

4. **Step 3 - Response based on Challenge:**

   (a) If the challenge bit e is 0, P returns $z \leftarrow (\pi, \vec{\delta})$. V then checks whether the following verification condition holds:

$$\mathsf{Verify}'(\pi(\vec{m}), \vec{\gamma}, \vec{\delta}) \overset{?}{=} 1$$

Optimized Σ-protocol of plaintext equality

```
Prover $\mathsf{P}((\mathbf{A},\mathbf{c}_1,\mathbf{c}_2),(\mathbf{e}_1,\mathbf{e}_2,\mathbf{s}_1,\mathbf{s}_2))$                Verifier $\mathsf{V}(\mathbf{A},\mathbf{c}_1,\mathbf{c}_2)$

$\mathbf{x} \xleftarrow{R} \mathcal{I}^v$
foreach $i$ in $\{1,2\}$ do
    $\pi_i \xleftarrow{R} \mathcal{S}_k, \mathbf{u}_i \xleftarrow{R} \mathcal{I}^\ell, \mathbf{f}_i \xleftarrow{R} \mathcal{I}^k$
    $\mathbf{v}_i \leftarrow \mathbf{u}_i \| \mathbf{x}$
    $(c_{i,0}, d_{i,0}) \xleftarrow{R} \mathsf{Com}'(\pi_i, \mathbf{t}_{i,0} = \mathbf{A}\cdot\mathbf{v}_i \oplus \mathbf{f}_i)$
    $(c_{i,1}, d_{i,1}) \xleftarrow{R} \mathsf{Com}'(\mathbf{t}_{i,1} = \pi_i(\mathbf{f}_i))$
    $(c_{i,2}, d_{i,2}) \xleftarrow{R} \mathsf{Com}'(\mathbf{t}_{i,2} = \pi_i(\mathbf{f}_i \oplus \mathbf{e}_i))$
$a \leftarrow (c_{i,0}, c_{i,1}, c_{i,2})_{i\in\{1,2\}}$

                              $\xrightarrow{a}$

                                                  $e \xleftarrow{R} \{0,1,2\}$

                              $\xleftarrow{e}$

if $e = 0$ then
    $z \leftarrow (\pi_i, \mathbf{t}_{i,0}, \mathbf{t}_{i,1}, d_{i,0}, d_{i,1}, \mathbf{v}_i)_{i\in\{1,2\}}$
elseif $e = 1$ then
    $\mathbf{v}'_i \leftarrow (\mathbf{u}'_i \| \mathbf{x}'_i) = \mathbf{v}_i \oplus \mathbf{s}_i$
    $z \leftarrow (\pi_i, \mathbf{t}_{i,0}, \mathbf{t}_{i,2}, d_{i,0}, d_{i,2}, \mathbf{v}'_i)_{i\in\{1,2\}}$
elseif $e = 2$ then
    $z \leftarrow (\mathbf{t}_{i,1}, \mathbf{t}_{i,2}, d_{i,1}, d_{i,2})_{i\in\{1,2\}}$

                              $\xrightarrow{z}$

                                                  if $e = 0$ then
                                                      foreach $i$ in $\{1,2\}$ do
                                                          $\mathsf{Verify}'(\mathbf{t}_{i,0}, c_{i,0}, d_{i,0}) \overset{?}{=} 1, \mathsf{Verify}'(\mathbf{t}_{i,1}, c_{i,1}, d_{i,1}) \overset{?}{=} 1$
                                                          $\mathbf{t}_{i,0} \oplus \pi_i^{-1}(\mathbf{t}_{i,1}) \overset{?}{=} \mathbf{A}\cdot\mathbf{v}_i, \pi_i \overset{?}{\in} \mathcal{S}_k$
                                                      $\mathbf{x}_1 \overset{?}{=} \mathbf{x}_2$ (where $(\mathbf{u}_i \| \mathbf{x}_i) = \mathbf{v}_i$)
                                                  if $e = 1$ then
                                                      foreach $i$ in $\{1,2\}$ do
                                                          $\mathsf{Verify}'(\mathbf{t}_{i,0}, c_{i,0}, d_{i,0}) \overset{?}{=} 1, \mathsf{Verify}'(\mathbf{t}_{i,2}, c_{i,2}, d_{i,2}) \overset{?}{=} 1$
                                                          $\mathbf{t}_{i,0} \oplus \pi_i^{-1}(\mathbf{t}_{i,2}) \oplus \mathbf{c}_i \overset{?}{=} \mathbf{A}\cdot\mathbf{v}'_i$
                                                      $\mathbf{x}'_1 \overset{?}{=} \mathbf{x}'_2$
                                                  if $e = 2$ then
                                                      foreach $i$ in $\{1,2\}$ do
                                                          $\mathsf{Verify}'(\mathbf{t}_{i,1}, c_{i,1}, d_{i,1}) \overset{?}{=} 1, \mathsf{Verify}'(\mathbf{t}_{i,2}, c_{i,2}, d_{i,2}) \overset{?}{=} 1$
                                                          $\|\mathbf{t}_{i,1} \oplus \mathbf{t}_{i,2}\|_1 \overset{?}{=} w$
```

FIGURE 5.4: Σ-protocol for $\mathcal{R} = \{((\mathbf{A},\mathbf{c}_1,\mathbf{c}_2),(\mathbf{e}_1,\mathbf{e}_2,\mathbf{s}_1,\mathbf{s}_2))\colon \mathbf{c}_1 = \mathbf{A}\cdot\mathbf{s}_1 \oplus \mathbf{e}_1 \wedge \mathbf{c}_2 = \mathbf{A}\cdot\mathbf{s}_2 \oplus \mathbf{e}_2 \wedge (\exists \mathbf{r}_1,\mathbf{r}_2 \in \mathcal{I}^\ell\ \exists \mathbf{m} \in \mathcal{I}^v\colon \mathbf{s}_1 = (\mathbf{r}_1\|\mathbf{m}) \wedge s_2 = (\mathbf{r}_2\|\mathbf{m}))\}$ (equality of plaintexts).

(b) If the challenge bit e is 1, P returns the composite permutation $\sigma' = \sigma \circ \pi^{-1}$, and P and verifier run the Σ-protocol of plaintext equality P_{eq}

depicted in Figure 5.4. The input to P in this case is the statement/witness pair:

$$((\vec{c}, \sigma'(\vec{\gamma})), (\sigma(\vec{m}), \vec{d}, \sigma'(\vec{\delta}))),$$

and V takes as input the statement $(\vec{c}, \sigma'(\vec{\gamma}))$. Indeed, this is because we want to prove the two commitments $\vec{c}$ and $\sigma'(\vec{\gamma})$ with the corresponding opening vectors $\vec{d}$ and $\sigma'(\vec{\delta})$, open to the same message $\sigma(\vec{m})$. If the plaintext equality check is successful, V returns 1; otherwise, it returns 0.

Σ-protocol of shuffle

Prover $\mathsf{P}((\vec{c}, \vec{m}), (\sigma, \vec{d}))$		Verifier $\mathsf{V}(\vec{c}, \vec{m})$
$\pi \xleftarrow{R} \mathcal{S}$		
$(\vec{\gamma}, \vec{\delta}) \xleftarrow{R} \mathsf{Com}(\pi(\vec{m}))$		
	$\xrightarrow{\gamma}$	
		$e \xleftarrow{R} \{0,1\}$
	$\xleftarrow{e}$	
if $e = 0$ **then**		
$z \leftarrow (\pi, \vec{\delta})$		
	$\xrightarrow{z}$	
		$\mathsf{Verify}'(\pi(\vec{m}), \vec{\gamma}, \vec{\delta}) \overset{?}{=} 1$
		$\pi \overset{?}{\in} \mathcal{S}$
elseif $e = 1$ **then**		
$z \leftarrow (\sigma' = \sigma \circ \pi^{-1})$		
	$\xrightarrow{z}$	
		$\sigma' \overset{?}{\in} \mathcal{S}$
	$\left\langle \mathsf{P}_{eq}((\vec{c}, \sigma'(\gamma)), (\sigma(\vec{m}), \vec{d}, \sigma'(\delta))), \mathsf{V}_{eq}(\vec{c}, \sigma'(\gamma)) \right\rangle$ $\longleftrightarrow$	

FIGURE 5.5: Interactive proof for $\mathcal{R} = \{((\vec{c}, \vec{m}), (\sigma, \vec{d})) : \mathsf{Verify}(\sigma(\vec{m}), \vec{c}, \vec{d}) = 1\}$ (proof of shuffle). P_{eq} denotes the Σ-protocol of plaintext equality described in Figure 5.4.

5.7.5 Security Analysis and efficiency

Security

The security of our proof of shuffle relies on robust theoretical foundations, as formally verified in Coq, in the research work [HM21] by Haines and Müller. Their paper presents a machine-verified security framework where an abstract version of our proof of shuffle was validated, treating the cryptographic primitives as black boxes. The foundation of our code-based proof of shuffle lies in two key components: the commitment scheme and the Σ-protocol for plaintext equality. In previous sections, we demonstrated that these components satisfy the necessary cryptographic properties, such as binding, hiding, and zero-knowledge. Consequently, our concrete implementation, which uses these primitives, achieves the desired security guarantees under the hardness of decoding random linear codes. To give an intuition on the other two security properties of a $\Sigma-$protocol, special honest verifier zero-knowledge (SHVZK) and Special Soundness, we provide the following intuition.

