UNIVERSITÉ DU LUXEMBOURG

PhD-FSTM-2023-057
The Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 17/08/2023 in Esch-sur-Alzette

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Ehsan ESTAJI
Born on 11 May 1984 in SABZEVAR, IRAN

INDIVIDUAL VERIFIABILITY FOR E-VOTING, FROM FORMAL VERIFICATION TO MACHINE LEARNING

Dissertation defence committee

Dr Peter Y.A. Ryan, dissertation supervisor
Professor, Université du Luxembourg

Dr Peter B. Rønne
Researcher, Loria Nancy

Dr MARCUS VÖLP, Chairman
Professor, Université du Luxembourg

Dr Jan WILLEMSON
Senior Researcher, Cybernetica Estonia,

Dr STEVE SCHNEIDER, Vice Chairman
Professor, University of Surrey, UK.

UNIVERSITY OF LUXEMBOURG

DOCTORAL THESIS

Individual Verifiability for E-Voting, From Formal Verification To Machine Learning

Author:
Ehsan ESTAJI

Supervisor:
Prof. Peter Y.A. RYAN
Daily Supervisor:
Dr. Peter Roenne

The author was supported by the FNR-SURCVS project

ABSTRACT

The cornerstone of secure electronic voting protocols lies in the principle of individual verifiability. This thesis delves into the intricate task of harmonizing this principle with two other crucial aspects: ballot privacy and coercion resistance.

In the realm of electronic voting, individual verifiability serves as a critical safeguard. It empowers each voter with the ability to confirm that their vote has been accurately recorded and counted in the final tally. This thesis explores the intricate balance between this pivotal aspect of electronic voting and the equally important facets of ballot privacy and coercion-resistance.

Ballot privacy, or the assurance that a voter's choice remains confidential, is a fundamental right in democratic processes. It ensures that voters can express their political preferences without fear of retribution or discrimination. On the other hand, coercion-resistance refers to the system's resilience against attempts to influence or manipulate a voter's choice.

Furthermore, this thesis also ventures into an empirical analysis of the effectiveness of individual voter checks in ensuring a correct election outcome. It considers a scenario where an adversary possesses additional knowledge about the individual voters and can strategically decide which voters to target. The study aims to estimate the degree to which these checks can still guarantee the accuracy of the election results under such circumstances.

In essence, this thesis embarks on a comprehensive exploration of the dynamics between individual verifiability, ballot privacy, and coercion-resistance in secure electronic voting protocols. It also seeks to quantify the effectiveness of individual voter checks in maintaining the integrity of election outcomes, particularly when faced with a knowledgeable adversary.

The first contribution of this thesis is revisiting the seminal coercion-resistant e-voting protocol by Juels, Catalano, and Jakobsson (JCJ) [78], examining its usability and practicality. It discusses the credential handling system proposed by Neumann et al. [92], which uses a smart card to unlock or fake credentials via a PIN code.

The thesis identifies several security concerns with the JCJ protocol, including an attack on coercion-resistance due to information leakage from the removal of duplicate ballots. It also addresses the issues of PIN errors and the single point of failure associated with the smart card.

To mitigate these vulnerabilities, we propose hardware-flexible protocols that allow credentials to be stored by ordinary means while still being PIN-based and providing PIN error resilience. One of these protocols features a linear tally complexity, ensuring efficiency and scalability for large-scale electronic voting systems.

The second contribution of this thesis pertains to the exploration and validation of the ballot privacy definition proposed by Cortier et. al. [41], particularly in the context of an adversarial presence.

Our exploration involves both the Selene and the MiniVoting abstract scheme [21]. We apply Cortier's definition of ballot privacy to this scheme, investigating how it holds up under this framework. To ensure the validity of our findings, we employ the use of tools for machine-checked proof. This method provides a rigorous and reliable means of verifying our results, ensuring that our conclusions are both accurate and trustworthy.

The final contribution of this thesis is a detailed examination and analysis of the Estonian election results. This analysis is conducted in several phases, each contributing to a comprehensive understanding of the election process.

The first phase involves a comprehensive marginal analysis of the Estonian election results. We compute upper bounds for several margins, providing a detailed statistical overview of the election outcome. This analysis allows us to identify key trends and patterns in the voting data, laying the groundwork for the subsequent phase of our research.

We then train multiple binary classifiers to predict whether a voter is likely to verify their vote. This predictive modeling enables an adversary to gain insights into voter behavior and the factors that may influence their decision to verify their vote. With the insights gained from the previous phases, an adversarial classification algorithm for verifying voters are trained. The likelihood of such an adversary is calculated using various machine learning models, providing a more robust assessment of potential threats to the election process.

Acknowledgements

I pursued my dreams and worked towards my second Ph.D. in information security. Writing a thesis was a challenging but rewarding experience for me, as it allowed me to delve deep into a topic that I was passionate about and contribute new knowledge to my field.

I would like to express my sincere and heartfelt gratitude to all those who have supported and assisted me throughout the course of my studies and the writing of this thesis. Without their help and encouragement, it would not have been possible for me to complete this important milestone in my academic journey.

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Peter Y. A. Ryan, my daily supervisor, Dr. Peter Roenne, and my committee members Prof. Steve Schneider, Prof. Jan Willemson, and Prof. Marcus Volp. Their expertise and insight have been essential in helping me to develop and refine my research, and I am deeply grateful for their contributions to my success. I would also like to extend my thanks to any additional advisors or committee members who may have played a role in this process, as their contributions have also been invaluable. I greatly appreciate their efforts in helping me to bring my research to fruition.

I would like to extend my sincere gratitude to the Fonds National de la Recherche for their generous financial support, which has made this research project possible. Without their assistance, it would not have been possible for me to undertake this important work. I am deeply grateful for their contribution and appreciate their commitment to advancing knowledge and understanding in my field of research.

I would also like to extend my big thanks to my classmates, cohort members, librarians, research assistants, laboratory assistants, and study participants for contributing to this project. Their support and assistance have been invaluable in helping me to complete my research and write this thesis.

I am filled with immense gratitude and appreciation for my parents, my wife, my daughter, and my friends who have provided me with an overwhelming amount of love and support throughout my studies. Their encouragement and understanding have been absolutely crucial in helping me to pursue my academic goals, and I am so grateful for their presence in my life. Their love and support have sustained me through the challenges and uncertainties of this journey, and I am forever thankful for their unwavering belief in me. I am blessed to have such wonderful people in my life, and I am grateful for the opportunity to make them proud. Thank you from the bottom of my heart.

Finally, I would like to thank all of these individuals for their contributions to my success and the completion of this thesis. I am deeply grateful to each and every one of them.

To my wife, **Nikta**, who has been my rock and a constant source of love and support throughout this journey. Your unwavering belief in me and your endless patience and encouragement have been invaluable. Thank you for being my partner in every sense of the word.

To my wonderful daughter, **Katie**, you make my life so much fun! You bring me more happiness than a clown juggling flaming bowling pins. Your endless curiosity makes me incredibly happy, and your innocence is as refreshing as an ice cream truck on a hot summer day. Thank you for being my adorable little tornado. I dedicate this thesis to both of you with all my love, some humor, and a lot of gratitude.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

Part I

Introduction and Background Knowledge

CHAPTER 1

INTRODUCTION

In this introductory chapter, we will provide a brief overview of the research topic and its significance, as well as a summary of the main research questions and objectives of the thesis. We will also provide a brief overview of the methods and data used in the thesis, as well as the structure of the rest of the thesis.

1.1 Motivation

The growing prevalence of electronic voting systems, or e-voting, heralds a new era of convenience where individuals can cast their votes remotely or via personal devices. Yet, this modern voting method also unveils a host of new concerns linked to the security and integrity of the voting process. To uphold public faith in the democratic process, it is critical to guarantee the precision and impartiality of e-voting systems.

A salient concern in the realm of e-voting is the necessity for individual verifiability, which provides each voter with the capacity to confirm the accurate recording and counting of their vote. This aspect becomes particularly crucial when voters are casting their votes remotely or through non-physical possession of devices, thus enabling them to validate that their vote has been accurately registered and remains free from interference.

In this thesis, formal verification is leveraged to substantiate that a minivoting abstract voting scheme adheres to a certain ballot privacy criterion. In addition, machine learning methodologies are deployed to scrutinize an attacker's ability to manipulate election results utilizing real-world data.

1.2 Aim

The aim of this thesis is to investigate ways to improve individual verifiability while preserving privacy or even coercion-resistance voting systems. To achieve this goal, we will pursue three main research paths.

1.2.1 Enhancing voter experience

We tackle the problem that has been previously acknowledged but not adequately resolved by Neumann et al [92]. This issue relates to the frequent occurrence of PIN typos, which can inadvertently invalidate a cast vote without the voter being aware of it. To address this, we devise innovative protocols that incorporate mechanisms to detect and rectify PIN errors, thereby safeguarding the voter's intention and ensuring their vote is accurately recorded.

1.2.2 Providing machine-checked proof for privacy

We aim to strengthen the assurance of privacy in the MiniVoting abstract scheme by exploring and validating the ballot privacy definition proposed by Veronique Cortier [41]. We specifically focus on evaluating the resilience of the scheme against adversarial presence, ensuring that the privacy guarantees hold up even in the face of potential attacks.

1.2.3 Using data analysis and machine learning

In our third research path, we embark on a journey to enhance the security and transparency of voting systems by harnessing the power of data analysis and machine learning techniques. Our primary objective is to develop a robust machine learning classifier capable of predicting whether a voter is likely to verify their vote or not. By pursuing this research direction, we aim to make significant contributions toward the advancement of secure and transparent voting systems that prioritize individual verifiability.

1.3 Thesis structure

The structure of this thesis is as follows:

Chapter 1: Introduction In this introductory chapter, we will provide a brief overview of the research topic and its significance, as well as a summary of the main research questions and objectives of the thesis. We will also provide a brief overview of the methods and data used in the thesis.

Chapter 2: Introduction to Cryptographic E-Voting This chapter consists of a comprehensive literature review aimed at examining prior research on the subject matter. The chapter provides a foundation for the subsequent research by highlighting areas requiring further investigation and structuring the research questions to be addressed in the rest of the thesis. The literature review ensures that the thesis builds upon existing knowledge and contributes to filling the identified gaps, leading to a well-informed and relevant research approach.

Chapter 3: Introduction to Formal Verification via the Hoare Logic This chapter provides an overview of the formal verification backgrounds, including Hoare Logic and Formal Verification. It introduces a small programming language, covering assignments, sequences, conditionals, and WHILE-commands, and summarizes their syntax. The chapter also discusses Hoare's notation and the axioms and rules of Hoare Logic, such as the assignment axiom, precondition strengthening, and postcondition weakening. This chapter establishes a foundation for the rest of the thesis, equipping readers with the necessary knowledge and tools to engage with the formal verification techniques discussed in subsequent chapters.

Chapter 4: Introduction to Machine-Learning This chapter introduces the essential materials and methodologies that lay the groundwork for our exploration of specific machine learning paradigms. While the subsequent sections provide a thorough overview of the tools, frameworks, and principles employed in this thesis for precise modeling, analysis, and implementation of complex machine learning systems,

readers interested in more detailed definitions and discussions about machine learning concepts are referred to [57].

Chapter 5: Practical and Usable Coercion-Resistant Remote E-Voting In this chapter, we examine the usability and security of the coercion-resistant e-voting protocol developed by Juels, Catalano, and Jakobsson (JCJ) [78]. Previous work by Neumann et al. [92] attempted to make this protocol more practical by using a smart card with a PIN code to handle cryptographic credentials, but we identify several vulnerabilities and problems with this approach, including an attack on coercion-resistance due to information leakage and the risk of invalidated votes due to PIN typos. We propose alternative protocols that address these issues and are hardware-flexible, allowing credentials to be stored using ordinary means, while still providing PIN-based protection and error resilience.

Chapter 6: Machine-Checked Proofs of Privacy Against Malicious Boards In this chapter, we use MiniVoting [21], a specific electronic voting scheme, as a case study to present a machine-checked proof of privacy. Our proof shows that MiniVoting meets a specific privacy definition, which is an essential step in improving the security and transparency of electronic voting systems. These systems play a crucial role in ensuring the integrity of democratic processes. Additionally, our work extends beyond MiniVoting to include the Selene electronic voting scheme. We offer a machine-checked proof of privacy for Selene, which is a remote electronic voting scheme known for its appealing combination of security properties and usability. Our formal verification of Selene's computational privacy is a novel contribution, as no prior work had formally verified it.

Chapter 7: Analysis of Estonian Election Margin In this chapter, we conducted a study of the Estonian election results, examining several measures of competitiveness for two elections in 2015 and 2019. Specifically, we calculated various metrics to assess the competitiveness of the elections and analyzed the results to understand trends and patterns. Our analysis provides valuable insights into the electoral landscape in Estonia and can inform future research and policy decisions.

Chapter 8: Identifying Voter Characteristics through ML-Assisted Models In this chapter, we will explore the use of ML-assisted models to identify voter characteristics and their potential impact on voting behavior. Through a combination of empirical analysis and theoretical modeling, we aim to gain a deeper understanding of this important topic and contribute to the development of more effective and transparent voting systems.

1.4 Contributions

This thesis is a culmination of original research papers, some of which are already published, submitted, or in preparation for submission. In this section, I outline my specific contributions to each of these papers, as well as the roles of my co-authors. The list of contributions is organized according to the following structure:

Contribution I: Revisiting Practical and Usable Coercion-Resistant Remote E-Voting

- **Co-authors:** Thomas Haines , Kristian Gjøsteen, Peter B. Rønne , Peter Y. A. Ryan, Najmeh Soroush
- **Publication status:** Published in Electronic Voting. E-Vote-ID 2020, Springer, Cham, 25 September Journal/Conference name, DOI: 10.1007/978-3-030-60347-2_4 (Used in the thesis)

- **Contribution:** In this paper, my co-authors and I critically analyzed the JCJ coercion-resistant e-voting protocol and identified several attacks and security problems with existing implementations. We further highlighted the practical issues associated with handling cryptographic credentials and presented several alternative protocols to repair these problems. My specific contributions include the development of hardware-flexible solutions that allow for storing credentials by ordinary means, ensuring PIN error resilience, and creating one protocol with linear tally complexity to ensure efficiency with a large number of voters. We aimed to make e-voting more practical, secure, and user-friendly while preserving essential properties such as coercion resistance.

Contribution II: Machine-Checked Proofs of Privacy Against Malicious Boards for Selene & Co

- **Co-authors:** Constantin Cătălin Drăgan, François Dupressoir, Kristian Gjøsteen, Thomas Haines, Peter Y. A. Ryan, Peter B. Rønne, Morten Rotvold Solberg
- **Publication status:** Published in 2022 IEEE 35th Computer Security Foundations Symposium (CSF), DOI: 10.1109/CSF54842.2022.00021 (Used in the thesis)
- **Contribution:** In this work, we tackle the complex challenge of privacy in electronic voting schemes. Recognizing the intricate nature of ensuring privacy in such systems, we define a new ballot privacy notion that is applicable to a broader class of voting schemes. This novel definition advances the field by considering the fact that verification in many voting systems occurs or must occur after the tally has been published, contrary to previous definitions. Through a comprehensive case study, we provided a machine-checked proof of privacy for Selene, a remote electronic voting scheme known for its balance of security and usability. Prior to our research, the computational privacy of Selene had not been formally verified. My specific contribution is that I proved that MiniVoting satisfies the original definition of ballot privacy for malicious ballot boxes.

Contribution III: "Just for the sake of transparency": Exploring Voter Mental Models Of Verifiability

- **Co-authors:** Marie-Laure Zollinger, Peter Y. A. Ryan, Karola Marky
- **Publication status:** Published in Sixth International Joint Conference, E-Vote-ID 2021, DOI: 10.1007/978-3-030-86942-7_11 (Not used in the thesis)
- **Contribution:** This paper delves into the human aspect of the Selene voting protocol, an electronic voting scheme that offers plaintext vote verification while maintaining privacy. Through interviews with 24 participants, we explored their mental models of Selene, categorizing them into four distinct levels: 1) understanding of the technology, 2) interpretation of the verification phase, 3) concerns related to security, and 4) perception of unnecessary steps within the process. Our analysis uncovers various misconceptions and provides unique insights into how voters perceive and interact with Internet voting technologies. The findings have led us to propose specific recommendations for the future implementation of Selene and for the design of Internet voting systems at large. Our research contributes to the field by bridging the gap between the technical complexities of verifiable voting schemes and the users' perceptions, aiding in the development of more intuitive and trustworthy electronic voting systems.

Contribution IV: Machine-Checked Proofs of Privacy Against Malicious Boards for Selene & Co (Journal Edition)

- **Co-authors:** Constantin Cătălin Drăgan, François Dupressoir, Kristian Gjøsteen, Thomas Haines, Peter Y. A. Ryan, Peter B. Rønne, Morten Rotvold Solberg
- **Publication status:** Accepted. (Used in the thesis)
- **Contribution:** This paper is an extended and revised version of a paper presented at CSF'22(Second Contribution).

Contribution V: Voter Verification and Adversarial Threats in Election with Machine Learning Models

- **Co-authors:** Peter B. Rønne, Jan Willemson, Peter. Y. A. Ryan
- **Publication status:** In preparation
- **Contribution:** In this research, we conducted a rigorous marginal analysis of the Riigikogu election in Estonia, a task vital to understanding the robustness of electoral systems. Recognizing the intricate nature of the candidate selection process, we segmented our analysis into various stages, allowing for a nuanced examination of the election margin, which signifies the number of votes needed to shift the outcome. Subsequently, we trained several binary classifiers to predict the likelihood of voter verification and constructed an adversarial model with the ability to modify the necessary quantity of votes to alter the election result. We evaluated the probability of such adversarial activities using a range of machine learning models. Our work aims to enhance the security and transparency of elections by analyzing margins, assessing vulnerabilities, and providing insights that can guide improvements in electoral processes and systems.

CHAPTER 2

INTRODUCTION TO CRYPTOGRAPHIC E-VOTING

In this chapter, we will recall the motivation and different security and cryptographic aspects of secure electronic voting. This will lay a necessary foundation for the rest of the thesis.

2.1 Cryptographic Primitives Backgrounds

Cryptography is the study of techniques for secure communication in the presence of adversarial parties. These techniques are based on mathematical algorithms and protocols that are designed to provide various levels of security, such as confidentiality, integrity, authentication, and non-repudiation.

Cryptography is an essential component of modern computer security systems and is used in a wide range of applications, such as secure communication, secure data storage, digital signatures, and access control. There are two main categories of cryptographic algorithms: symmetric and asymmetric.

Symmetric algorithms use the same key for both encryption and decryption. They are fast and efficient but require that both the sender and the recipient have a copy of the same key, which must be kept secret. Examples of symmetric algorithms include AES, DES, and Blowfish.

In the context of e-voting, it is important to note that we mostly use asymmetric crypto. Asymmetric algorithms, such as RSA, ECC, and ElGamal, employ different keys for encryption and decryption. The encryption key is made public, allowing anyone to encrypt data, while the decryption key is kept private. Although asymmetric algorithms are slower and more complex compared to symmetric algorithms, they eliminate the need for exchanging a secret key. Therefore, when it comes to e-voting, the utilization of asymmetric cryptography is prevalent.

Cryptography is a constantly evolving field, and new algorithms and protocols are being developed all the time. It is crucial to thoroughly evaluate the security of any cryptographic algorithm or protocol before implementing it in a real-world application. Furthermore, it is worth noting that there is a long history of security failures in cryptography. However, advancements have been made in terms of formal verification methods, such as the symbolic model. In this thesis, we will employ a combination of paper-based and

machine-checked proofs within the computational model to ensure the robustness and reliability of our cryptographic approach. For more details on this, please refer to section ?????

2.1.1 Interactive Zero Knowledge Proofs

Zero-knowledge proofs are a cryptographic concept that allows one party (the prover) to prove to another party (the verifier) that they know a certain piece of information, without revealing what that information is. This is achieved through the use of interactive protocols, in which the prover and verifier engage in a series of exchanges. The prover's goal is to convince the verifier that they have the necessary information, while the verifier's goal is to determine whether the prover is telling the truth.

The concept of zero-knowledge proofs was first introduced by Shafi Goldwasser, Silvio Micali, and Charles Rackoff in a paper published in 1985 [58]. The key idea is to use a series of interactions between the prover and the verifier to convince the verifier that the prover has the necessary information, without actually revealing that information. This is achieved through the use of special protocols and mathematical techniques that ensure that the information remains hidden even during interactions.

The concept of zero-knowledge proofs has been extensively studied and applied in a variety of settings, including cryptography, computer science, and mathematics. It has been used to design secure protocols for a wide range of applications, including digital signatures, password authentication, and secure multiparty computation.

Three properties are necessary for this kind of proof:

- **Completeness:** If the assertion is true, the prover can convince the verifier of its truth with high probability.
- **Soundness:** If the assertion is false, the verifier will not be convinced of its truth, even if the prover tries to cheat.
- **Zero-knowledge:** In the course of the interaction, the verifier learns nothing about the assertion except whether it is true or false. This means that even if the verifier is convinced of the truth of the assertion, they will not be able to extract any additional information about the claim.

In general, a protocol is said to have "zero knowledge" if, regardless of the result of the verifier following the protocol, it could have generated the same outcome without engaging with the prover. This means that the verifier does not learn any additional information about the claim being proved beyond what was already known before the protocol was executed.

Interactive zero-knowledge proofs are used at several stages in voting protocols. For example, authorities may use them to demonstrate their honesty to each other in order to proceed with the protocol.

2.1.2 Non-Interactive Zero-Knowledge Proofs

In the realm of cryptographic systems, particularly in the context of secure digital voting, the utilization of non-interactive zero-knowledge proofs (NIZKPs) is of paramount importance. NIZKPs are a subclass of zero-knowledge proofs that allow a prover, say P, to demonstrate to a verifier, say V, that they know a secret value x, without revealing any information about x other than the fact that they know it. Formally, given a relation R, a NIZKP system enables P to convince V that there exists a witness w such that $(x, w) \in R$, without any interaction between P and V after the proof is generated.

In the context of voting systems, these proofs are utilized to ensure the integrity and transparency of the voting process. Given a set of ciphertext votes committed to a bulletin board, any individual can verify that the tally is correctly determined by using the proof. Let $c_1, c_2, ..., c_n$ denote the set of ciphertext

votes and let T denote the tally. A NIZKP can be used to prove that T is a correct aggregation of the votes, i.e., there exists a set of plaintext votes $m_1, m_2, ..., m_n$ such that $Enc(m_i) = c_i$ for all i, and $T = \sum_{i=1}^{n} m_i$, without revealing any information about individual m_i's.

By guaranteeing both the correctness of the tally and the privacy of individual votes, NIZKPs enhance trust in the voting process, ensuring the reliability and accuracy of digital voting systems.

They may also be used between authorities and voters, such as when people want to be convinced of the validity of the tally (universal verifiability) or when a voter wants to be convinced of the correctness of their vote's encryption (individual verifiability).

2.1.3 Secret Sharing

Secret sharing [112, 110, 72] is a method in cryptography where a secret is divided into parts, giving each participant its own unique part. To reconstruct the original secret, a minimum number of parts is required. In the threshold scheme of secret sharing, this minimum number is less than the total number of parts. Otherwise, all participants are needed to reconstruct the original secret.

The concept of secret sharing was introduced by Adi Shamir in 1979 in a method now known as Shamir's Secret Sharing. It is a form of the threshold scheme and is based on polynomial interpolation.

Let's consider a secret S that we want to share among n participants in such a way that any k of them can reconstruct the secret, but $k-1$ or fewer of them have no information about S. The procedure in Shamir's Secret Sharing scheme is as follows:

1. Choose a prime number p such that $p > \max(n, S)$.
2. Choose $k-1$ random coefficients $a_1, a_2, \ldots, a_{k-1}$ from $\mathbb{Z}_p$.
3. Define a polynomial $f(x) = S + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1}$.
4. For each participant i ($1 \le i \le n$), compute $f(i)$ and give him the pair $(i, f(i))$.

Now, if any group of k participants pool their pairs, they can reconstruct the polynomial $f(x)$ using polynomial interpolation, and hence retrieve the secret $S = f(0)$.

The basis of the scheme is that any k points on a polynomial of degree $k-1$ uniquely define the polynomial. Therefore, the polynomial (and hence the secret) can be reconstructed given k points. However, $k-1$ points give absolutely no information about the polynomial, making the secret safe if fewer than k participants combine their shares.

Shamir's Secret Sharing is a highly sophisticated and potent scheme, offering an ideal balance between the confidentiality of the information and the capability to reassemble it when needed. However, a critical point of consideration is its lack of participant authentication measures; anyone possessing k shares can reconstruct the secret. This aspect necessitates the integration of supplementary security measures in real-world applications to verify participant identities.

One such commonly employed measure is threshold decryption, particularly in voting systems. In threshold decryption, a minimum number of participants (threshold) is required to decrypt a message. This adds an extra layer of security by ensuring that no single participant can access the information without the necessary threshold of consensual participants. Thus, while Shamir's Secret Sharing forms the backbone of data security, its effective application often necessitates additional layers of security such as threshold decryption to ensure participant authenticity and data integrity.

2.1.4 Public Key Encryption Scheme

In the conventional paradigm of secure communication, commonly referred to as symmetric key encryption, the involved parties, Alice and Bob, utilize a shared secret key for both the encryption and decryption processes. Consequently, they must exchange this secret key in advance and safeguard its confidentiality throughout their desired period of secure communication. In their influential paper [94], Diffie and Hellman introduced the concept of a public-key encryption scheme, which distinguishes itself from symmetric encryption schemes by employing two distinct keys for encryption and decryption operations.

Within a public-key encryption scheme, the encryption key is represented by the public key, enabling anyone possessing knowledge of this key to encrypt a message. On the other hand, the private key functions as the decryption key, granting exclusive access to the original message solely to the owner of the private key. Remarkably, an adversary attempting to decipher ciphertexts encrypted with the public key finds it futile to utilize the encryption key. Consequently, the public key can be published or broadcasted without concern for eavesdropping, facilitating secure communication without necessitating a private channel for key distribution [80].

Definition 1. *A public-key encryption scheme is a tuple of three probabilistic polynomial-time algorithms* $\Pi^{pke}(K_{gen}, Enc, Dec)$ *applied to three sets, the keyspace* K*, the message space* M*, and the ciphertext space* C*, such that:*

- $Kgen(1^{\ell}) \rightarrow (PK, SK)$*: The key generation algorithm takes the security parameter,* ℓ*, as an input and returns a pair of keys* (PK, SK) *from the keyspace* K*. We refer to the first component,* PK*, as a public key, which defines a message space,* M*, and the second* SK *as private (or secret key). It requires that both keys have a length polynomial in terms of the security parameter.*
- $Enc(PK, m) \rightarrow c$*: The encryption algorithm is a probabilistic polynomial-time algorithm that takes as input a message* $m \in M$ *and the public key and returns a ciphertext* $c \in C$*.*
- $Dec(SK, c) \rightarrow M \cup \{\perp\}$*: The decryption algorithm is a deterministic algorithm that takes the ciphertext* c *and the secret key* SK *as inputs and returns the message* m' *from the message space or* $\perp$ *as denoting failure.*

Security Requirements. A public-key encryption scheme must possess the following properties:

- **Correctness:** The output of the decryption algorithm should be the original message, except with negligible probability over the randomness of the key generation and encryption algorithms:
- **Security:** When considering security within a public-key encryption scheme, it is necessary to address the "security guarantee" and the "adversarial model," which define the adversary's capabilities. By doing so, precise definitions for different flavors of security notions can be established, such as adaptive versus non-adaptive attacks, and chosen-plaintext versus chosen-ciphertext attacks.

The basic notion of security for a Public Key Encryption (PKE) is semantic security, which states that no probabilistic polynomial-time (PPT) adversary, given two messages m_0 and m_1, and a ciphertext ct, can guess whether ct is the encryption of m_0 or m_1 better than a random guess.

Semantic security is the fundamental security concept in a public-key system, asserting that an adversary, who selects messages m_0 and m_1, and receives the ciphertext ct, cannot differentiate whether ct corresponds to the encryption of m_0 or m_1. Hence, taking into account the adversary's power, we define the following security notions [9]:

IND-CPA-Adaptive adversary

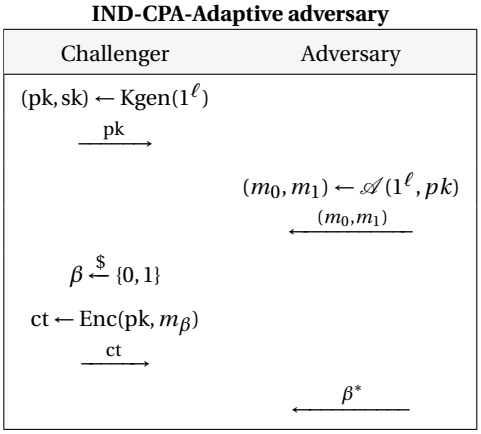


FIGURE 2.1: Experiments for CPA security

IND-CCA-Adaptive adversary

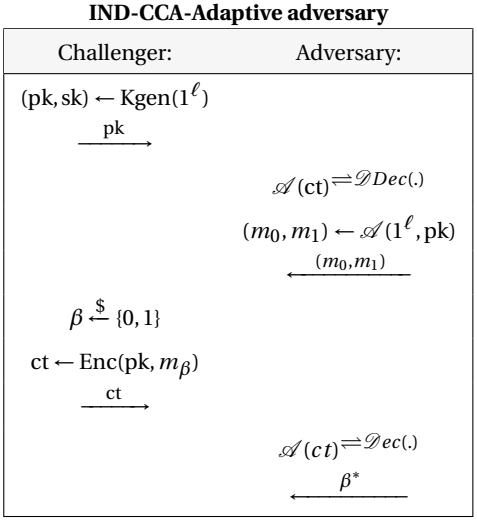


FIGURE 2.2: Experiments for CCA security

- **IND-CPA Security against an adaptive adversary:** An adaptive adversary selects messages m_0 and m_1 after receiving the public key of the encryption scheme. A public-key encryption scheme is indistinguishable against chosen plaintext attacks (IND-CPA for short) if the advantage of any probabilistic polynomial-time (PPT) adversary in $\mathrm{Exp}^{\text{cpa-nA}}_{\mathscr{A}}(1^\ell)$ (as shown in Figure 2.1) is negligible with respect to the security parameter.

- **IND-CCA Security against an adaptive adversary** ensures a high level of security in which the adversary has access to a decryption oracle. It means that the adversary can query some string ct to the decryption oracle and receive some message m such that $\mathrm{Dec}(ct) = m$ or $\perp$. An adaptive adversary can query after determining the message m_0 and m_1, whereas non-adaptive adversaries can only have access to the decryption oracle before selecting the messages. A public-key encryption scheme is indistinguishable against chosen plaintext attacks (IND-CPA for short) if the advantage of any PPT adversary in the experiment in Figure 2.2 would be negligible in terms of the security parameter.

The ElGamal encryption system, named after its inventor Taher ElGamal, is one such public key encryption scheme. It is based on the Diffie-Hellman key exchange protocol and is considered secure against passive adversaries if the computational Diffie-Hellman problem is unsolvable.

The ElGamal system involves three steps:

1. **Key Generation:** A large prime number p and a generator g of the multiplicative group of integers modulo p are chosen. The private key x is a randomly chosen integer from $\{1, ..., p-2\}$, and the public key y is computed as $y = g^x \mod p$. The public key (p, g, y) is then published.

2. **Encryption:** To encrypt a message m into a ciphertext c, an ephemeral key k is randomly chosen from $\{1, ..., p-2\}$. The ciphertext c is then a pair (c_1, c_2), where $c_1 = g^k \mod p$ and $c_2 = m * y^k \mod p$.

3. **Decryption:** To decrypt a ciphertext (c_1, c_2), the recipient uses their private key x to compute $s = c_1^x \mod p$, then computes $m = c_2 / s \mod p$.

The security of the ElGamal encryption system relies on the difficulty of the discrete logarithm problem. If an adversary can solve this problem efficiently, they could compute the private key from the public key and decrypt any message.

For a more detailed description of the ElGamal encryption system, please see [49].

2.1.5 Homomorphic Encryption

Homomorphic encryption is a cryptographic scheme that allows computations to be performed on encrypted data without decrypting it. In other words, it enables performing mathematical operations directly on encrypted data, producing an encrypted result that, when decrypted, corresponds to the desired result of the operations. This property makes homomorphic encryption a powerful tool for privacy-preserving computation.

Let's consider a simplified representation of a homomorphic encryption scheme using the Paillier cryptosystem which is additively homomorphic.. In this scheme [99] , the public key consists of a modulus N and a public encryption key g, while the private key consists of a secret factorization of N into its prime factors. Given a plaintext message m, the encryption process involves randomly selecting a non-zero integer r and computing the ciphertext c as follows:

$$c = g^m \cdot r^N \mod N^2$$

To perform computations on encrypted data, we can use the properties of the Paillier cryptosystem. One important property is the homomorphic addition property, which states that multiplying two ciphertexts together corresponds to adding their respective plaintexts modulo N. Mathematically, given ciphertexts c_1 and c_2 representing plaintexts m_1 and m_2, respectively, the multiplication of ciphertexts c_1 and c_2 yields a new ciphertext c_3 representing the sum of m_1 and m_2:

$$c_3 = c_1 \cdot c_2 \mod N^2$$

Similarly, we have the homomorphic scalar multiplication property, which allows multiplying a ciphertext c by a plaintext m without decrypting the ciphertext. This operation results in a new ciphertext c' representing the product of the plaintext m and the original plaintext represented by c:

$$c' = c^m \mod N^2$$

These properties enable performing computations on encrypted data without revealing the plaintexts, ensuring privacy while obtaining valid results. In summary, homomorphic encryption allows computations to be performed on encrypted data by leveraging mathematical properties of the encryption scheme. This capability enables secure and private data processing in scenarios where sensitive information needs to be kept confidential while still allowing useful computations to be carried out.

As another example, we can mention the Boneh-Goh-Nissim (BGN) cryptosystem which is a pairing-based cryptosystem that supports the evaluation of quadratic multivariate polynomials on encrypted data. The BGN cryptosystem is semantically secure under the Decisional Composite Residuosity Assumption (DCRA).

The BGN cryptosystem is defined over a bilinear group of composite order. Let $N = pq$ be a composite number where p and q are two large primes. Let G_1, G_2, and G_T be groups of order N, and let $e : G_1 \times G_1 \rightarrow G_T$ be a bilinear map. The BGN cryptosystem is defined as follows:

1. **Key Generation:** Choose a random generator $g \in G_1$ and a random element $h = g^s \in G_1$ for some random $s \in \mathbb{Z}_N^*$. The public key is (g, h) and the private key is s.
2. **Encryption:** To encrypt a message $m \in \mathbb{Z}_N$, choose a random $r \in \mathbb{Z}_N^*$ and compute the ciphertext $c = g^m h^r \in G_1$.
3. **Decryption:** To decrypt a ciphertext $c \in G_1$, compute $m = \log_g(c/h^s) \mod N$.

The BGN cryptosystem supports the homomorphic evaluation of quadratic polynomials. Given the encryptions of $m_1, m_2 \in \mathbb{Z}_N$, one can compute an encryption of $m_1 m_2 \mod N$ as $e(c_1, c_2) \in G_T$. For more details, you can refer to the original paper [25].

2.1.6 Re-Encryption

Re-encryption is the process of generating a new encrypted version of a message from an existing encrypted version, without the need to decrypt the original message. This property is a direct result of the homomorphic property, and it can be useful for electronic voting schemes that need to ensure the privacy of individual votes e.g. for mix-nets, see next subsection.

For notation let, $E_r(m)$ represent the encryption of a plain-text message m using a secret key r, and $D(c)$ to represent the decryption of an encrypted message c. The re-encryption algorithm is defined as $\mathrm{RE}_r(c) = c \bigotimes \mathrm{E}_r(1)$, where 1 is the plain-text identity element of the group of plain-texts. This means that, if c is an encrypted message and m is the corresponding plain-text message, then $\mathrm{RE}_r(c)$ is a new encrypted version of m that can be decrypted to obtain the same plain-text message m. Importantly, this new ciphertext is indistinguishable from fresh encryption of the same plaintext, a property that can be incredibly useful. This property can be useful for ensuring the privacy of individual votes in an electronic voting system because it allows a voter to prove that they have cast a vote without revealing the actual vote.

2.1.7 Mix-net

A mix-net is a type of network protocol that is often used in electronic voting systems to protect the anonymity of voters. This system was initially proposed by David Chaum [34] in 1981 to ensure the privacy of communications over public networks. Subsequently works including [1, 3, 26, 32, 60] have used Mix-net or its variant in their constructs. Mix-nets work by routing data, such as ballots, through a network of authorities, known as mix-servers, which shuffle the data before forwarding it to the next authority. This process, known as mixing, conceals the order of the data, which protects the anonymity of the voters who cast the ballots.

There are several different types of mix-nets, each with its own specific structure and features. Some mix-net protocols use zero-knowledge proofs, re-encryption, or secret sharing to protect the anonymity of the data being shuffled.

In most mix-net protocols, the setup and operation of the mix-net follow a similar pattern. First, the mix servers are chosen and their public keys are made available. Before the election, ballots are generated using the mix-servers' public keys. During the election, voters cast their ballots, which are encrypted and sent through the mix-net. As the ballots pass through the mix-servers, they are decrypted using the mix-servers' secret keys and re-encrypted in a different order. This shuffling process protects the anonymity of the voters. Finally, the ballots are tallied, normally by verifiable decryption, i.e. decryption with zero-knowledge proofs of correct decryption (see above section), and the results of the election are determined.

2.2 Introduction to E-Voting Backgrounds

We are exploring the history and evolution of voting methods and providing an overview of the components of electronic voting systems that will be discussed in more detail in the following chapters.

2.2.1 Contemporary Voting Systems

The paper-based election method is one of the most widely used voting mechanisms for consulting people on key choices. In this method, voters arrive at a polling station on election day and are authenticated before receiving a ballot. They then cast their vote in a transparent ballot box in front of a representative of the election authority and sign a register to confirm their vote. After the voting phase is complete, the authority publicly opens the ballot box, counts the ballots, and publishes the election results.

While this method has been effective in many cases, it is still vulnerable to scams and other forms of fraud. However, many people believe that a paper-based election system has certain qualities that make it trustworthy, such as the anonymity of the vote. Despite its limitations, this method continues to be used in many elections around the world.

The standard procedure for paper-based elections can be time-consuming, particularly the process of counting and tallying the ballots. This can delay the publication and implementation of election results. In order to speed up this process and maintain certain requirements such as voter privacy, electronic voting systems have been developed. These systems allow for a more rapid count of the ballots, with results often available within an hour of the end of voting. This can greatly reduce the time it takes to determine the winner of an election and make decisions based on the results.

Since the 1980s, electronic voting systems have been the subject of extensive research. As a result, a large number of protocols for electronic voting have been developed. However, as the number of protocols has increased, so have the security requirements that these protocols must meet. As a result, researchers continue to explore new and existing cryptographic techniques in order to develop secure electronic voting algorithms. This is an ongoing area of study, as the security of electronic voting systems is of critical importance in ensuring the integrity of the electoral process.

2.2.2 Fundamental Knowledge

We will provide an overview of the key characteristics that electronic voting systems should possess. There is some disagreement among researchers and writers about the specific requirements for these systems, but the majority agree on the need for secrecy and authenticity in electronic voting protocols. These characteristics are similar to those of paper-based elections, and are essential for ensuring the integrity of the electoral process. Some of the key features that electronic voting systems should strive to achieve include:

- **Correctness:** The system should be able to accurately determine the outcomes of an election and provide verifiable results.
- **Privacy:** The system should protect the privacy of voters by concealing their identities and ensuring that their choices cannot be traced back to them.
- **Receipt-freeness:** The system should not provide any evidence or confirmation of a voter's choices, in order to prevent vote-selling and coercion.
- **Robustness:** The system should be able to withstand various forms of attack or manipulation, including errant behavior.
- **Verifiability:** The system should be trustworthy and transparent, allowing interested parties to verify its operation and the accuracy of the results.
- **Democracy:** The system should ensure that only registered voters are able to cast their ballots, and that each individual is only able to vote once.
- **Fairness:** The system should not provide any information about the election to voters during the voting phase, in order to prevent them from being influenced by partial or incomplete information.

2.2.3 Security Properties

Developing a viable voting protocol is a challenging task, as there are many security requirements that must be met in order to ensure the reliability and integrity of the system. However, it is difficult to design a protocol that meets all of these requirements simultaneously, as some of them may be conflicting or inconsistent. For example, as discussed in [35], it is generally impossible to achieve both universal verifiability of the tally and unconditional privacy of the votes unless all registered voters participate in the election. The design of electronic voting systems must carefully balance the various security requirements in order to provide a reliable and trustworthy system. This can be a difficult task, but it is essential for ensuring the integrity of the electoral process and maintaining the trust of voters.

For example, it is demonstrated that it is impossible to achieve

- Universal verifiability of the tally means that any interested party can verify the accuracy of the election results, while unconditional privacy of the votes means that the identity of the voter is protected and cannot be traced back without any trust or computational. It is impossible to achieve both of these properties concurrently unless all registered voters participate in the election. This is because if some voters do not participate, their lack of participation could potentially reveal information about their vote, compromising the privacy of the voting process(See [35]).
- Universal verifiability of the tally means that any interested party can verify the accuracy of the election results, while receipt-freeness means that the system does not provide any confirmation or evidence of a voter's choices. It is difficult to achieve both of these properties unless the voting procedure includes interactions between multiple voters and possibly the voting authority. This is because receipt-freeness requires that the system does not provide any confirmation of a voter's choices, which makes it challenging to verify the accuracy of the tally without additional interactions.

To begin, we will provide a an informal definition for each security property. This will allow us to understand the various interpretations of these properties and how they are applied in different contexts.

2.2.3.1 Privacy

The concept of privacy in elections refers to the ability of voters to cast their votes without fear of retaliation or coercion. In this context, the principle of secrecy of the ballot is crucial for ensuring that voters can exercise their right to vote freely and without fear.

2.2.3.2 Receipt-freeness

Intuitively, an election protocol is considered receipt-free when a voter , referred to as A, is unable to provide conclusive evidence to a potential coercer or vote-buyer, denoted as C, regarding the specific manner in which she cast her vote. In this context, we assume that A is willing to collaborate E.g. getting money from C. Maybe also define that the interaction is after voting. The concept of receipt-freeness ensures that such collaboration would be futile because C would be unable to acquire any proof concerning A's voting choices [46].

Receipt-freeness bears resemblance to the notion of privacy, which asserts that an unauthorized individual cannot obtain information regarding A's voting behavior. However, receipt-freeness goes a step further by incorporating an additional assumption: A's willingness to cooperate with an intruder, denoted as C, by sharing her secret key and other confidential information generated throughout the election protocol. Consequently, receipt-freeness implies the guarantee of privacy, in that A's voting choices remain undisclosed even when she voluntarily engages with C during the voting process.

2.2.3.3 Coercion-Resistence

The interpretation of coercion-resistance in scholarly works is shaped by the complexities of assessing an attacker's abilities and the permissible degree of coercion. Historically, the concept of coercion-resistance was largely associated with receipt-freeness, as demonstrated by numerous definitions [15, 95, 76, 45]. Nevertheless, receipt-freeness does not cover all possible coercive attacks. As a result, it's crucial to embrace a broader understanding of coercion-resistance that includes all potential coercive actions that could influence an election's outcome. Given the lack of a universally accepted definition for coercion and coercion-resistance [64], we've opted to employ a comprehensive definition of coercion-resistance that covers a broad spectrum of attacks.

This viewpoint is inspired by Juels et al. [78], who suggested that a voting system is deemed coercion-resistant if it's virtually impossible for an attacker to verify whether a voter has complied with a coercer's demand. They further elaborated that a coercion-resistant voting system should effectively combat three coercion tactics:

- Compelling the voter to refrain from participating in elections.
- Pressuring the voter to submit an invalid vote.
- Manipulating the voter's credentials to cast a legitimate vote on their behalf.

In a remote environment where the adversary can persistently observe the voter from registration to the conclusion of voting, achieving coercion-resistance is extremely challenging. However, it's unlikely that intrusive attacks of this kind can be effectively scaled. Therefore, a more pragmatic approach involves examining coercion within election-related subprocesses.

Benaloh [13] divides coercion into three temporal phases: before voter registration, between registration and voting, and post-voting. It's important to recognize that these phases impose varying constraints on the adversary. For example, averting coercion when the voter can be coerced before registration is difficult, as the coercer might obtain valid voting credentials. If coercion is attempted post-registration but pre-voting, the success of the coercion attack hinges on the attacker's abilities and the

anti-coercion mechanisms incorporated in the voting system. However, if coercion is attempted post-voting, a coercion-resistant voting system should protect both the attacker and the voter from being able to confirm the voting choice, thereby preserving the vote's privacy. Therefore, the key characteristic of coercion-resistance is the assumptions that limit the adversary's abilities.

2.2.3.4 Verifiability

In general, verifiability in the context of voting refers to the ability of participants or third parties to verify that an election has been conducted properly and that the results accurately reflect the votes cast. There are two main types of verifiability: universal (or public) verifiability, which allows anyone to verify the authenticity of individual votes and the final tally, and individual verifiability, which allows each eligible voter to verify that their vote was properly counted. Ensuring that a voting system has these qualities can help to ensure the integrity of the election and provide confidence in the results.

Different authors have proposed slightly different definitions of verifiability in the context of voting. For example, FOO [55] defines verifiability as the ability to ensure that nobody can alter the outcome of an election. Weber [125] defines universal verifiability as the ability of anyone to verify that the voting procedure has processed and counted all legitimate votes, and defines individual verifiability as the ability of each voter to confirm that their own vote was accurately included in the final result. These definitions all emphasize the importance of being able to verify the integrity and accuracy of the voting process. Tuinstra and Benaloh [16] define verifiability in terms of the ability of voters to be satisfied that the election results truly reflect the sum of all votes cast. They propose that an election system is considered to be verifiable if a specifically designated output shared by all participants who follow proper procedures produces a common correct tally with a probability of at least $1 - \frac{m}{2^N}$ for a given security parameter N. This definition emphasizes the importance of ensuring that the election results accurately reflect the votes cast, and that voters can have confidence in the integrity of the election. Lee, Boyd, and others [85] define verifiability in terms of the ability to ensure that all legitimate votes have been included in the final result. They propose that in order to achieve this, all essential communications must be made public and the correctness of all procedures (including voting, mixing, and tallying) must be independently verified. This definition emphasizes the importance of transparency and independent verification in ensuring the integrity of the election and the accuracy of the results.

Radwin [44] defines individual verifiability in terms of a voter's ability to confirm that their vote was correctly received by the voting authority. This enables the voter to have confidence that their vote was counted properly and provides them with documentation that can be used to file a complaint if their vote is miscounted. Radwin also discusses the importance of universal verifiability, which allows any voter or third party to verify that an election was conducted correctly at a later date. This enables easy auditing of the election and is considered desirable as long as the cost is not too high. This definition emphasizes the importance of both individual and universal verifiability in ensuring the integrity of the voting process.

Cohen and Fischer [39, 12] define verifiability in terms of the confidence with which a voting scheme can be verified. They propose that a method is considered confidently verifiable if the check function meets certain criteria for random runs using a security parameter N. Specifically, if the government is honest, there should be a good chance that the check will be valid and the government will disclose an accurate count. Regardless of the government's integrity, the joint probability that the check will provide accurate results and the government will disclose an accurate tally (or release any tally at all) should be low. Verifiability is defined as the ability of a scheme to be verified with confidence δ, where δ is a function of the security parameter N and is defined as $\delta = \frac{1}{p(N)}$ for some non-constant polynomial p with a positive leading coefficient. This definition emphasizes the importance of having a high level of confidence in the

verifiability of a voting scheme, and the need to ensure that the check function meets certain criteria for random runs.

Rjaskova [104] defines individual verifiability in terms of each qualified voter's right to check that their vote was properly tallied. She also discusses universal verifiability, which allows any participant or passive spectator to verify that the election is fair and that the final tally announced is the true total of votes cast. This definition emphasizes the importance of both individual and universal verifiability in ensuring the integrity of the voting process and the accuracy of the election results.

CHAPTER 3

INTRODUCTION TO FORMAL VERIFICATION VIA THE HOARE LOGIC

The chapter covers some background on formal verification, including Hoare Logic and introduces a small programming language. It includes an examination of assignments, sequences, conditionals, and WHILE-commands, as well as a summary of their syntax. The chapter also discusses Hoare's notation and the axioms and rules of Hoare Logic, including the assignment axiom, precondition strengthening, and postcondition weakening. This information provides a foundation for Chapter 6 where we use EASYCRYPT proof assistant to show MiniVoting scheme satisfy some privacy definition.

3.1 Hoare Logic and Formal Verification

Hoare logic, also known as Floyd–Hoare logic or Hoare rules, is a formal system that contains a set of logical rules for reasoning rigorously about the correctness of computer programs. It was proposed in 1969 by the British computer scientist [73] and logician Tony Hoare, and subsequently refined by Hoare and other researchers. The original ideas were seeded by the work of Robert W. Floyd, who had published a similar system

The central feature of Hoare logic is the Hoare triple. A triple describes how the execution of a piece of code changes the state of the computation. A Hoare triple is of the form $\{P\}\ C\ \{Q\}$ where P and Q are assertions and C is a command. P is named the precondition and Q the postcondition: when the precondition is met, executing the command establishes the postcondition. Assertions are formulae in predicate logic.

Hoare logic provides axioms and inference rules for all the constructs of a simple imperative programming language. In addition to the rules for the simple language in Hoare's original paper, rules for other language constructs have been developed since then by Hoare and many other researchers. There are rules for concurrency, procedures, jumps, and pointers.

Formal proof techniques, such as induction or program transformation, can be used for this purpose. Hoare logic is a significant tool in program verification, providing a systematic method to prove program

correctness, and enabling the detection and rectification of errors prior to program deployment. It is also a powerful tool for specifying and verifying the behavior of complex systems, including distributed and concurrent programs.

Example: Let's consider a simple program that increments a variable x by 1. The precondition P is $x = n$ (where n is a natural number), the postcondition Q is $x = n+1$, and the program $\mathscr{P}$ is "x = x + 1". According to Hoare logic, the Hoare triple $\{P\}\ \mathscr{P}\ \{Q\}$ is true, which means if $x = n$ holds before the execution of "x = x + 1", then $x = n+1$ will hold after the execution.

3.1.1 A little programming language

In this context, a program is a set of instructions that are executed by a computer in order to perform a specific task. These instructions can include assignments, which assign a value to a variable; conditionals, which execute a specific block of code based on the truth value of a given condition; and other types of statements such as loops and function calls.

The phrases "program" and "command" are often used interchangeably, with the term "program" typically reserved for a set of commands that together form an algorithm. The term "statement" refers to conditions on program variables that appear in correctness requirements. However, there is a potential for ambiguity, as some authors may use this term to refer to individual commands. The syntax of a programming language refers to the rules for constructing and formatting valid instructions in the language. The semantics of a programming language, on the other hand, refers to the meaning of these instructions and how they are executed by the computer.

It is important to introduce these concepts to provide a foundation for explaining Hoare logic, which is a formal system used to reason about the correctness of programs. By understanding the fundamental components and terminology of programs, we can delve into the application of Hoare logic and its significance in ensuring program correctness.

The following notations are used to represent the different elements of a program:

- The notations V, $V_1, \ldots, V_n$ represent arbitrary variables.
- E, $E_1, \ldots, E_n$ represent arbitrary expressions (or terms). These are symbols, such as $\mathrm{X}+1$ or $\sqrt{2}$, that represent values (often numerals).
- S, $S_1, \ldots, S_n$ represent arbitrary assertions. These are conditions that are either true or false, such as $\mathrm{X} < \mathrm{Y}$ and $\mathrm{X}^2 = 1$.
- C, $C_1, \ldots, C_n$ represent the arbitrary instructions in a programming language. These may include assignments, conditionals, loops, and other types of statements. The specific syntax and semantics of these instructions will depend on the particular programming language being used.

3.1.2 Assignments

An assignment statement in a programming language is a command that allows you to modify the value of a variable. The syntax of an assignment statement is represented by the formula $V := E$, where V represents a variable and E represents an expression. The semantics of the assignment statement indicate that it changes the state of the program by assigning the value of the expression E to the variable V.

For example, the assignment statement $\mathrm{X} := \mathrm{X}+1$ increases the value of the variable X by one. This statement is interpreted as follows: the current value of the variable X is retrieved, the value of $\mathrm{X}+1$ is calculated, and then the value of $\mathrm{X}+1$ is assigned back to the variable X.

It's worth noting that the scope of a variable refers to the part of the program where the variable is visible and can be accessed. A global variable is visible and can be accessed throughout the entire program, while a local variable is only visible and accessible within a specific block of code, such as a function or loop.

3.1.3 Sequences

The syntax and semantics described above correspond to a sequence of instructions in a programming language. The syntax of a sequence of instructions is given by the formula $C_1; \cdots ; C_n$, where $C_1, \ldots, C_n$ represent individual instructions. The semantics of a sequence of instructions indicate that the instructions $C_1, \ldots, C_n$ are performed in the given order.

In the example given, the sequence of instructions R:=X; X:=Y; Y:=R uses the temporary variable R to reverse the values of the variables X and Y. This sequence of instructions is interpreted as follows: the value of X is assigned to R, the value of Y is assigned to X, and then the value of R is assigned to Y. As a result, the values of X and Y are swapped, and the value of R is changed to the previous value of X.

3.1.4 Conditionals

The syntax and semantics described above correspond to an if-then-else statement in a programming language. The syntax of an if-then-else statement is given by the formula IF S THEN C_1 ELSE C_2, where S represents an assertion (a condition that is either true or false) and C_1 and C_2 represent instructions. The semantics of an if-then-else statement indicate that if the assertion S is true in the current state, the instruction C_1 is performed. If S evaluates to false, the instruction C_2 is performed instead.

In the example given, the if-then-else statement IF X < Y THEN MAX := Y ELSE MAX := X assigns the maximum of the values of X and Y to the variable MAX. This if-then-else statement is interpreted as follows: if the value of X is less than the value of Y, the value of Y is assigned to MAX; otherwise, the value of X is assigned to MAX.

3.1.5 WHILE-commands

A while loop in a programming language is a control flow statement that allows you to repeatedly execute a block of code as long as a certain condition is true. The syntax of a while loop is given by the formula WHILE S DO C, where S represents an assertion (a condition that is either true or false) and C represents an instruction. The semantics of a while loop indicate that if the assertion S is true in the current state, the instruction C is performed, and then the while loop is repeated. If S evaluates to false, the while loop ends and the instruction C is not performed.

For example, the while loop WHILE $\neg(X = 0)$ DO X := X − 2 decreases the value of X by 2 repeatedly until the value of X becomes zero. This while loop is interpreted as follows: if the value of X is not equal to zero, the value of X is decreased by 2 and the while loop is repeated; otherwise, the while loop ends. Therefore, if the value of X is an even, non-negative integer, this while loop will terminate (with X having the value 0). In every other state, the while loop will continue to run indefinitely. It's important to ensure that the assertion S will eventually evaluate to false, otherwise the while loop will run forever (resulting in an infinite loop).

3.1.6 Summary of syntax

The *Backus-Naur Form* (BNF), named after its creators John Backus and Peter Naur [79], is a widely used notation for formal description of syntax in computer languages. First introduced in the report for ALGOL 60 programming language in 1960, BNF provides a grammatical framework that can represent the structure of any string generated by a context-free language.

BNF uses a series of rules, which are definitions of a syntax notation. Each rule is expressed as a sequence of terminal symbols (concrete symbols present in the language) and non-terminal symbols (abstract symbols that represent sets of strings). A BNF grammar is a set of derivation rules, consisting of a sequence of these symbols, often separated by a "`:=`" to denote that the sequence on the left can be replaced by the sequence on the right.

Here is a simple example of a BNF grammar for a small subset of English:

```
<sentence> := <subject> <verb> <object>
<subject> := I | You | Bob | Alice
<verb> := love | hate
<object> := hamburgers | programming
```

In this example, `<sentence>`, `<subject>`, `<verb>`, and `<object>` are non-terminal symbols that represent different sets of strings in the language. "I", "You", "Bob", "Alice", "love", "hate", "hamburgers", and "programming" are terminal symbols.

This BNF notation summarizes the syntax of a small programming language that includes assignment statements, sequences of commands, if-then-else statements, and while loops. The notation consists of four rules, each of which defines a different type of command in the language:

- The first rule <variable>:=<term> defines an assignment statement, which has the form <variable>:=<term>, where <variable> represents a variable and <term> represents an expression.
- The second rule <command>; . . . ; <command> defines a sequence of commands, which can be written as <command>; . . . ; <command>, where ; separates each individual command.
- The third rule IF <statement> THEN <command> ELSE <command> defines an if-then-else statement, which has the form IF <statement> THEN <command> ELSE <command>, where <statement> represents an assertion and <command> represents an instruction.
- The fourth rule WHILE <statement> DO <command> defines a while loop, which has the form WHILE <statement> DO <command>, where <statement> represents an assertion and <command> represents an instruction.

3.1.7 Hoare's notation

Hoare introduced the notation $\{P\}C\{Q\}$, known as a Hoare triple, for specifying the behavior of a program in an influential paper [73]. In this notation, C represents a program written in the programming language for which specifications are being made, and P and Q are conditioned on the variables used in the program. A Hoare triple $\{P\}C\{Q\}$ is considered true if, when C is executed in a state fulfilling P, the execution of C concludes in a state satisfying Q.

To illustrate the use of Hoare triples, consider the following example: X > 0 Y := X Y > 0. In this case, P is the condition that X is greater than 0, Q is the condition that Y is greater than 0, and C is the assignment instruction Y := X (i.e., "Y becomes X"). The Hoare triple is true because if X is greater than 0, then Y will also be greater than 0 after the assignment instruction is executed.

Hoare triples provide a formal method for specifying the behavior of a program and can be used to prove the correctness of a program. This is particularly useful in ensuring the reliability of programs used in safety-critical systems. Additionally, Hoare triples allow for abstract reasoning about the behavior of a program, which can aid in the development of complex software and in identifying and correcting errors in program design.

A partial correctness specification only requires that the postcondition Q holds if the program C terminates, while a total correctness specification requires both that the program C terminates if it is launched from a state fulfilling P, and that the postcondition Q holds after termination. Total correctness specifications are stronger than partial correctness specifications, as they require not only that the postcondition is satisfied, but also that the program terminates in the expected way.

To give an example, consider a program C that is intended to sort a list of numbers. A partial correctness specification for this program might be $\{P\}C\{Q\}$, where P is the precondition that the input list is a finite, non-empty list of numbers, and Q is the postcondition that the output list is a permutation of the input list and is sorted in non-decreasing order. This specification only requires that the output list is sorted if the program terminates, but does not guarantee that the program will terminate if it is launched from a state fulfilling P.

On the other hand, a total correctness specification for the same program might be $[P]C[Q]$, where P and Q have the same meaning as in the partial correctness specification. This specification requires not only that the output list is sorted, but also that the program terminates if it is launched from a state fulfilling P. This stronger requirement can be useful, for example, if the program is intended to be used in a real-time system where it is important that the program always terminates. Total correctness requires both that the program terminates if it is launched from a state fulfilling the precondition, and that the postcondition holds after termination. Partial correctness only requires that the postcondition holds if the program terminates.

It is important to note that total correctness does not imply partial correctness. For example, a program that always terminates but produces an incorrect result would be considered to have total correctness (if the precondition does not hold), but not partial correctness. On the other hand, a program that satisfies the postcondition only under certain conditions but may or may not terminate would be considered to have partial correctness, but not total correctness.

3.1.8 Axioms and rules of Hoare logic

In summary, the formal proof is a logical argument that demonstrates the truth of a statement, called a theorem, by showing that it can be derived from a set of axioms and rules of inference. The process of constructing a formal proof involves writing a series of statements, each of which is either an axiom or follows logically from previous statements using an inference rule. If a theorem has been formally proved, we write $\vdash S$ to indicate that it has proof. In some cases, it may be necessary to simply claim that a theorem can be proved without providing complete formal proof. This may be because the proof is too complex or involves too many steps to be written out in full, or because the proof relies on assumptions or claims that are taken to be true without being formally proved. In such cases, it is important to be explicit about any assumptions that are being made and to provide as much detail as possible about the proof. However, in order to maintain the highest level of rigor, it is ultimately necessary to provide a complete formal proof of any theorem that is claimed to be true. In Hoare logic, the axioms are templates that can be filled in with specific information to create assertions about the behavior of programs. The inference rules of Hoare logic allow us to deduce new assertions from previously proven assertions using logical reasoning. The notation $\frac{\vdash S_1, \vdash S_2, \ldots, \vdash S_n}{\vdash S}$ is used to express an inference rule, which states that if we can prove the statements $\vdash S_1, \ldots, \vdash S_n$, then we can conclude that $\vdash S$ is also true. The statements $\vdash S_1, \ldots, \vdash S_n$ are

called the hypotheses of the rule, and can be either Hoare logic theorems or mathematical theorems, or a combination of both.

For example, the sequencing rule is an inference rule in Hoare logic that allows us to deduce an assertion about a sequence of two statements from assertions about the individual statements. The sequencing rule is often expressed in the form $\frac{\{P\}S_1\{Q\} \quad \{Q\}S_2\{R\}}{\{P\}S_1;S_2\{R\}}$, which states that if we can prove that the precondition P implies the postcondition Q for the first statement S_1, and that the precondition Q implies the postcondition R for the second statement S_2, then we can conclude that the precondition P implies the postcondition R for the sequence of statements $S_1;S_2$.

3.1.9 The assignment axiom

In Hoare logic, the assignment axiom is an axiom scheme that specifies the partial correctness of an assignment statement of the form "x := E", where x is variable and E is an expression. The assignment axiom has the following form:

$$\{P[E/x]\}x := E\{P\}$$

This axiom states that if we can prove that the precondition P holds after replacing every occurrence of the variable x with the expression E, then we can conclude that the precondition P holds before and after the assignment statement "x := E" is executed. In other words, the assignment axiom states that if the expression E is evaluated and assigned to the variable x, the resulting assignment will not change the truth of the precondition P. This means that the assignment statement preserves any properties of the program that are stated in the precondition. For example, consider the following assignment statement:

$$x := x + 1$$

Suppose we want to prove that the precondition P holds before and after this assignment is executed. We can use the assignment axiom as follows:

$$\{P[(x+1)/x]\}x := x + 1\{P\}$$

If we can prove that the precondition P holds after replacing every occurrence of x with the expression $x+1$, then we can conclude that the precondition P holds before and after the assignment is executed.

The assignment axiom is a useful tool for demonstrating the partial correctness of assignment statements in Hoare logic. It allows us to reason about the effects of assignment statements on the program's behavior without having to consider the specific details of the expressions being assigned.

3.1.10 Precondition strengthening

In the context of Hoare logic, a program doesn't "satisfy" a precondition per se. Instead, a precondition is a condition that should be satisfied (i.e., hold true) before the execution of a program or a specific part of the program (e.g., a function or a block of code). A program is said to "meet" a precondition if the state of the system (the values of variables, the state of memory, etc.), when the program starts execution, satisfies the precondition.

Once a precondition is met, the program promises to deliver a certain result or bring the system into a certain state, which is expressed as a postcondition. The relationship between preconditions, the program, and postconditions is often expressed in the form of a Hoare triple: {P} C {Q}, where P is the precondition, C is the command (or program), and Q is the postcondition.

Here's a simple example of a function that increments a number:

$$\{x = n\} \text{ increment(x) } \{x = n + 1\} \tag{3.1}$$

In the above Hoare triple, the precondition is "x = n", which means that before the execution of the "increment(x)" function, the variable "x" should be equal to "n". The postcondition is "x = n + 1", which means that after the execution of "increment(x)", the variable "x" should be equal to "n + 1". If the precondition is met before the function executes, and if the function is correct, then the postcondition will be true after the function executes.

The precondition strengthening rule is a rule that allows you to strengthen the precondition of a program or a program block. This rule states that if a program or program block satisfies its original precondition, then it also satisfies a stronger precondition. The precondition strengthening rule can then be stated as follows:

If $\{P\}C\{Q\}$ and P' implies P, then $\{P'\}C\{Q\}$. One way to understand the precondition strengthening rule is to think of the precondition as a set of conditions that must be satisfied before a program or program block can be executed. The stronger precondition P' is a larger set of conditions that includes the original precondition P as a subset. The precondition strengthening rule is useful in program verification, as it allows you to make the preconditions of a program or program block stronger without changing the program or program block itself. This can be useful in situations where you want to make sure that a program or program block is only executed under certain conditions, or where you want to make sure that certain conditions are satisfied before a program or program block is executed.