- **SHVZK**: The information that P sends to V in each protocol run, depends on the challenge bit e provided by V. Based on this value, P either reveals the individual links between the commitment vectors (open $\mathsf{Com}(\pi(\vec{m}))$) or the individual links between the intermediate commitments and the plaintext messages (plaintext equality test of the commitment vectors $\mathbf{c}$ and $\sigma'(\vec{\gamma})$). This selective exposure maintains the verifier's inability to deduce the hidden permutation, even with repeated interactions.

- **Special Soundness**: The protocol guarantees that a cheating prover P^* cannot succeed without possessing a valid witness for the shuffle relation. Indeed, if V challenges the prover with both possible challenge values of 0 and 1 for the same instance, the prover's respective responses can be used to reconstruct the witness.

Efficiency

To achieve a target soundness error of 2^{-16}, for the soundness error of $\frac{2}{3}$ that our proof of shuffle inherits from the underlying $\Sigma-$protocol for linear relations, considering y repetitions, we need to solve the following inequality.

$$\left(\frac{2}{3}\right)^y \leq 2^{-16}.$$

which implies

$$n \geq \frac{\log(2^{-16})}{\log\left(\frac{2}{3}\right)}.$$

A simple calculation shows that y has to be minimum 28. Similarly, for soundness error 2^{-32}, the number of repetitions, y, is 55.

5.8 Verifiable Code-Based E-Voting Scheme

In this section, we explain our proposed e-voting scheme designed to be quantum resistant. This is the first e-voting protocol that ensures both verifiability and vote privacy through the computational difficulty of decoding random linear codes, marking it as the inaugural verifiable code-based e-voting system.

It is important to clarify that *code-based e-voting* should not be confused with *code voting*, as the term *code* in these contexts refers to entirely different concepts. In code-based e-voting, the term relates to code-based cryptography (see Section 2.4.2). Conversely, in code voting, the term refers to *return codes*, which serve as mechanisms for verifying votes. Examples of code voting systems include Pretty Good Democracy [RT09], Surevote [Cha], and the system described in [Gjø13]. In general, return codes function as tokens provided to voters after they cast their votes. These tokens, rooted in cryptographic principles, do not reveal the voter's choice but enable recorded-as-cast verifiability by allowing voters to confirm that their votes have been accurately recorded (see Section 2.2.4).

In the introduction of this thesis and also in the previous chapters, we mentioned that a secure e-voting system strives for both public verifiability and vote privacy at the same time. Our secure e-voting system utilizes the mixing approach for tallying the ballot. See Section 5.3 for the entities running the e-voting system and their roles. The high-level idea of our proposal is that the voters first commit to their votes and then publish them on PBB and send the opening values privately to the tallier T. Later, in the tally phase, the tallier permutes the votes with a secret permutation, publishes the result on PBB, and provides proof of correct shuffle on PBB for the public to verify the final result of the election, which is an aggregation of the publicly available shuffled votes on the public bulletin board.

Cryptographic Primitives

The cryptographic primitives that we use as a building block of our e-voting system have all been explained in Section 5.4. More precisely, we use the following constructions: xLPN-based CS from Section 5.5.1, optimized xLPN-based NIZK of well-formedness from Section 5.5.3 denoted by Π_{wf}, our xLPN-based NIZKP of shuffle denoted by Π_{sh} from Section 5.7, and last but not least, an arbitrary code-based IND-CCA secure PKE (e.g., based on KEM BIKE or HQC explained in Section 2.4.2).

5.8.1 E-Voting Phases

Building on the security measures described, this section breaks down the stages of our e-voting protocol, detailing each phase: setup, submission, tally, and verification. At each stage, specific parties engage with carefully chosen cryptographic techniques to ensure post-quantum security.

Setup Phase

EA and T perform the following tasks.

- Parameter Setup (EA):
  - Determines key system parameters:
    - Security parameter: $\ell \in \mathbb{N}$.
    - Noise parameter: $0 < \tau < 0.25$.
    - Message length: $v \in \mathbb{N}$.
    - Parameter: $k \in \Theta(\ell + v)$.
    - Weight: $w = [\tau k]$.
  - Assumes these parameters as implicit inputs to all algorithms in the CS.
- Public Key Generation for the CS (EA):
  - Runs the key generation algorithm KeyGen to generate the public key $\mathbf{A}$ for the commitment scheme.
  - Publishes the following on the public bulletin board (PBB):
    - The parameters: ℓ, τ, v, k, w.
    - The public key $\mathbf{A}$.
- Key Generation (T):

  - Executes the public-key encryption scheme's key generation algorithm KeyGen_{pk} to generate:
    - Public encryption key: pk.
    - Private decryption key: sk.
  - Publishes pk on PBB for use in the system, while securely retaining sk.

Submission Phase

Each voter V_i performs the following tasks.

1. Parameter Initialization: Retrieve all necessary parameters from PBB.

2. Favorite Choice Selection: Select her preferred candidate $\mathbf{m}_i \in \mathcal{I}^v$.

3. Commitment Generation: Commit to $\mathbf{m}$ by computing the commitment/opening pair $(\mathbf{c}_i, (\mathbf{r}_i, \mathbf{e}_i)) \leftarrow \mathsf{Com}(\mathbf{A}, \mathbf{m}_i)$.

4. Proof of Well-Formedness: Verify validity of the commitment, by generating the NIZKP of well-formedness π^i_{wf} as $\pi^i_{wf} \leftarrow \Pi_{wf}(\mathbf{A}, \mathbf{c}_i, \mathbf{m}_i, \mathbf{r}_i, \mathbf{e}_i)$.

5. Encryption: Using pk, encrypt $(\mathbf{m}_i, \mathbf{c}_i, \mathbf{r}_i)$ as a ciphertext $\mathbf{ct}_i \leftarrow \mathsf{Enc}(pk, (\mathbf{m}_i, \mathbf{c}_i, \mathbf{r}_i))$.

6. Ballot submission: Form the final ballot as $b_i \leftarrow (i, \mathbf{c}_i, \pi^i_{wf}, \mathbf{ct}_i)$ and publish it on PBB.

Tally Phase

The tallier T performs the following tasks.

1. Reading Encrypted Ballots: Retrieves all ballots $b_i = (i, \mathbf{c}_i, \pi^i_{wf}, \mathbf{ct}_i)$ from PBB.

2. Initial Filtering:

   - Discards ballots that:
     - Do not have the correct format.
     - Contain duplicate entries.
     - Contain invalid proofs of well-formedness (π^i_{wf}).[5]

3. Decryption and Validation:

   - For each remaining ballot, performs the following steps.
     - Uses its secret key sk to decrypt the ciphertext $\mathbf{ct}_i$, obtaining the plaintext.
     - Checks if the plaintext is a valid opening of the ballot's commitment $\mathbf{c}_i$.
     - Discards the ballot if this validation fails.

4. Result Computation:

   - After filtering, computes a uniformly random permutation $\sigma \in \mathcal{S}_k$ to anonymize the votes.

[5]The last two steps prevent replay attacks, where a voter's ballot is duplicated to amplify its choice, compromising vote privacy [MMR23].

- Determines the final result as $res \leftarrow (\mathbf{m}_{\sigma(i)})_{i\in[n]}$.[6]

5. Proof of Shuffle: Uses the opening values $(\mathbf{r}_i, \mathbf{e}_i)_{i\in[n]}$ of all commitments and the permutation σ to compute a proof of shuffle:

$$\pi_{sh} \leftarrow \Pi_{sh}(((\mathbf{c}_i)_{i\in[n]}, (\mathbf{m}_i)_{i\in[n]}), (\sigma, (\mathbf{r}_i, \mathbf{e}_i)_{i\in[n]})).$$

6. Publishing the Result: Posts the final result res and the proof of shuffle π_{sh} on PBB.

Verification Phase

For individual verifiability, every voter V_i can check whether their ballot b_i appears on PBB as an input to the tally phase. For public verifiability, anyone including third parties can confirm whether the proof of shuffle π_{sh} generated by T is correct w.r.t. the committed votes $(\mathbf{c}_i)_{i\in[n]}$ and the final result $res \leftarrow (\mathbf{m}_{\sigma(i)})_{i\in[n]}$.

This concludes the explanation of the e-voting phases. Next, we will discuss the security and performance of our proposed electronic voting system.

5.8.2 Security

As we explained earlier, a secure electronic voting system must provide both public verifiability and vote privacy. We also previously highlighted that the underlying commitment scheme and zero-knowledge proofs (ZKPs) used to instantiate our voting system ensure the required security properties. Furthermore, in Section 5.2 of [Ara+13], the authors formally verify an abstract version of our e-voting system, treating the cryptographic primitives as black boxes. These two observations, taken together, lead to the conclusion that our code-based e-voting system achieves the features we have claimed.

Next, we explain why our system provides public verifiability and ballot privacy.

5.8.2.1 Verifiability

Our e-voting protocol guarantees verifiability under two main assumptions:

- Trust in the Public Bulletin Board PBB: The PBB is assumed to be honest, meaning that it accurately posts all ballots without modification or omission. This assumption is essential for ensuring that all voters' submissions are publicly available and verifiable. In this assumption, we consider the voters V and the tallier T can be malicious.

- xLPN Hardness Assumption: explained in Sections 2.4.2 and 5.4.

The protocol ensures that even if voters act maliciously, they cannot compromise the integrity of the voting process. This protection is achieved through: the soundness of well-formedness proofs π^i_{wf} generated by each voter V_i. This validation step prevents the submission of incorrectly formatted ballots from being included in the final count and ensures that the submitted ballots are correctly structured according to the protocol's rules.

In addition, our protocol protects against malicious T by ensuring that T cannot tamper with the result even if he acts maliciously. This protection is achieved

[6]For notation simplicity, we assume all ballots are correct.

through: the soundness of proof of shuffle π^i_{wf} generated by T. This validation step ensures that up to a permutation σ, the list of votes in the final result matches the list of messages in the input commitments.

5.8.2.2 Privacy

Our e-voting protocol guarantees vote privacy under the following assumptions.

- Trust in the Public Bulletin Board PBB and tallier T: We assume that these two parties are both trusted, while the voters could still be malicious.
- xLPN Hardness Assumption of the CS and the ZKPs
- IND-CCA of the PKE
- ZK Property of well-formedness proofs
- Special Soundness Property of Well-formedness Proofs
- ZK Property of Proof of Shuffle

More precisely, the vote privacy is followed by the computationally (xLPN) hiding property of the commitment scheme, which ensures no one can learn the vote from the commitment, secrecy (IND-CCA) of the public-key encryption scheme, which keeps the plaintext confidential, zero-knowledge property of the proof of well-formedness π_{wf} which guarantees confidentiality of the secret, special soundness of the proof of well-formedness π_{wf}, which ensures that ballots are mutually independent, and last but not least, the zero-knowledge property of the proof of shuffle π_{sh}, which keeps the permutation secret.

5.8.3 Performance

In this section, we analyze the computational workload for both voters and the trustee in our e-voting system based on the parameter selection we explained in Section 5.5.3. We demonstrate that the workload is efficient and practical, even for large-scale elections, with optimizations making the process feasible for real-world applications.

5.8.3.1 Voters' Workload

Each voter is responsible for creating a commitment to their vote and generating a non-interactive zero-knowledge proof of knowledge (NIZKPoK) to prove the correctness of their commitment. Below are the performance details for the voter:

- **Commitment:** According to the Table 4 in [Bel+20]:
  - The size of the commitment is 0.17 Kilobytes.
  - The computation of the commitment takes less than 0.2 milliseconds.
- **NIZKPoK:** According to Table 5 in [Bel+20]:
  - The size of the NIZKP is 35.42 Kilobytes.
  - The computation time for the NIZKP is 28 milliseconds, based on the parameters for a soundness error of 2^{-16}.

  - Verifying the correctness of the NIZKP is equally efficient, requiring less than 28 milliseconds with our optimizations.

- **Ciphertext Size and Speed:** The size and computational time for encrypting the voter's opening values depend on the public-key encryption (PKE) scheme used. Efficient implementations, such as those based on BIKE or HQC KEMs, could make this process practical.

- **Summary:** The overall workload for voters is minimal and feasible even for devices with limited computational resources or poor Internet connections.

5.8.3.2 Tallier's Workload

The tallier T's primary task is to compute the NIZKP of shuffle. This involves significant computation but can be optimized through pre-computation in an offline phase and the efficient protocols we proposed. The following are the details.

- **Baseline Performance:**
  - According to Section 5.7, the prover (the tallier T here) can process about 17 messages per second for the shuffle proof.

- **Pre-Computation Optimizations:**
  - T can pre-compute several components in an offline phase, which are independent of the messages (votes) that need to be shuffled:
    - In the commitment phase of the underlying Σ-protocol of plaintext equality (Figure 5.4), the permutation, random vectors to hide the committed messages, the first two commitments, and randomness for the third commitment can be computed beforehand.
    - In the commitment phase of the $\Sigma-$protocol for proof of shuffle (Figure 5.5), the permutation and randomness for commitments can also be pre-computed.
  - These pre-computation steps significantly reduce the online computational burden during the actual voting process.

- **Performance Improvements:**
  - Using the performance results in [Bel+20], the pre-computation helps the tallying process to accelerate by a factor of about 3.
  - As an example, T can create a NIZKP for over $1,000,000$ voters in less than 6 hours.
  - Without using our optimized version of the ZKPs, it would take over $1,300$ hours to verify a NIZKP for the same number of voters. While using our improved verifier, the same size NIZKP can be verified in less than 12 hours, ensuring practicality for real-world applications.

These performance characteristics demonstrate the practicality of the proposed e-voting system for large-scale elections, making it suitable for real-world deployment while maintaining efficiency and security.

Chapter 6

Conclusion

In this chapter, we conclude the thesis by summarizing the key highlights of the research papers discussed in detail in Chapters 3 ,4 , and 5. This is followed by a discussion of potential directions for future work.

6.1 Summary

In this section, we provide a summary of our contributions throughout this thesis. This work focuses on the design and analysis of post-quantum e-voting schemes. We began by presenting the necessary background on secure e-voting, cryptographic primitives, and post-quantum cryptography, setting the stage for the research. In the introduction, we justified why we chose the code-based cryptography approach to design a post-quantum e-voting system. We then continued with our SoK paper on the subject of everlasting privacy in secure e-voting. We explained in the introduction of the thesis that a system with everlasting privacy provides a stronger privacy guarantee than the one with post-quantum privacy. In Chapter 3, we explained our findings related to critically studying the papers in the literature striving for secure e-voting that guarantees everlasting privacy. We identified the preferred approach. Building on this insight, in the next Chapter 4, we designed an e-voting system with everlasting privacy, which leverages the Hyperion e-voting system as a core component. We call this e-voting system *everlasting Hyperion*. In addition to everlasting privacy, everlasting Hyperion achieves individual and universal verifiability, everlasting receipt-freeness, and provides coercion mitigation. We provided a game-based security proof for each of these security properties. In Chapter 5, the last chapter, we designed a post-quantum (code-based) e-voting system with mixing tally, which uses an optimized and efficient version of a code-based Sigma protocol (ZKP of well-formedness) and a code-based proof of shuffle as its building blocks.

6.2 Future Work

While our works make substantial progress, it also highlights several exciting challenges and opportunities for future research in the context of e-voting, and even a broader cryptographic area, as we summarize next.

- Everlasting Privacy (Based on Chapter 3 and Chapter 4):
  - Formal Protocol Analysis: It is an open problem to formally verify the promising approaches [Cra+96; CPP13; CGG19] on the protocol level. Particularly, formal analysis of EP in Belenios needs to be done.
  - Deployable E-Voting System: Among the promising protocols mentioned in Section 3.5, only Belenios is B-ANON has been deployed. It is an open

problem to develop full-fledged deployable versions of the other promising approaches in the superior class, B-ID.

- Weaker Trust for Arbitrary Ballot Types: There are three promising approaches that can handle complex ballots: [CPP13; CGG19; LH15]. These protocols all need full trust in the EA or all talliers to achieve EP. It is an open problem to mitigate this trust.
- **Receipt-Freeness**: To achieve EP and RF at the same time in a secure e-voting system was an open problem at the time of writing this paper. Now it has been addressed in the literature, including our paper, which is explained in detail in Chapter 4 (see also Section 3.6 for a full list of solutions).
- As outlined in the game-based proof of Everlasting Privacy for the Everlasting Hyperion system in Section 4.7, it is valuable to examine potential privacy leaks in voter verification processes. This includes defining a new experiment that ensures the success or failure of voter verification does not compromise the overall privacy guarantees of the protocol by leaking information about how voters voted.
- It is interesting to conduct formal verification of Everlasting Hyperion, similar to its predecessors Selene, SeleneRF, and Hyperion [Bal+23].

• Post-Quantum Security (Based on Chapter 5):

  - Lightweight Code-Based Proofs of Shuffle:
    One important direction is to design new code-based proofs of shuffle that are more lightweight in terms of computational power and memory requirements. Achieving this could make such proofs more suitable for devices with limited resources, broadening their practical applications. It is interesting to check whether the generic plaintext equality proof by Blazy et al. in [Bla+21] can be applied in our setting, and to compare its efficiency.
  - Decentralized Trust Models:
    Another open problem is the development of a code-based e-voting protocol where the trust in the tallier for ensuring vote privacy is distributed among multiple parties. Such a protocol would enhance trustworthiness and reduce the reliance on a single trusted entity.
  - Scalability and Performance: Exploring methods to further improve the scalability and performance of code-based ZKPs, especially for large-scale applications like national elections remains a significant challenge.

Bibliography

[Ada04] Colin Conrad Adams. *The knot book: an elementary introduction to the mathematical theory of knots*. American Mathematical Soc., 2004.

[Adi08] Ben Adida. "Helios: Web-based Open-Audit Voting". In: *USENIX Security Symposium*. 2008, pp. 335–348.

[Agu+18] Carlos Aguilar-Melchor, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. "Efficient encryption from random quasi-cyclic codes". In: *IEEE Transactions on Information Theory* 64.5 (2018), pp. 3927–3943.

[Ale03] Michael Alekhnovich. "More on average case vs approximation complexity". In: *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.* IEEE. 2003, pp. 298–307.

[AM16] Syed Taha Ali and Judy Murray. "An overview of end-to-end verifiable voting systems". In: *Real-World Electronic Voting* (2016), pp. 189–234.