3.1.11 Postcondition weakening

The postcondition weakening rule states that if a program C satisfies a stronger postcondition Q', and Q' implies Q, then C also satisfies the weaker postcondition Q. In other words, if we can prove that C satisfies a stronger postcondition, we can also prove that it satisfies a weaker postcondition. The postcondition weakening rule is often used in conjunction with the precondition strengthening rule, which allows the programmer to strengthen the precondition of a program while preserving its correctness. Together, these rules allow the programmer to modify the precondition and postcondition of a program in order to make it easier to verify. It is important to note that the postcondition weakening rule should be used with caution, as it can potentially weaken the correctness of a program. It is always important to ensure that the weakened postcondition is still satisfied by the program being verified.

3.1.12 Specification conjunction and disjunction

The specification conjunction and disjunction rule allow the combining of multiple partial correctness statements into a single statement using conjunction (and) or disjunction (or). The specification conjunction rule states that if a program C satisfies two partial correctness statements $\{P1\}C\{Q1\}$ and $\{P2\}C\{Q2\}$, then C also satisfies the conjunction of these statements, $\{P1 and P2\}$ C $\{Q1 and Q2\}$. In other words, if a program satisfies two partial correctness statements, it also satisfies the conjunction of these statements. The specification disjunction rule states that if a program C satisfies either of two partial correctness statements $\{P1\}$ C $\{Q\}$ or $\{P2\}$ C $\{Q\}$, then C also satisfies the disjunction of these statements, $\{P1 or P2\}$ C $\{Q\}$. In other words, if a program satisfies either of two partial correctness statements, it also satisfies the disjunction of these statements.

These rules can be useful in situations where we want to specify multiple conditions under which a program is correct, or where they want to specify multiple possible outcomes of a program. For example, consider the following program:

```
int max(int x, int y) {
  if (x > y) {
    return x;
  } else {
    return y;
  }
}
```

We might want to specify two partial correctness statements for this program: one for the case where $x > y$, and one for the case where $y >= x$. Using the specification conjunction and disjunction rule, we could combine these statements into a single statement as follows:

```
{(x > y) or (y >= x)} max(x, y) {return value == max(x, y)}
```

This single statement specifies that the program max is correct for all possible input values of x and y, and that it returns the maximum of x and y. In general, the specification conjunction and disjunction rule can be useful for specifying complex conditions and outcomes in a program.

3.1.13 The sequencing rule

In Hoare logic, the sequencing rule (also known as the composition rule) states that if we have two program statements, P and Q, and if the execution of statement P establishes a certain precondition, R, and the execution of statement Q establishes a certain postcondition, S, then the execution of the sequence $P;Q$ establishes the precondition R and the postcondition S. The sequencing rule can be written formally as follows:

$$\frac{\{R\}P\{S\}\ ,\ \{S\}Q\{T\}}{\{R\}P;Q\{T\}}$$

Here, the curly braces denote the precondition and postcondition, respectively. The symbol ";" represents the sequencing of two statements, P and Q.

The sequencing rule is a fundamental rule in Hoare logic, as it allows us to reason about the correctness of programs by composing smaller, simpler program statements. It is often used in conjunction with other rules, such as the assignment rule, the conditional rule, and the loop rule, to prove the correctness of more complex programs.

3.1.14 The conditional rule

The conditional rule states that if we have a program statement of the form "if $(B)P$ else Q," and if the execution of statement P establishes a certain postcondition, S, when the Boolean condition B is true, and the execution of statement Q establishes a certain postcondition, T, when the Boolean condition B is false, then the execution of the entire conditional statement establishes the postcondition S when B is true, and the postcondition T when B is false.

$$\frac{\{\text{B}\}\ \text{P}\ \{\text{S}\}\ \text{if B is true} \qquad \{\text{B}\}\ \text{Q}\ \{\text{T}\}\ \text{if B is false}}{\{\text{B}\}\ \text{if (B) P else Q}\ \{\text{S if B else T}\}}$$

The symbol "if B is true" and "if B is false" are used to separate the two cases of the conditional statement (when B is true and when B is false).

The conditional rule is a fundamental rule in Hoare logic, as it allows us to reason about the correctness of programs that contain conditional statements. It is often used in conjunction with other rules, such as the assignment rule and the sequencing rule, to prove the correctness of more complex programs.

3.1.15 The WHILE-rule

the WHILE-rule (also known as the loop rule) is a rule that is used to reason about the correctness of programs that contain while loops. A while loop is a programming construct that allows a program to repeatedly execute a block of statements as long as a particular condition (called the loop invariant) is true. The WHILE-rule can be written formally as follows:

$$\frac{\{\text{I}\}\ \text{P}\ \{\text{I}\} \qquad \text{I} \wedge \text{B} \Rightarrow \{\text{I}\}\ \text{Q}\ \{\text{I}\}}{\{\text{I}\}\ \text{while (B)}\ \{\ \text{P; Q}\ \}\ \{\text{I} \wedge \text{!B}\}}$$

The symbol $\wedge$ represents the logical "and" operator, and the symbol "!" represents the logical "not" operator. The symbols "$\{I\}$" and "$\{I \wedge !B\}$" denote the loop invariant, which is a condition that must be satisfied at the beginning and end of each iteration of the loop, respectively.

The WHILE-rule states that if the loop invariant I is satisfied at the beginning of the loop, if the execution of statement P preserves the loop invariant (that is, it establishes the loop invariant as its postcondition), and if the execution of statement Q preserves the loop invariant and the Boolean condition B is true, then the execution of the entire while loop will preserve the loop invariant and terminate when the Boolean condition B becomes false.

The WHILE-rule is a fundamental rule in Hoare logic, as it allows us to reason about the correctness of programs that contain while loops. It is often used in conjunction with other rules, such as the assignment rule and the sequencing rule, to prove the correctness of more complex programs.

3.2 EASYCRYPT Backgrounds

EASYCRYPT is a proof assistant for cryptographic algorithms and imperative programs. It is a formal demonstration tool that helps users create and validate proofs. It doesn't create a proof independently, but instead assists in its formation and enables machine-checked verification that each step logically follows from the last one. EASYCRYPT offers a language to write definitions, programs, and theorems, and an environment to develop machine-checked proofs [6, 38].

Formal verification, a technique used to prove that a piece of code correctly implements a specification, is one of the primary applications of EASYCRYPT. Formal verification and formal methods have been around since the 1950s, and today they are employed in various ways: from automating the checking of security proofs to automating checks for functional correctness and the absence of side-channels attacks. Code verified using such formal verification has been deployed in popular products like Mozilla Firefox and Google Chrome [38].

EASYCRYPT operates based on a detailed process. Given a description of an algorithm in a natural language, the goal is to produce two proofs: one that shows that the algorithm has the desired security properties and another that verifies the correct implementation of the algorithm. This is achieved in four steps: turning the algorithm and its security goals into a formal specification, using formal analysis to prove that the algorithm attains the specified properties, using formal verification to prove that the implementation correctly implements the algorithm, and using formal verification to prove that the implementation has additional properties like memory safety, efficiency, etc. [38].

EASYCRYPT was originally designed for mechanizing the generation of proofs of game-based security of cryptographic schemes and protocols. However, it has been extended to mechanize proofs of the security of cryptographic protocols within the universally composable (UC) security framework. This makes it possible to mechanize and formally verify the entire sequence of steps needed for proving simulation-based security in a modular way. These steps include specifying a protocol and the desired ideal functionality, constructing a simulator and demonstrating its validity via reduction to hard computational problems, and invoking the universal composition operation, and demonstrating that it indeed preserves security [30].

3.2.1 Foundations

EASYCRYPT is a proof assistant that uses the objective-directed proof methodology, which is a technique for constructing formal proofs in a logical system. The objective-directed proof methodology involves breaking down the proposition that needs to be proven into smaller, more manageable subproblems, and then using tactics to solve these subproblems until the original proposition has been proven. The objective-directed proof methodology is often used in the Coq proof assistant, which is a popular tool for constructing and verifying formal proofs in computer science. EASYCRYPT is an extension of Coq that is specifically designed for proving the security of cryptographic systems. In the objective-directed proof methodology, the user inputs a proposition that they want to prove, along with a name for future reference, using the command "lemma". The EASYCRYPT system then presents the formula as a goal to be proven, possibly providing a context of local facts that may be used to prove the goal. The user then inputs a command to break down the goal into smaller subgoals, ideally down to axioms or formulas that are already known to be true. The EASYCRYPT system then produces a list of subgoals that need to be proven, and the process is repeated until there are no more subgoals remaining. Once all of the subgoals have been proven, the proof is complete, and the user can save the lemma using the command "qed". Before defining a lemma to prove, the user may describe the structures he will work with, axioms he assumes, or import libraries containing such definitions. EASYCRYPT includes a typed expression language based on polymorphic typed lambda calculus for this purpose. In EASYCRYPT, types are non-empty collections of values, whereas operators are typed functions on these collections. EASYCRYPT's internal kernel contains built-in types such as bool, int, real, and unit (the type occupied by tt or ()). Formalization of lists, arrays, sets, finite sets, maps, finite maps, distributions, etc., are included in the standard libraries. EASYCRYPT's theory system enables the user to reorganize related types, predicates, operators, modules, axioms, and lemmas. EASYCRYPT enables users to describe and specify their data-types and operators, such as inductive data-types and operators defined by pattern matching. Types and operators without definitions are considered abstract and may be seen as context parameters. Also, a powerful feature known as *theory cloning* allows for the instantiation and specialization of theories, which can be abstracted over types, operators, predicates, and proofs.

3.2.2 Probability Distributions

A sub-distribution is a function that assigns probabilities to events in a set. In the case of a discrete sub-distribution, the set is a discrete set of possible outcomes, and the probabilities are assigned to each individual outcome. The type "t distr" represents the type of real discrete sub-distributions, and "$\mathscr{D}(t)$" represents the set of all sub-distributions on a type "t". The mass function "f" is a non-negative function that assigns probabilities to the outcomes in the set "t", and is defined over a discrete support. The mass function must also satisfy the condition that the sum of the probabilities assigned to all of the outcomes in the set "t" is less than or equal to 1.

3.2.3 Modules in EASYCRYPT

In EASYCRYPT, programs are defined as modules, which are stateful "objects" that consist of global variables and processes. Global variables are accessible outside of the module and determine the module's internal state at all times, while processes are collections of local variable declarations and a series of instructions. The language used to define these programs is called pWhile, and is a straightforward imperative probabilistic programming language. EASYCRYPT's module system also supports higher-order modules, which enables a cryptographer to design a game by specifying an opponent and a cryptosystem as parameters. Before creating a game that accepts modules as parameters, the user must declare a module type for each type of parameter. The semantics of an instruction sequence "c" in EASYCRYPT is defined as a function that maps program memories to sub-distributions on program memories. This function, denoted "$[[c]]$", takes a program memory "$\&m$" as input and produces a sub-distribution on program memories as output. If one of the potential executions of "c" does not terminate, the resulting sub-distribution will have a total probability of less than 1. The probability of an event occurring in a program "c" with an initial memory "$\&m$" is defined as the total of the masses of all memories in the distribution "$[[c]]\langle m\rangle$" that satisfy the event's boolean expression "ψ". This probability can be expressed using the notation "$\Pr[c, m : \psi]$", or equivalently as "$\Pr[c@\&m : \psi]$".

Adversary as a module, Quantification over all adversaries In EASYCRYPT, adversaries are modeled as abstract modules of a specified module type. The module type specifies the procedure type, but the code of the abstract module is unknown. This allows the user to argue about the behavior of an adversary without knowing its exact implementation. EASYCRYPT provides three different classes of probabilistic program facts, known as judgments, that allow the user to make assertions about the behavior of probabilistic programs. These judgments are:

HL (Hoare Logic) with probabilistic programs: This judgment allows the user to prove properties of probabilistic programs using traditional Hoare Logic techniques.

pHL (probabilistic Hoare Logic): This judgment enables the user to prove properties about the likelihood of a procedure's execution yielding a post-condition that holds.

pRHL (probabilistic Relational Hoare Logic): This judgment allows the user to bind a pair of processes and prove properties about their interactions.

3.2.3.1 Hoare Logic(HL)

In Hoare Logic (See 3.1), a Hoare judgment is a statement about the behavior of a (probabilistic) program. It consists of a program "c", and two predicates "ϕ" and "ψ". The Hoare judgment "$[c : \phi \Rightarrow \psi]$" asserts that for all memories "$\&m$" that meet the precondition "ϕ", all memories in the support of the subdistribution "$[[c]]\langle m\rangle$" will satisfy the postcondition "ψ". If the program "c" is deterministic, then the subdistribution "$[[c]]\langle m\rangle$" will support no more than one element. This means that the program will have a unique behavior, and the Hoare judgment can be used to prove the properties about this behavior.

3.2.3.2 Probabilistic Hoare Logic(pHL)

Probabilistic Hoare Logic is an extension of traditional Hoare Logic that allows the user to reason about the likelihood of a program satisfying a postcondition. A probabilistic Hoare judgment is a quintuplet "$[c : \phi \Rightarrow \psi] \diamond p$" that consists of a program "c", two predicates "ϕ" and "ψ", a relation on reals "$\diamond$" (which can be either "$\leq$", "$=$", or "$\geq$"), and a real "p". The probabilistic Hoare judgment "$[c : \phi \Rightarrow \psi] \diamond p$" asserts that, for all memories "$\&m$", if "$\phi\langle m\rangle$" holds, then the probability of "c" satisfying the postcondition "ψ" when run with initial memory "m" will be related to "p" according to the relation "$\diamond$". In other words, the probability of "c" satisfying "ψ" will be either less than, equal to, or greater than "p", depending on the

value of "$\diamond$". Probabilistic Hoare Logic is a useful tool for reasoning about the likelihood of a program satisfying a given postcondition and is particularly useful in the context of probabilistic programs where the behavior of the program is not deterministic. It allows the user to make assertions about the probability of a program satisfying a postcondition, rather than just the existence of a single execution that satisfies the postcondition.

3.2.3.3 Probabilistic Relational Hoare Logic

The Probabilistic Relational Hoare Logic (pRHL) is an extension of Hoare Logic designed to facilitate reasoning about the interactions between two programs. A pRHL judgment takes the form of a quadruplet "$[C_1 \sim C_2 : \Phi \Rightarrow \Psi]$" comprising two programs, C_1 and C_2, and two relations, Φ and Ψ. The precondition Φ and postcondition Ψ are expressed as first-order formulas constructed using relational expressions that represent relations on the program memory. These expressions may incorporate program variables distinguished by the labels "$< 1 >$" or "$< 2 >$" to differentiate between the two programs, as well as logical variables that are only present when quantified.

Given an initial pair of memories, denoted as m_1 and m_2, satisfying the precondition Φ, the distributions "$[[C_1]] < m_1 >$" and "$[[C_2]] < m_2 >$" resulting from the execution of C_1 and C_2, respectively, must conform to the lifting operator L applied to the postcondition Ψ. The lifting operator L transforms a binary relation on states into a binary relation on sub-distributions over states [77].

CHAPTER 4

INTRODUCTION TO MACHINE-LEARNING

This chapter introduces the essential materials and methodologies that lay the groundwork for our exploration of specific machine learning paradigms. While the subsequent sections provide a thorough overview of the tools, frameworks, and principles employed in this thesis for precise modeling, analysis, and implementation of complex machine learning systems, readers interested in more detailed definitions and discussions about machine learning concepts are referred to [57].

4.1 Assessing Data Balance in Datasets

Imbalanced datasets are datasets in which one class is significantly more prevalent than the other classes. This can cause problems when training machine learning models, as the models may be biased towards the majority class and have difficulty accurately predicting the minority class. One common problem with imbalanced datasets is that the accuracy of the model is not a reliable evaluation metric, as the model can achieve high accuracy by simply predicting the majority class all the time. This is known as the "accuracy paradox." To address this issue, other evaluation metrics such as precision, recall, and the F1 score should be used, as these metrics take into account the balance of the classes in the dataset. Another common problem with imbalanced datasets is that the model may be biased towards the majority class, as it has more data points to learn from. This can lead to poor performance in the minority class, which is known as the "class imbalance problem." To address this issue, various techniques such as oversampling, undersampling, and class weighting can be used to balance the classes in the dataset. Overall, imbalanced datasets can be challenging to work with, but there are various techniques that can be used to address the problems they can cause when training machine learning models. It is important to carefully evaluate the performance of the model using appropriate evaluation metrics and to consider techniques to balance the classes in the dataset.

4.1.1 Oversampling

Oversampling is a technique used in machine learning to balance the class distribution in a dataset. It is commonly used when the dataset is imbalanced, meaning that one class is significantly larger than the other. This can be a problem because an imbalanced dataset can lead to a model that is biased toward the larger class, leading to poor performance in the smaller class. One way to address this issue is to use oversampling, which involves creating additional synthetic samples of the smaller class in order to balance the dataset. Simple random oversampling involves simply replicating the samples of the smaller class at random until the class distribution is balanced. This can be effective in some cases, but it can also lead to overfitting since it does not consider the underlying structure of the data. Oversampling is important because it can help to improve the performance of machine learning models on imbalanced datasets. By balancing the class distribution, it can help to reduce bias and improve the model's ability to accurately classify samples from the smaller class. We can then use the oversampled data to train a machine-learning model. The model will be trained on a balanced dataset, which can help to improve its performance. In summary, oversampling is a technique used to balance the class distribution in a dataset.

4.1.2 Undersampling

One way to address the issue of the imbalanced dataset is to use undersampling, which involves reducing the number of samples in the larger class in order to balance the dataset. Simple random undersampling involves randomly selecting a smaller number of samples from the larger class until the class distribution is balanced. This can be effective in some cases, but it can also lead to the loss of valuable information since it does not consider the underlying structure of the data. Undersampling is important because it can help to improve the performance of machine learning models on imbalanced datasets. Balancing the class distribution can help to reduce bias and improve the model's ability to accurately classify samples from the smaller class. We can then use the undersampled data to train a machine-learning model. The model will be trained on a balanced dataset, which can help to improve its performance. In summary, undersampling is a technique used to balance the class distribution in a dataset. It can help to improve the performance of machine learning models on imbalanced datasets and reduce bias towards the larger class.

4.1.3 Limitation

Both oversampling and undersampling can be effective techniques for addressing the class imbalance problem, but they also have their limitations. Oversampling can increase the risk of overfitting, as it increases the complexity of the model. Undersampling can reduce the amount of data available for the model to learn from, which can lead to a decrease in performance. Another approach is to use a different evaluation metric, such as precision, recall, or the F1 score, which takes into account the class distribution in the dataset. These metrics can be more appropriate for imbalanced datasets, as they place more emphasis on the performance of the classifier on the minority class. Additionally, it is important to choose an appropriate classification algorithm for the imbalanced dataset. Some algorithms, such as decision trees and support vector machines, are more sensitive to class imbalance than others, such as k-nearest neighbors and naive Bayes. Choosing an appropriate algorithm can help to improve the performance of the classifier on the minority class. In summary, imbalanced datasets can be a challenge when training a binary classifier, as the classifier may be biased towards the larger class. Techniques such as oversampling, undersampling, and using different evaluation metrics can help to address this issue, as can choosing an appropriate classification algorithm.

4.2 Feature Engineering

Feature engineering is the process of selecting and constructing relevant features for a machine learning model. It is a crucial step in the modeling process, as the quality and relevance of the features can greatly affect the performance of the model. There are two main approaches to feature engineering: feature selection and feature extraction. Feature selection involves selecting a subset of the available features to use in the model, while feature extraction involves creating new features from the available data.

4.2.1 Feature Selection

Feature selection is the process of selecting a subset of relevant features for use in model construction. It is an important step in the data preprocessing phase because it helps to improve the performance and interpretability of the resulting model. There are several benefits to performing feature selection. First, it reduces the complexity of the model, which can improve the model's performance on unseen data. Second, it can reduce the amount of time and resources required to train the model. Third, it can improve the interpretability of the model by reducing the number of features that need to be considered when interpreting the model's predictions. There are several different techniques for performing feature selection. One popular method is called mutual information gain, which measures the mutual dependence between each feature and the target variable. Features with high mutual information gain are likely to be more relevant to the target variable and should be included in the model. Other techniques for feature selection include forward and backward selection, which involves iteratively adding or removing features from the model based on their performance. In summary, feature selection is an important step in the data preprocessing phase that can improve the performance and interpretability of the resulting model. By selecting only the most relevant features, it is possible to build more accurate and interpretable models with less complexity and computational cost.

4.2.1.1 Using Pearson Correlation

Pearson correlation is a measure of the linear relationship between two variables. It is used to determine how closely two variables are related and to predict one variable based on the other. The Pearson correlation coefficient, also known as the Pearson r, is a statistical measure that ranges from -1 to 1, where -1 indicates a strong negative relationship, 0 indicates no relationship, and 1 indicates a strong positive relationship. To calculate the Pearson correlation between two variables, you need to find the covariance between the two variables and divide it by the product of the standard deviations of each variable. This will give you the Pearson correlation coefficient, which you can interpret to understand the strength and direction of the relationship between the two variables. For example, if you have data on the heights and weights of a group of people, you can use Pearson correlation to determine whether there is a relationship between height and weight. If the Pearson correlation coefficient is close to 1, it indicates a strong positive relationship between height and weight, meaning that as height increases, weight also tends to increase. On the other hand, if the coefficient is close to -1, it indicates a strong negative relationship, meaning that as height increases, weight tends to decrease. In addition to its use in predicting one variable based on the other, Pearson correlation is also used in a variety of other applications, such as determining the strength of a linear relationship between two variables in a regression analysis or comparing the results of different experiments to determine whether there is a significant difference between them. Overall, the Pearson correlation is a widely used and powerful tool for understanding the relationship between two variables.

4.2.1.2 Chi-square test

The chi-square test is a statistical method used to ascertain if a significant difference exists between expected and observed frequencies in one or more categories. This test is essential for hypothesis testing, either to evaluate how well a model fits the observed data or to test the independence of two variables within a contingency table. The chi-square statistic is derived by summing up the squared differences between observed and expected frequencies for each category and dividing by the expected frequency. The chi-square statistic is then compared to a critical value derived from the chi-square distribution, dependent on the degrees of freedom and the desired significance level. If the chi-square statistic exceeds the critical value, the null hypothesis is rejected, indicating that observed and expected frequencies significantly differ. While the chi-square test is versatile and widely used in various fields like psychology, sociology, and biology, it has assumptions like random sampling and sufficiently large expected frequencies that must be accounted for before applying the test.

Suppose you are conducting a study to assess the relationship between age groups and beverage preferences among customers in a café. A random sample of 100 customers is surveyed, and the data is displayed in Table 1.

TABLE 4.1: Table 1: Observed frequencies of beverage preference by age group

	Coffee	Tea	Total
Youth	30	20	50
Adult	35	15	50
Total	65	35	100

The first step in conducting a chi-square test is to calculate the expected frequencies for each cell in Table 1. This involves multiplying the row total by the column total for each cell and dividing by the total number of observations. For instance, the expected frequency for Youth who prefer Coffee would be (50*65)/100=32.5(50*65)/100=32.5. The expected frequencies are presented in Table 2.

TABLE 4.2: Table 2: Expected frequencies of beverage preference by age group

	Coffee	Tea	Total
Youth	32.5	17.5	50
Adult	32.5	17.5	50
Total	65	35	100

Subsequently, you would compute the chi-square statistic by summing the squared differences between observed and expected frequencies for each cell and dividing by the corresponding expected frequency. In this example, the calculated chi-square statistic would be 0.92. This value is then compared to the critical value from the chi-square distribution with one degree of freedom at a 0.05 level of significance. If the chi-square statistic surpasses this critical value, the null hypothesis is rejected, and a significant relationship between age group and beverage preference is established. In this case, however, the chi-square value suggests that no significant relationship exists between the two variables.

4.2.1.3 Mutual Information Gain

Mutual information gain is a measure of the information shared between two random variables X and Y. It quantifies the degree of dependence between the variables and is commonly used in machine learning for feature selection.

The mutual information gain $I(X;Y)$ between two variables X and Y is calculated as the difference between the entropy $H(X)$ and $H(Y)$ of each variable alone and the joint entropy $H(X,Y)$ of the two variables together:

$$I(X;Y) = H(X) + H(Y) - H(X,Y)$$

Here, entropy $H(X)$ is a measure of uncertainty and is defined for a discrete random variable X as:

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$

The mutual information gain essentially quantifies the reduction in uncertainty for one variable given the knowledge of the other.

For example, consider a more realistic dataset with three variables for eye color E: blue (B), green (G), brown (Br), and two variables for hair color H: blonde (Bl), black (Blk).

Let the probabilities for eye colors be $P(B) = 0.4$, $P(G) = 0.1$, $P(Br) = 0.5$, and for hair colors, let $P(Bl) = 0.7$ and $P(Blk) = 0.3$.

The entropy for eye color E would then be calculated as:

$$H(E) = -(0.4 \log_2 0.4 + 0.1 \log_2 0.1 + 0.5 \log_2 0.5) \approx 1.36$$

Similarly, the entropy for hair color H would be:

$$H(H) = -(0.7 \log_2 0.7 + 0.3 \log_2 0.3) \approx 0.88$$

Assuming the joint probabilities as $P(B,Bl) = 0.28$, $P(B,Blk) = 0.12$, $P(G,Bl) = 0.07$, $P(G,Blk) = 0.03$, $P(Br,Bl) = 0.35$, and $P(Br,Blk) = 0.15$, the joint entropy $H(E,H)$ would be:

$$H(E,H) = -(0.28 \log_2 0.28 + 0.12 \log_2 0.12 + 0.07 \log_2 0.07 + 0.03 \log_2 0.03 + 0.35 \log_2 0.35 + 0.15 \log_2 0.15) \approx 2.45$$

Therefore, the mutual information gain $I(E;H)$ would be $1.36 + 0.88 - 2.45 = -0.21$, indicating a small but negative relationship between eye color and hair color.

4.2.2 Feature Extraction

Feature extraction is the process of transforming raw data into a set of features that can be easily and reliably used in machine learning and other data analysis tasks. It is an important step in the data preprocessing phase because it helps to improve the performance and interpretability of the resulting model. There are several benefits to performing feature extraction. First, it can improve the performance of the model by transforming the raw data into a more useful representation. For example, if the raw data consists of images, feature extraction can be used to extract the edges, corners, and other important features from the images, which can be used as inputs to a machine-learning model. Second, it can reduce the amount of time and resources required to train the model by reducing the dimensionality of the data. Third, it can improve the interpretability of the model by providing a more intuitive representation of the data. There are several different techniques for performing feature extraction. One popular method is called principal component analysis (PCA), which is used to reduce the dimensionality of the data by projecting it onto a lower-dimensional space. Other techniques for feature extraction include independent component analysis (ICA) and non-negative matrix factorization (NMF), which are used to decompose the data into a set of underlying factors.

4.2.2.1 Principal Component Analysis (PCA)

Principal component analysis (PCA) is a statistical procedure that is used to analyze the interrelationships among a large number of variables and to explain these variables in terms of a smaller number of underlying factors. The aim of PCA is to reduce the dimensionality of a data set, i.e. to simplify it by extracting the most important information from the data and expressing it in a smaller number of variables. PCA is a widely used technique in the fields of statistics and machine learning, and it has several important applications. For example, PCA can be used to reduce the number of variables in a data set for the purposes of data visualization, data compression, and feature extraction. In these cases, PCA can help to make the data more manageable and easier to analyze. Another important application of PCA is in the field of predictive modeling. In this case, PCA can be used to identify the underlying factors that are most important for predicting the outcome of a particular event. By reducing the dimensionality of the data, PCA can help to improve the performance of predictive models, and it can also help to identify potential relationships between different variables that might not be obvious from the raw data. The importance of PCA lies in its ability to extract the most important information from a data set and to express it in a smaller number of variables. This can make the data easier to understand and analyze, and it can also help to improve the performance of predictive models. Additionally, PCA can be used to identify relationships between different variables in a data set, which can provide valuable insights for further analysis. One simple example of PCA in Python is to use the technique to reduce the number of variables in a data set. For instance, suppose we have a data set with 100 variables, but we want to reduce it to just 10 variables. This can make the data easier to analyze and can also improve the performance of predictive models.

4.2.2.2 Independent Component Analysis (ICA)

Independent component analysis (ICA) is a statistical technique used to identify and separate underlying sources of variation in a data set. ICA is based on the assumption that the observed data is a linear combination of underlying independent sources, and the goal of ICA is to estimate these independent sources and separate them from the observed data. ICA has several important properties that make it a useful tool for data analysis and signal processing. First, ICA is a non-parametric method, which means that it does not make any assumptions about the underlying distribution of the data. This makes ICA a flexible and versatile technique that can be applied to a wide range of data sets. Second, ICA is an unsupervised learning method, which means that it does not require any prior knowledge or labeling of the data. This makes ICA particularly useful for exploring and discovering hidden patterns in data sets that have not been previously studied. Third, ICA is a blind source separation method, which means that it can separate independent sources of variation in the data even when these sources are mixed together or when their statistical properties are unknown. This property of ICA makes it a powerful tool for analyzing complex and noisy data sets. Overall, ICA is a valuable tool for data analysis and signal processing, and it has many important applications in fields such as neuroscience, engineering, and finance. By estimating and separating independent sources of variation in data, ICA can help to uncover hidden patterns and relationships, and it can provide valuable insights for further analysis.

4.3 Machine-Learning Classifiers

This section of the thesis will concentrate on using the previously created datasets to train a machine learning model that functions as an ML classifier. Subsequently, we will assess the model's generated

reports and its overall performance. The evaluation process will encompass measuring the model's accuracy, precision, recall, and other critical performance metrics, which will assist us in gauging its efficacy. To enhance the performance of our machine learning model, we will also include a hyperparameter tuning step in the aforementioned process. Hyperparameter tuning involves optimizing the model's hyperparameters, which are the adjustable settings that determine its behavior and performance. We will utilize various techniques and tools to explore the optimal values of the hyperparameters, including grid search. By adjusting the hyperparameters and evaluating the model's performance, we can find the best possible configuration for our ML classifier. After performing hyperparameter tuning, we will retrain the model with the new hyperparameter settings and assess its performance using the same evaluation metrics. This step will allow us to fine-tune the model and achieve the best possible results. In the following, we aim to provide a concise but informative explanation of the different machine-learning classification algorithms that we utilized in our thesis. We recognize the significance of selecting the appropriate algorithm to achieve the best possible results, and thus we will offer an overview of the key characteristics and applications of each algorithm. We employed a range of algorithms such as logistic regression, decision trees, k-nearest neighbors, support vector machines, and neural networks, to name a few. We will describe the underlying principles, benefits, and drawbacks of each algorithm, as well as how they compare and contrast with one another.

4.3.1 Logistic Regression

Logistic regression is a widely used supervised machine-learning technique for classification tasks. It is a statistical model that is used to predict the probability of an outcome being one of two possible classes, typically referred to as 0 and 1. For example, it can be used to predict whether a patient has a certain disease (class 1) or not (class 0), or whether an email is spam (class 1) or not (class 0). The basic idea behind logistic regression is to find the best linear boundary that separates the classes in the feature space. This boundary is called the decision boundary. Logistic regression uses an iterative process to learn the coefficients (weights) of the features in the training data, which can then be used to make predictions on unseen data. One key advantage of logistic regression is that it can be easily implemented and trained, even on large datasets. It is also a robust model that can handle a variety of features and is not sensitive to the scaling of the features. However, it can only model binary classification tasks and may not perform well on datasets with non-linear patterns or multiple classes. Upon training our machine learning model with various datasets and conducting hyperparameter tuning to optimize its performance, we have obtained a comprehensive report on the logistic regression classifier. This report provides valuable insights into the efficacy of the algorithm and its performance on our specific data. Our report includes various evaluation metrics, including accuracy, precision, recall, and F1 score, to assess the model's effectiveness in accurately predicting the outcomes. We also examine the confusion matrix to analyze the model's ability to correctly classify instances.

4.3.2 Decision Trees

Decision tree classifiers are a type of supervised machine learning algorithm that can be used for both classification and regression tasks. They are called "decision trees" because they build a tree-like model of decisions based on the features of the data. Each internal node in the tree represents a feature, and the branches represent the decision based on that feature. The leaves of the tree represent the final classification or prediction. Decision trees are a popular choice for machine learning because they are easy to understand and interpret, and they can handle both continuous and categorical data. They are also

relatively efficient and fast to train, making them a good choice for large datasets. One key aspect of decision tree classifiers is the ability to handle missing values in the data. When training a decision tree, the algorithm can handle missing values by simply treating them as another possible feature value, and it will create branches in the tree to account for them. This is a useful property, as real-world datasets often have missing or incomplete data. One potential issue is that they can be prone to overfitting, especially if the tree is allowed to grow too deep. Overfitting occurs when the model is too closely tied to the training data, and as a result, it may not generalize well to unseen data. To address this issue, it is often necessary to prune the tree or use other techniques to prevent overfitting.