[Ara+13] Myrto Arapinis, Véronique Cortier, Steve Kremer, and Mark Ryan. "Practical Everlasting Privacy". In: *Principles of Security and Trust*. 2013, pp. 21–40.

[Ara+21] Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, Tjerand Silde, and Thor Tunge. "Lattice-Based Proof of Shuffle and Applications to Electronic Voting". In: *CT-RSA*. Vol. 12704. 2021, pp. 227–251.

[Ara+22] N. Aragon, P. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Gueron, T. Guneysu, C.A. Melchor, R. Misoczki, E. Persichetti, N. Sendrier, J.-P. Tillich, G. Zemor, V. Vasseur, S. Ghosh, and J. Richter-Brokmann. *BIKE: Bit-Flipping Key Encapsulation*. 2022. URL: https://bikesuite.org/files/v5.0/BIKE_Spec.2022.10.10.1.pdf.

[Ara+23] Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, and Tjerand Silde. "Verifiable Mix-Nets and Distributed Decryption for Voting from Lattice-Based Assumptions". In: *SIGSAC*. 2023, pp. 1467–1481.

[BAF08] Bruno Blanchet, Martín Abadi, and Cédric Fournet. "Automated verification of selected equivalences for security protocols". In: *J. Log. Algebraic Methods Program.* 75.1 (2008), pp. 3–51.

[Bal+21a] Sevdenur Baloglu, Sergiu Bursuc, Sjouke Mauw, and Jun Pang. "Election Verifiability Revisited: Automated Security Proofs and Attacks on Helios and Belenios". In: *IEEE CSF*. 2021, pp. 1–15.

[Bal+21b] Sevdenur Baloglu, Sergiu Bursuc, Sjouke Mauw, and Jun Pang. "Provably Improving Election Verifiability in Belenios". In: *E-Vote-ID Proceedings*. 2021, pp. 1–16.

[Bal+23] Sevdenur Baloglu, Sergiu Bursuc, Sjouke Mauw, and Jun Pang. "Election Verifiability in Receipt-Free Voting Protocols". In: *CSF*. 2023, pp. 59–74.

[Bal23] Sevdenur Baloglu. “Formal Verification of Verifiability in E-Voting Protocols”. English. PhD thesis. Unilu - University of Luxembourg, Esch-sur-Alzette, Luxembourg, 2023.

[Bar+23] Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, and Jean-Pierre Tillich. “Revisiting algebraic attacks on MinRank and on the rank decoding problem”. In: *Des. Codes Cryptogr.* 91 (2023), pp. 3671–3707.

[Bar22] Jessica Bariffi. *What is Lattice-Based Cryptography?* `https://user.math.uzh.ch/bariffi/slides_PhDSeminar_lattice.pdf`. 2022.

[Bat+24] Christopher Battarbee, Delaram Kahrobaei, Ludovic Perret, and Siamak F Shahandashti. “A subexponential quantum algorithm for the semidirect discrete logarithm problem”. In: *International Conference on Post-Quantum Cryptography*. 2024, pp. 202–226.

[BDG13] Johannes Buchmann, Denise Demirel, and Jeroen van de Graaf. “Towards a Publicly-Verifiable Mix-Net Providing Everlasting Privacy”. In: *FC 2013, Revised Selected Papers*. 2013, pp. 197–204.

[Bel+20] Emanuele Bellini, Philippe Gaborit, Alexandros Hasikos, and Víctor Mateu. “Enhancing Code Based Zero-Knowledge Proofs Using Rank Metric”. In: *Cryptology and Network Security - CANS*. Vol. 12579. 2020, pp. 570–592.

[Bel+98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. “Relations among notions of security for public-key encryption schemes”. In: *Advances in Cryptology — CRYPTO*. 1998, pp. 26–45.

[Ben06] Josh Benaloh. “Simple Verifiable Elections”. In: *USENIX/ACCURATE Electronic Voting Technology Workshop*. 2006.

[Ben87] Josh Daniel Cohen Benaloh. “Verifiable secret-ballot elections”. PhD thesis. 1987.

[Ber+15] David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. “SoK: A comprehensive analysis of game-based ballot privacy definitions”. In: *2015 IEEE Symposium on Security and Privacy*. IEEE. 2015, pp. 499–516.

[Ber+19] Daniel J Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. “The SPHINCS+ signature framework”. In: *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 2019, pp. 2129–2146.

[Ber+22] D.J. Bernstein, T. Chou, C. Cid, J. Gilcher, T. Lange, V. Maram, I. von Maurich, R. Misoczki, R. Niederhagen, E. Persichetti, C. Peters, N. Sendrier, J. Szefer, C.J. Tjhai, M. Tomlinson, and W. Wang. *Classic McEliece: Conservative Code-Based Cryptography*. 2022. URL: `https://classic.mceliece.org/mceliece-spec-20221023.pdf`.

[Ber73] E. Berlekamp. “Goppa codes”. In: *IEEE Transactions on Information Theory* 19.5 (1973), pp. 590–592.

[Bet+22] Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Yann Connan, and Philippe Gaborit. “A gapless code-based hash proof system based on RQC and its applications”. In: *Des. Codes Cryptogr.* 90 (2022), pp. 3011–3044.

[Beu+21] Ward Beullens, Jan-Pieter D’Anvers, Andreas T. Hülsing, Tanja Lange, Lorenz Panny, Cyprien de Saint Guilhem, and Nigel P. Smart. *Post-Quantum Cryptography: Current state and quantum mitigation*. ENISA, 2021.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "(Leveled) fully homomorphic encryption without bootstrapping". In: *ACM Transactions on Computation Theory (TOCT)* 6.3 (2014), pp. 1–36.

[BH18] Xavier Boyen and Thomas Haines. "Forward-Secure Linkable Ring Signatures". In: *ACISP*. 2018, pp. 245–264.

[BHM20] Xavier Boyen, Thomas Haines, and Johannes Müller. "A Verifiable and Practical Lattice-Based Decryption Mix Net with External Auditing". In: *ESORICS*. Vol. 12309. 2020, pp. 336–356.

[BHM21] Xavier Boyen, Thomas Haines, and Johannes Müller. "Epoque: Practical End-to-End Verifiable Post-Quantum-Secure E-Voting". In: *EuroS&P*. 2021, pp. 272–291.

[Bid+22] Loïc Bidoux, Philippe Gaborit, Mukul Kulkarni, and Nicolas Sendrier. "Quasi-Cyclic Stern Proof of Knowledge". In: *International Symposium on Information Theory, ISIT*. 2022, pp. 1459–1464.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. "Noise-tolerant learning, the parity problem, and the statistical query model". In: *J. ACM* 50 (2003), pp. 506–519.

[BL] Daniel J. Bernstein and Tanja Lange. *Post-quantum cryptography conference*. Tech. rep. URL: https://pqcrypto.org/conferences.html.

[BL17] Daniel J Bernstein and Tanja Lange. "Post-quantum cryptography". In: *Nature* 549.7671 (2017), pp. 188–194.

[Bla+21] Olivier Blazy, Xavier Bultel, Pascal Lafourcade, and Octavio Perez-Kempner. "Generic Plaintext Equality and Inequality Proofs". In: *Financial Cryptography and Data Security - 25th International Conference, FC 2021*. 2021, pp. 415–435.

[BLP08] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. "Attacking and Defending the McEliece Cryptosystem". In: *Post-Quantum Cryptography*. Springer Berlin Heidelberg, 2008, pp. 31–46.

[BMV78] Elwyn Berlekamp, Robert McEliece, and Henk Van Tilborg. "On the inherent intractability of certain coding problems (corresp.)" In: *IEEE Transactions on Information theory* 24.3 (1978), pp. 384–386.

[Bou+20] Samuel Bouaziz-Ermann, Sébastien Canard, Gautier Eberhart, Guillaume Kaim, Adeline Roux-Langlois, and Jacques Traoré. "Lattice-based (Partially) Blind Signature without Restart". In: *IACR Cryptol. ePrint Arch.* (2020), p. 260.

[BPW12] David Bernhard, Olivier Pereira, and Bogdan Warinschi. "How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios". In: *Advances in Cryptology–ASIACRYPT, Proceedings 18*. Springer. 2012, pp. 626–643.

[BS20] Dan Boneh and Victor Shoup. "A graduate course in applied cryptography". In: *Draft 0.5* (2020).

[BT94] Josh Cohen Benaloh and Dwight Tuinstra. "Receipt-free secret-ballot elections (extended abstract)". In: *ACM Symposium on Theory of Computing*. 1994, pp. 544–553.

[CCK12] Rohit Chadha, Ştefan Ciobâcă, and Steve Kremer. "Automated Verification of Equivalence Properties of Cryptographic Protocols". In: *ESOP Proceedings*. Ed. by Helmut Seidl. Vol. 7211. 2012, pp. 108–127.

[CD23] Wouter Castryck and Thomas Decru. "An Efficient Key Recovery Attack on SIDH". In: *Advances in Cryptology - EUROCRYPT*. Springer, 2023, pp. 423–447.

[CFG15] Véronique Cortier, Georg Fuchsbauer, and David Galindo. "BeleniosRF: A Strongly Receipt-Free Electronic Voting Scheme." In: *IACR Cryptol. ePrint Arch.* (2015), p. 629.