4.3.3 Random Forest

Random forest classifiers are a type of ensemble machine-learning algorithm that can be used for both classification and regression tasks. They are called "random forests" because they are made up of a large number of decision trees, each trained on a different subset of the data, and the predictions of the individual trees are combined to make the final prediction. The basic idea behind random forests is to build a large number of decision trees, each trained on a different subset of the data, and then average the predictions of the individual trees to make the final prediction. This process helps to reduce the variance of the model, and as a result, random forests are often more accurate than individual decision trees. There are several algorithms that can be used to build random forests, including Breiman's original algorithm and the randomForest package in R. These algorithms differ in the way the decision trees are trained and the features are selected, but they all follow a similar overall process. To train a random forest classifier, the algorithm first selects a random subset of the data and a random subset of the features to use at each node of the decision tree. It then builds the decision tree using these subsets of the data and features. This process is repeated multiple times, and the resulting decision trees are combined to form a random forest. One key advantage of random forest classifiers is their ability to handle a large number of features and handle missing values in the data. They are also relatively fast to train and are resistant to overfitting, which makes them a good choice for large and complex datasets. Despite their many advantages, random forest classifiers are not without their limitations. One potential issue is that they can be difficult to interpret, as the individual decision trees are combined to make the final prediction. This can make it challenging to understand how the model is making its predictions and identify any potential biases.

4.3.4 Naive Bayes

Naive Bayes classifiers are a type of supervised machine learning algorithm that can be used for both classification and regression tasks. They are called "naive" because they make the assumption that all of the features in the data are independent of each other, which is often not the case in real-world datasets. Despite this assumption, naive Bayes classifiers are often surprisingly effective and are widely used in a variety of applications.

The basic idea behind naive Bayes classifiers is to use Bayes' theorem to estimate the probability of an event based on the probabilities of related events. In the case of a classification task, the event is the class label (e.g. spam or not spam), and their related events are the features of the data (e.g. certain words or phrases in an email). The algorithm uses the training data to estimate the probabilities of each class and the probabilities of each feature given a particular class, and it combines these probabilities using Bayes' theorem to make the final prediction.

There are several algorithms that can be used to build naive Bayes classifiers, including the Gaussian naive Bayes, Bernoulli naive Bayes, and multinomial naive Bayes algorithms. These algorithms differ in the way they handle the features and make the predictions, but they all follow a similar overall process.

One key advantage of naive Bayes classifiers is their simplicity and efficiency. They are fast to train and relatively easy to implement, making them a good choice for large datasets. They are also robust to irrelevant features and are not sensitive to the scaling of the data, which can be a problem for some other machine-learning algorithms. Despite their many advantages, naive Bayes classifiers are not without their limitations. One potential issue is that they rely on the assumption that all of the features are independent, which is often not the case in real-world datasets. This can lead to less accurate predictions, especially if there are strong dependencies between the features. This algorithm takes in a training dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ and a test data point x, and it returns a predicted class $\hat{y}$. It first calculates the prior probabilities $P(y_i)$ for each class y_i in the training data and then calculates the likelihoods $P(x_j|y_i)$ for each feature x_j given each class y_i. Finally, it uses Bayes' theorem to calculate the posterior probability $P(y_i|x)$ for each class y_i given the test data x, and it returns the class with the highest posterior probability as the predicted class $\hat{y}$.

4.3.5 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a type of supervised machine learning algorithm that can be used for both classification and regression tasks. It is based on the idea of using the information from the closest neighbors to make a prediction. In the case of a classification task, the algorithm assigns the class label of the majority of the nearest neighbors to the test data point. In the case of a regression task, the algorithm takes the average of the values of the nearest neighbors to make the prediction. One key advantage of KNN is its simplicity and flexibility. It is easy to implement and does not require any training, as it simply uses the information from the nearest neighbors to make the prediction. KNN is also relatively robust to noise and can handle missing values in the data. To determine the number of nearest neighbors to consider, the algorithm uses a parameter called k, which is typically chosen through cross-validation. A larger value of k will smooth the decision boundary and reduce the variance of the model, but it may also increase the bias. A smaller value of k will make the model more sensitive to the local structure of the data and may increase the variance, but it may also decrease the bias. Despite their many advantages, KNN classifiers are not without their limitations. One potential issue is that they can be computationally intensive to make predictions, especially on large datasets, as the algorithm must consider the information from all of the data points in the dataset. KNN is also sensitive to the scale of the features, and it is generally recommended to scale the features before training the model.

Part II

Contributions

CHAPTER 5

PRACTICAL AND USABLE COERCION-RESISTANT REMOTE E-VOTING

In this chapter, we delve into the practical and usable aspects of coercion-resistant remote e-voting systems, with a primary focus on the JCJ protocol [78, 37]. We begin by providing an introduction to the JCJ protocol and its main structure, including the functions and the coercion-resistant election protocol framework. Subsequently, we discuss the usability aspects of e-voting schemes and evaluate their effectiveness. In order to demonstrate the importance of designing secure e-voting systems, we present an overview of the NV12 scheme [91], highlighting the attacks and security vulnerabilities identified in previous works. We then provide detailed information about the protocol, including electronic voting protocols based on the Paillier [99] and BGN [25] encryption schemes respectively. Lastly, we discuss the concept of PIN space covering as an essential component in the process of achieving coercion-resistance in the PIN-based system we design here. The chapter concludes with a summary of the findings and a brief outlook on potential future research directions in the field of coercion-resistant remote e-voting.

5.1 Introduction

The absence of a voting booth in remote electronic voting makes it highly vulnerable to coercion-attacks, which is a major concern. In a seminal paper by Juels, Catalano, and Jakobsson [78], they provided a formal definition of coercion-resistance and developed a protocol (JCJ) that meets this security requirement. The JCJ protocol relies on a coercion-free setup phase, wherein the voter receives a credential, essentially a cryptographic key. To cast a valid ballot, the voter must accurately enter this key along with their vote. In the event of coercion, the voter has the option to provide the coercer with a fake random credential and even vote alongside the coercer using this false credential. However, during the tallying process, the corresponding vote associated with the fake credential will be disregarded.

However, the tally process of weeding out the ballots with fake credentials and duplicates suffers from a quadratic complexity problem in the number of voters and cast ballots. Several papers are devoted to reducing the tally complexity in JCJ, see e.g. [107, 62, 118]; however, each with its drawbacks. Moreover, JCJ and similar constructions- we consider them under the term JCJ-type-protocols, however also suffer from usability deficits, see also [90].

Moreover, the handling and storing of long credentials is a notorious usability problem, getting even harder with a coercer present. The usability was analyzed by Neumann et. al. [92], which led to a protocol using smart cards for handling voter's credentials. The stored credential is combined with a PIN code to produce the full credential, compared with the credential stored by the authorities on the bulletin board. In [51], we revisit this protocol and present several attacks on coercion-resistance and verifiability, but also possible repairs.

Whereas the smart card provides a solution to the usability problem, it also comes with strong trust assumptions and problems such as:

- Their security model assumes that the smart card generally needs to be trusted. A malicious card could e.g. use the wrong credential invalidating the cast ballot without detection, and we cannot let the voter check if the ballot is correct without introducing coercion threats.
- The coercer can take the smart card away from the voter to force abstention.
- It is more expensive, less flexible, and harder to update than a pure software solution.
- One of the attacks we found is that a coercer can use the smart card to cast ballots. This not only endangers coerced voter's real vote, but due to a leak of information in the weeding phase, the coercer can also detect, with non-negligible probability, whether the coerced voter has cast an independent ballot against his instructions.

We will present protocols that repair or at least diminish the attack probability of, the last point's attack probability by constructing new duplicate removal methods in JCJ. Furthermore, the protocols constructed in this chapter are hardware-independent: they could use a smart card or be implemented using a combination of a digitally stored cryptographic length key and a PIN only known by the voter. The long credential could be stored in several places – or hidden via steganography.

At ballot casting time, the software will take as input the digital key and the password to form the credential submitted with the vote. Depending on the level of coercion, the coerced voter can either fake the long credential or, for stronger levels of coercion, the voter can reveal the digitally stored credential to the coercer but fake the PIN. Due to our improved tally, the coercer will not know if he got fake credentials or PINs.

Another major problem with the original construction, already discussed as an open problem in [92], is the high chance of users making a PIN typo error which will invalidate the vote and remain undetected. Note that naively giving feedback on the correctness of the PIN is not possible for coercion-resistance as it would allow the coercer to check whether he got a fake PIN or not. Instead, we will define a set of allowed PIN errors (e.g. chosen by the election administrator), and we will consider a ballot as valid both if it has a correct PIN or an allowed PIN error, but invalid for other PINs. We construct protocols that at the tally time secretly check whether a given PIN is in the set of allowed PINs and will sort out invalid ballots. The protocols can accommodate general PIN error policies, however, Wiseman et. al. [126] studied usual errors in PIN entries. Two frequent errors are transposition errors (i.e. entering "2134" instead of "1234") and wrong digit number errors (i.e. entering "1235" instead of "1234"). However, correcting for both of these errors is however problematic, as we will see since the set of independent PINs becomes small.

5.2 Usability aspects of E-Voting Schemes

The importance of usability in voting systems is often overlooked in addition to the critical issues of accuracy and security. To ensure the integrity of elections, it is essential that voters are able to cast their ballots as intended. Incidents of unintentional undervotes, overvotes, or votes for incorrect candidates can significantly impact the outcome of elections, as demonstrated in the 2000 Florida election upset. According to [124], the butterfly ballot used in Palm Beach County, Florida resulted in over 2000 votes being cast for Pat Buchanan rather than Al Gore, ultimately leading to George W. Bush's victory. Therefore, it is important that voting systems prioritize usability to ensure that voters are able to cast their ballots accurately and efficiently. Poor ballot design, confusing instructions, and other usability issues can lead to mistakes, mistrust in the election's results, and long lines at the polls. By addressing usability issues, we can improve the overall experience of voting for both voters and election officials. On the other hand, a positive voting experience can encourage individuals to participate in future elections. Good usability in voting systems can enhance voters' confidence in the election's outcome and make the process of voting a more enjoyable experience. This can lead to increased voter turnout and ultimately strengthen the democratic process. In conclusion, usability is a critical aspect of voting systems that should not be overlooked. Ensuring that voting systems are easy to understand and use is essential for maintaining the integrity of elections, encouraging voter participation, and fostering trust in the democratic process. By considering usability in the design and implementation of voting systems, we can create systems that are not only secure and accurate but also user-friendly and accessible to all voters.

Based on ISO 9421-11, In order for a voting system to be considered usable, it must allow users to complete their task of casting a vote effectively, efficiently, and with satisfaction. This means that the system must be easy to understand and navigate, allow voters to cast their votes without errors or confusion, and provide a positive experience for the user. Usability is crucial for voting systems because it ensures that all eligible voters are able to cast their ballots and have confidence in the election process. Poor usability can lead to confusion, frustration, and ultimately lower voter participation. It is therefore important for voting systems to be designed with usability in mind, in order to ensure that elections are fair, accurate, and representative of the will of the people. For instance, authors in [90] found bad usability in JCJ since the voter need to handle long crypto-size credential and be able to fake them.

5.3 Overview of the NV12 Scheme

The NV12 scheme [92] is an improvement on the proposed scheme by Shirazi et al in their work [114]. The NV12 scheme addresses some of the abstract assumptions and credential management abstractions of the JCJ/Civitas system through the use of smart cards, as proposed by Mendes in [116] and Neumann and Volkamer in [28]. In the NV12 scheme, several entities are involved: A supervisor who is in charge of running the election and declaring election authorities; the voter who intends to cast her vote; the voter's smart card that serves as a trusted device between the voter and the JCJ / Civitas system; a registrar who administrates the electoral register; a supervised registration authority and a set of registration tellers that provide the voter with her credential; a set of tabulation tellers that are in charge of the tallying process; a set of ballot boxes to which voters cast their votes; and a bulletin board that is used to publish information. This part of the work summarizes the NV12 approach and includes a modification to address side-channel threats. NV12 generally follows JCJ but with some tweaks for usability. The main idea is to store JCJ credentials on a smartcard, but adding a PIN such that using a wrong PIN will result in a fake credential.

Setup Phase. In the NV12 scheme, the administrator organizes the election and makes information about the ballot layout available. The registrar releases the electoral register along with the public keys of the voters. The tabulation tellers construct the election key pair in a distributed manner and broadcast the accompanying public key pk_{EK}. After that, each registration teller creates random private credential shares for all eligible voters. They then encrypt these private credential shares using the public election key, resulting in public credential shares, which they publish along with the voter's record in the published electoral register. In more technical terms, the registration teller i publishes $S_i = \{\sigma_{RT_i}\}^r_{pk_{EK}}$ for a particular voter, where σ_{RT_i} is the voter's credential share.

Registration Phase. The NV12 scheme has a different registration process compared to the original JCJ/Civitas plan. It involves an offline phase, during which a voter visits a supervised registration authority (SRA) in person. The SRA ensures that the voter is not being coerced and asks them to insert their smart device (a smart card) into the reader. The voter is then asked to create a personal identification number (PIN) for their smart card. After the voter creates their PIN, the supervised registration authority saves the voter's private credential share σ^v_{SRA} from the secure group on their smart card. Along with the private credential share, the authority generates a designated verifier re-encryption proof (DVRP) that convinces the voter's smart card that the public credential share published on the bulletin board during the setup phase is a re-encryption of the voter's private credential share. The smart card then calculates the coefficient c_p that will be used to transfer the PIN (p) to the shared private credential (σ^v_{SRA}), using the equation $c_p = \frac{\sigma^v_{SRA}}{p}$. These calculations are performed in a finite field of prime order, with the selected order being n-times ($n \geq 1$) as large as the order of the set containing the private credential. The smart card stores the calculated coefficient during this phase. The voter then exits the supervised registration authority, completing the offline registration process. During the online phase of the voting process, the voter begins by connecting to the election website and completing the registration process. As part of this process, the voter is asked to select their preferred registration agents from a list of available options. This selection is then sent to the voter's smart card, which prompts the voter to confirm the selection through their smart card reader. Once this confirmation has been received, the smart card is preloaded with the IDs of the selected, trusted registration agents (denoted as TRT for the voter v). The smart card then establishes secure connections with these trusted registration agents via the voter's computer and obtains private credential shares $\sigma^v_{RT_i}$, as well as encrypted versions of these shares (denoted as $S'_i = \{\sigma_{RT_i}\}^{r'}_{pk_{EK}}$). The card also receives DVRPs (digital verifiable random proofs) from each individual agent, which serve to confirm that the S'_i and S_i contain the same message. After obtaining all of the necessary private credential shares and validating the DVRPs, the voter's smart card computes and saves the voter's credential factor. This information is then used to securely cast the voter's ballot during the voting process. Also, note that the encryption versions on the bulletin board are added homomorphically under encryption.

$$c_{factor} = c_p \times \prod_{i \in TRT(v)} \sigma_{RT_i}$$

Voting Phase. After completing the registration process, the voter can begin the voting process by accessing the election website and viewing the available options. The voter makes their selection using the provided JavaScript interface. Once the voter has made their selection, it is sent to their smart card for encryption. To ensure the integrity of the process, the voter has the option to audit the JavaScript using a "cut-and-choose" method, such as the Benaloh challenge (as described in NV12 [14]). This allows the voter to verify that their selection was conveyed correctly. Once the voter has confirmed the validity of their selection, they are prompted to enter their voting PIN into the smart card reader to finalize their vote. The smart card then multiplies this PIN by the credential factor generated during the registration process (which was saved on the card). This computation is performed within a finite field of prime order,

as previously mentioned. If the voter correctly enters their PIN, the card generates a true credential that is associated with the voter's vote. If the voter enters an incorrect PIN, a randomly generated (invalid) credential is used instead. The resulting credential is as follows:

$$c = \text{PIN} \times c_{factor}$$

This process helps to ensure the security of the voting process by allowing the voter to enter their PIN correctly while protecting against side-channel attacks. Specifically, the voter's smart card generates a ballot with the following structure:

$$\langle \{\text{c}\}_{pk_{EK}}, \{\text{vote}\}_{pk_{EK}}, \sigma, \phi \rangle$$

Notations $\{c\}_{pkEK}$ and $\{vote\}_{pkEK}$ refer to private credentials, which are both encrypted using a public election key. The term σ is a well-formedness proof that indicates that the encrypted vote $\{vote\}_{pk_{EK}}$ includes a valid choice, while ϕ is a zero-knowledge proof demonstrating that the submitter knows both c and $vote$ in order to prevent replay attacks. The voter's smart card uses these elements to compute and output the hash value

$$\text{hash}(\langle \{c\} pk_{EK}, \{vote\}_{pk_{EK}}, \sigma, \phi \rangle)$$

on the smart card reader. It is important to note that, while the use of a smart card for this process may provide a more realistic implementation in comparison to JCJ/Civitas systems, the NV12 method does not provide any proof of integrity once the voter has entered their PIN. This is a practical issue that must be addressed, as humans are known to make mistakes such as mistyping or forgetting their PINs and passwords (as noted in [54]) and this is what we will improve on below. After computing the hash value, the smart card anonymously distributes the prepared ballot to all available ballot boxes. Each ballot box computes the hash value of the received ballot and posts it on a bulletin board upon receipt.

Tallying Phase. During the counting phase, all tabulation tellers collect the ballots from all ballot boxes, as well as the public credentials that have been posted on the bulletin board. They then verify the zero-knowledge proofs, eliminate any duplicate votes (which may be the result of vote altering), and identify any illegitimate votes (which may be the result of forged credentials). Finally, the encrypted credentials of the remaining ballots are discarded and the corresponding encrypted votes are decrypted in a distributed manner (See [78]). Each step of the tabulation process can be independently verified using a series of zero-knowledge proofs. This helps to ensure the accuracy and integrity of the vote count.

5.3.1 Attacks

In [51] we demonstrate many attacks and issues with the protocol's security, most notably a coercion-resistance attack caused by information leakage caused by the removal of duplicate votes. Moreover, they discuss how to repair these issues. These shortcomings include the Benaloh challenge problem, Brute force attack, Leaky duplicate removal, Fake election identifier, and Smart card removal. Finally, to motivate our protocol in this section, we describe these attacks that our protocol will repair.

In this part we are presenting the attacks that we found in [51]. In order to ensure the security and integrity of an electronic voting system, it is essential to identify potential vulnerabilities and implement appropriate measures to protect against potential attacks. In the following, we will examine some of the potential attacks that may be encountered in this voting scheme and discuss ways to safeguard against them.

Benaloh challenge problem: The described attack focuses on exploiting a potential weakness in the NV12 voting scheme with regard to the Benaloh challenge, which is designed to provide individual verifiability. The problem lies in the fact that the hash of the encrypted vote, committed to by the smart card and reader, is not checked for the cast ballot. Instead, what is checked is the hash of the entire ballot, which includes the encryption of the credential and zero-knowledge proofs (ZKPs).

Let's expand on this attack strategy:

The smart card, in this scenario, acts dishonestly. During the initial stages of the voting process, it behaves properly, encrypting all votes as expected. This creates the illusion of integrity and honesty because the Benaloh challenge at this stage will be passed. The smart card reader's screen will display the correct hash of the encrypted vote, and the voter will be under the impression that their vote has been correctly recorded.

However, once the Personal Identification Number (PIN) is entered to cast the ballot, the smart card switches its strategy. It encrypts a different vote choice—potentially one that wasn't chosen by the voter—and includes this in the ballot.

The crucial part of this attack is that it goes undetected even in an honest verification environment. This is because the verification process doesn't check the hash of the individual encrypted vote, but rather checks the hash of the entire ballot, which includes the new dishonestly encrypted vote, the credential, and the ZKPs. If the dishonest smart card maintains the integrity of the rest of the ballot, the hash check will pass, and the altered vote will be accepted as part of a valid ballot. This effectively violates the trust assumption, undermining the individual verifiability of the voting process.

Repair: In order to properly verify the integrity of the vote, it is necessary to hash of the vote encryption committed in the Benaloh challenge and compare both the hash of the vote encryption and the hash of the entire ballot to values computed from the ballot received by the ballot box. This can be a usability issue, as it requires the voter to verify two different hashes, which may not be straightforward. For example, an adversary may precompute hashes that are difficult for the voter to identify, such as matching on the leading component.

Another option is to commit to the entire ballot in the Benaloh challenge process, but this would require the voter to input their PIN for each challenge. Given that it is common for legitimate voters to rarely perform verification checks in electronic voting systems, requiring a PIN for each challenge may further compromise the security of the Benaloh challenge. Additionally, a voter is typically only able to challenge once, so an adversary could potentially cheat after the first challenge.

Attack using brute force: Coercion-resistance and verifiability may be compromised if an adversary is able to gain access to the voter's smart card, either by requesting access directly or by stealthily accessing the card. In these cases, the adversary could potentially guess the voter's PIN and cast a vote on their behalf. Due to the anonymity of the voting process, the voter would not be aware of this attack and would not be able to detect it. Unfortunately, it is not possible to simply increase the size of the PIN space to protect against this type of attack, as there is a limit to the voter's ability to remember and accurately input a longer PIN. This means that it is important to find other ways to protect against such attacks, such as implementing additional security measures or implementing a system that allows the voter to verify that their vote was cast as intended. The probability of an adversary successfully guessing the voter's PIN is not be negligible, especially if the adversary is able to vote multiple times. Additionally, even if the voter is using a properly designed smart card reader, it is possible for an adversary to create a malicious reader that could be used to automate the ballot-casting process and increase the probability of a successful attack. This type of attack is not practical, however, as it would take a significant amount of time to fill the PIN space through brute force guessing. For example, according to [52], the process of voting typically takes around 13 seconds, including network delay. While the theoretical value for voting with current

smart cards is likely to be lower, it would still take a significant amount of time to fill the PIN space through brute-force guessing. It is also worth noting that the success of this type of attack depends on the vote update policy in place and the timing of the vote. An adversary may be able to optimize their strategy by casting their vote last, for example.

Repair: One way to combat the risk of brute force attacks is to add a short pause or delay between each voting attempt. This delay can slow down an attacker trying to guess the voter's PIN, as it would take a lot longer to try all possible combinations.

However, we have to be careful about the length of this delay. If it's too long, someone who's trying to force a voter to abstain might notice it. On the other hand, if it's too short, it may not slow down a brute-force attack enough to make a difference.

So, it's a bit of a balancing act to find just the right amount of delay. It has to be long enough to slow down an attacker, but not so long that it would be noticed by someone trying to force a voter to abstain or make voting inconvenient for the legitimate voter. We need to experiment with different lengths of delays and test their effectiveness to find the optimal solution.

Leaky duplicate removal: This is an attack on coercion-resistance but can also be an attack on verifiability. In the simplest form, the coercer uses the smart card to cast a vote with some trial PIN. The coercer wants to determine if this trial PIN is correct. According to the protocol, the voter will cast her true vote using the correct PIN at some secret point during the voting phase. However, in the tally phase, credentials are weeded using plaintext equivalence tests (PETs) of the encrypted credentials directly on the submitted ballots. If the coercer now sees an equivalence with his submitted trial ballot, he can guess that it was the voter casting the other ballot, probably with the correct PIN. Thus he has determined the correct PIN and that the voter defied his instructions in one go. To boost the attack he can simply try several PINs. In standard JCJ, such an attack would not work since the submitted trial credential would have the same probability of being identical to the coerced voter's credential as for it to be identical to any other voter's credential. Furthermore, the probability would be negligible.

A local adversary getting access to the smart card could also follow this strategy to try to know the PIN and cast valid votes. The voter might actually detect this if the voter checks the weeding on the bulletin board and sees a duplicate of his own vote (note this was also mentioned in [105]), but the voter is not instructed to do this in the protocol. Thus the PIN is not protecting against unauthorized use of the smart card.

It is actually surprisingly hard to make a tally protocol that does not leak information to prevent this attack. The original JCJ protocol relies on guessing the real full credential can only happen with a negligible chance. A first repair could be to mix the ballots before weeding but after verifying the Zero-knowledge proofs. This makes it difficult to implement certain policies, like the last valid vote counts; however, it fits nicely with the policy that a random selection from the valid votes counts. Unfortunately, this does not prevent the attack. The coercer could mark his ballot by casting it a certain number of times which is likely to be unique. He then checks if he sees this number of duplicates or one more. Even if mix between each duplicate removal, which would be horrible from an efficiency perspective, we do not get a leak-free tally. The distribution of time until a PET reveals a duplicate will depend on whether the PIN was correct or not. Especially the coercer could cast a lot of votes with the same trial PIN, which would make detecting this more visible. There are other methods to limit the information leak in the tally which we will present below.

The protocol we will present in this chapter does not leak information about the number of duplicates per voter and has linear tally complexity (compared to the quadratic in JCJ) but has an obfuscated form of participation privacy.

Repair: It can be challenging to design a tally protocol that does not leak information and is able to protect against this type of attack. The original JCJ methodology relies on the assumption that it is unlikely that an adversary will be able to guess the true complete credential. One potential measure that could be implemented to address this issue is to mix the ballots prior to weeding, but after confirming the zero-knowledge proofs (ZKPs). This would make it more difficult for an adversary to determine which ballots are equivalent and deduce the voter's actual vote. However, this approach has the potential to complicate the implementation of certain rules, such as the last legitimate vote count. Despite these efforts, it is still possible for an adversary to attempt this type of attack and potentially deduce the voter's actual vote. To further protect against this type of attack, the coercer may mark their ballot by casting it a specific number of times, which is likely to be unique. They can then check to see if there are any other duplicate ballots and thus learn if the trial PIN was correct. While mixing the ballots between each duplicate removal may help to complicate the process, it is not a foolproof solution and may not be sufficient to prevent information leakage in the tally. One potential issue with this approach is that the time interval between plaintext equivalency tests (PETs) revealing a duplicate may vary depending on whether the PIN was accurate or not. If the coercer uses the same trial PIN to cast a large number of votes, it may be easier to identify this and deduce the voter's actual vote. There may be other techniques that can be implemented to limit information leakage in the tally. We are presenting one based on multi-party computation below, but where voter participation is only obfuscated. It may also be possible to develop a technique that does not disclose the number of duplicates per voter, has a linear tally complexity (as opposed to the quadratic complexity in JCJ), and provides an obfuscated form of participation privacy. These methods could help to further strengthen the security of the voting system and protect against attacks.

Fake election identifier: This attack targets the verifiability of the voting system by attempting to present the voter with an incorrect election identifier. As stated in the original JCJ article, zero-knowledge proofs must have a unique election identifier to prevent ballots from being copied between elections. If an incorrect identifier is used, the proofs will not be validated. However, before voting, the voter's smart card must be updated with the correct election identifier. If an adversary is able to manipulate the voter's computer and present them with the wrong credential, it could potentially compromise the integrity of the voting process.

Repair: To protect against this type of attack, it is important to ensure that the correct election identifier is properly transmitted to the voter's smart card and that the voter is able to verify its accuracy. This may involve implementing additional security measures, such as encryption or authentication, to protect against tampering or manipulation. One potential solution to this issue is for the voter to manually enter the election identifier to ensure its accuracy. However, this approach may be prone to mistakes, as the voter may accidentally input the wrong identifier. An alternative approach is for the voter to verify that the submitted ballot contains a zero-knowledge proof (ZKP) that certifies the ballot's authenticity in relation to the genuine election identity. This can be done when the hash of the entire ballot is verified, but it may require the voter to wait a little longer in order to complete the process. By implementing this additional verification step, it may be possible to increase the security of the voting system and protect against attacks that seek to compromise the integrity of the voting process. It is important to carefully consider the trade-offs between security and usability when implementing these types of measures to ensure that they do not unduly burden the voter or undermine the voting process.

Removing the smart card: An obvious forced abstention attack is when the coercer simply demands to hold the smart card during the election period. This problem seems quite inherent to the smart card approach. We could let the voter hold several smart cards. However, holding several cards would be physical evidence that a voter with a local coercer probably would not want to risk. Furthermore, the number of cards allowed per voter could necessarily not be bounded. If each voter were allowed to hold

e.g. 5 cards, the coercer would simply ask for five cards. If this is troublesome it seems better to leave the smart card only approach and allow the voter to also hold the credential as a piece of data as in standard JCJ. This can more easily be hidden (steganography could be an option here) even though theoretically this also has problems [113]. Our protocols below can be implemented with or without smart cards.

Additionally to the above attacks, some problems with the protocol do not fall under the category of attacks. The main usability and verifiability problem with the protocol is that PIN entry is error-prone, as was already stressed in the papers by Neumann et al. An obvious solution is to have a PIN check, e.g. a checksum check. However, this would mean that only certain PINs are valid PINs, and for a voter to present a fake PIN to a coercer, she would first have to prepare a valid fake PIN, which is less usable.

An option with higher usability is to have a policy of allowing PIN errors and accepting full credentials corresponding to the PIN being entered with allowed errors. This is the approach we will essentially follow in this paper; however, our solutions will also work for checksum checks.

If JCJ had a method of verifying the cast votes, we would also be able to detect such PIN errors. Such a verification mechanism was suggested in [74] using the Selene approach. However, this check can only be made after vote casting has ended; thus too late to update a PIN typo.

Another problem is the assumption that the smart card is trustworthy. This does not seem like a valid assumption, at least for important elections. The smart card could simply use the wrong credential in a ballot, invalidating the vote. Further, this cannot be detected since the smart card is the only credential holder. At least the PIN encryption could be Benaloh tested, but not the credential. Furthermore, the smart card reader is also trusted. However, this might not be reasonable in practice. As an example, if the middleware on the reader allows the voter's computer or the network to display messages on the screen, e.g. to say it is waiting for a connection, then it could e.g. try to display fake hash values. A corrupted smart card could also easily break privacy by using encryption as a subliminal channel for vote choice. The smart card can also be seen as *a single point of failure* in light of this. We will thus focus on hardware-independent protocols.

Repair: The utilization of smart cards in the voting process introduces challenges in safeguarding against coercion and other forms of interference. Allowing voters to possess multiple smart cards could potentially address this issue; however, practical difficulties arise, such as secure storage and management of multiple cards. Alternatively, it may be necessary to abandon the reliance on smart cards altogether and enable voters to possess their credentials as digital data, similar to the conventional JCJ method. This approach offers greater flexibility in countering coercion, as the credentials can be more easily concealed using techniques like steganography [113]. Nonetheless, this alternative approach brings its own set of challenges, such as ensuring the security and integrity of the credentials to prevent tampering or unauthorized access.

5.3.2 Security Vulnerabilities

There are several vulnerabilities in the protocol that are not considered attacks. One of the main issues with the protocol is that it is prone to errors when it comes to inputting the PIN. Neumann et al. have noted this issue in their articles. One solution to this problem is to include a PIN check, such as a checksum, to ensure that only certain PINs are valid. This would make it more difficult for a voter to give a fake PIN to a coercer, as they would have to create a valid fake PIN first. A more user-friendly solution is to implement a policy that allows for PIN errors and accepts complete credentials that match the allowed PIN errors. This is the approach we will take in this chapter, though our solutions will also be applicable to checksum checks.

If JCJ had a mechanism for verifying cast votes, we could also identify such PIN issues. Iovino et al. proposed a method for doing this using the Selene approach in their article [74]. However, this check can

only be performed after voting has ended, so it is too late to correct a PIN error. Another major concern with the protocol is the reliability of the smart card. In large elections, it may not be reasonable to assume that the smart card will always function properly. The smart card may accidentally include the wrong credential in the ballot, rendering the vote invalid. This is difficult to detect because the smart card is the only entity with access to the credential.

While the encryption of the PIN can be checked using the Benaloh challenge, the credential itself cannot be checked in this way. Additionally, the trustworthiness of the smart card reader is assumed. However, this may not be sufficient in practice. For example, if the reader's middleware allows the voter's computer or network to display messages on the screen, such as indicating that it is waiting for a connection, the voter's computer or network may attempt to display fraudulent hash values. Additionally, a compromised smart card may compromise privacy by using the encryption option as a covert channel for transmitting the vote selection. As a result, the smart card may be seen as a single point of failure. Therefore, we will focus on hardware-independent protocols.