[CGG19] Véronique Cortier, Pierrick Gaudry, and Stéphane Glondu. "Belenios: A Simple Private and Verifiable Electronic Voting System". In: *Foundations of Security, Protocols, and Equational Reasoning*. 2019, pp. 214–238.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. "A secure and optimally efficient multi-authority election scheme". In: *European transactions on Telecommunications* 8.5 (1997), pp. 481–490.

[Cha] David Chaum. *Surevote: technical overview*. Tech. rep. URL: http://www.iavoss.org/mirror/wote01/pdfs/surevote.pdf.

[Cha04] David Chaum. "Secret-ballot receipts: True voter-verifiable elections". In: *IEEE security & privacy* 2.1 (2004), pp. 38–47.

[Cha81] David L Chaum. "Untraceable electronic mail, return addresses, and digital pseudonyms". In: *Communications of the ACM* 24.2 (1981), pp. 84–90.

[Cha83] David Chaum. "Blind Signature System". In: *CRYPTO*. 1983, p. 153.

[Cha88] David Chaum. "The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability". In: *J. Cryptol.* (1988), pp. 65–75.

[Che+10] Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré. "On Some Incompatible Properties of Voting Schemes". In: *Towards Trustworthy Elections, New Directions in Electronic Voting*. Springer, 2010, pp. 191–199.

[Che+16] Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. "NIST: Report on Post-Quantum Cryptography". In: *NIST, Tech. Rep.* (2016).

[Che+20] Hao Chen, Lynn Chua, Kristin E. Lauter, and Yongsoo Song. "On the Concrete Security of LWE with Small Secret". In: *IACR Cryptol. ePrint Arch.* (2020), p. 539.

[CLW20] Véronique Cortier, Joseph Lallemand, and Bogdan Warinschi. "Fifty shades of ballot privacy: Privacy against a malicious board". In: *33rd Computer Security Foundations Symposium (CSF)*. IEEE. 2020, pp. 17–32.

[CMM17] Núria Costa, Ramiro Martínez, and Paz Morillo. "Proof of a Shuffle for Lattice-Based Cryptography". In: *Secure IT Systems*. Vol. 10674. 2017, pp. 280–296.

[CMM19] Núria Costa, Ramiro Martínez, and Paz Morillo. "Lattice-Based Proof of a Shuffle". In: *Financial Cryptography and Data Security*. Vol. 11599. 2019, pp. 330–346.

[Coo23] Stephen A Cook. "The complexity of theorem-proving procedures". In: *Logic, automata, and computational complexity: The works of Stephen A. Cook*. ACM, 2023, pp. 143–152.

[Cor+16] Véronique Cortier, David Galindo, Ralf Küsters, Johannes Müller, and Tomasz Truderung. "SoK: Verifiability Notions for E-Voting Protocols". In: *2016 IEEE Symposium on Security and Privacy (SP)*. 2016, pp. 779–798.

[Cor+23] Véronique Cortier, Alexandre Debant, Pierrick Gaudry, and Stéphane Glondu. “Belenios with cast as intended”. In: *Voting 2023 - 8th Workshop on Advances in Secure Electronic Voting*. 2023.

[Cor+24] Véronique Cortier, Alexandre Debant, Anselme Goetschmann, and Lucca Hirschi. “Election Eligibility with OpenID: Turning Authentication into Transferable Proof of Eligibility”. In: USENIX Association, 2024.

[CPP13] Edouard Cuvelier, Olivier Pereira, and Thomas Peters. “Election verifiability or ballot privacy: Do we need to choose?” In: *Computer Security–ESORICS, Proceedings 18*. Springer. 2013, pp. 481–498.

[Cra+96] Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. “Multi-Autority Secret-Ballot Elections with Linear Work”. In: *EUROCRYPT*. 1996, pp. 72–83.

[CRS05] David Chaum, Peter Y. A. Ryan, and Steve Schneider. “A Practical Voter-Verifiable Election Scheme”. In: *Computer Security – ESORICS*. 2005, pp. 118–139.

[Dam+24] Aditya Damodaran, Simon Rastikian, Peter B. Rønne, and Peter Y A Ryan. *Hyperion: Transparent End-to-End Verifiable Voting with Coercion Mitigation*. Cryptology ePrint Archive, Paper 2024/1182. 2024. URL: https://eprint.iacr.org/2024/1182.

[Dam+95] Ivan Damgård, Oded Goldreich, Tatsuaki Okamoto, and Avi Wigderson. “Honest Verifier vs Dishonest Verifier in Public Coin Zero-Knowledge Proofs”. In: *Advances in Cryptology - CRYPTO ’95, 15th Annual International Cryptology Conference Proceedings*. 1995, pp. 325–338.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. “Non-malleable cryptography”. In: *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*. 1991, pp. 542–552.

[De 17] Luca De Feo. “Mathematics of isogeny based cryptography”. In: *arXiv preprint arXiv:1711.04062* (2017).

[Deh11] Max Dehn. “Über infinite discontinuous groups”. In: *Mathematical Annals* 71.1 (1911), pp. 116–144.

[Dem+13] Denise Demirel, Maria Henning, Jeroen van de Graaf, Peter Y. A. Ryan, and Johannes Buchmann. “Prêt à Voter Providing Everlasting Privacy”. In: *VoteID*. 2013, pp. 156–175.

[Dem13] Denise Demirel. “Universally verifiable poll-site voting schemes providing everlasting privacy”. PhD thesis. Darmstadt University of Technology, 2013.

[Dep] Chambre des Deputes GRAND-DUCHE DE LUXEMBOURG. *The history of elections*. Tech. rep. URL: https://www.chd.lu/fr/lhistoire-des-elections.

[DER21] Ivan Damgård, Daniel Escudero, and Divya Ravi. “Information-Theoretically Secure MPC Against Mixed Dynamic Adversaries”. In: *TCC 2021*. 2021, pp. 591–622.

[DGS12] Denise Demirel, Jeroen van de Graaf, and Roberto Samarone dos Santos Araújo. “Improving Helios with Everlasting Privacy Towards the Public”. In: *Electronic Voting Technology Workshop*. USENIX Association, 2012.

[DH76] Whitfield Diffie and Martin E. Hellman. “New directions in cryptography”. In: *IEEE Trans. Inf. Theory* 22 (1976), pp. 644–654.

[DKR06] Stéphanie Delaune, Steve Kremer, and Mark Ryan. “Coercion-Resistance and Receipt-Freeness in Electronic Voting”. In: *19th IEEE Computer Security Foundations Workshop, CSFW-19*. 2006, pp. 28–42.

[DMV23] Xuan-Thanh Do, Dang Truong Mac, and Quoc-Huy Vu. “zk-SNARKs from Codes with Rank Metrics”. In: *IMACC*. 2023, pp. 99–119.

[DPP24] Thi Van Thao Doan, Olivier Pereira, and Thomas Peters. “Encryption Mechanisms for Receipt-Free and Perfectly Private Verifiable Elections”. In: *ACNS*. 2024, pp. 257–287.

[Dra+22] Constantin Catalin Dragan, Francois Dupressoir, Ehsan Estaji, Kristian Gjoesteen, Thomas Haines, Peter YA Ryan, Morten Rotvold Solberg, et al. “Machine-checked proofs of privacy against malicious boards for Selene & Co”. In: *35th Computer Security Foundations Symposium (CSF)*. IEEE. 2022, pp. 335–347.

[DY09] Jintai Ding and Bo-Yin Yang. “Multivariate public key cryptography”. In: *Post-quantum cryptography* (2009), pp. 193–241.

[Dya18] Mikhail Dyakonov. *The Case Against Quantum Computing*. Tech. rep. 2018. URL: https://spectrum.ieee.org/the-case-against-quantum-computing.

[Est+20] Ehsan Estaji, Thomas Haines, Kristian Gjøsteen, Peter B. Rønne, Peter Y. A. Ryan, and Najmeh Soroush. “Revisiting Practical and Usable Coercion-Resistant Remote E-Voting”. In: *E-Vote-ID*. Vol. 12455. 2020, pp. 50–66.

[FOO92] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. “A Practical Secret Voting Scheme for Large Scale Elections”. In: *AUSCRYPT*. 1992, pp. 244–251.

[FP01] Pierre-Alain Fouque and David Pointcheval. “Threshold cryptosystems secure against chosen-ciphertext attacks”. In: *Advances in Cryptology-ASIACRYPT, Proceedings 7*. Springer. 2001, pp. 351–368.

[FS86] Amos Fiat and Adi Shamir. “How to prove yourself: Practical solutions to identification and signature problems”. In: *Conference on the theory and application of cryptographic techniques*. 1986, pp. 186–194.

[Gam83] Taher El Gamal. “A Subexponential-Time Algorithm for Computing Discrete Logarithms over GF(p^2)”. In: *CRYPTO 1983*. Plenum Press, 1983, pp. 275–292.

[Gam85] Taher El Gamal. “A public key cryptosystem and a signature scheme based on discrete logarithms”. In: *IEEE Trans. Inf. Theory* 31 (1985), pp. 469–472.

[Ge+19] Huangyi Ge, Sze Yiu Chau, Victor E. Gonsalves, Huian Li, Tianhao Wang, Xukai Zou, and Ninghui Li. “Koinonia: verifiable e-voting with long-term privacy”. In: *ACSAC*. 2019, pp. 270–285.