5.4 Pin-based JCJ Protocol

To overcome the unsatisfying state-of-affairs described before in [51] we propose a practical and usable e-voting scheme, **PIN-JCJ** that satisfies verifiability, and coercion-resistance properties.

Compared to the previous electronic voting technique, our protocol has two significant innovations. First, it is a hardware-independent protocol, as we replace the smart card with the voter's supporting device, which can be easily duplicated. Second, we construct it in such a way that some human error is tolerated.

5.4.1 The intuition behind the PIN

In a nutshell, in our voting scenario, the voter has two keys: a long key (a.k.a. long credential) which is stored on her supporting device (smart card or another device), and a short key, namely PIN, which should be memorized by the voter. Our goal is to design a voting protocol that tolerates some level of human error. In other words, the validity of a ballot and the result of the decryption algorithm tolerate human errors, such as typos (fat finger errors) regarding the PIN. The naive solution to do this is for each PIN, "a" we generate an $\mathsf{ErrorList}_a$, and in the tally phase, to verify the legitimacy of the ballot, check whether the input PIN is in the set of error lists or not.

$$\mathbf{pin} = a: \ \mathsf{ErrorList}_a = \{a_1 = a, a_2, \ldots, a_k\}, \left|a\right| = \left|a_i\right| \tag{5.1}$$

This method has several significant privacy and efficiency issues, and to overcome these drawbacks, we propose the following new approach:

Our approach is based on polynomial evaluation, which allows us to determine whether or not a PIN is legitimate efficiently. This is accomplished by generating a list of approved PINs based on the user's PIN a and the election policy. From now on, we refer to this list as an ***Error List***. We emphasize that to have a constant degree polynomial for all voters, the error list must have the same number of PINs, which might contain duplicates. From this, we generate the ***pin-Polynomial***, as follows, which has all $\mathsf{ErrorList}_a$ members as its root:

$$\text{pin-polynomial:} \ \mathsf{poly}_{\mathbf{pin}}(x) = \prod_{i=1}^{k}(x - a_i) = \sum_{i=0}^{k} p_i x^i \tag{5.2}$$

In order to check the validity of the PIN typed by the voter, it is then sufficient to check whether the polynomial value on this **pin** is equal to zero or not.

It is obvious that this polynomial must be kept secret at all times to prevent the coercer from recovering the PIN by factorizing it. As a result, we must operate with encrypted polynomials, which brings us to our next challenge: polynomial evaluation under this encryption. Namely, given the polynomial encryption as its encrypted coefficient,

$$\mathsf{poly}_{\mathbf{pin}}(x) = \sum_{i=0}^{k} p_i x^i \Rightarrow \mathsf{Enc}(\mathsf{poly}_{\mathbf{pin}})(x) = \sum_{i=0}^{k} \mathsf{cp}_i x^i,$$

as well as a ciphertext $\mathsf{CT}_{\mathbf{pin}} = \mathsf{Enc}(\hat{a})$ that is the encryption of the entered PIN, we need to compute $\mathsf{Enc}(\mathsf{poly}_{\mathbf{pin}}(\hat{a}))$.

Therefore in the next step, we have to find a way to prove (publicly) that the individual voter's polynomial is correctly evaluated without endangering the coercion-resistance.

Further, while solving this problem, we will also focus on efficient protocols to obtain a practical JCJ scheme with (almost) linear tally time in the number of voters. To obtain this we need to sacrifice perfect privacy. We only have participation privacy in the first scheme by obfuscation inspired by [61]. Here ballots are submitted with an ID, and homomorphic Paillier encryption can then be used to evaluate the polynomial. However, everybody, e.g. an independent authority, can cast votes labeled with ID, which will later be discarded as invalid. Thus the actual participation of the voter is obfuscated, and the voter can deny having participated in the election. Optionally, we could also follow the JCJ alternative method in [61] to achieve perfect privacy; however, the cost will be that the voters twice have to defy the coercer and interact with the voting system. In the second scheme using BGN encryption, the information leak from duplicate removal will not be negligible but bounded, and this scheme does not satisfy linear tally efficiency.

The next section will present the protocol description by introducing the cryptographic building blocks and their algorithm. Also, for simplicity, we describe the protocol with a single trusted party, but it is possible to run this protocol distributively. We will also not specify all parts of the distributed registration phase and the Benaloh challenges; this can be implemented as in the NV12 scheme with some obvious modifications and the repairs presented in [51].

5.5 Protocol Description; Participants, Primitives and Framework

We now present the **PIN-JCJ** e-voting protocol on the conceptual level. In sections 5.6 and 5.7, we will then instantiate our protocol by introducing concrete cryptographic primitives to demonstrate the efficiency and usability of the protocol.

5.5.1 Protocol Participants

The **PIN-JCJ** protocol is run among the following participants: *Election authority* in charge of running elections and declaring elections. *Election trustees* are in charge of the tallying process, and they are the only parties in possession of the election secret keys. n_{v} *voters*, each having their own *voting supporting device*,vsd. A set of *supervised registrars* administrate the electoral register and provide the voters with their credentials. A set of mix-servers (mix-net) and an append-only bulletin board $\mathfrak{BB}$.

We also presume that all parties communicate with one another via authenticated channels. An authenticated channel from each voter to the bulletin board allows the voter to post data on the bulletin

board, for example, cast her ballot, or file a complaint. To describe the **PIN-JCJ** protocol, we will first go through the cryptographic primitives that are employed in the scheme. We separate two sets of primitives for instantiation: those that use standard public encryption schemes and those that use the public functional encryption scheme.

5.5.2 Cryptographic Primitives

We use the following cryptographic primitives:

- **An IND-CPA-secure public-key encryption scheme:** $\Pi^{\mathsf{pke}} = (\mathsf{Kgen}, \mathsf{Enc}, \mathsf{Dec})$. Our protocol requires performing a polynomial evaluation on encrypted data; namely, the underlying public-key cryptosystem must support re-encryption. To this end, we consider three approaches: A public-key encryption scheme that allows multiplication on the ciphertext or a homomorphic public-key encryption scheme, or a functional encryption scheme. We detail the instantiation of the protocol with these two cryptosystems in sections 5.6 and 5.7. However, for the time being, we are only considering a public-key cryptosystem with IND-CPA-secure property as the protocol's security and privacy are dependent upon it.

  Additional Note. As mentioned above, the underlying public-key cryptosystem needs to have an extra algorithm, as a re-encryption algorithm. On the other hand, for security analysis, we require the cryptosystem to be non-malleable, which contradicts the re-encryption requirement. We employ a public-key cryptosystem with a re-encryption algorithm to address this issue, w, but we require that each ciphertext contain a zero-knowledge proof of knowledge. We refer to [21] for more detail on the CCA-secure cryptosystem for e-voting protocol.

- **Non-interactive zero-knowledge proof:** For the protocol's verifiability, we use a non-interactive zero-knowledge proof system to prove the correctness and plaintext knowledge for all encrypted data, such as "Proof of Plaintext Knowledge" and "Proof of Decryption Validity." We may utilize either a zero-knowledge proof system or a witness-indistinguishable proof system for this aim. Formally we will use the following proofs in our protocol:

  1. A *proof of correct key generation*, i.e., a *non-interactive zero-knowledge proof* (NIZKP) π_{Kgen} for proving the correctness of a public key pk w.r.t. Π^{pke}. The underlying relation $\mathscr{R}\mathsf{KeyGen}$ is

  $$\big(x = \mathsf{pk}, w = (\mathsf{random}, \mathsf{sk})\big) \in \mathscr{R}\mathsf{KeyGen} \Leftrightarrow (\mathsf{pk}, \mathsf{sk}) = \mathsf{Kgen}(\mathsf{random}). \tag{5.3}$$

  2. The *proof of correct encryption* π_{Enc}, i.e., a NIZK proof of knowledge for proving the correctness of a ciphertext ct w.r.t. election key pk, voter public key pk_{v}.
  3. A *proof of shuffle*, π_{Shuffle} with respect to the cryptosystem Π^{pke}, i.e., NIZK proof of knowledge for the following relation $\mathscr{R}\mathsf{shuffle}$:

  $$\begin{gathered}\big(x = ((\mathsf{ct}_i)_{i=1}^n, (\mathsf{ct}'_i)_{i=1}^n), w = ((\mathsf{r}_i)_{i=1}^n, \sigma)\big) \in \mathscr{R}\mathsf{shuffle} \\ \Updownarrow \\ \big(\sigma\colon [n] \to [n] \text{ bijective }\big) \wedge \big(\forall i \in [n]\colon \mathsf{ct}'_i = \mathsf{ReEnc}(\mathsf{pk}, \mathsf{ct}_{\sigma(i)}; \mathsf{r}_i)\big).\end{gathered} \tag{5.4}$$

  In other words, the public statement of proof of shuffle consists of two ciphertext vectors $(\mathsf{ct}_i)_{i=1}^n$ and $(\mathsf{ct}'_i)_{i=1}^n$ of the same size, and if the prover's output is valid, then this implies that the prover knows random coins $(\mathsf{r}_i)_{i=1}^n$ and a permutation σ over $[n]$ such that ct'_i is a re-encryption of $\mathsf{ct}_{\sigma(i)}$ (using randomness r_i).

4. A *proof of correct decryption* π_{Dec} with respect to Π^{pke}, i.e., a NIZK for the following relation $\mathscr{R}\mathsf{dec}$:

$$\begin{gathered}\big(x = (\mathsf{pk}, m, \mathsf{ct}), w = \mathsf{sk}\big) \in \mathrm{R}_{\mathrm{dec}} \\ \Updownarrow \\ \big(m = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})\big) \wedge (\mathsf{pk}, \mathsf{sk}) \leftarrow \Pi^{\mathsf{pke}}.\mathsf{Kgen}.\end{gathered} \tag{5.5}$$

   For all the above relations we can use either a zero-knowledge proof system or a witness-indistinguishable proof system for this purpose.

- **EUF-CMA-secure signature scheme:** The EUF-CMA-secure signature scheme, or Existential Unforgeability under Chosen Message Attacks, is a crucial security property for digital signature schemes. Mathematically, a signature scheme (Gen, Sign, Verify) is said to be EUF-CMA secure if for all probabilistic polynomial-time adversaries A, the probability $\Pr[\mathrm{Forge}(A, (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Verify}))]$ is negligible. Here, Forge denotes a successful forgery event, which is defined as follows:

  A forges a signature if after a learning phase, where A has access to a signing oracle that signs any message of A's choice, A can output a pair (m, σ) such that $\mathrm{Verify}(\mathsf{pk}, m, \sigma) = 1$ and m was not queried to the signing oracle during the learning phase.

  We assume that every message encrypted by a protocol participant contains some election parameters such as the election identifier, and to avoid the heavy notation, we use the following convention: Signing some message m implies that the signature is computed on the tuple $(m; \mathsf{pp})$ where the second components are public election parameters including an election identifier.

5.5.3 Protocol Framework

A protocol run consists of the following phases:

- **Setup Phase.** This procedure is carried out in the same manner as a standard e-voting protocol. The election authority sets up the election, establishes all the policies, such as the re-voting policy and the type of error that the protocol will tolerate, and publishes details about the ballot design and other procedures. For example, the re-vote policy specifies whether voters can cast only one ballot or many ballots. Furthermore, which ballot will be counted as the voter vote in the tally phase in the latter case?

  Then the election trustees distributively generate the election key through distributed threshold secret sharing and publish the public part of it on the bulletin board, together with proof of data correctness.

- **Registration Phase.** During this phase, the voter, v, personally consults a so-called supervised registrar (SR). This authority verifies that the voter is not being coerced directly and then generates a unique credential. This credential is then split into two parts: the long credential and the short credential, also referred to as the PIN.

  Following that, the supervised registration authority develops the PIN's error list(as in 5.1) according to the election policy and determines the pin polynomial (as defined in 5.2). Finally, the voter's long credential and encryption of the pin-polynomial are transmitted to (stored on) the voter's supporting device. Notably, the PIN is not stored in the voting device or on the bulletin board, and it is necessary to be memorized by the voter. Along with the voter credential, the registrar provides proof that satisfies only voter, v, that all associated data was generated correctly, whether published on the bulletin board or stored on the voter's device.

- **Voting Phase.** In this phase, the voter device asks the voter to enter her PIN, $\hat{a}$, and her preferred choice, vote. Then it generates the ballot of the form:

$$\begin{aligned} \mathsf{ballot} = (\mathsf{CT}_{\mathsf{crd}}, \mathsf{CT}_{\mathsf{vote}}, \pi_{\mathsf{ballot}}) : \\ \mathsf{CT}_{\mathsf{crd}} = \mathsf{Enc}(\mathsf{voterCredential}) = (\mathsf{CT}_1, \mathsf{CT}_2), \\ \mathsf{CT}_{\mathsf{vote}} = \mathsf{Enc}(\mathsf{vote}) \end{aligned} \tag{5.6}$$

  together with some election's public parameters, such as the election identifier. In the ballot form, CT_1 and CT_2 denote the encryption of the voter long credential and the pin she entered, with respect to the election public key, respectively. Moreover, π_{ballot} denotes the proof of the ciphertexts' well-formedness, which consists of proof of plaintext knowledge of both the credential and vote. In the next step, the voter can choose to either audit her ballot her vsd and conduct the *Benaloh Challenge* [11] procedure to ensure that her ballot has the correct vote and pin or to submit it.

  Device Auditing: If the voter wishes to audit option, the smart card commits to encryption of the ballot by displaying $\mathsf{H}(\mathsf{ballot}; \{\mathsf{random}\})$. The voter notes down this hash, and if the encryption is challenged, the smart card releases all the randomness numbers $\{\mathsf{random}\}$ to the voter's computer. The voter can verify the hash indeed was consistent with the vote choice via a third device. This challenging procedure can be reiterated.

  Ballot Casting: When voters get sufficient confidence in the honesty of their device, they sign the ballot 5.6 and submit it via an authentic anonymous channel to the bulletin board.

- **Tally phase.** This includes several steps. First, any exact duplicate ballots and ballots with incorrect proofs or signatures are removed. The teller should then detect duplicate ballots from a single voter, weeding the duplicates while ensuring the remaining ballot matches the valid pin (if existing). The weeding step is done according to the re-vote policy of the election. For example, suppose the re-vote policy specifies that each voter's last (according to the timeline) valid ballot must be counted in the tally phase. In that case, this step must ensure that the last valid ballot remains on the bulletin board and the rest are removed.

  The several mix-net servers carry out the following phase: each mix-net receives a list of ballots as input, re-encrypts each ballot, and then performs a secret permutation on the re-encrypted list to produce the new list. Finally, the decryption trustees process the output of the last mix-net to compute the election outcome. We emphasize that the in-charge parties must provide proof of correctness in all of the preceding procedures.

- **Public verification phase.** In this phase, every participant, including the voters or external observers can verify the correctness of the previous phases, particularly the correctness of all NIZKPs published during the setup, registration, voting, and tallying phases.

  In sections 5.6 and 5.7, we propose two instantiations of our protocol. The main difference in these instantiations is how we perform the tally phase.

This the next section we provide two instantiations of the generic **PIN-JCJ** protocol with concrete cryptographic primitives. While all three institutions adhere to the protocol framework 5.5.3, we emphasize that they differ in some steps, particularly the tally phase.

We will explain only the basic building blocks and their algorithm and suppress some details about ballot integrity and non-malleability from the zero-knowledge proofs, e.g. the inclusion of election identifiers and the correct form of the Fiat-Shamir transformations. Also, for simplicity, we describe the protocol with a single election trustee, but it is possible to run this protocol distributively. Finally, we also will not specify all parts of the distributed registration phase and the Benaloh challenges, this can; be implemented as in the NV12 scheme with some obvious modifications and with the repairs mentioned in [51].

Before proceeding, we will establish some specific notations for this chapter. First, we refer to the $\mathsf{ct}[m]$ as the cipher-text containing the message m, i.e. $\mathsf{ct} = \mathsf{Enc}(m)$, and by $\mathsf{ballot}[\mathsf{crd}, \mathsf{vote}]$, we mean a ballot that contains both the crd's and the vote's encryption.

5.6 Instantiation with Paillier cryptosystem

The first instantiation relies on the standard threshold variant Paillier public-key cryptosystem[93] whose security is based on the hardness of the decisional composite residuosity assumption.

The main reason for choosing Paillier [99] instead of exponential ElGamal [65] (as in the original JCJ protocol) is that its plaintext-homomorphic property allows us to evaluate the polynomial without decrypting the coefficients of the polynomials. Further, it allows an efficient multi-party computation protocol to compare and (hence sort) ciphertexts by plaintext values without decryption [87]. Furthermore, this algorithm is linear in the bit length, i.e. logarithmic in the security parameter, and can be made public verifiable [84]. Using this technique allows us to empower this instantiation to secure and efficient the weeding process but at the cost of all ballots submitted with a voter identifier. Thus, obfuscating votes [61, 83] must also to be cast to achieve participation privacy.

We also use the secure multi-party computation protocol MPC_{min} to compare and (hence sort) ciphertexts by plaintext values without decryption introduced by Lipmaa and Toft [87] which secretly evaluates the equality of two secret integers. In addition, the MPC protocol that is used in tally-hiding e-voting protocol such as Ordinos [84]. We refer to [84] for the public verifiable multi-party computation details.

Additional Note. We consider the protocol in the context of a single trusted party, namely a single election trustee running the key generation algorithm, rather than distributing trust among a group of election trustees; however, it can be implemented in a distributed fashion, as was already mentioned above, using the methods introduced in [93, 67]. Furthermore, we emphasize that in the Paillier cryptosystem, proof of the correct key generation, π_{Kgen}, is actually the bi-primary test of the modulus n. Numerous methods for bi-primary testing have been proposed in single- and multi-party settings, including the method introduced by Boneh and Franklin in [24]. Additionally, Vitto [123] introduces a protocol for efficiently generating certificates of semi-primality, which enables us to generate distributively semi-primes with unknown factorization.

For the proof of knowledge of the vote and the well-formedness of the ballot, we use a non-interactive version of the sigma protocol for Paillier encryption scheme. Finally, we can use any EUF-CMA-secure signature scheme.

The detail of the protocol is as follows:

- ***Set Up Phase:*** The election authority runs the key generation algorithm of Paillier encryption scheme, $\Pi^{\mathsf{Pai}}.\mathsf{Kgen}$ to generate the pair of keys and publish them on the bulletin board along with the election parameter:

$$\mathsf{k}_{\mathsf{election}} = (\mathsf{PK}, \mathsf{SK}); \mathsf{PK} = \left(n = pq, \mathbb{G}, g, \mathsf{pp}_{\mathsf{election}}\right), \mathsf{SK} = (p, q)$$

- ***Registration Phase:*** At the beginning of this phase, an empty list of voter's credentials, VoterCredentialList is initiated. Then for voter $\mathtt{v}_{\mathtt{id}}$ the registrar does the following steps:

  1. Pick a random number, $\mathsf{crd} \xleftarrow{\$} \mathbb{Z}_n$. according to the uniform distribution such that $\mathsf{crd} \notin$ VoterCredentialList. For the sake of privacy, we can store a hash value of the credential in this list rather than the credential. Append the new credential crd to the VoterCredentialList and store it on the voter's device. We refer to crd as the voter's long credential.
  2. Pick randomly a number $a \xleftarrow{\$}$ **PinSpace** and declare it to the voter $\mathtt{v}_{\mathtt{id}}$. We refer to a as a voter's short credential or voter's pin.
  3. A registrar, based on the error-tolerance policy of the election, computes the set $\mathsf{ErrorList}_a$ as in (5.1)
  $$\mathsf{poly}_{\mathtt{id},a} = \prod_{i=1}^{k}(x - \mathsf{crd} - a_i) = \sum_{i=0}^{k} p_i x^i \tag{5.7}$$
  We refer to the set $\mathsf{ErrorList}_a$ as the error list for **pin** $= a$ and $\mathsf{poly}_{\mathtt{id},a}$ the pin-polynomial for voter $\mathtt{id}$. If the voter's $\mathtt{id}$ is clear from the context we will omit the index $\mathtt{id}$.
  4. Encrypt the polynomial with respect to the election public key. We stress that by encryption of the polynomial, we mean the encryption of its coefficients, throughout this chapter. Namely, we consider the encryption as a vector with lent $k+1$.
  $$\mathsf{Enc}(\mathsf{poly}_{(\mathtt{id},a)}) = \vec{\mathsf{CP}} = (\mathsf{cp}_0, \ldots, \mathsf{cp}_k) \,:\, \mathsf{cp}_i = \mathsf{Enc}(p_i) \text{ for } i = 0, \ldots k. \tag{5.8}$$
  5. Provide a proof π_{poly} for the following relation:
  $$\begin{aligned} \mathscr{R}\mathsf{poly} = \Big\{ & (x, w), x = \big(\mathtt{id}, a, \mathsf{crd}, \mathsf{ErrorList}_a, \vec{\mathsf{CP}} = (\mathsf{cp}_i)_{i\in[k]}\big) \\ & w = \big((\mathsf{random}_i)_{i\in[k]}\big) : \\ & \mathsf{cp}_i = \mathsf{Enc}(\mathsf{PK}, p_i; \mathsf{random}_i) = g^{p_i} \cdot {\mathsf{random}_i}^n, \\ & \mathsf{poly}_{\mathtt{id},a} = \sum_{i=0}^{k} p_i x^i = \prod_{i=0}^{k}(x - \mathsf{crd} - a_i), \\ & i = 1, \ldots, k : a_i \in \mathsf{ErrorList} \Big\} \end{aligned} \tag{5.9}$$
  The registrant presents π_{poly} the voter as proof of the polynomial's validity (well-formedness and correctness). This proof is done in a designated way.
  6. Store $(\mathsf{crd}, \mathsf{CP})$ on voter device.
  7. Publish $\mathtt{v}_{\mathtt{id}} : \big(\mathsf{CP} = (\mathsf{cp}_0, \ldots, \mathsf{cp}_k), \mathsf{Enc}(\mathsf{crd})\big)$ on bulletin board.

- ***Casting Ballot:*** In this step, every voter v_i decides to abstain from the election or vote for some candidate. In the latter scenario, the voter indicates her choice after entering her PIN, $\hat{a}$ to her vsd device. Then the vsd encrypts the vote, the voter credential, and the entered $\hat{a}$, under the public key of the election (Paillier public key) using random coins random_j, resulting in the following

ciphertexts: $\mathsf{CP}^* = (\mathsf{cp}_0^*, \ldots, \mathsf{cp}_k^*)$, $\mathsf{ct}[\mathsf{vote}]$ and $\mathsf{ct}[\mathsf{crd}]$.

$$\begin{aligned} i \in [k] : \mathsf{cp}_i^* &= \mathsf{cp}_i^{(\hat{a}+\mathsf{crd})^i} \cdot \mathsf{random}_i^{*\,n} = (g^{p_i} \cdot \mathsf{random}_i^n)^{(\hat{a}+\mathsf{crd})^i} \cdot \mathsf{random}_i^{*\,n} \\ &= (g^{p_i \cdot (\hat{a}+\mathsf{crd})^i} \cdot \left(\mathsf{random}_i^{(\hat{a}+\mathsf{crd})^i} \cdot \mathsf{random}_i^*\right)^n \\ &= \mathsf{Enc}(\mathsf{PK}, p_i \cdot (\hat{a}+\mathsf{crd})^i) \\ \mathsf{ct}[\mathsf{vote}] &= \mathsf{Enc}(\mathsf{PK}, \mathsf{vote}; \mathsf{random}) = g^{\mathsf{vote}} \cdot \mathsf{random}^n, \\ \mathsf{ct}[\mathsf{crd}] &= \mathsf{Enc}(\mathsf{PK}, \mathsf{crd}) = g^{\mathsf{crd}} \cdot \mathsf{random}'^n \end{aligned} \tag{5.10}$$

As previously stated, the voter's vsd contains the voter's long credential (plain) but not the polynomial. In fact, it only has the encryption of the polynomial. However, because of the homomorphic property of the Paillier encryption scheme, the vsd is capable of performing a blind multiplication of the polynomial coefficient with the $(\mathsf{crd} + \hat{a})$.

In the next step, vsd provides a proof, π_{ballot}, (also proof of knowledge), ensures that everything was generated honestly, for relation $\mathscr{R}\mathsf{ballot}^{\mathsf{paillier}}$.

$$\begin{aligned} \mathscr{R}\mathsf{ballot}^{\mathsf{paillier}} = \Big\{ & (x, w), x = \left(\mathsf{ct}[\mathsf{vote}], \mathsf{ct}[\mathsf{crd}], \vec{\mathsf{CP}} = (\mathsf{cp}_i)_{i\in[k]}, \vec{\mathsf{CP}}^* = (\mathsf{cp}_i^*)_{i\in[k]}\right) \\ & w = \left(\mathsf{vote}, \mathsf{random}_{\mathsf{vote}}, \hat{a}, \mathsf{crd}, \mathsf{random}_{\mathsf{crd}}, \{\mathsf{random}_i^*\}_{i\in[k]}\right) : \\ & \mathsf{ct}[\mathsf{vote}] = g^{\mathsf{vote}} \cdot \mathsf{random}^n, \mathsf{vote} \in \mathsf{cList}, \\ & \mathsf{ct}[\mathsf{crd}] = g^{\mathsf{crd}} \cdot \mathsf{random}'^n, \\ & i = 1, \ldots, k : \mathsf{cp}_i^* = \mathsf{cp}^{(\mathsf{crd}+\hat{a})^i} \cdot \mathsf{random}_i^{*\,n} \Big\} \end{aligned} \tag{5.11}$$

This proof can be implemented efficiently using Sigma protocols and relies on the DDH assumption and it can be made non-interactive using the strong Fiat-Shamir heuristic.

At the end of this phase, the voter signs her ballot on the following form:

$$\mathsf{ballot}_{\mathsf{v_{id}}} = (\mathsf{ct}[\mathsf{crd}], \mathsf{ct}[\mathsf{vote}], \vec{\mathsf{CP}}^* = (\mathsf{cp}_1^*, \ldots, \mathsf{cp}_k^*), \pi_{\mathsf{ballot}}) \tag{5.12}$$

and submits it to the bulletin board. Note that the hash should contain all parts of the ballot.

Re-Vote Policy In this instantiation, each voter is permitted to submit multiple ballots, and in the tally phase, one of the valid ballots is randomly picked and counted.

Additional Note. We stress that we do not describe some steps in detail here, such as the "Benaloh challenge" to audit the voter's supporting device. Furthermore, in this instantiation, we use the obfuscate method introduced in [62, 83] to protect the voter against the coercer who compels a voter not to vote (abstain from voting) i.e. a separate (distributed) authority will submit dummy ballots on behalf of all voters. Since this authority does not know the long credential all of these dummy ballots will be invalid but will obfuscate whether a specific voter did vote or not.

- ***Tally Phase:*** Using the Paillier encryption scheme allows us to efficiently sort the ciphertexts based on plaintext values without decrypting them using MPC among the Tally Tellers (see [87] for details). Additionally, this algorithm can be implemented in a multi-party computation, to strengthen the elections. In our protocol, we take advantage of this multi-party computation to sort the encryption of polynomial evaluation.

We refer by $\mathsf{MPC}_{\min}$ the algorithm that takes as input the ciphertexts:

$$\mathsf{ct}_1 = \mathsf{Enc}(m_1), \mathsf{ct}_2 = \mathsf{Enc}(m_2), \ldots, \mathsf{ct}_t = \mathsf{Enc}(m_q),$$

and outputs the index i^* such that

$$\mathsf{ct}_{i^*} = \mathsf{Enc}(m_{i^*}) : m_{i^*} = \min\{m_1, \ldots, m_t\}.$$

We use this algorithm in the Tally phase.

1. *Ballot Validity check:* We remove exact ballot duplicates and all ballots with invalid proof and the signature in the first step. In the next step, we need to remove extra ballots for each voter, making sure a valid ballot is kept if existing.
2. *Weeding:* Since each voter will be associated with possibly more than one ballot, we need to weed them in a way that at least one of the ballots associated with the valid credential and pin will remain - if existing.

   Assume the voter $\mathsf{v}_{\mathtt{id}}$ casts q ballots, each includes choices $\mathsf{vote}_{\mathtt{id},i}$:

   $$\mathsf{v}_{\mathtt{id}} : \mathsf{ballot}_{\mathtt{id},1} : \left(\vec{\mathsf{CP}}_1^*, \mathsf{ct}[\mathsf{vote}_{\mathtt{id},1}]\right), \ldots, \mathsf{ballot}_{\mathtt{id},q}\left(\vec{\mathsf{CP}}_q^*, \mathsf{ct}[\mathsf{vote}_{\mathtt{id},q}]\right)$$

   each contains $\vec{\mathsf{CP}}_t^* = \left(\mathsf{cp}^*_{(t,1)}, \ldots, \mathsf{cp}^*_{(t,k)}\right)$ and the pin $\hat{a}_t$.

   We homomorphically combine the public ciphertext cp_0 with the submitted encryptions to obtain an encrypted polynomial evaluation for each ballot:

   $$\begin{aligned} &\text{For } t = 1, \ldots q: \\ &\mathsf{cp}_0 \cdot \prod_{j=1}^{k} \mathsf{cp}^*_{t,j} = g^{p_0} \cdot \mathsf{random}_0^n \cdot \prod_{i=0}^{k} g^{p_i \cdot (\hat{a}_t + \mathsf{crd})^i} \cdot \mathsf{random}_i^n \\ &\qquad = g^{\sum_{i=0}^{k} p_i (\mathsf{crd} + \hat{a}_t)^i} \cdot \mathsf{random} \\ &\qquad = \mathsf{Enc}(\mathsf{PK}, \mathsf{poly}_{\mathtt{id},a}(\mathsf{crd} + \hat{a}_t)) \end{aligned} \tag{5.13}$$

   Note that the above ciphertext would be the encryption of zero if the ballot has a valid credential and pin.

   $$\mathsf{Enc}(\mathsf{PK}, 0) \iff \hat{a}_t \in \mathsf{ErrorList}_a$$

   We now verifiably mix the pairs:

   $$\mathsf{ct}_t[\mathsf{poly}_{\mathtt{id},a}(\hat{a}_t + \mathsf{crd})], \mathsf{ct}[\mathsf{vote}_{\mathtt{id},t}]$$

   and run the $\mathsf{MPC}_{\min}$ protocol:

   $$\mathsf{MPC}_{\min}\left(\mathsf{ct}_{\sigma(j)}[\mathsf{poly}_{\mathtt{id},a}(\hat{a}_{\sigma j} + \mathsf{crd})]\right)_{j \in [q]} \mapsto \mathsf{ct}[\mathsf{poly}_{\mathtt{id},a}(\hat{a}_m + \mathsf{crd})] \tag{5.14}$$

   which securely outputs the minimum value:

   $$\forall\, t = 1, \ldots, q : \mathsf{poly}_{\mathtt{id},a}(\mathsf{crd} + \hat{a}_m) \leq \mathsf{poly}_{\mathtt{id}}(\mathsf{crd} + \hat{a}_t),$$

We keep this ciphertext and the corresponding encrypted vote and discard the rest. Note that $\mathsf{MPC}_{\min}$ selects a valid ballots having $\mathsf{poly}_{\mathtt{id},a}(\hat{a}_m + \mathsf{crd}) = 0$ if it exists.[1]

3. *Ballot anonymization:* We observe that at this stage, there is only one ballot left on the bulletin board for each voter, which is composed of two components:

$$\begin{aligned}\mathsf{v}_{\mathtt{id}} : \tilde{\mathsf{ballot}}_{\mathtt{id},m} = (\tilde{\mathsf{ct}}_{\mathtt{id}}, \tilde{\mathsf{ct}}'_{\mathtt{id}}) : \\ \tilde{\mathsf{ct}}_{\mathtt{id}} = \mathsf{ct}_{\mathtt{id}}[\mathsf{poly}] = \mathsf{Enc}(\mathsf{poly}_{\mathtt{id},a}(\hat{a}_m + \mathsf{crd})), \\ \tilde{\mathsf{ct}}'_{\mathtt{id}} = \mathsf{ct}_{\mathtt{id}}[\mathsf{vote}_{\mathtt{id},m}].\end{aligned}$$

In fact, $\mathsf{ballot}_{\mathtt{id},m}$ is the output of the MPC_{min} protocol. To avoid the heavy notation, we will no longer use the index m and we refer to $\mathsf{ballot}_{\mathtt{id},m}$ by $\mathsf{ballot}_{\mathtt{id}}$.