[GHS20] Kristian Gjøsteen, Thomas Haines, and Morten Rotvold Solberg. “Efficient mixing of arbitrary ballots with everlasting privacy: How to verifiably mix the PPATC scheme”. In: *Secure IT Systems: NordSec, Proceedings 25*. 2020, pp. 92–107.

[Gjø+22] Kristian Gjøsteen, Thomas Haines, Johannes Müller, Peter B. Rønne, and Tjerand Silde. “Verifiable Decryption in the Head”. In: *ACISP*. 2022, pp. 355–374.

[Gjø13] Kristian Gjøsteen. “The Norwegian Internet Voting Protocol”. In: *IACR Cryptol. ePrint Arch.* (2013), p. 473.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. “A digital signature scheme secure against adaptive chosen-message attacks”. In: *SIAM Journal on computing* 17.2 (1988), pp. 281–308.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. “The Knowledge Complexity of Interactive Proof Systems”. In: *SIAM J. Comput.* 18 (1989), pp. 186–208.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. “Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design (Extended Abstract)”. In: IEEE Computer Society, 1986, pp. 174–187.

[Gol+02] Solomon W Golomb, Elwyn Berlekamp, Thomas M Cover, Robert G Gallager, James L Massey, and Andrew J Viterbi. “Claude elwood shannon”. In: *Notices of the AMS* 49.1 (2002).

[Gol23] Shafi Goldwasser. *Zero-Knowledge Proof MOOC*. `https://zk-learning.org`. 2023.

[Gov24] Luxembourg Government. *Conseil Supérieur du Vivre-Ensemble Interculturel*. Accessed: 2025-01-10. 2024. URL: `https://mfsva.gouvernement.lu/fr/le-ministere/attributions/integration/conseilsuperieur.html`.

[GPS22] Shay Gueron, Edoardo Persichetti, and Paolo Santini. “Designing a Practical Code-Based Signature Scheme from Zero-Knowledge Proofs with Trusted Setup”. In: *Cryptogr.* 6 (2022), p. 5.

[Gra09] Jeroen van de Graaf. “Voting with unconditional privacy: CFSY for booth voting”. In: *IACR Cryptol. ePrint Arch.* (2009).

[Gra10] Jeroen van de Graaf. “Anonymous One-Time Broadcast Using Non-interactive Dining Cryptographer Nets with Applications to Voting”. In: *Towards Trustworthy Elections, New Directions in Electronic Voting*. 2010, pp. 231–241.

[Gro+18] Panagiotis Grontas, Aris Pagourtzis, Alexandros Zacharakis, and Bingsheng Zhang. “Towards Everlasting Privacy and Efficient Coercion Resistance in Remote Electronic Voting”. In: *FC International Workshops*. 2018, pp. 210–231.

[Hai+23] Thomas Haines, Johannes Mueller, Rafieh Mosaheb, and Ivan Pryvalov. “Sok: Secure e-voting with everlasting privacy”. In: *Privacy Enhancing Technologies Symposium (PETS)*. 2023.

[Hai+25] Thomas Haines, Rafieh Mosaheb, Johannes Müller, and Reetika Reetika. “Zero-Knowledge Proofs from Learning Parity with Noise: Optimization, Verification, and Application”. In: *IEEE Computer Security Foundations (CSF) Symposium*. 2025.

[Hai19] Thomas Haines. “Cronus: Everlasting Privacy with Audit and Cast”. In: *NordSec*. 2019, pp. 53–68.

[Har82] Juris Hartmanis. “Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson)”. In: *Siam Review* 24 (1982), p. 90.

[Hau+20] Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen. “Lattice-Based Blind Signatures, Revisited”. In: *CRYPTO*. 2020, pp. 500–529.

[HB16] Thomas Haines and Xavier Boyen. “VOTOR: Conceptually simple remote voting against tiny tyrants”. In: *ACM ACSWM*. 2016, p. 32.

[Hen24] Nadia Heninger. “Lessons from Pre-Quantum Cryptography”. In: *Post-Quantum Algebraic Cryptography Workshop, IHP* (2024).

[HM20] Thomas Haines and Johannes Müller. “SoK: Techniques for Verifiable Mix Nets”. In: *CSF*. 2020, pp. 49–64.

[HM21] Thomas Haines and Johannes Müller. “A Novel Proof of Shuffle: Exponentially Secure Cut-and-Choose”. In: *ACISP*. Vol. 13083. 2021, pp. 293–308.

[HMQ23] Thomas Haines, Johannes Müller, and Iñigo Querejeta-Azurmendi. “Scalable Coercion-Resistant E-Voting under Weaker Trust Assumptions”. In: *ACM/SIGAPP Symposium on Applied Computing, SAC*. 2023, pp. 1576–1584.

[HMS21] Javier Herranz, Ramiro Martínez, and Manuel Sánchez. “Shorter Lattice-Based Zero-Knowledge Proofs for the Correctness of a Shuffle”. In: *Financial Cryptography and Data Security*. Vol. 12676. 2021, pp. 315–329.

[HMT13] Rong Hu, Kirill Morozov, and Tsuyoshi Takagi. “Proof of plaintext knowledge for code-based public-key encryption revisited”. In: *CCS*. 2013, pp. 535–540.

[HSB21] Lucca Hirschi, Lara Schmid, and David A. Basin. “Fixing the Achilles Heel of E-Voting: The Bulletin Board”. In: *IEEE CSF*. 2021, pp. 1–17.

[HSS23] Patrick Hough, Caroline Sandsbråten, and Tjerand Silde. “Concrete NTRU Security and Advances in Practical Lattice-Based Electronic Voting”. In: *IACR Cryptol. ePrint Arch.* (2023), p. 933.

[HV08] Benjamin Hosp and Poorvi L. Vora. “An information-theoretic model of voting systems”. In: *Math. Comput. Model.* (2008), pp. 1628–1645.

[Iov+17] Vincenzo Iovino, Alfredo Rial, Peter B. Rønne, and Peter Y. A. Ryan. “Using Selene to Verify Your Vote in JCJ”. In: *FC 2017 International Workshops*. 2017, pp. 385–403.

[Jai+12] Abhishek Jain, Stephan Krenn, Krzysztof Pietrzak, and Aris Tentes. “Commitments and Efficient Zero-Knowledge Proofs from Learning Parity with Noise”. In: *Advances in Cryptology - ASIACRYPT*. Vol. 7658. 2012, pp. 663–680.

[Jao+22] D. Jao, R. Azarderakhsh, M. Campagna, C. Costello, L. De Feo, B. Hess, A. Hutchinson, A. Jalali, K. Karabina, B. Koziel, B. LaMacchia, P. Longa, M. Naehrig, G. Pereira, J. Renes, V. Soukharev, and D. Urbanik. *Supersingular Isogeny Key Encapsulation*. 2022. URL: https://sike.org/files/SIDH-spec.pdf.

[JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson. “Coercion-resistant electronic elections”. In: *Workshop on Privacy in the Electronic Society - WPES*. ACM, 2005, pp. 61–70.

[JD11] David Jao and Luca De Feo. "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies". In: *Proceedings of 4th International Workshop PQCrypto*. Springer. 2011, pp. 19–34.

[Kah96] David Kahn. *The Codebreakers: The comprehensive history of secret communication from ancient times to the internet*. Simon and Schuster, 1996.

[Kai+21] Guillaume Kaim, Sébastien Canard, Adeline Roux-Langlois, and Jacques Traoré. "Post-quantum Online Voting Scheme". In: *FC International Workshops, Revised Selected Papers*. 2021, pp. 290–305.

[Kal20] Gil Kalai. "The argument against quantum computers, the quantum laws of nature, and Google's supremacy claims". In: *arXiv:2008.05188* (2020).

[KFN+23] Delaram Kahrobaei, Ramon Flores, Marialaura Noce, et al. "Group-based cryptography in the quantum era". In: *Not. Am. Math. Soc* 70 (2023), pp. 752–763.

[Kil+21] Christian Killer, Markus Knecht, Claude Müller, Bruno Rodrigues, Eder J. Scheid, Muriel Figueredo Franco, and Burkhard Stiller. "Æternum: A Decentralized Voting System with Unconditional Privacy". In: *IEEE ICBC*. 2021.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[KTV11] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. "Verifiability, privacy, and coercion-resistance: New insights from a case study". In: *IEEE Symposium on Security and Privacy*. IEEE. 2011, pp. 538–553.

[Küs+20] Ralf Küsters, Julian Liedtke, Johannes Müller, Daniel Rausch, and Andreas Vogt. "Ordinos: A verifiable tally-hiding e-voting system". In: *IEEE European Symposium on Security and Privacy (EuroS&P)*. 2020, pp. 216–235.

[Lam79] Leslie Lamport. "Constructing digital signatures from a one way function". In: *Microsoft Research* (1979).

[LH15] Philipp Locher and Rolf Haenni. "Verifiable Internet Elections with Everlasting Privacy and Minimal Trust". In: *VoteID 2015*. 2015, pp. 74–91.

[LH16] Philipp Locher and Rolf Haenni. "Receipt-free remote electronic elections with everlasting privacy". In: *Ann. des Télécommunications* (2016), pp. 323–336.

[LHK16] Philipp Locher, Rolf Haenni, and Reto E. Koenig. "Coercion-Resistant Internet Voting with Everlasting Privacy". In: *FC*. 2016, pp. 161–175.

[LLL82] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. "Factoring polynomials with rational coefficients". In: *Mathematische annalen* 261 (1982), pp. 515–534.

[LW14] Steven P. Lalley and E. Glen Weyl. "Nash Equilibria for a Quadratic Voting Game". In: *CoRR* abs/1409.0264 (2014).

[Lyn07] Ben Lynn. "On the implementation of pairing-based cryptosystems". In: *Stanford University Stanford* (2007).

[Lyu+21] V. Lyubashevsky, L. Ducas, E. Kiltz, T. Lepoint, P. Schwabe, G. Seiler, D. Stehle, and S. Bai. *CRYSTALS-DILITHIUM*. 2021. URL: `https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf`.

[McE78] Robert J McEliece. "A public-key cryptosystem based on algebraic". In: *Coding Thv* 4244 (1978), pp. 114–116.

[Mel+23] Carlos Aguilar Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. "The Return of the SDitH". In: *Advances in Cryptology - EUROCRYPT*. 2023, pp. 564–596.

[Mer79] Ralph Charles Merkle. *Secrecy, authentication, and public key systems.* Stanford university, 1979.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of lattice problems: a cryptographic perspective*. Vol. 671. Springer Science & Business Media, 2002.

[MMR23] David Mestel, Johannes Müller, and Pascal Reisert. "How efficient are replay attacks against vote privacy? A formal quantitative analysis". In: *J. Comput. Secur.* 31 (2023), pp. 421–467.

[MN06] Tal Moran and Moni Naor. "Receipt-Free Universally-Verifiable Voting with Everlasting Privacy". In: *Advances in Cryptology - CRYPTO*. Ed. by Cynthia Dwork. 2006.

[MN07] Tal Moran and Moni Naor. "Split-ballot voting: everlasting privacy with distributed trust". In: *ACM CCS 2007*. 2007, pp. 246–255.

[Mos+24] Rafieh Mosaheb, Peter B. Rønne, Peter Y. A. Ryan, and Sara Sarfaraz. "Direct and Transparent Voter Verification with Everlasting Receipt-Freeness". In: *E-Vote-ID*. Vol. 15014. 2024, pp. 124–140.

[MPP24] Johannes Müller, Balázs Pejó, and Ivan Pryvalov. "DeVoS: Deniable Yet Verifiable Vote Updating". In: *Proc. Priv. Enhancing Technol.* 2024 (2024), pp. 357–378.

[MU10] Jörn Müller-Quade and Dominique Unruh. "Long-Term Security and Universal Composability". In: *J. Cryptol.* (2010), pp. 594–671.

[Mül22] Johannes Müller. "Breaking and Fixing Vote Privacy of the Estonian E-Voting Protocol IVXV". In: *Workshop on Advances in Secure Electronic Voting*. 2022.

[Nef04] C Andrew Neff. "Practical high certainty intent verification for encrypted votes". In: *Citeseer*. 2004.

[Pan20] Lorenz Panny. "Guess what?! On the impossibility of unconditionally secure public-key encryption". In: *Mathematical Cryptology* 1 (2020), pp. 1–7.

[Ped91] Torben P. Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing". In: *CRYPTO 1991*. 1991, pp. 129–140.

[Pet05] RA Peters. "A secure bulletin board". MA thesis. Technische Universiteit Eindhoven, 2005.

[Pin+17] Rafaël del Pino, Vadim Lyubashevsky, Gregory Neven, and Gregor Seiler. "Practical Quantum-Safe Voting from Lattices". In: *SIGSAC Conference on Computer and Communications Security, CCS*. 2017, pp. 1565–1581.

[PKG24] Sieglinde M.-L. Pfaendler, Konstantin Kosnon, and Franziska Greinert. "Advancements in Quantum Computing - Viewpoint: Building Adoption and Competency in Industry". In: *Datenbank-Spektrum* 24.1 (2024), pp. 5–20.

[Poi24] David Pointcheval. "Efficient universally-verifiable electronic voting with everlasting privacy". In: *International Conference on Security and Cryptography for Networks*. 2024, pp. 323–344.

[PR19] Olivier Pereira and Peter B. Rønne. "End-to-End Verifiable Quadratic Voting with Everlasting Privacy". In: *FC International Workshops*. 2019, pp. 314–329.

[Pra+20] Miguel Ángel Prada-Delgado, Iluminada Baturone, Gero Dittmann, Jens Jelitto, and Andreas Kind. "PUF-derived IoT identities in a zero-knowledge protocol for blockchain". In: *Internet Things* 9 (2020).

[Pre+20] T. Prest, P.-A. Fouque, J. Høstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang. *Falcon: Fast-Fourier Lattice-Based Compact Signatures Over NTRU*. 2020. URL: https://falcon-sign.info/falcon.pdf.

[Que+20] Iñigo Querejeta-Azurmendi, David Arroyo Guardeño, Jorge L Hernández-Ardieta, and Luis Hernández Encinas. "NetVote: A Strict-Coercion Resistance Re-Voting Based Internet Voting Scheme with Linear Filtering". In: *Mathematics* (2020), p. 1618.

[Reg05] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*. 2005, pp. 84–93.

[Reg09] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *Journal of the ACM (JACM)* 56.6 (2009), pp. 1–40.

[Riv90] Ronald L. Rivest. "Cryptography". In: *Handbook of Theoretical Computer Science*. Vol. 1. 1990. Chap. 13, pp. 717–755.

[Riv92] Ronald Rivest. *The MD5 message-digest algorithm*. Tech. rep. 1992.

[Rob23] Damien Robert. "Breaking SIDH in Polynomial Time". In: *Advances in Cryptology - EUROCRYPT*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. Lecture Notes in Computer Science. Springer, 2023, pp. 472–503.

[Roe16] Peter Roenne. "JCJ with improved verifiability guarantees". In: *E-Vote-ID*. 2016.

[Rot06] Ron Roth. "Codes in the Lee Metric". In: *Introduction to Coding Theory*. Cambridge University Press, 2006, pp. 298–332.

[RRI16] Peter Y. A. Ryan, Peter B. Rønne, and Vincenzo Iovino. "Selene: Voting with Transparent Verifiability and Coercion-Mitigation". In: *Financial Cryptography and Data Security - FC*. Vol. 9604. 2016, pp. 176–192.

[RRS21] Peter B Rønne, Peter YA Ryan, and Ben Smyth. "Cast-as-intended: A formal definition and case studies". In: *FC 2021 International Workshops*. 2021, pp. 251–262.

[RS06a] Alexander Rostovtsev and Anton Stolbunov. "Public-key cryptosystem based on isogenies". In: *Cryptology ePrint Archive* (2006).

[RS06b] Peter Y. A. Ryan and Steve A. Schneider. "Prêt à Voter with Re-encryption Mixes". In: *ESORICS*. 2006, pp. 313–326.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. "How to Leak a Secret". In: *ASIACRYPT 2001*. 2001, pp. 552–565.

[RT09] Peter Y. A. Ryan and Vanessa Teague. “Pretty Good Democracy”. In: *Security Protocols XVII, 17th International Workshop*. Vol. 7028. 2009, pp. 111–130.

[Rya05] Peter Y. A. Ryan. “A variant of the Chaum voter-verifiable scheme”. In: *Proceedings of the 2005 Workshop on Issues in the Theory of Security*. 2005, pp. 81–88.

[Sch+21] P. Schwabe, R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J.M. Schanck, G. Seiler, and D. Stehle. *CRYSTALS-Kyber*. 2021. URL: https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf.

[Sch87] Claus-Peter Schnorr. “A hierarchy of polynomial time lattice basis reduction algorithms”. In: *Theoretical computer science* 53.2-3 (1987), pp. 201–224.

[Sch91] Claus-Peter Schnorr. “Efficient signature generation by smart cards”. In: *Journal of cryptology* 4 (1991), pp. 161–174.

[SGS23] Maryam Sheikhi, Rosario Giustolisi, and Carsten Schürmann. “Receipt-Free Electronic Voting from zk-SNARK”. In: *SECRYPT*. 2023, pp. 254–266.

[Sha49] Claude E Shannon. “Communication theory of secrecy systems”. In: *The Bell system technical journal* 28.4 (1949), pp. 656–715.

[Sha79] Adi Shamir. “How to share a secret”. In: *Communications of the ACM* 22 (1979), pp. 612–613.

[Sho94] Peter W Shor. “Algorithms for quantum computation: discrete logarithms and factoring”. In: *Proceedings 35th annual symposium on foundations of computer science*. Ieee. 1994, pp. 124–134.

[Sil22] Tjerand Silde. “Short Paper: Verifiable Decryption for BGV”. In: *Financial Cryptography and Data Security*. Vol. 13412. 2022, pp. 381–390.

[SK95] Kazue Sako and Joe Kilian. “Receipt-Free Mix-Type Voting Scheme - A Practical Solution to the Implementation of a Voting Booth”. In: *EUROCRYPT*. Vol. 921. 1995, pp. 393–403.

[Smi07] Warren D. Smith. “Three Voting Protocols: ThreeBallot, VAV, and Twin”. In: *USENIX/ACCURATE Electronic Voting Technology Workshop*. 2007.

[SN93] National Institute of Standards and Technology (NIST). *Secure Hash Standard (SHS)*. Federal Information Processing Standards Publication FIPS PUB 180. SHA-0. U.S. Department of Commerce, 1993-05. URL: https://csrc.nist.gov/publications/fips/fips180/fips180.pdf.