Two steps must be completed in order to anonymize the ballots:

Step 1. Raise the first component to the power r_1 and re-encrypt the second component:

$$\tilde{\mathsf{ct}}_{\mathtt{id}} \mapsto \mathsf{ct}_{\mathtt{id}} = \tilde{\mathsf{ct}}_{\mathtt{id}}^{\mathsf{r}_1} = \mathsf{Enc}\big(\mathsf{poly}_{\mathtt{id},a}(\hat{a}_m + \mathsf{crd})^{\mathsf{r}_1}\big) \tag{5.15}$$

and provide a proof $\pi_{\mathsf{randomization}}$ for the relation $\mathscr{R}\mathsf{randomization}$:

$$\begin{aligned}\mathscr{R}\mathsf{randomization} = \Big\{(x, w), x = (\tilde{\mathsf{ballot}}, \mathsf{ballot}), w = \mathsf{r}_1 : \\ \tilde{\mathsf{ballot}} = (\tilde{\mathsf{ct}}, \mathsf{ct}'), \mathsf{ballot} = (\mathsf{ct}, \mathsf{ct}'), \\ \mathsf{ct} = \tilde{\mathsf{ct}}^{\mathsf{r}_1} \\ \Big\}\end{aligned} \tag{5.16}$$

We multiplicatively randomize the polynomial evaluation to avoid coercion side channels but still be able to determine if a ballot was valid by checking if the value is zero. At the end of this step, we obtain the following list, which includes a single ballot for each voter:

$$\mathbf{L}_1 = \Big[\big(\mathsf{ct}_{\mathtt{id}} = \mathsf{ct}[\mathsf{poly}_{\mathtt{id},a}(\hat{a}_{\mathtt{id}} + \mathsf{crd}_{\mathtt{id}})], \mathsf{ct}'_{\mathtt{id}} = \mathsf{ct}[\mathsf{vote}_{\mathtt{id}}]\big)\Big]_{\mathtt{id} \in \mathsf{idSet}}$$

Step 2. In this step first, we delete the voter identifier. Then we insert the list $\mathbf{L}_1$ into a verifiable parallel mix-net to apply the re-encryption and permutation procedure on the list $\mathbf{L}_1$. It is required that each mix-net provides proof of shuffle, π_{Shuffle}, for the relation, $\mathscr{R}\mathsf{shuffle}$ (5.4). At the end of this step we obtain $(\mathbf{L}_2, \pi_{\mathsf{Shuffle}})$ where:

$$\mathbf{L}_2 = \Big[(\mathsf{ct}_{\sigma(\mathtt{id})}[\mathsf{poly}_{\sigma(\mathtt{id}),a}(\hat{a}_{\sigma(\mathtt{id})} + \mathsf{crd}_{\sigma(\mathtt{id})})^{\mathsf{random}}], \mathsf{ct}'[\mathsf{vote}_{\sigma(\mathtt{id})}])\Big]_{\mathtt{id} \in \mathsf{idSet}}$$

4. *Final PIN-Credential Check:* In this step, first we decrypt the first component of each ballot:

$$\mathsf{ct}_{\mathtt{id}}[\mathsf{poly}_{\sigma(\mathtt{id}),a}(\hat{a}_{\sigma(\mathtt{id})} + \mathsf{crd}_{\sigma(\mathtt{id})})^{\mathsf{random}}]$$

to retrieve the polynomial evaluation. All ballots with non-zero polynomial evaluation will be discarded.

5. *Vote Extraction:* Decrypt the remaining vote ciphertexts and provide proof of decryption 5.5.

[1] This will give a random correct vote. The "Last valid vote counts" policy can be implemented by adding the received order to the plaintext.

6. *Result computation:* Obtain the election result by applying the f_{res} over all plaintext votes.

Error tolerance property in this instantiation results from the correctness of the encryption scheme and the MPC_{min}. First, consider the following computation:

$$\left.\begin{array}{l} 5.8: \mathsf{cp}_i = g^{p_i} \cdot \mathsf{r}_i^n \\ 5.10: \mathsf{cp}_i^* = \mathsf{cp}_i^{(\hat{a}+\mathsf{crd})^i} \cdot \mathsf{r}_i^{*n} \end{array}\right\} \Rightarrow \mathsf{cp}_i^* = g^{(\hat{a}+\mathsf{crd})^i p_i} \cdot \mathsf{r}_i'^n$$

$$\Rightarrow \mathsf{ct}[\mathsf{poly}_{\mathtt{id},a}(\hat{a}_t + \mathsf{crd})] := \mathsf{cp}_0 \cdot \prod_{i=1}^{k} \mathsf{cp}_i^*$$

$$= g^{\sum_{i=0}^{n} (\hat{a}+\mathsf{crd})^i p_i} \cdot \mathsf{r}^n$$

$$= g^{\mathsf{Poly}_{\mathtt{id}}(\mathsf{crd}+\hat{a})} \cdot \mathsf{r}^n$$

Due to the definition of **pin**-polynomial (5.2):

$$\mathsf{ct}[\mathsf{poly}_{\mathtt{id},a}(\hat{a}_t + \mathsf{crd})] = \mathsf{Enc}(0) \text{ if and only if } \mathsf{crd}+_{\hat{a}_t} \in \mathsf{ErrorList}_a + \mathsf{crd}_{\mathsf{registered}}$$

which with the overwhelming probability implies that $\hat{a} \in \mathsf{ErrorList}_a$. Additionally, the correctness property of the multi-party computation protocol $\mathsf{MPC}_{\min}$ in (5.14) guarantees that the ballot with a minimum value of $\mathsf{poly}_{\mathtt{id},a}(\hat{a}_t + \mathsf{crd})$ remains in the weeding step.

Note that $\mathsf{poly}_{\mathtt{id},a}$ has a range in non-negative integers. Therefore if there is any ballot with valid credentials and PIN, the output of $\mathsf{MPC}_{\min}$ will be a valid ballot, namely $\mathsf{ct}[\mathsf{poly}_{\mathtt{id},a}(\hat{a}_t + \mathsf{crd})] = \mathsf{Enc}(0)$.

Additional Note. The main advantage of this instantiation is sorting the ciphertexts without decrypting them (using $\mathsf{MPC}_{\min}$ protocol). Because this approach does not reveal whether any ballot has a valid PIN or not, thus sidestepping the attack on the standard duplicate removal.

5.7 Instantiation with BGN cryptosystem

The second instantiation is based on composite order groups introduced by [25] and the Groth-Sahai NIWI-proof system [63] with security based on the Subgroup decision assumption.

The main point of using those primitives in this instantiation is, BGN is a homomorphic encryption scheme that supports a single multiplication which is what we need and also it can be efficiently implemented in a bilinear group. Having a bilinear map allows us to do the polynomial evaluation in an efficient and secure way and also have the efficient NIWI-proof system.

Definition 2. ***BGN Cryptosystem*** *works as follows. Its Key-Generation algorithm,* Kgen *outputs a pair of keys:* $\big(\mathsf{pk} = (n, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, h = g'^q), \mathsf{sk} = (p, q)\big)$ *where* $\mathbb{G} = \langle g \rangle$ *and* $\mathbb{G}_T$ *are two groups of order n and the secret key consists of two primes* p, q *such that* $n = pq$. $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ *is bilinear* $(\forall a, b \in \mathbb{Z}, g \in \mathbb{G} : \mathbf{e}(g^a, g^b) = \mathbf{e}(g, g)^{ab})$, *non-degenerate* $(\mathbb{G} = \langle g \rangle \Rightarrow \mathbf{e}(g, g) \neq 1_{\mathbb{G}_T})$ *and commutable map. A ciphertext on message* $m \in [T]$, *for* $T < q$ *has the form* $\mathsf{CT} = g^m h^{\mathsf{random}} \in \mathbb{G}$ *for some random number* random. *Decryption: raise the ciphertext to power p and compute the discrete log.*

The detailed protocol is as follows:

- ***Set Up Phase:***

1. The election authority runs the key generation algorithm of BGN encryption scheme, $\Pi^{\mathsf{BGN}}.\mathsf{Kgen}$ to generate the pair of keys:

$$(\mathsf{sk}_{\mathsf{BGN}} = p, q, \mathsf{pk}_{\mathsf{BGN}} = (n, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, h)$$

2. Then chooses a random group element $f \xleftarrow{\$} \mathbb{G}$. Note that $\mathbb{G} = \langle g \rangle$ is a cyclic group so there exists a unique integer z such that $f = g^z$.
3. Set the election key as:

$$\mathsf{PK}_{\mathsf{election}} = (n, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, h)\ ,\ \mathsf{SK}_{\mathsf{election}} = (p, f)$$

4. Publish the public key on the bulletin board along with the election parameter.

- ***Registration Phase:*** The registrar performs the following steps to generate the credential for the voter, $\mathsf{v}_{\mathtt{id}}$

1. Generate credential and PIN: crd, a as in the Paillier instantiation.
2. Generate the list of errors and the pin-polynomial. Note that in this instantiation the polynomial $\mathsf{poly}_{\mathtt{id},a}$ only contains the polynomial, not the long credential:

$$\mathsf{ErrorList}_a = \{a_1 = a, a_2, \ldots, a_k\}$$
$$\mathsf{poly}_{\mathtt{id},a} = \prod_{i=1}^{k}(x - a_i) = \sum_{i=0}^{k} p_i x^i$$

3. Run the encryption algorithm, $\Pi^{\mathsf{bgn}}.\mathsf{Enc}$ to obtain the following ciphertexts:

$$\forall i \in [k] : \mathsf{cp}_i = \mathsf{Enc}(p_i) = g^{p_i} h^{\mathsf{random}_i},$$
$$\mathsf{cp}_0 = g^{p_0} \cdot f^{\mathsf{crd}} h^{\mathsf{random}}$$
$$= \mathsf{Enc}(p_0 + \mathsf{crd} \times z).$$

Note that, technically cp_0 is the encryption of $p_0 + \mathsf{crd} \times z$. Although z is not a known value to any party, the registrar can compute cp_0 without knowing its exact value.

4. Generates the proof of validity of the polynomial $\mathsf{poly}_{\mathtt{id},a}$ and $\left(\mathsf{cp}_i\right)_{i=0,\ldots k}$, similar to the Paillier instantiation.
5. Store

$$\left(\mathsf{CP} = (\mathsf{cp}_0, \mathsf{cp}_1, \ldots, \mathsf{cp}_k), \mathsf{CRD} = g^{\mathsf{crd}}\right)$$

in the user device.

6. Publish $\mathsf{v}_{\mathtt{id}} : \left(\mathsf{Enc}(\mathsf{crd}) = g^{\mathsf{crd}} \cdot h^{\mathsf{random}}, \mathsf{CP}\right)$ on bulletin board.

- ***Casting ballot:*** During this phase, the voter can perform an audit on her device to ensure its accuracy. This process is similar to the Paillier instantiation, therefore we skip the details and only describe the ballot creation step.

Voter, $\mathsf{v}_{\mathtt{id}}$, indicates her choice after entering her PIN, $\hat{a}$ to her vsd device. Then the vsd encrypts the vote, the voter credential, and the entered $\hat{a}$, under the public key of the election(See 3) using random coins random_j, resulting in the following ciphertexts:

1. Compute:
$$\mathsf{CT}_{\mathsf{vote}} = \mathsf{Enc}(\mathsf{vote}), \mathsf{CT}_{\mathsf{crd}} = \mathsf{Enc}(\mathsf{crd}) = \mathsf{CRD} \cdot h^{\mathsf{random}}$$

2. Encrypt the PIN:
$$\text{For } i = 1, \ldots, k : \mathsf{ca}_i = \mathsf{Enc}(\hat{a}^i).$$

3. Re-randomize the polynomial coefficients:
$$\text{For } i : 0, 1, \ldots, k : \mathsf{cp}_i^* = \mathsf{cp}_i \cdot h^{\mathsf{random}_i^*}$$

4. Set
$$\begin{aligned} \mathsf{CA} &= (\mathsf{ca}_1, \ldots, \mathsf{ca}_k), \\ \mathsf{CP}^* &= (\mathsf{cp}_0^*, \ldots, \mathsf{cp}_k^*) \end{aligned}$$
and provide a proof (proof of knowledge), π_{ballot} for the following relation, including a joint proof of plaintext-knowledge for all the other ciphertexts in the ballot and include the rest of the ballot in the hash for non-malleability. This proof can be generated using the Groth-Sahai technique.
$$\begin{aligned} \mathscr{R}ballot^{\mathsf{bgn}} = \Big\{ (x, w) : \\ & x = \big(\mathsf{pp}_{\mathsf{election}}, \mathsf{CT}_{\mathsf{vote}}, \mathsf{CT}_{\mathsf{crd}}, \mathsf{CA}\big), \\ & w = \big(\mathsf{vote}, \mathsf{random}_{\mathsf{vote}}, \mathsf{CRD}, \mathsf{r}_{\mathsf{crd}}, \hat{a}, \{\mathsf{r}_i\}_{i \in [k]}\big) : \\ & \mathsf{CT}_{\mathsf{vote}} = g^{\mathsf{vote}} \cdot h^{\mathsf{random}_{\mathsf{vote}}}, \\ & \mathsf{vote} \in \texttt{List of candidats}, \\ & \mathsf{CT}_{\mathsf{crd}} = \mathsf{CRD} \cdot h^{\mathsf{r}_{\mathsf{crd}}}, \\ & \{\mathsf{ca}_i = g^{(\hat{a})^i} \cdot h^{\mathsf{r}_i}\}_{i \in [k]} \\ \Big\} \end{aligned} \tag{5.17}$$

5. At the end of this phase, the voter signs her ballot of the following form:
$$\mathsf{ballot} = (\mathsf{CT}_{\mathsf{vote}}, \mathsf{CT}_{\mathsf{CRD}}, \mathsf{CA}, \mathsf{CP}^*, \pi_{\mathsf{ballot}})$$
and submits it to the bulletin board.

Re-Vote Policy In this instantiation, each voter is permitted to submit multiple ballots, and in the tally phase, unlike to Paillier instantiation, it is possible to consider the last valid ballot cast by the voter.

1. *Ballot Validity check:* In the first step, we remove exact ballot duplicates and then we check the validity of the proofs and the signatures. In case of any failure, the ballot will be discarded. In the next step, we need to remove extra ballots for each voter, making sure a valid ballot is kept if existing.
2. *Polynomial evaluation:* Compute the encrypted polynomial evaluation as follows:
$$\begin{aligned} \mathsf{Enc}_T(t) &= \mathbf{e}(\mathsf{CT}_{\mathsf{crd}}, f)^{-1} \cdot \mathbf{e}(\mathsf{cp}_0^*, g) \cdot \mathbf{e}(\mathsf{cp}_1^*, CA_1) \cdots \mathbf{e}(\mathsf{cp}_k^*, CA_k) \\ &= \mathsf{Enc}_T(\mathbf{e}(g^{\mathsf{poly}_{\mathtt{id}, a, g}(\hat{a}_{\mathtt{id}})}, g)) \end{aligned} \tag{5.18}$$

We refer to this value by $\mathsf{Enc}_T(t)$ with t being the polynomial evaluation which can be seen as encryption in the target space (group $\mathbb{G}_T$).

3. *Mixing:* Now verifiably mix the tuples $(\mathsf{CT}_{\mathsf{crd}}, \mathsf{CT}_{\mathsf{vote}}, \mathsf{Enc}_T(t))$.

4. For each ballot we create $\mathsf{Enc}_T(\mathsf{crd}+t)$:

$$\begin{aligned}\mathbf{e}(\mathsf{CT}_{\mathsf{crd}}, g)\cdot \mathsf{Enc}_T(t) &= \mathsf{Enc}_T(\mathbf{e}(g^{\mathsf{crd}}, g))\cdot \mathsf{Enc}(\mathbf{e}(g^{\mathsf{poly}_{\mathtt{id},a}(\hat{a})}, g))\\ &= \mathsf{Enc}_T(\mathbf{e}(g^{\mathsf{crd}+\mathsf{poly}_{\mathtt{id},a}(\hat{a})}, g))\end{aligned}$$

Now, we need to remove ballots having the same value $\mathbf{e}(g^{\mathsf{crd}+\mathsf{poly}_{\mathtt{id},a}(\hat{a})}, g)$, which basically means the same credential and same $\hat{a}$. This can be done by running a PET or using some hash function. At the end of this step, we have a list of ballots of the following form:

$$\left(\mathsf{Enc}_T(\mathbf{e}(g^{\mathsf{crd}+\mathsf{poly}_{\mathtt{id},a}(\hat{a})}, g)), \mathsf{CT}_{\mathsf{vote}}\right) \tag{5.19}$$

Additional Note. If we have a small number of voters, it is possible (efficient) to do this via PETs. Further, we can mix between each duplicate removal. For a larger number, we suggest splitting the board in two, removing duplicates separately, then mixing and doing duplicate removal again. This will decrease the information from the distribution of confirmed duplicates to a coercer carrying out the *leaky duplicate removal attack*. Additionally, for the sake of privacy, we can replace the first component in (5.19) with $\mathsf{H}_{\mathsf{hk}}(\mathbf{e}(g^{\mathsf{crd}+\mathsf{poly}_{\mathtt{id},a}(\hat{a})}, g)(\hat{a}))$ where H_{k} is a keyed-hash function with secret key hk.

5. We now need to select eligible valid votes. We mix the above list and the list of registered encrypted credentials. To this end, compare $\mathsf{crd}+\mathsf{poly}_{(\mathtt{id},a)}(\hat{a})$ to the list of registered credentials. Any ballot that has a match with a registered credential would be a ballot with a valid credential where its polynomial evaluation is equal to zero with overwhelming probability. There is only a negligible chance that invalid crd and invalid PIN generate an invalid full credential. Next, we do a further PET against the short credentials. This will reveal malicious authorities creating valid polynomials on their own. If this is positive too, we decrypt the vote and continue to the next registered credential.

Error tolerance property: The following computation shows the correctness of equation 5.18, in other words, it shows how to evaluate the polynomial on the input value $\hat{a}$:

$$\begin{aligned}&\mathbf{e}(\mathsf{CT}_{\mathsf{crd}}, f)^{-1}\cdot\mathbf{e}(\mathsf{cp}_0^*, g)\cdot\mathbf{e}(\mathsf{cp}_1^*, CA_1)\cdots\mathbf{e}(\mathsf{cp}_k^*, CA_k) =\\
&\mathbf{e}(\mathsf{CRD}\cdot h^{\mathsf{r}}, f)^{-1}\cdot\mathbf{e}(g^{p_0}(f)^{\mathsf{crd}}h^{\mathsf{r}_0}, g)\cdot\mathbf{e}(g^{p_1}h^{r_1}, g^{\alpha_i}h^{\gamma_i})\cdot\ldots\mathbf{e}(g^{p_k}h^{\mathsf{r}_k}, g^{\alpha_k}h^{\gamma_k}) =\\
&\mathbf{e}(\mathsf{CRD}, f)^{-1}\cdot\mathbf{e}(h, f)^{-\mathsf{r}}\mathbf{e}(g^{p_0}f^{\mathsf{crd}}h^{\mathsf{r}_0}, g)\cdot\mathbf{e}(g^{p_1}h^{\mathsf{r}_1}, g^{a^i}h^{\gamma_i})\cdot\ldots\mathbf{e}(g^{p_k}h^{\mathsf{r}_k}, g^{a^k}h^{\gamma_k}) =\\
&\mathbf{e}(\mathsf{CRD}, f)^{-1}\mathbf{e}(f, h^{\mathsf{r}})\mathbf{e}(f, \mathsf{CRD})\mathbf{e}(g^{p_0}h^{\mathsf{r}_0}, g)\cdot\mathbf{e}(g^{p_1}h^{\mathsf{r}_1}, g^{a^i}h^{\gamma_i})\cdot\ldots\mathbf{e}(g^{p_k}h^{\mathsf{r}_k}, g^{a^k}h^{\gamma_k}) =\\
&\mathbf{e}(h^{\mathsf{r}}, f)(\prod_{i=0}^{k}\mathbf{e}(g^{p_i}, g^{\alpha_i}))\cdot(\prod_{i=0}^{k}\mathbf{e}(g^{p_i}, h^{\gamma_i}))\cdot(\prod_{i=0}^{k}\mathbf{e}(g^{\alpha_i}, h^{\mathsf{r}_i}))(\prod_{i=0}^{k}\mathbf{e}(h^{\gamma_i}, h^{\mathsf{r}_i}))\\
&\mathbf{e}(g, g^{\sum_{i=0}^{k}p_i\alpha_i}))\cdot\mathbf{e}(g, h^{\mathsf{r}}) = \mathbf{e}(g, g^{\mathsf{poly}_a(\hat{a})})\cdot\mathbf{e}(g, h^{\mathsf{r}})\end{aligned}$$

As a result, if we raise the above term to the power p, the result is 1 if and only if $\mathsf{poly}_{\mathtt{id},a}(\hat{a}) = 0$. Also, due to the secret f and zero-knowledge proof, π_{ballot}, malicious voters cannot construct a fake polynomial resulting in zero evaluation.

Additional Note. The main advantage of this instantiation is that voters can vote anonymously; thus, unlike the first instantiation, no identity obfuscation is required. Additionally, by utilizing Groth-Sahai proof techniques, the implementation becomes more efficient. Particularly in an election with many candidates and voters, compared to the Sigma protocol, the proof of membership will be more effective. Moreover, we can implement the mix-net components using the Groth-Bayer mix-net servers [7].

5.8 PIN Space Covering

People are prone to errors in many aspects of life, including when entering numbers. The effects of these errors can be disastrous. Designing better systems may help to prevent these errors however, in order to do this we need to understand far more about the types of errors being made, and their causes.

Reason [102] defines two types of errors: slips (or lapses) and mistakes. Mistakes occur when a person has incorrect or absent knowledge of the task they are aiming to complete. Slips occur when a person has the knowledge needed to perform a task but for some reason takes the wrong actions in completing the task — this may be during the execution stage or during planning. Errors due to mistakes are remedied by providing training to people in order to complete their knowledge about the task and how to perform it. Slip errors however are more difficult to avoid — they occur even when we are very skilled at a task. Before we can handle to prevent these slip errors from occurring, we must first understand the errors being made.

They presented a corpus of real-life errors, specifically number entry errors, gathered in experimental conditions and a method for classifying and organizing the errors collected in order to facilitate further research into causes and solutions. Unfortunately, there are very few documented examples of number entry errors and thus many of the studies conducted so far rely upon modeled, not real-world data. Authors in [126] studied different types of errors. They provide a list of common errors e.g. “Digit wrong”, “Anagram”, “Skipped”, “Digit Missing”, etc.

In our voting system, we ensure that the voter’s credentials are verified even if they make a mistake when entering their PIN. This could be a transposition error or a single incorrect digit, for example. From a security perspective, it is interesting to consider how much this reduces the entropy of the PINs. We are going to determine the minimum number of PINs an attacker would need to try in order to cover the entire PIN space. This defines the security against a brute force attack conducted by an attacker who has a legitimate credential, such as a smart card.

We will not solve this exactly but will provide some upper and lower bounds. Additionally, as indicated by PIN frequency studies, users are often poor at selecting random PINs. As a result, we recommend that the PIN be generated randomly and not chosen by the voter. We start by considering the scenario in which we allow for PIN swaps and a single-digit error. Let’s refer to the PIN as $p_1 p_2 \cdots p_k$. We begin by calculating the number of PINs covered by a PIN attempt.

Let us begin with the example when k equals two. The term $[p_1 p_2]$ refers to the range of numbers covered by this PIN. Clearly, $[p_1 p_2] = \{p_1 p_2, p_2 p_1, p_1 *, * p_2\}$, where $* \in \{01, 2, \ldots, 9\}$. After deleting the repeated occurrences, $|[p_1 p_2]| = 20$ in the case $p_1 \neq p_2$, and 19 in the case $p_1 = p_2$. Indeed, given $2r$ different digits $p_1, \ldots, p_{2r}$, it is possible to verify that the r 2-digit numbers $p_1 p_2, p_3 p_4, \ldots, p_{2r-1} p_{2r}$ will cover a total of $20r - 2\binom{r}{2}$ PINs.

Additionally, the formula may be used to get an upper limit on the number of PINs covered by r, from which we see that, indicating that the attacker needs at least eight PINs to cover the whole PIN space for all 2-digit numbers. Given that the attacker’s goal is to cover the PIN space with the fewest available tries, a suitable technique seems to be to augment the basis with PINs with different digits. If no new PIN with different digits is available, we will add a PIN that increases the size of the existing basis the most, and so

on until the PIN space is covered. We built an algorithm in Python that is based on this concept, but uses random sampling to identify the next most efficient element. We found that 9 PIN attempts are enough for the situation of a two-digit PIN, which is near the theoretical lowest limit.

We now move on to the situation of three-digit PINs. The maximum size of any PIN $p_1p_2p_3$ is 30 (when we allow swapping and wrong digit errors). As a result, 34 is a lower constraint on the base size of PIN space in this situation. Assume that no mistakes other than swappings will be accepted. There, the base size is 55, as determined by the Python code[2]. For $k \geq 3$, the upper limit on the cover of a single PIN is k (including itself), implying a lower bound of $\frac{10^k}{k}$. The results in Table 5.1 show the range of possible theoretical bounds for PIN lengths of 2 to 5. For PIN lengths 2, 3, and 4, the code was run 1000 times to obtain these results. For a PIN length of 5, the code was only run once.

These results give us an idea of the potential range of theoretical bounds for different PIN lengths.

PIN Length	2	3	4	5
S+W Lower Bound	8	34	250	2000
S+W Upper Bound	9	78	713	6490
S Upper Bound	55	465	4131	

TABLE 5.1: S+W means the system accepts swapping errors and wrong digit errors, whereas S means a system that just tolerates swapping errors.

5.9 Conclusions and Outlook

In summary, we have presented an e-voting scheme that is resistant to coercion attacks and typo errors in PINs that is usable by using smart cards to store credentials and PINs to unlock these. The scheme uses partially homomorphic Paillier encryption and distributed plaintext equivalency tests to ensure the privacy and security of the voting process. The scheme also includes mechanisms to detect and prevent malicious authorities from generating fake polynomial evaluations. Future work may focus on improving the efficiency of mistake correction and incorporating biometric data into the scheme.

Incorporating the tally system from Ordinos [84] into our Paillier-based system is a logical next step, as both protocols are based on the Paillier encryption system. The Ordinos system only reveals the winner or candidate rankings in the election, which can help to resist coercion in cases where candidates are expected to receive few or no votes. Another possibility for both protocols is the use of risk-limiting tallies [75], which offer voters plausible deniability. It may also be worth exploring the space of possible PINs in more detail, and it is likely that more precise findings could be discovered.

Intriguingly, the likelihood of a single-digit error occurring in a k-digit Personal Identification Number (PIN) has a mathematical relationship with Rook polynomials on a k-dimensional chessboard. This relationship is derived from the concept of Rook polynomials, which are used in combinatorics to count the number of ways a certain number of non-attacking rooks can be placed on a chessboard. In the context of PINs, each digit can be thought of as a rook on a k-dimensional chessboard, where k is the number of digits in the PIN. A one-digit error in the PIN is analogous to a possible place in the chessboard which is under attack by the rook. On that problem, we are trying to put minimum rooks in the chessboard such that all places are under attack. This mathematical model provides a unique perspective on the probability of PIN errors and can be used to develop more effective error detection and correction strategies [5].

[2] Here the code for this analysis https://github.com/ehsanestaji/JCJ-Pin

In addition to this mathematical perspective, there are several socio-technical research issues that need to be addressed. These issues are particularly relevant in the context of secure voting systems, where voters are required to enter their PINs without receiving any feedback on the correctness of their input.

The first issue pertains to the types of errors voters are likely to make when entering their PINs in a voting environment. Understanding these error patterns can help in designing more user-friendly voting systems and in developing effective error correction strategies. For instance, are voters more likely to transpose digits, forget digits, or enter extra digits? Are certain digits more prone to errors than others? These are some of the questions that need to be explored.

The second issue is related to the development of an optimal PIN strategy that can correct the maximum number of PIN errors while maintaining a high level of entropy in the PIN space. High entropy is desirable as it makes the PINs harder to guess, thereby enhancing the security of the voting system. This requires a careful balance between user-friendliness (which may necessitate simpler PINs) and security (which requires more complex PINs).

The third issue is about the education of voters in the management, forging, and concealment of secret keys, especially in scenarios where smart cards are not utilized. Even when smart cards are used, voters still need to be educated on how to handle and protect their secret keys. This involves not only technical education but also awareness about the potential threats and the importance of maintaining the secrecy of their keys. The effectiveness of such education programs is a crucial factor in the overall security of the voting system.

CHAPTER 6

MACHINE-CHECKED PROOFS OF PRIVACY AGAINST DELAY-USE MALICIOUS BOARDS

This chapter offers a detailed analysis of concepts related to ballot privacy, including privacy modeling, established definitions, key terminology, and core properties. It also traces the historical evolution of ballot privacy, highlighting important elements such as the BPRIV definition, strong consistency, strong correctness, and re-voting strategies.

Additionally, this chapter introduces Labeled MiniVoting and voting friendliness and discusses the game-based security approach against a dishonest ballot box using the mechanism of mb – BPRIV.

We further explore the mb – BPRIV model, where a tally is taken only if the adversary's board passes the validity test and all voters are satisfied after completing a specific verification procedure. This precondition requires the verification process to be completed before the tally, thereby rendering mb – BPRIV incompatible with voting systems such as Selene. In Selene, the tally is done prior to verification to mitigate coercion risks. This incompatibility underscores the need for a new privacy definition that can handle potentially ballot box (tally is honest) and also support voting protocols where the verification is done post-tally computation.

In response to this need, we introduce a new definition of ballot privacy against a potentially dishonest ballot box, termed as delay-use malicious ballot box ballot privacy (du–mb–BPRIV). This definition extends the applicability of voting systems to those that permit verification after tallying and those that necessitate a secret key during the verification stage. While sharing similarities with mb – BPRIV, our definition diverges in allowing the adversary to view the tally only after the validity of his board is confirmed, and subsequently, the results of the verification phase are disclosed.

Two sections 6.3 and 6.5 are closely related to sections III and V in [47]. The remaining parts give more explanations than presented in [47] especially on the proof of Theorem 1 of the Mini-Voting scheme in

Section 6.4. [1]

6.1 Ballot Privacy Evolution

The privacy of votes has been a longstanding concern in democratic elections, especially as universal suffrage has been implemented in various countries. With the introduction of electronic technologies into the voting process, there are new challenges and concerns related to preserving the privacy of ballots. Cryptographic voting protocols seek to address these issues by developing security models and constructing schemes to fulfill these models. These protocols aim to provide ballot secrecy in electronic voting.

Cryptographic voting schemes can generally be classified into two categories: purely electronic schemes, in which voters cast ballots from the privacy of their own computers (e.g., Helios [3, 4] or Civitas [37]), and hybrid schemes, in which paper ballots and computers are used to facilitate the tally (e.g., ThreeBallot [103], Prêt-à-Voter [109], and Scantegrity [33]).

In the typical practice of electronic voting, votes are usually encrypted using a public key, while the decryption key, necessary for revealing the votes and calculating the final tally, is shared among several trustees. The implication of this method is that a minimum number of trustees are required for decoding the votes. A significant design consideration is to withhold the decryption key from the ballot box or, more accurately, the server assigned the task of collecting all the votes. This strategy is implemented with the intention of safeguarding voter privacy, assuring that no information about the votes can be accessed by the server, even when its trustworthiness is questionable.

Current definitions of ballot privacy, such as those found in [17, 22, 36], posit a scenario where control over the votes of honest parties is in the hands of the adversary, but the resulting ballots remain beyond their influence. Once posted on the bulletin board, these ballots are immune to modification or removal prior to tallying. This means that the existing security notions can only guarantee the security of such schemes under the assumption of an honest bulletin board, despite these schemes being intended to withstand a deceitful voting server.

This mismatch between the desired security objectives and the existing security definitions has been recently confirmed in the case of Helios [3] by Roenne [106]. As it has been explained in [41], it was revealed that voter privacy can be compromised by a dishonest bulletin board if the option for voters to revote is available. Furthermore, it seems to be non-trivial to prevent such a scenario, even when additional checks are implemented by voters and independent auditors, such as the prohibition of duplicate ballots, commonly referred to as 'weeding.' In order to detect such an attack, impractical preventive measures would need to be adopted, such as requiring each voter to meticulously keep a record of every one of their ballots, including those that were unsuccessful in reaching the ballot box.

To address this issue, a new approach is proposed by Cortier *et al.* [41]. We will begin by providing an overview of the historical contexts surrounding ballot privacy definitions. Subsequently, we will delve into the novel definition of ballot privacy specifically tailored for the dishonest ballot box scenario, considering the presence of an adversary.

In [40], an attempt was made to model privacy against a dishonest board, assuming that all honest voters adhere to the specified protocol verifications. However, this assumption proved unrealistic as it is more plausible in a real-world scenario that only a fraction of honest voters perform the required tests while others cease their participation after casting their votes.

[1] The EasyCrypt proof of Theorem 1 was mainly written by me whereas the EasyCrypt proof of Theorem 2 was mainly done by Morten Solberg.

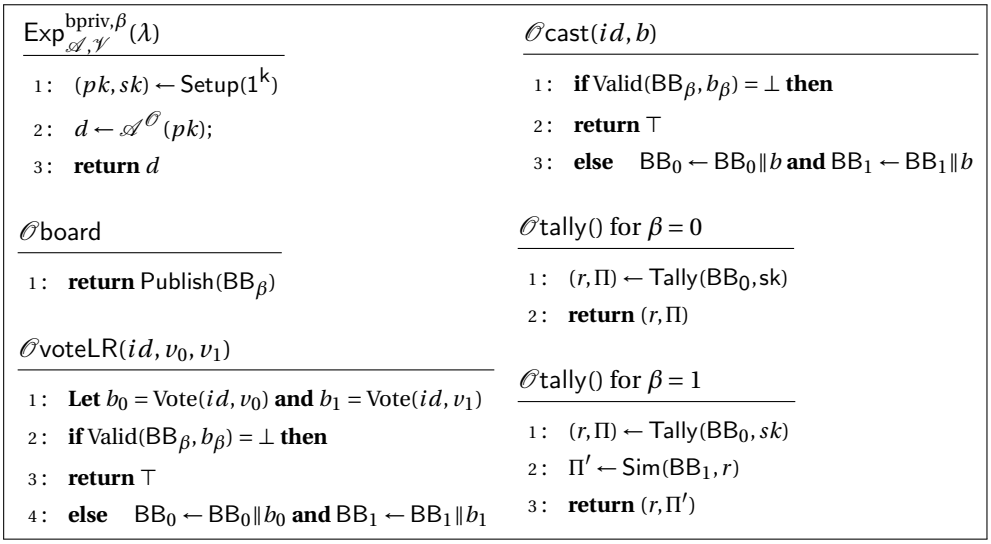


FIGURE 6.1: In the experiments $\mathsf{Exp}^{\mathrm{bpriv},\beta}_{\mathscr{A},\mathscr{V}}(\lambda)$ defined above for $\beta = 0,1$ adversary $\mathscr{A}$ has access to the set of oracles $\mathscr{O} = \{\mathscr{O}\mathrm{cast}, \mathscr{O}\mathrm{board}, \mathscr{O}\mathrm{voteLR}, \mathscr{O}\mathrm{tally}\}$. The adversary is allowed to query $\mathscr{O}$tally only once. For $\beta = 1$ the experiment also depends on Sim.

The first game-based definition concerning a malicious board was proposed by Bernhard and Smyth [20]. An extension of Benaloh's methodology [17] was used in their definition. Here, the adversary provides a board and vote tallying occurs only if the subtally of ballots from honest voters does not deviate in the "left" and "right" worlds. This somehow corresponds to one possible instance of our recovery algorithm in mb-BPRIV, where the attacker may remove any honest vote and add an arbitrary number of votes, independently of whether honest voters do check their ballot and independently of the number of dishonest voters.

Recent security studies on encryption schemes, where ballots can be partially altered, were carried out by Bursuc, Dragan, and Kremer [29]. A variant of BPRIV (A precise definition will be provided in the subsequent parts of this section.) was proposed that considers such situations. For cases with a malicious board, equated to instances where ballots can be fully manipulated, they defined vote privacy in [29] as an instance of mb – BPRIV where arbitrary tampering of honest ballots is permitted by the recovery algorithm. However, this approach would label all schemes as insecure, making the [29] model appear unfit for a comprehensive discussion on malicious boards.

Instead, their work focuses on a subset of schemes where security is guaranteed by the portion of the ballot securely delivered to the (honest) board, despite tampering attempts on other parts of the ballot by an adversary.

6.1.1 Original BPRIV notion

We consider a finite set $\mathscr{I} = \mathsf{H} \cup \mathsf{D}$ of voter identities, partitioned into the sets H and D of honest and dishonest voters. H is further partitioned into sets $\mathsf{H}_{\mathsf{check}}$ and $\mathsf{H}_{\overline{\mathsf{check}}}$, meant to contain the identities of voters who verify their vote (resp. do not verify).

Basically, we will relate two notions of vote privacy, showing that one implies the other. Having fixed the sets of honest and corrupted voters beforehand will then let us know that this implication holds when considering the same number of adversary-controlled voters for both notions, which would not be the case if the voters were dynamically corrupted. With dynamic corruption, one might need to resort to complexity-leveraging arguments with a corresponding loss in the estimate of the adversarial advantage.

An additional mild assumption is made regarding the format of the ballots: they are assumed to be pairs of bitstrings.

Definition 3. *(Bulletin board) A bulletin board* BB *is a list of pairs of bitstrings that we call ballots. For a ballot* (p,b)*, we call the bitstring* p *a public credential, and the bitstring* b *is a ciphertext.* $\mathsf{BB}[j]$ *denotes the* j*th element of* BB.

We will also call an extended bulletin board a board where elements are associated with an identity, i.e. a list of elements of the form $(id,(p,b))$.

While the exact nature of the ballots, i.e. public credentials and ciphertexts, depends on the protocol considered, we intuitively intend the public credential to link to the identity of the voter who casts the ballot, and the ciphertext to contain the vote expressed by the ballot.

We call counting functions the functions which calculate the result of an election. To be more general, we no longer assume a partial tallying property. In addition, we will no longer explicitly encode the revote policy in the games defining properties (e.g. having ballots cast by oracles replace previous ballots, to encode that the last vote counts).

Instead, the revote policy will be encoded in the counting function. That is, we now consider counting functions that 1. take the identities of voters into account, 2. apply to lists rather than multisets of votes. This way, any revote policy can be expressed by a counting function. We will express our results generically (for an arbitrary unspecified counting function ρ), so that they are independent of the choice of a particular revote policy.

Definition 4. *(Counting function) . A counting function* ρ *is a mapping that takes as input a sequence* S *of pairs* (id,v)*, where* $id \in \mathscr{I}$ *and* v *is a vote, and returns a value (bitstring) intended to represent the result of tallying the votes in* S*. It may use the ids contained in* S *to apply a revote policy. We assume a special value* $\perp$*, intended to represent an invalid identity and vote (e.g. obtained by decrypting a ballot that was incorrectly generated), that should not be counted. Formally, we require that counting functions ignore this value, i.e. that for all* l*,* l_0*,* $\rho(l||\perp||l_0) = \rho(l||l_0)$.

The original concept of BPRIV represents an honest ballot box, whereas the new variation considers a malicious one. To differentiate these two concepts, we refer to the original as hb – BPRIV security, denoting an "honest ballot box", while the concept introduced in the following section is labeled as mb – BPRIV, indicative of a "malicious ballot box".

hb – BPRIV: In the hb – BPRIV scenario, the adversary interacts with the execution of the voting protocol and observes either the real ballots or fake ones containing fraudulent votes, subject to a secret bit, β. With the aid of oracles, the adversary is able to select the values of the real and fake votes and cast any ballot they can craft, ostensibly on behalf of a corrupted voter. Eventually, regardless of the value of β, the adversary is presented with the result of the real ballot tally. They are then tasked with guessing β, with the privacy stipulation preventing them from doing so accurately. Essentially, this understanding of ballot privacy suggests that all honest voters' ballots might contain fraudulent votes during the election process, and the adversary wouldn't be able to detect this, hence avoiding any information leakage about the votes.

Nonetheless, a challenge arises: many protocols have a Tally algorithm that not only unveils the result but also related data, such as zero-knowledge proofs meant to validate that the result aligns correctly with the tally on the bulletin board. As previously explained, the adversary invariably observes the result of the real ballot tally. Consequently, presenting the adversary with a legitimate tally proof would enable them to differentiate between the two situations. In fact, if the adversary witnesses fake ballots, the result they

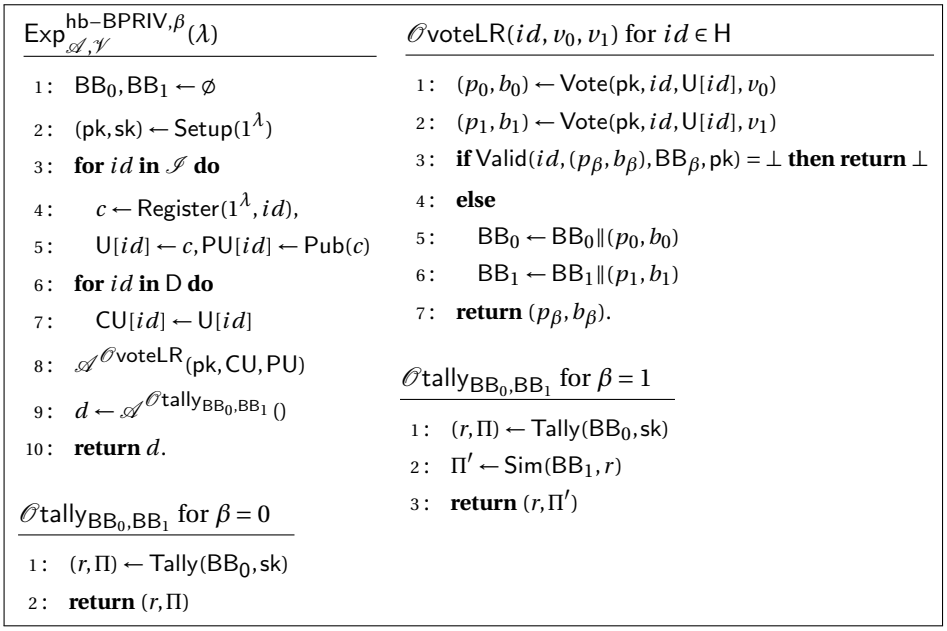


FIGURE 6.2: The hb-BPRIV game.

observe wouldn't align with the bulletin board's tally that they witness, which would become apparent if the adversary had access to the authentic proofs of correct tallying. To mitigate this issue, the authors in [22] mandate that the challenger in the hb – BPRIV game should be capable of simulating these proofs. Essentially, there should exist an algorithm $Sim(\mathsf{BB}, r)$ that intuitively yields a simulated proof that r is the accurate result for BB. This algorithm is employed to create a fake proof to show to the adversary when they come across fake ballots. It's crucial to note that the *Sim* algorithm is not explicitly assumed to generate a convincing fake proof, however, should it fail to do so, the hb – BPRIV property would likely be unsatisfiable. Moreover, the *Sim* algorithm only accesses public data visible to the adversary and does not have access to the election's private key, ensuring it cannot leak such confidential information to the adversary. Generally we can, the challenger can implement such a simulator for zero-knowledge proofs by manipulating the random oracle within the random oracle model. In this context, we gloss over these specifics and merely insist on the existence of the *Sim* algorithm, regardless of how it's implemented.

Formally, the hb – BPRIV property can be defined as:

Definition 5. *(*hb – BPRIV *[22]) Assume $\mathscr{V}$ to be a voting scheme. Consider the game* $\mathsf{Exp}^{\mathsf{hb-BPRIV},\beta}_{\mathscr{A},\mathscr{V}}$ *defined in Figure 6.2,* hb – BPRIV *is fulfilled by V if, for any polynomial adversary $\mathscr{A}$,*

$$|\mathbb{P}(\mathsf{Exp}^{\mathscr{A},\mathscr{V}}_{\mathsf{hb-BPRIV},0}(\lambda) = 1) - \mathbb{P}(\mathsf{Exp}^{\mathscr{A},\mathscr{V}}_{\mathsf{hb-BPRIV},1}(\lambda) = 1)|$$

is negligible in λ.

In the study [22], the authors elaborate on two additional attributes, strong consistency, and strong correctness, of voting systems. While these attributes are important to truly ensure vote privacy alongside hb – BPRIV, they do not directly contribute to the discussion in our context.

Their inclusion would further establish that alongside mb – BPRIV, they can prove an ideal functionality, rendering mb – BPRIV a noteworthy and sufficiently robust definition. However, it's important to note that such proof hasn't been established for du–mb–BPRIV. This presents an important avenue for future work.

The following part in which the voting scheme is defined leads to the next part where a new definition for ballot privacy for malicious ballot boxes by Cortier *et al.* [41], denoted as mb – BPRIV, is introduced.

Definition 6. *(Voting scheme) states that a voting scheme is an assembly of seven algorithms:*

$$\mathscr{V} = (\mathsf{Setup}, \mathsf{Register}, \mathsf{Pub}, \mathsf{Vote}, \mathsf{ValidBoard}, \mathsf{Tally}, \mathsf{Verify})$$

.

- *A pair of election keys* $(\mathsf{pk}, \mathsf{sk})$ *is computed by* $\mathsf{Setup}(1^\lambda)$*, given a security parameter* λ*.*
- *A private credential* c *for voter id is generated by* $\mathsf{Register}(1^\lambda, id)$*, and the correspondence* (id, c) *is stored in a list,* U*, for modeling purposes.*
- *The public credential,* pub *associated with a credential* c *is returned by* $\mathsf{Pub}(c)$*.*
- *A ballot* (p, b) *for user id with private credential* c*, containing vote* v*, is constructed by* $\mathsf{Vote}(\mathsf{pk}, id, c, v)$ *using the public election key* pk*. A state is also returned to the voter, modeling what a voter should record, such as her ballot. This state can be viewed as any information a voter would need to record, for example, to verify if the ballot has been cast.*
- *The validity of the board* BB *is checked by* $\mathsf{ValidBoard}(\mathsf{BB}, \mathsf{pk})$*.*
- *The result of the election, and potentially proofs of good tallying, are computed by* $\mathsf{Tally}(\mathsf{BB}, \mathsf{sk})$ *using the board* BB *and the secret election key* sk*.*
- *The checks that a voter id, with local state, should perform on a board* BB *to ensure her vote is counted are represented by* $\mathsf{Verify}(id, state, \mathsf{BB})$*.*

Let us now delve into the new ballot privacy definition given by Cortier *et al.* [41] and denoted as mb – BPRIV.

6.2 mb-BPRIV

Cortier *et al.* in [41] adapts definitions of ballot privacy and strong consistency property for taking into account the case of malicious ballot boxes.

6.2.1 Ballot Privacy In The Presence of Adversary: mb – BPRIV

Similar to hb – BPRIV, the adversary possesses partial information regarding the votes of honest users. For each such user, the adversary selects a left-or-right challenge consisting of two votes, v_0 and v_1. The game calculates the corresponding ballots for these votes, but it only reveals to the adversary the ballot corresponding to v_β, where β is a hidden bit that the adversary needs to determine. The game keeps track of both BB_0 and BB_1, which represent the ordered lists of ballots computed in response to the adversary's queries. Essentially, the adversary only sees BB_β.

The adversary proceeds to create a public bulletin board BB by utilizing the honest votes as well as arbitrary additional votes created by itself. This scenario models the adversary's control over the ballot box, which differentiates it from hb – BPRIV. In this case, the adversary has the freedom to manipulate

the contents of the ballot box in any manner by submitting any sequence of ballots it can construct, rather than being limited to adding only its own ballots.

The verification steps of the protocol are now incorporated. Prior to the tallying phase, the steps are performed by voters in $\mathsf{H}_{\mathsf{check}}$ to ensure that the board BB provided by the adversary accurately represents their votes. If any of these verifications fail, the election is terminated, and no tally is computed. The significance of this step lies in the fact that the ballot box is under the control of the adversary, who has the potential to manipulate its content, including the removal of ballots. These verifications guarantee that if the adversary desires to view the result, caution must be exercised to avoid upsetting the voters in $\mathsf{H}_{\mathsf{check}}$. If no verification steps are desired, $\mathsf{H}_{\mathsf{check}}$ can be set as an empty set. Assuming all voters in $\mathsf{H}_{\mathsf{check}}$ are satisfied with BB, the game proceeds to the tallying phase.

The defining aspect of the game lies in how it computes the tally returned to the adversary. In the "real" world, where the adversary observes BB_0, the game simply tallies BB, akin to $\mathsf{hb}-\mathsf{BPRIV}$. However, the case of the "fake" world, where the adversary sees BB_1, is considerably more intricate. In $\mathsf{hb}-\mathsf{BPRIV}$, the intuition was that the adversary should still receive the tally from the "real" board. Nevertheless, there is no longer a real and fake board distinction, as was the case in $\mathsf{hb}-\mathsf{BPRIV}$. The only available board is the one generated by the adversary.

As a result, the focus of this fundamental idea lies in determining how the votes on BB_1 were manipulated by the adversary to create the board that undergoes tallying. The primary objective is to efficiently establish the honest ballots that were cast, along with their corresponding positions on the bulletin board, and identify the removed ones. Once this transformation is determined, it is applied to BB_0 by the game, enabling the tallying process on the resulting board, and subsequently returning the outcome to the adversary. From a technical standpoint, the transformation that describes how the adversary constructs BB from BB_1 is represented as a selection function, and the recovery of this transformation is formalized as a recovery algorithm.

Definition 7 (Selection function)**.** *For $m, n \geq 1$, a selection function for m, n is any mapping*

$$\pi : [\![1, n]\!] \to [\![1, m]\!] \cup (\{0,1\}^* \times \{0,1\}^*)$$

Intuitively, π represents the steps that an opponent would take to create a bulletin board BB with n ballots from a board BB_1 with m ballots. For $i \in [\![1, n]\!]$, $\pi(i)$ shows how to construct $\mathsf{BB}[i]$:

- *$\pi(i) = j \in [\![1, m]\!]$ means this element is the jth from BB_1;*
- *$\pi(i) = (p, b)$ means that this element is (p, b).*

In more formal language:

Definition 8 (Applying a selection function to a board)**.** *Consider a function called π for $m, n \geq 1$. The function $\overline{\pi}$ is associated with a mapping that takes an extended board BB_0 with a length of m and maps it to a board $\overline{\pi}(\mathsf{BB}_0)$ with a length of n. This is done in such a way that for any j between 1 and n,*

$$\overline{\pi}(\mathsf{BB}_0)[j] = \begin{cases} (p, b) & \text{if } \pi(j) = i \text{ and } \mathsf{BB}_0[i] = (id, (p, b)) \\ (p, b) & \text{if } \pi(j) = (p, b) \end{cases} \tag{6.1}$$

The "recovery" algorithm, which is designed to discover the selection function employed by a potential adversary, requires two boards and several other data elements. This is because this information contains the intrinsic connection between voter identities and public credentials. However, it is important to elaborate on the broader context and intuition underlying the function of this recovery algorithm.

- In a scenario where a board is compromised, or malicious, privacy is no longer a guarantee. For instance, a manipulative voter could eliminate all votes except their own. Consequently, the identity of this voter could be learned from the tally. This exemplifies a situation where privacy could be breached in reality.
- Therefore, it's essential to consider and define specific manipulations that the adversary is permitted to enact. While these manipulations could potentially breach privacy in real-world situations, they are a necessary evil - for example, the deletion of votes.
- Our aim is to define privacy, accounting for these 'allowed' attacks. In other words, we want to ascertain if any undesirable outcomes occur, beyond the attacks we have already accepted as part of the system.
- In defining this privacy, the recovery function plays a crucial role. It is designed to detect permitted manipulations and generate a board where these do not result in attacks. Nevertheless, other manipulations, which are not a part of the allowed set, can still lead to attacks. The recovery function, therefore, ensures that the only breaches of privacy are the ones we've already accepted as a given risk, and mitigates any other potential breaches that could arise from unauthorized manipulations.

Definition 9 (Recovery algorithm)**.** *An algorithm that is capable of recovery is referred to as a recovery algorithm. This algorithm is able to, given a board* BB*, an extended board* BB_1*, and some additional data* d*, return a selection function for* $|\mathsf{BB}_1|$*,* n *for some* n*.*

Ballot privacy for a certain scheme $\mathscr{V}$ is formalized in this definition in a situation where the ballot box is controlled by the adversary $\mathscr{A}$. It's acknowledged that a fixed set of voter identities $\mathscr{I}$, divided into two sets H and D of honest and dishonest voters, is considered. A fixed subset $\mathsf{H}_{\mathsf{check}}$ of H is also presumed, whose users perform the checks the scheme anticipates to be executed before tallying.

The execution as shown in Figure 6.3 begins with the generation of the public key for the election and its related secret key (for tallying) (as before). Subsequently, several voters from an arbitrary set $\mathscr{I}$ are registered. It's assumed that the registration algorithm/protocol is performed for each user id in $\mathscr{I}$, and only the secret credential c and its associated public credential $\mathsf{Pub}(c)$ are recorded. Arrays U and PU are used to record these, while array CU is used to record the secret credentials for some arbitrary set of dishonest users D.

The public data pk, PU, and the corrupt credentials CU are given as input to the adversary. Access to a left-right voting oracle is also provided. Upon input of an identity id and two potential votes v_0 and v_1, two ballots are calculated by the oracle for id. The first ballot is recorded in list BB_0 and the second in list BB_1. It's important to note that a stateful voting algorithm is modeled, which is considered a necessary feature for considering voters who perform additional actions after voting. The resulting state is stored in arrays V_0 and V_1 for each user.

The voting phase is represented by this stage, where the ballots are submitted by the users. A bulletin board BB is prepared by the adversary, which is intended for tallying. If the bulletin board fails the validity test, tallying does not take place, and the adversary must make a guess at this point.

Control over the users who check via the oracle $\mathscr{O}\mathsf{verify}$ is given to the adversary. Arbitrary identities can be submitted by it to the oracle. The set of users who have checked is recorded in the variable $\mathsf{Checked}$,

$\mathsf{Exp}^{\mathsf{mb-BPRIV},\mathsf{Recover},\beta}_{\mathscr{A},\mathscr{V}}(\lambda)$

1: $\mathsf{V}_0, \mathsf{V}_1, \mathsf{Checked}, \mathsf{Happy} \leftarrow \emptyset$
2: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$
3: **for** id **in** $\mathscr{I}$ **do**
4: $\quad c \leftarrow \mathsf{Register}(1^\lambda, \mathrm{id})$,
5: $\quad \mathsf{U}[\mathrm{id}] \leftarrow c, \mathsf{PU}[\mathrm{id}] \leftarrow \mathsf{Pub}(c)$
6: $\mathsf{BB} \leftarrow \mathscr{A}^{\mathscr{O}\mathsf{voteLR}}(\mathsf{pk}, \mathsf{CU}, \mathsf{PU})$
7: **if** $\mathsf{H}_{\mathsf{check}} \not\subseteq \mathsf{V}_0, \mathsf{V}_1$ **then return** $\perp$
8: **if** $\mathsf{Valid}(\mathsf{BB}, \mathsf{pk}) = \perp$ **then**
9: $\quad d \leftarrow \mathscr{A}()$; **return** d
10: $\mathscr{A}^{\mathscr{O}\mathsf{verify}_{\mathsf{BB}}}()$
11: **if** $\mathsf{H}_{\mathsf{check}} \not\subseteq \mathsf{Checked}$ **then return** $\perp$
12: **if** $\mathsf{H}_{\mathsf{check}} \not\subseteq \mathsf{Happy}$ **then return** $d \leftarrow \mathscr{A}()$;
13: **if** $\mathsf{H}_{\mathsf{check}} \subseteq \mathsf{Happy}$ **then** $d \leftarrow \mathscr{A}^{\mathscr{O}_{\mathsf{BB},\mathsf{BB}_0,\mathsf{BB}_1}}()$
14: **return** d.

$\mathscr{O}\mathsf{verify}_{\mathsf{BB}}(\mathrm{id})$ for $\mathrm{id} \in \mathsf{H}_{\mathsf{check}}$

1: $\mathsf{Checked} \leftarrow \mathsf{Checked} \cup \{\mathrm{id}\}$
2: **if** $\mathsf{Verify}(\mathrm{id}, \mathsf{V}_\beta[\mathrm{id}], \mathsf{BB}) = \top$ **then**
3: $\quad \mathsf{Happy} \leftarrow \mathsf{Happy} \cup \{\mathrm{id}\}$

$\mathscr{O}\mathsf{voteLR}(\mathrm{id}, v_0, v_1)$

1: **if** $\mathrm{id} \notin \mathsf{H}$ **then return** $\perp$
2: **else**
3: $\quad (p_0, b_0, \mathsf{state}_0) \leftarrow \mathsf{Vote}(\mathsf{pk}, \mathrm{id}, \mathsf{U}[\mathrm{id}], pc, v_0)$
4: $\quad (p_1, b_1, \mathsf{state}_1) \leftarrow \mathsf{Vote}(\mathsf{pk}, \mathrm{id}, \mathsf{U}[\mathrm{id}], pc, v_1)$
5: $\quad \mathsf{V}_0[\mathrm{id}] \leftarrow \mathsf{state}_0, \mathsf{V}_1[\mathrm{id}] \leftarrow \mathsf{state}_1$
6: $\quad \mathsf{BB}_0 \leftarrow \mathsf{BB}_0 \| (\mathrm{id}, (p_0, b_0))$
7: $\quad \mathsf{BB}_1 \leftarrow \mathsf{BB}_1 \| (\mathrm{id}, (p_1, b_1))$
8: $\quad$ **return** (p_β, b_β).

$\mathscr{O}\mathsf{tally}_{\mathsf{BB},\mathsf{BB}_0,\mathsf{BB}_1}$ for $\beta = 0$

1: $(r, \Pi) \leftarrow \mathsf{Tally}(\mathsf{BB}, \mathsf{pd}, \mathsf{sd})$
2: **return** (r, Π)

$\mathscr{O}\mathsf{tally}_{\mathsf{BB},\mathsf{BB}_0,\mathsf{BB}_1}$ for $\beta = 1$

1: $\pi \leftarrow \mathsf{Recover}_{\mathsf{U}}(\mathsf{BB}_1, \mathsf{BB})$
2: $\mathsf{BB}' \leftarrow \overline{\pi}(\mathsf{BB}_0)$
3: $(r, \Pi) \leftarrow \mathsf{Tally}(\mathsf{BB}', \mathsf{sk})$
4: $\Pi' \leftarrow \mathsf{SimProof}(\mathsf{BB}, r)$
5: **return** (r, Π')

FIGURE 6.3: The mb-BPRIV game.

and the set of users for whom the check was successful is recorded in the variable Happy. The game is stopped without allowing $\mathscr{A}$ to make a guess if Checked does not contain $\mathsf{H}_{\mathsf{check}}$.

If the voters who should check do check successfully, the tally of the election is shown to the adversary. However, if some voters are not satisfied with their verifications, the election is called off, which means the adversary must make a guess without seeing the tally.

The way the experiment calculates the tally is an important part of this definition. In the real execution (i.e., $\beta = 0$), the tally is executed on BB. In the fake execution (i.e., $\beta = 1$), the Recover algorithm is employed first by the tally to determine how the votes it has seen (i.e., BB_1) have been tampered with by the adversary to produce the board it asks to be tallied. The influence of the choice of the Recover algorithm on the security level guaranteed by this definition will be explained later. Then, the transformation obtained this way is applied to BB_0 by the game. The resulting board is tallied, and the result, along with a simulated proof (as in hb – BPRIV), is returned to the adversary.

Definition 10. *(*mb – BPRIV *with respect to a recovery algorithm [41]). Let $\mathscr{V}$ represent a voting scheme, and let* Recover *be a recovery algorithm. The game* $\mathsf{Exp}^{\mathsf{mb-BPRIV},\mathsf{Recover},\beta}_{\mathscr{A},\mathscr{V}}$*, as depicted in Figure 6.3, is considered. $\mathscr{V}$ is said to satisfy* mb – BPRIV *with respect to* Recover *if, for any polynomial adversary $\mathscr{A}$, the difference between the following probabilities is negligible in λ.*

$$\left|\mathbb{P}(\mathsf{Exp}^{\mathsf{mb-BPRIV},\mathsf{Recover},0}_{\mathscr{A},\mathscr{V}}(\lambda) = 1) - \mathbb{P}(\mathsf{Exp}^{\mathsf{mb-BPRIV},\mathsf{Recover},1}_{\mathscr{A},\mathscr{V}}(\lambda) = 1)\right|$$

6.3 Delay-Use Malicious Ballot Privacy: du – mb – BPRIV

In the mb – BPRIV, a tally is taken if and only if the adversary's board passes the validity test and all voters are satisfied after carrying out a specific verification procedure. This precondition necessitates that the verification process be completed before the tally, rendering mb – BPRIV incompatible with voting systems such as Selene [108]. In Selene, the tally is taken before the verification to mitigate coercion risks. Consequently, there is a need for a new privacy definition that can cope with potentially dishonest voting servers (and by extension, the ballot box), and also supports voting protocols where the verification is done after the tally has been computed.

In this section, we introduce a new definition of ballot privacy against a potentially dishonest ballot box, which we label as delay-use malicious ballot box BB ballot privacy (du–mb–BPRIV)(See [47]). Fundamentally, this definition expands the range of voting systems to include those that allow verification after tallying and those that require a secret key during the verification step. This new definition bears similarities to mb – BPRIV, with the most difference being that in our definition, the adversary is only allowed to view the tally after the validity of his board is confirmed, and subsequently, the outcome of the verification phase is revealed. Figure 6.4 provides a detailed account of the security game for du–mb–BPRIV.

The security experiment begins with the generation of some public and secret data pd and sd (which typically includes the public and secret keys used to encrypt and decrypt the votes). Then, a number of voters in a set $\mathscr{I}$ are registered. In the registration phase, public and secret credentials pc and sc are generated for each voter, and stored in finite maps PU and U, respectively. Furthermore, we store the secret credentials of a set D of dishonest voters in a finite map CU.

The adversary is now given pd, PU and CU as input. In addition, he gets access to a vote oracle, that on input (id, v_0, v_1) computes two ballots for this user. The first ballot is stored in a list BB_0 and the second ballot is stored in a list BB_1. The vote oracle also records a state, containing any information the voter later needs to verify that her vote was correctly cast and counted; we split the state into two components. The first component ($\mathsf{state}_{\mathsf{pre}}$) covers information checked which is generated before tallying and the second component ($\mathsf{state}_{\mathsf{post}}$) covers information generated after tallying; this is necessary due to recovery which ensures that information after tallying (including the tally) always acts as if $\beta = 0$. The adversary may also call on a publish oracle, allowing him to see, essentially, BB_β for a secret bit β.

Using this information, the adversary creates a public bulletin board BB. If the board is invalid, we output a random bit. If the board is valid, we allow the adversary to make an initial guess at the secret bit β, based on the information he has seen so far. This guess is stored in a variable d^*, possibly to be returned by the experiment at a later point.

The adversary now gets access to the tally and is allowed to add some extra information to the bulletin board, namely a result r^* and a proof Π^* that this result corresponds to the votes. If this result and proof fail to pass verification, we output a random bit.

Otherwise, the adversary gets to make a guess d, given access to a verification oracle $\mathscr{O}$verify. The verification oracle records the users who have verified in a set called Checked, and the users who are happy with the verification are recorded in the set Happy. If anyone who we expect should verify actually does not verify, we output a random bit. If a voter is unhappy with the verification process, we output the initial guess d^* the adversary made before seeing the tally. Otherwise, the experiment outputs the guess d that the adversary made after calling the verify oracle.