[SN95] National Institute of Standards and Technology (NIST). *Secure Hash Standard (SHS)*. Federal Information Processing Standards Publication FIPS PUB 180-1. SHA-1. U.S. Department of Commerce, 1995-04. URL: https://csrc.nist.gov/publications/fips/fips180-1/fips180-1.pdf.

[SS24] Siyon Singh and Eric Sakk. “Implementation and Analysis of Shor’s Algorithm to Break RSA Cryptosystem Security”. In: *Authorea Preprints* (2024).

[SSW20] Peter Schwabe, Douglas Stebila, and Thom Wiggers. “Post-Quantum TLS Without Handshake Signatures”. In: *ACM CCS*. 2020, pp. 1461–1480.

[Ste93] Jacques Stern. "A New Identification Scheme Based on Syndrome Decoding". In: *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference Proceedings*. Ed. by Douglas R. Stinson. 1993, pp. 13–21.

[TW10] Björn Terelius and Douglas Wikström. "Proofs of Restricted Shuffles". In: *Progress in Cryptology - AFRICACRYPT*. Springer, 2010, pp. 100–113.

[Wan+24] Chao Wang, Qi-Di Wang, Chun-Lei Hong, Qiao-Yun HU, and Zhi Pei. *Quantum Annealing Public Key Cryptographic Attack Algorithm Based on D-Wave Advantage*. `https://cjc.ict.ac.cn/online/onlinepaper/wc-202458160402.pdf`. 2024.

[Wik11] Douglas Wikström. "How to implement a stand-alone verifier for the Verificatum mix-net". In: *Unpublished draft, December* (2011).

[YC17] Nan Yang and Jeremy Clark. "Practical Governmental Voting with Unconditional Integrity and Privacy". In: *FC 2017 International Workshops*. 2017, pp. 434–449.

[Zal99] Christof Zalka. "Grover's quantum searching algorithm is optimal". In: *Physical Review A* 60.4 (1999), p. 2746.

[ZGP17] Alexandros Zacharakis, Panagiotis Grontas, and Aris Pagourtzis. "Conditional Blind Signatures". In: *IACR Cryptol. ePrint Arch.* (2017), p. 682. URL: `http://eprint.iacr.org/2017/682`.

Appendix

In this Appendix, we provide a brief overview of the Hyperion voting system, which serves as the foundation for our e-voting system, that ensures everlasting privacy and everlasting receipt-freeness, introduced in Chapter 4. For a more in-depth technical analysis and details of Hyperion, we refer the reader to the full version of the paper [Dam+24].

Hyperion

Hyperion is an enhanced version of Selene [RRI16], a tracker-based voting system designed to streamline individual verification by presenting verification data in plaintext. In contrast to Selene, Hyperion uses a commitment instead of a plain tracker to identify the voters' votes. This means that low-entropy trackers are replaced by cryptographic terms, which we refer to as "tracking terms" instead of trackers to emphasize this distinction. This is the key feature that results in Hyperion adapting better to our goal of achieving everlasting receipt-freeness, which we elaborate on in Chapter 4.

As noted in [RRI16], a limitation of Selene is that in the coercion evasion mechanism, the voter might inadvertently point to the coercer's tracker, in case the coercer is also a voter himself. A version of Hyperion proposed in [Dam+24] overcomes the tracker collision incident by providing individual view of the BB to each voter. This individual view is the result of re-randomization and shuffle of the rows of BB per voter, in a verifiable manner. As the authors noted in [Dam+24], the trade-off for this advantage is its inefficiency, which makes it unsuitable for large-scale elections.

Both Selene and Hyperion have been formally verified in ProVerif in [Bal23]. The authors consider seven corruption scenarios for the four parties running the election (excluding the case that all four parties are corrupted at the same time). Their conclusion for Hyperion is that it satisfies E2E verifiability for honest voters who verify their ballot, meaning that it provides the same level of verifiability as in Selene, but is computationally more efficient and can overcome tracker collision problem.

In this section, we will first provide an overview of the parties involved in Hyperion, followed by a description of the cryptographic primitives utilized, and finally, an exploration of the phases of the basic Hyperion protocol.

Parties.

The protocol runs between the following parties.

- Voters (V): n voters $\mathsf{V}_1, \mathsf{V}_2, \ldots, \mathsf{V}_n$ with unique identities $\mathsf{ID}_1, \mathsf{ID}_2, \ldots, \mathsf{ID}_n$ holding two pairs of keys: signing key (sk_i, vk_i) and ephemeral trapdoor key (x_i, pk_i).
- Bulletin Board (BB): an append-only secure broadcast channel. In this basic version, all participants have a consistent view of BB.

- Election Authority (EA): generates the election parameters and sets up the BB.
- Adm or Registration Authority (Adm): responsible for checking the eligibility of the voters and publishing their public keys on BB.
- Talliers (T): k talliers $\mathsf{T}_1, \mathsf{T}_2, \ldots, \mathsf{T}_k$ who threshold share the secret key and cooperate to decrypt the ciphertexts.
- Mix-servers (M): N mix servers $\mathsf{M}_1, \mathsf{M}_2, \ldots, \mathsf{M}_N$ responsible for verifiable mixing of the votes in the tally phase.

Cryptographic primitives.

There main cryptographic primitives are being used in the Hyperion e-voting scheme, which we explain bellow. We refer readers for other cryptographic primitives to Section 2.3.

- ZKPoK and ZKP (Π): We refer the reader to Section 2.3.1. For simplicity, the authors of [Dam+24] removed the binding information to the voters' ID and election identifier from the hash function.
- Encryption (Enc): ElGamal encryption scheme, as explained in Section 2.3.2, is used.
- Shuffle: A verifiable mixing approach is used to shuffle the encryption of voters' votes and encryption of re-randomized public-keys of the voters in parallel. This is an exponentiation mix, which also permutes the random exponents of the public-keys. This is instantiated in Hyperion using Verificatum [Wik11].

Election Phases.

We outline all the four phases of Hyperion, highlighting the roles of the aforementioned parties and the previously discussed cryptographic primitives.

Setup Phase

The EA, Ts, and Vs perform the following tasks.

1. EA: Generate the election parameters prm including the cryptographic group setting on BB.
2. T_i: Generate a threshold key pair $(pk_{\mathsf{EA}}, sk_{\mathsf{EA}})$ and send pk_{EA} to BB.
3. V_i: Each voter has a signing key pair (vk_i, sk_i) and a unique ID_i where vk_i and ID_i are public on BB.

Ballot Submission Phase

Each voter V_i has to perform the following tasks.

1. Generate trapdoor key $x_i \in \mathbb{Z}_q$ and compute the public trapdoor key $pk_i := g^{x_i}$ using her device.
2. Choose her vote v_i and compute encryption of v_i as $e_i = \mathsf{Enc}_{pk_{\mathsf{EA}}}(v_i)$.

3. Generate $\mathsf{NIZKPoK}(x_i)$ and ZKP of well-formedness of e_i and concatenate them as a single proof Π_i.

4. Generate the signature $\sigma_i := \mathsf{Sign}_{sk_i}(e_i, pk_i, \Pi_i)$.

5. Send $\mathsf{ID}_i, e_i, pk_i, \Pi_i$, and σ_i to BB.

Tally Phase

In this phase, each party including the Adm, Ts, M, and EA perform the following tasks.

1. Adm: Check validity of σ_i and Π_i for each voter V_i.

2. Talliers T_j perform the following tasks together.

   (a) Compute a fresh secret $r_i \in_R \mathbb{Z}_q$.
   (b) Read pk_i from BB and compute $e_i^T := \mathsf{Enc}_{pk_{EA}}(pk_i{}^{r_i})$.
   (c) Compute $\Pi_i^T := \mathsf{ZKPoK}(r_i)$
   (d) Send e_i^T, Π_i^T to BB, in the row that belongs to ID_i.

3. The mix-servers M_j perform the following tasks.

   (a) Shuffle the encryptions $\{e_i, e_i^T\}$ in parallel and send the result as $\{e_i', e_i'^T\}$ to BB.
   (b) Compute ZKP of correct mixing as Π^{M} and send it to BB.

4. The election authority EA performs the following tasks.

   (a) Decrypt $\{e_i', e_i'^T\}$ as $\{v_i', pk_i'^{r_i}\}$.
   (b) Compute ZKP of correct decryption as Π^{EA} and send it to BB.

Notification and Verification Phase

In this phase, the talliers send verification information to the voters within the notification period. They perform the following tasks.

1. Ts: Choose a random time within the notification period and send $\alpha_i := g^{r_i}$, referred to as **alpha term** or **Hyperion term**, to V_i via a private channel.

2. V_i: Compute $(\alpha_i)^{x_i} = pk_i^{r_i}$ and find the matching pair $(v_i, pk_i^{r_i})$ on BB.

Coercion Mitigation.

In the event of coercion, if the coercer compels the voter V_i to cast a vote for candidate v_k, V_i can find a row on BB containing the pair $(v_k, pk_k^{r_k})$. She can then utilize her trapdoor key x_i to compute a fake verification term, defined as $\bar{\alpha}_i := (pk_k^{r_k})^{-x_i}$. She then provides $\bar{\alpha}_i$ as her receipt to the coercer.