When given access to the tally oracle, the adversary can call this oracle only once, and the behavior of the tally oracle depends on whether we are in the left world ($\beta = 0$) or in the right world ($\beta = 1$). If we are in the left world, the tally is performed directly on the board BB created by the adversary. The adversary then gets to see a real result and proof of the correct tally. In the right world, however, we first run the recovery algorithm to detect how the adversary has tampered with the ballots in BB_1, to create BB. We

$\mathsf{Exp}^{\mathsf{du\text{–}mb\text{–}BPRIV},\mathsf{Recover},\beta}_{\mathscr{A},\mathscr{V},\mathsf{Sim}}(\lambda)$

1: $\mathsf{Checked}, \mathsf{Happy} \leftarrow \emptyset$
2: $\mathsf{V}, \mathsf{PU}, \mathsf{U}, \mathsf{CU} \leftarrow$ empty
3: $(\mathsf{pd}, \mathsf{sd}) \leftarrow \mathsf{Setup}(\lambda)$
4: **for** id **in** $\mathscr{I}$ **do**
5: $\quad (pc, sc) \leftarrow \mathsf{Register}(\mathrm{id})$,
6: $\quad \mathsf{U}[\mathrm{id}] \leftarrow sc, \mathsf{PU}[\mathrm{id}] \leftarrow pc$
7: $\quad$ **if** id $\in \mathsf{D}$ **then** $\mathsf{CU}[\mathrm{id}] \leftarrow \mathsf{U}[\mathrm{id}]$
8: $\mathsf{BB} \leftarrow \mathscr{A}^{\mathscr{O}\mathsf{voteLR}}(\mathsf{pd}, \mathsf{CU}, \mathsf{PU}, \mathsf{H}_{\mathsf{check}})$
9: **if** $\mathsf{H}_{\mathsf{check}} \not\subseteq \mathsf{V}$ **then** $d \leftarrow_{\$} \{0,1\}$; **return** d
10: **if** $\mathsf{ValidBoard}(\mathsf{BB}, \mathsf{pk}) = \bot$ **then**
11: $\quad d \leftarrow_{\$} \{0,1\}$; **return** d
12: $d^* \leftarrow \mathscr{A}()$;
13: $(r^*, \Pi^*) \leftarrow \mathscr{A}^{\mathscr{O}\mathsf{tally}_{\mathsf{BB},\mathsf{BB}_0,\mathsf{BB}_1}()}$
14: **if** $\mathsf{VerifyTally}((\mathsf{pd}, pbb, r^*), \Pi^*) = \bot$ **then**
15: $\quad d \leftarrow_{\$} \{0,1\}$; **return** d
16: $d \leftarrow \mathscr{A}^{\mathscr{O}\mathsf{verify}}()$
17: **if** $\mathsf{H}_{\mathsf{check}} \not\subseteq \mathsf{Checked}$ **then**
18: $\quad d \leftarrow_{\$} \{0,1\}$; **return** d
19: **if** $\mathsf{H}_{\mathsf{check}} \not\subseteq \mathsf{Happy}$ **then return** d^*
20: **return** d

$\mathscr{O}$board

1: **return** $\mathsf{Publish}(\mathsf{BB}_\beta)$

$\mathscr{O}\mathsf{voteLR}(\mathrm{id}, v_0, v_1)$

1: **if** id $\in \mathsf{H}$ **then**
2: $\quad (pc, b_0, \mathsf{state}_{\mathsf{pre},0}, \mathsf{state}_{\mathsf{post},0}) \leftarrow \mathsf{Vote}(\mathsf{pd}, pc, v_0)$
3: $\quad (pc, b_1, \mathsf{state}_{\mathsf{pre},1}, \mathsf{state}_{\mathsf{post},1}) \leftarrow \mathsf{Vote}(\mathsf{pd}, pc, v_1)$
4: $\quad \mathsf{V}[\mathrm{id}] \leftarrow \mathsf{V}[\mathrm{id}] \| (\mathsf{state}_{\mathsf{pre},\beta}, \mathsf{state}_{\mathsf{post},0}, v_0)$
5: $\quad \mathsf{BB}_0 \leftarrow \mathsf{BB}_0 \| (\mathrm{id}, (pc, b_0))$
6: $\quad \mathsf{BB}_1 \leftarrow \mathsf{BB}_1 \| (\mathrm{id}, (pc, b_1))$

$\mathscr{O}\mathsf{tally}_{\mathsf{BB},\mathsf{BB}_0,\mathsf{BB}_1}$ for $\beta = 0$

1: $(r, \Pi) \leftarrow \mathsf{Tally}(\mathsf{BB}, \mathsf{pd}, \mathsf{sd})$
2: **return** (r, Π)

$\mathscr{O}\mathsf{tally}_{\mathsf{BB},\mathsf{BB}_0,\mathsf{BB}_1}$ for $\beta = 1$

1: $\mathsf{BB}' \leftarrow \mathsf{Recover}(\mathsf{BB}, \mathsf{BB}_0, \mathsf{BB}_1)$
2: $(r, \Pi) \leftarrow \mathsf{Tally}(\mathsf{BB}', \mathsf{pd}, \mathsf{sd})$
3: $\Pi' \leftarrow \mathsf{Sim}(\mathsf{pd}, \mathsf{Publish}(\mathsf{BB}), r)$
4: **return** (r, Π')

$\mathscr{O}\mathsf{verify}$ for id $\in \mathsf{H}_{\mathsf{check}}$

1: $\mathsf{Checked} \leftarrow \mathsf{Checked} \cup \{\mathrm{id}\}$
2: **if** $\mathsf{VerifyVote}(\mathrm{id}, \mathsf{state}_{\mathsf{pre}}, \mathsf{state}_{\mathsf{post}}, \mathsf{BB}, pc, sc) = \top$ **then**
3: $\quad \mathsf{Happy} \leftarrow \mathsf{Happy} \cup \{\mathrm{id}\}$
4: **return** Happy

FIGURE 6.4: The new security notion for ballot privacy against a dishonest ballot box.

then change the ballots on BB_0 accordingly, yielding a new board BB', which we tally. The adversary then gets to see the result r corresponding to BB' and a simulated proof Π' of correct tally, with respect to r and the adversarial board BB.

Definition 11. *Let $\mathscr{V}$ be a voting system, and let* Recover *be a recovery algorithm. We say that $\mathscr{V}$ satisfies* du–mb–BPRIV *with respect to* Recover *if there exists an efficient simulator* Sim*, such that for any efficient adversary $\mathscr{A}$,*

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

is negligible in the security parameter λ, where $\mathsf{Exp}^{\mathsf{du\text{–}mb\text{–}BPRIV},\mathsf{Recover},\beta}_{\mathscr{A},\mathscr{V},\mathsf{Sim}}$ is the game defined in Figure. 6.4.

6.3.1 Recovery function

The mb–BPRIV definition is well-defined for many recovery functions, but three are suggested in the body of [41]. All three recovery functions when used with mb–BPRIV are only satisfied by schemes that fulfill strong constraints on their verification procedure, and two of them presume that voters are authenticated.

We introduce a simple new recovery function (Fig. 6.5) which we call $\mathsf{Recover}_{\mathsf{U}}^{\mathsf{del,reorder'}}$ which is essentially identical to $\mathsf{Recover}_{\mathsf{U}}^{\mathsf{del,reorder}}$ and $\mathsf{Recover}_{\mathsf{U}}^{\mathsf{del,reorder,change}}$ in [41], but does not require ballot authentication and without explicit constraints on the verification procedure. mb – BPRIV with the recovery functions originally suggested implies the satisfying scheme is equivalent to some ideal functionalities. Future versions of du–mb–BPRIV may wish to prove similar results in which a different recovery function is likely necessary.

$\mathsf{Recover}_{\mathsf{U}}^{\mathsf{del,reorder'}}(\mathsf{BB}_1, \mathsf{BB})$

1: $L \leftarrow [\,]$
2: **for** $(pc, c) \in \mathsf{BB}$:
3: **if** $\exists j, \mathrm{id} : \mathsf{BB}_1[j] = (\mathrm{id}, (pc, c))$:
4: $L \leftarrow L \parallel j$ (if several such j exist, pick the first one)
5: **else** :
6: $L \leftarrow L \parallel (pc, c)$
7: **return** $(\lambda i.\ L[i])$

FIGURE 6.5: The $\mathsf{Recover}_{\mathsf{U}}^{\mathsf{del,reorder'}}$ algorithm.

6.3.2 Comparison of du–mb–BPRIV to mb–BPRIV

In this section, we briefly analyze the differences between the mb – BPRIV definition and our new du–mb–BPRIV definition. As mentioned above, the main difference is that the verification oracle is first available after the tally oracle has been called. This accommodates schemes where the verification first happens after tally and allows a secret key to be used for the verification process, however, it naturally also applies to schemes with early verification. We have changed some parts of the definition to adapt to the delayed use of the verification, but also to make it optimized and precise enough for EASYCRYPT.

The main differences are:

- We only have one voter map V but the state stored depends on secret bit β, see line 4 in the definition of $\mathcal{O}$voteLR in Figure 6.1. However, the state is split into a part relevant before tallying and a post-tally part (only relating to $\beta = 0$ which is the board used for tallying). This is necessary for the state handling in EASYCRYPT oracles and was not necessary for mb – BPRIV since the stateful verification happened before tallying.

- If the adversary does not output a valid board, the experiment outputs a random guess bit, whereas mb – BPRIV allows the adversary to output a guess but without tally access. In both cases, this corresponds to real life, where a board will not be tallied if it is not valid. Outputting a random bit makes our proofs in EASYCRYPT slightly easier, and it is actually equivalent to mb – BPRIV unless the ValidBoard algorithm always outputs $\perp$.

- In our definition the experiment also outputs a random guess bit if the verification of the tally fails via the algorithm VerifyTally. Again, this corresponds to not allowing any adversarial advantage when the tally fails publicly. This case was not explicitly considered in mb – BPRIV but is natural in our case where the verification step will not proceed on an invalid tally output.

- In du–mb–BPRIV the verification status of the honest voters contained in Happy is directly output to the adversary. In mb – BPRIV this is not defined precisely. In both definitions, the appearance of failed verifications inside the set of checking honest voters $\mathsf{H}_{\mathsf{check}}$ will in any case imply that the

guess bit has to be determined without knowing the tally result. This is to punish the adversary for creating a board with verifications failing which would cause complaints in real-life protocols.

Consider a voting scheme where the verification step does not depend on a secret key and can be done before the tally. For such schemes, both privacy definitions can be applied to the scheme. We claim that under reasonable conditions our definition is stronger. Further, for voting schemes where the outcome of the individual verifications can be computed by the adversary using the data from the bulletin board, as e.g. happens in Helios, we have the equivalence of the two definitions. We now sketch why this is the case.

We show that du–mb–BPRIV privacy implies mb – BPRIV assuming that ValidBoard does not constantly output $\perp$ and that VerifyTally never fails on an honestly computed tally. Finally, we also assume that the verification status Happy is not output to the adversary in mb – BPRIV or that the verification status can be computed using public data, as e.g. happens in Helios. To prove the implication we assume that we have an attack algorithm for mb – BPRIV. We use the vote choices from the mb – BPRIV algorithm. If the attack algorithm outputs an invalid board, we change the board that is being output to a valid board which we have assumed exists. Since the tally also does not fail we are allowed to output a guess and we use the one from the attack algorithm and will win with the same advantage since in this case no verification will be done in mb – BPRIV. If the attack algorithm outputs a valid board, we use the same board in the du–mb–BPRIV experiment. In du–mb–BPRIV line 12 the adversary has to output a guess bit d^* which will be used if verification fails for the honest verifier set $\mathsf{H}_{\mathsf{check}}$. Here we use what will be output from the mb – BPRIV attack algorithm in case of failed verifications, which by assumption either does not depend on Happy, which we do not yet have access to at this stage in the du–mb–BPRIV experiment, or it can be computed from the public data on the board. In the experiment du–mb–BPRIV line 13 we now get the tally before verification (and the tally verifies by assumption) but we ignore this at first and choose the same verifying voters as in the attack algorithm. At this point, the two experiments will be equivalent. If the verification fails we are forced to go back to d^* but this was from the attack algorithm and will be equivalently successful. If no verification fails in $\mathsf{H}_{\mathsf{check}}$ then we output the guess from the attack algorithm using the tally result we got earlier as input. The advantage will be the same as for the mb – BPRIV attack algorithm. This was the important implication direction since it demonstrates that our definition is not too weak, i.e. if an early verification scheme is declared du–mb–BPRIV then it is also mb – BPRIV private which in turn has ideal functionality implications under assumptions such as strong consistency [41].

Considering whether mb – BPRIV privacy implies du–mb–BPRIV, the main problem is that the choice of verifying voters in du–mb–BPRIV could depend on the tally output. However, for voting schemes where the outcome of the honest voters' individual verification can be computed by the adversary, we can prove the implication. We thus assume we have an attack algorithm for du–mb–BPRIV with a non-negligible advantage. In the experiment mb – BPRIV we use the votes and output the board from this attack algorithm. If the board is not valid, we simply output a random bit in the experiment mb – BPRIV, which is equivalent to what happens in du–mb–BPRIV. If the board is valid, the next step in mb – BPRIV is to use the verification oracle. Here we simply let all the required honest voters $\mathsf{H}_{\mathsf{check}}$ perform verifications. These will also all have to verify in the attack against du–mb–BPRIV to get an advantage since the experiment will otherwise output a random bit. If a verification fails we will use the output d^* in the attack algorithm which was computed without tally access. If none of the verifications fail, we will get tally access in the experiment mb – BPRIV. If more voters were chosen to verify in the attack algorithm we can compute the outcomes by assumption. If we encounter a failure in the verification here, we again output d^*, otherwise the output from the attack algorithm. The experiments will be equivalent at this point.

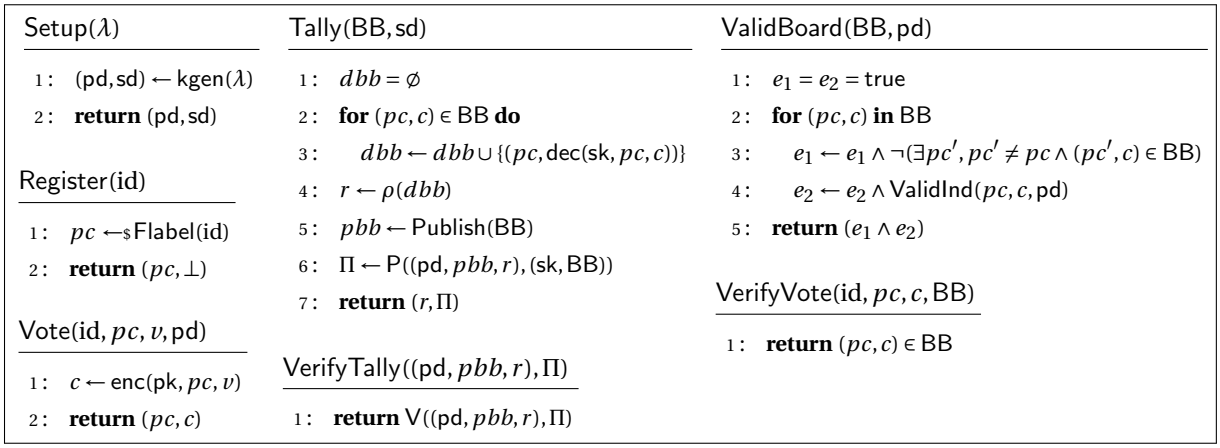


FIGURE 6.6: Algorithms defining the Labeled-MiniVoting scheme for the labeled PKE $\mathsf{E} = (\mathsf{kgen}, \mathsf{enc}, \mathsf{dec})$, and the proof system $\Sigma = (\mathsf{P}, \mathsf{V})$.

6.4 Labeled-MiniVoting

As we delve into the section on Labeled-MiniVoting, we need to discuss several concepts that contribute to a fuller understanding of this topic. It is essential to grasp these procedures before we venture into Labeled Public Key Encryption (LPKE), which extends the traditional notion of Public Key Encryption by adding an additional, non-malleable data known as a label. Furthermore, we will also delve into the specifics of 'indistinguishability under chosen ciphertext attack with one parallel decryption query' (IND–1–CCA), an important security notion for labeled PKE.

In addition, we will dissect the concept of Proof Systems, a pair of efficient algorithms used in Selene to ascertain that various operations are executed correctly. The proof system involves a prover and a verifier and forms a cornerstone of the cryptographic procedures in the Labeled-MiniVoting scheme.

All these will allow us to form an understanding of the Labeled-MiniVoting scheme before we move to the formal definitions.

6.4.1 Labeled Public Key Encryption

A *labeled public key encryption scheme (LPKE)* extends the notion of a "regular" PKE by adding some additional, non-malleable data called a *label* [115]. One important property for a labeled PKE is that decrypting a ciphertext using a different label than the one used for encryption, should not reveal anything about the original plaintext. Formally, a labeled PKE is defined similarly to how we define a PKE in Section 2.1.4, but a label ℓ is given as additional input in the encryption and decryption algorithms.

The security of the schemes we analyze relies on a security notion for labeled public key encryption called *indistinguishability under chosen ciphertext attack with one parallel decryption query* (IND–1–CCA) [10]. Informally, this notion captures that any efficient adversary is unable to distinguish between encryptions of two messages of the same length when given access to a batch decryption oracle that can be called once.

A similar security notion, namely poly–IND–1–CCA, allows the adversary to make up to n challenge queries, for some polynomially bounded integer n. This notion is formalized in Figure 6.7, where an adversary $\mathscr{B}$ is given access to an encryption oracle $\mathscr{O}_{\mathsf{enc}}$ and a decryption oracle $\mathscr{O}_{\mathsf{dec}}$. The adversary can make n challenge queries to $\mathscr{O}_{\mathsf{enc}}$, who encrypts one of two plaintexts, depending on the bit β. The adversary can query $\mathscr{O}_{\mathsf{dec}}$ at most once, and the oracle then decrypts a list of ciphertexts. For any ciphertexts created by the encryption oracle, the decryption oracle returns $\perp$.

$\mathsf{Exp}^{\mathsf{poly\text{–}ind1cca},\beta}_{\mathscr{B},E,n}(\lambda)$

1: $encL \leftarrow [\,]$
2: $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{kgen}(\lambda)$
3: $\beta' \leftarrow \mathscr{B}^{\mathscr{O}_{\mathsf{enc}},\mathscr{O}_{\mathsf{dec}}}(\mathsf{pk})$
4: **return** β'

$\mathscr{O}_{\mathsf{enc}}(\ell, m_0, m_1)$

1: $c \leftarrow \perp$
2: **if** $|\mathsf{encL}| < n$ **then**
3: $\quad c \leftarrow \mathsf{enc}(\mathsf{pk}, \ell, m_\beta)$
4: $\quad \mathsf{encL} \leftarrow \mathsf{encL} + [(c,\ell)]$
5: **return** c

$\mathscr{O}_{\mathsf{dec}}(cL)$

1: $mL \leftarrow [\,]$
2: **for** $(c,\ell) \in cL$ **do**
3: $\quad$ **if** $(c,\ell) \notin \mathsf{encL}$ **then**
4: $\quad\quad mL \leftarrow mL + [\mathsf{dec}(\mathsf{sk},\ell,c)]$
5: $\quad$ **else** $mL \leftarrow mL + [\perp]$
6: **return** mL

FIGURE 6.7: Security experiment for poly–IND–1–CCA [43]

The advantage of a poly–IND–1–CCA adversary $\mathscr{B}$ against a labeled public key encryption scheme E is defined as

$$\mathsf{Adv}^{\mathsf{poly\text{–}ind1cca}}_{\mathscr{B},\mathsf{E},n}(\lambda) = \left|\Pr\left[\mathsf{Exp}^{\mathsf{poly\text{–}ind1cca},0}_{\mathscr{B},\mathsf{E},n}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{poly\text{–}ind1cca},1}_{\mathscr{B},\mathsf{E},n}(\lambda) = 1\right]\right|,$$

and we say that the labeled PKE E is n-challenge poly–IND–1–CCA-secure if the advantage defined above is negligible in λ for all efficient adversaries $\mathscr{B}$.

As noted in [43], if $n = 1$, poly–IND–1–CCA security is essentially reduced to IND–1–CCA security. Indeed, it is possible to prove, through a hybrid argument, that a labeled PKE is IND–1–CCA secure if and only if it is poly–IND–1–CCA secure. This fact was also verified in EASYCRYPT [43], and we were able to reuse this framework in our formalization of the ballot privacy of Selene.

6.4.2 Proof Systems

We now describe proof systems that are used in Selene to ensure that various operations are performed correctly. We say that a binary relation $\mathscr{R}$ is a subset $\mathscr{R} \subseteq X \times W$, where X is a set of *statements* and W is a set of *witnesses*. A *proof system* for the relation $\mathscr{R}$ is a pair of efficient algorithms (P,V), where P is called the *prover* and V is called the *verifier*. The prover and verifier work on a common input $x \in X$, and the prover has some additional input $w \in W$. In a *non-interactive proof system*, P uses his input to compute a proof Π. He sends the proof to V, who, on input (x,Π) produces a verification output in $\{0,1\}$.

- A proof system is said to be *complete* if the prover can produce a valid proof whenever the statement is true. More formally, for any $(x,w) \in \mathscr{R}$, if Π is a proof output by $\mathsf{P}(x,w)$, then $\mathsf{V}(x,\Pi)$ outputs 1 with probability 1.
- A proof system is *sound* if a prover is unable to convince a verifier that a false statement is true.
- A proof system is *zero-knowledge* if the proof leaks no information beyond the fact that the relation holds. More formally, we demand the existence of an efficient algorithm Sim, called the *simulator*, that produces valid-looking proofs for a statement $x \in X$ without access to the witness w. Formally, we consider a zero-knowledge adversary $\mathscr{B}$ in the following experiments:

$\mathsf{Exp}^{zk,0}_{\mathscr{B},\mathsf{P},\mathscr{R}}(\lambda)$	$\mathsf{Exp}^{zk,1}_{\mathscr{B},\mathsf{Sim},\mathscr{R}}(\lambda)$
1 : $(x, w, \mathsf{state}) \leftarrow \mathscr{B}(\lambda)$	1 : $(x, w, \mathsf{state}) \leftarrow \mathscr{B}(\lambda)$
2 : $\Pi \leftarrow \bot$	2 : $\Pi \leftarrow \bot$
3 : **if** $(\mathscr{R}(x, w))$ **then**	3 : **if** $(\mathscr{R}(x, w))$ **then**
4 : $\quad \Pi \leftarrow \mathsf{P}(x, w)$	4 : $\quad \Pi \leftarrow \mathsf{Sim}(x)$
5 : $\beta' \leftarrow \mathscr{B}(\mathsf{state}, \Pi)$	5 : $\beta' \leftarrow \mathscr{B}(\mathsf{state}, \Pi)$
6 : **return** β'	6 : **return** β'

FIGURE 6.8: Games for zero-knowledge adversary

The advantage of the zero-knowledge adversary $\mathscr{B}$ over the proof system (P, V) and simulator Sim is defined as (See Figure 6.8)

$$\mathsf{Adv}^{\mathsf{zk}}_{\mathscr{B},\mathsf{P},\mathsf{Sim},\mathscr{R}}(\lambda) = \left|\Pr\left[\mathsf{Exp}^{zk,0}_{\mathscr{B},\mathsf{P},\mathscr{R}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{zk,1}_{\mathscr{B},\mathsf{Sim},\mathscr{R}}(\lambda) = 1\right]\right|,$$

and we say that a proof system is zero-knowledge if, for any adversary $\mathscr{B}$, there exists a simulator Sim such that the advantage defined above is negligible.

6.4.3 Voting-Friendly Relation

In the voting systems, proof systems are used to compute and validate proofs of the correct tally. In our analysis and EASYCRYPT formalization, we keep the relation $\mathscr{R}$ abstract, and thus, we need to ensure that the relation is compatible with the result of the election. For this, we adopt the notion of voting-friendly relations, as defined by Cortier *et al.* [43] and generalize it a bit, so that it also accommodates schemes like Selene. Very informally, a relationship is voting friendly if for any adversarially chosen bulletin board, it is possible to find a corresponding tally such that the pair (bulletin board and tally) are in the language. The relation being compatible with the result of the election means that if $\mathscr{V}$ is a voting system, $(\mathsf{pd}, \mathsf{sd})$ is public and secret data generated by the Setup algorithm, the result r of the election corresponds to the tally of the votes obtained by decrypting the ciphertexts in the ballot box BB, and if pbb is the public part of BB, then the relation $\mathscr{R}((\mathsf{pk}, pbb, r), (\mathsf{sk}, \mathsf{BB}))$ holds. In other words, it is possible to prove that r is the correct result. More formally, a voting-friendly relation is defined as follows:

Definition 12. *This definition introduces the concept of a voting-friendly relation with regard to a labeled public-key encryption scheme $\mathscr{E}$ and a proof system $\sum_{\mathscr{R}}$ for some relation $\mathscr{R}$. Given the abstract algorithms ρ and* Publish*, the definition specifies an experiment as in Figure 6.9. We say that $\mathscr{R}$ is a voting-friendly relation with respect to ρ and* Publish*, if, for any efficient adversary $\mathscr{B}$, this experiment returns 1 with negligible probability. In the above experiment, the algorithm* Dec^* *decrypts the ballot box* BB *line by line using the secret data* sd*, and returns a list $[(a_1, v_1), \ldots, (a_n, v_n)]$ of votes v_i and some additional information a_i, e.g. voter identities as in MiniVoting or tracking numbers as in Selene.*

The pivotal point of this definition lies in the interaction between the efficient adversary $\mathscr{B}$ and the voting-friendly relation. The adversary $\mathscr{B}$ takes on the role of creating the ballot box BB. Without the assumption of a voting-friendly relation, we could not consistently tally the votes and output valid proof. This emphasizes the significance of the voting-friendly relation.

$\mathsf{Exp}^{\mathsf{vfr}}_{\mathscr{B},\mathscr{E},\Sigma_{\mathscr{B}},\rho,\mathsf{Publish}}(\lambda)$

1: $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
2: $\mathsf{BB} \leftarrow \mathscr{B}(1^\lambda,\mathsf{pk})$
3: $dbb \leftarrow \mathsf{Dec}^*(\mathsf{sk},\mathsf{BB})$
4: $r \leftarrow \rho(dbb)$
5: $pbb \leftarrow \mathsf{Publish}(\mathsf{BB})$
6: **return** $\neg\mathscr{R}((\mathsf{pk},pbb,r),(\mathsf{sk},\mathsf{BB}))$

FIGURE 6.9: Voting-Friendly Experiment.

6.4.4 Formal Definition

The initial concept of the MiniVoting scheme, proposed by Bernhard *et al.* [21], serves as a general representation of various constructions found in the existing literature. This scheme relies on two fundamental components: public key encryption and a zero-knowledge proof system. In this scheme, a ballot is represented as (id, c), where id represents the voter's identity and c denotes a ciphertext that encrypts the voter's choice or votes v in a straightforward manner.

Cortier *et al.* [43] further refined this model, introducing the *Labeled-MiniVoting* scheme. This development extended the class of encapsulated voting schemes by incorporating public information related to voters, known as *labels*. These labels established a strong connection between a voter's identity and their ciphertext in a specific election by employing labeled public key encryption.

Labels could signify general election information, such as in Helios, or details relating to a voter's public identity, like pseudonyms or public verification keys, as in Belenios. In the Labeled-MiniVoting scheme, the ballot takes the form (id, ℓ, c), where id stands for the voter's identity, and (ℓ, c) represents the label-ciphertext pair generated by the labeled public key encryption scheme.

A defining characteristic of the MiniVoting scheme family is the enforcement of unique label-ciphertext pairs. This is achieved via a process called 'weeding' [42], conducted by the ValidBoard algorithm, to counter trivial privacy attacks where an adversary could replicate ciphertexts (and their labels) and monitor the alterations in the election outcome.

A fine-tuned version of Labeled-MiniVoting could be fined in Figure 6.6. Note that we treat the public credential pc as the label and consider the following ballot format (pc, c). The removal of id doesn't have a significant impact, as we use the operation Flabel to model the link between identity and public credential:

$$pc \leftarrow \mathsf{Flabel}(id).$$

Common implementations for this operator vary based on the voter's identity assumptions and the level of separation desired in the public credential. In this context, we treat the voter's identity as either a pseudonym or a public encryption key, thereby setting Flabel as the identity function $\mathsf{Flabel}(x) = x$, a strategy similarly employed by Selene. Alternate strategies might regard the real voter's identity (e.g. email, name) as input and develop a pseudonym or a public credential, particularly if signatures are involved. Regardless of the method, an injectivity assumption must be made about Flabel, which is automatically met by the identity function.

Labeled-minivoting is further parameterized by three other classic operators:

$\mathsf{ValidInd}(pc, c, \mathsf{pd})$: $\{0,1\}$ This operator validates whether the label-ciphertext pair (pc, c) is correctly formed.

$\rho((pc_i, v_i)_{i\in\mathscr{I}})$ (where $\mathscr{I}$ is the set of voters) This function delivers the election result in a predetermined format, such as a lexicographic order for mixnet tally or a specific value for the homomorphic tally.

The process first involves selecting which votes to retain using Policy, followed by calculating the outcome from the votes preserved by Policy using Count.

- In the context of both Labeled-MiniVoting and Selene, Policy is set to "last vote counts" for an individual voter.
- For Labeled-MiniVoting, Count remains abstract, whereas it is specifically defined for Selene.

Publish(BB) : $\{0,1\}^*$ This function serves as an abstract representation of the public bulletin board, which most of the time mirrors the ballot box.

Definition 13. *Assuming* E *as a* poly–IND–1–CCA *secure labeled PKE and* $\Sigma = (\mathsf{P},\mathsf{V})$ *as a zero-knowledge proof system. Given the operators* Flabel, ValidInd, ρ, Publish *previously defined, we formulate the Labeled-MiniVoting scheme as*

$$\begin{aligned}&\mathsf{MiniVoting}(\mathsf{E},\Sigma,\mathsf{Flabel},\mathsf{ValidInd},\rho,\mathsf{Publish}) =\\ &\qquad(\mathsf{Setup},\mathsf{Register},\mathsf{Vote},\mathsf{ValidBoard},\\ &\qquad\mathsf{Tally},\mathsf{VerifyVote},\mathsf{VerifyTally},\mathsf{Publish})\end{aligned}$$

This single-pass voting scheme's algorithms are described in Figure 6.6, and we offer an informal explanation below:

Setup(λ) : (pd, sd). On receiving the security parameter λ, it generates the output (pd, sd) from the encryption scheme's key creation algorithm.

Register(**id**) : (pc, sc). It applies Flabel to id to create the public credential pc, and it does not contemplate a secret credential, i.e., $sc \leftarrow \perp$.

Vote(**id**, pc, v, pd) : (pc, c). It encrypts a vote v with the public credential pc acting as the label and delivers the ciphertext along with the voter's public credential. For simplicity, the secret credential sc is omitted from the input as it is not used.

ValidBoard(BB, pd) : $\{0,1\}$. Returns true if all ballots (pc, c) are well-structured, according to ValidInd, and each ciphertext is consistently used with the same public credential (abiding by the weeding property); otherwise, it returns false.

Tally(BB, sd) : (r, Π). It calculates the election result r by applying the counting function ρ to the list of $(pc, v_\perp)$, where $v_\perp$ is the decrypted version of (pc, c) from the ballot box. It also offers a proof of correct decryption Π by using the P algorithm of the proof system Σ.

VerifyVote(**id**, pc, c, BB) : $\{0,1\}$. It confirms if the most recent ballot (pc, c) is present in the ballot box BB. The voter may have a state with all cast ballots, but verification is done in relation to the last vote cast.

Verifytally((pd, pbb, r), Π) : $\{0,1\}\}$. It triggers the V algorithm of the proof system to ensure the tally proof aligns with the provided statement.

Publish(BB) : It executes the Publish operator on the ballot box BB.

6.4.5 mb – BPRIV for MiniVoting

We have also transposed the original mb – BPRIV definition [41] into EASYCRYPT and demonstrated that Theorem 1 is still applicable in context with this privacy definition. The approach to proving follows a similar strategy, but the proofs required modifications due to the discrepancies between the definitions,

```
lemma mb_bpriv &m :
  BP.setidents{m} = BP.setH{m} `|` BP.setD{m} =>
  (** The mb-BPRIV advantage of some adversary A upper bounded by the sum of: **)
 `|Pr[OMB_BPRIV_L(MV(E,P,Ve,C),A,HRO.ERO,G).main() @ &m : res]
   - Pr[OMB_BPRIV_R(MV(E,P,Ve,C),A,HRO.ERO,G,S,Recover').main() @ &m : res]|
   (** - the advantages of BVFR(G) and BVFR(S) in breaking the Voting Friendly Relation; **)
   <=  Pr[VFR(E, BVFR(MV(E,P,Ve,C), A), R(E), HRO.ERO, G).main() @ &m : res]
   + Pr[VFR(E, BVFR(MV(E,P,Ve,C), A), R(E), HRO.ERO, S).main() @ &m : res]
   (** - the ZK advantage of adversary BZK against the underlying NIZK; and **)
   + `|Pr[ZK_L(R(E,HRO.ERO), P, BZK(E,P,C,Ve,A,HRO.ERO), G).main() @ &m : res]
       -  Pr[ZK_R(R(E,HRO.ERO), S, BZK(E,P,C,Ve,A,HRO.ERO)   ).main() @ &m : res]|
   (** - the IND1-CCA advantage of adversary BCCA against the ballot-encryption scheme **)
   + `|Pr[Ind1CCA(E,BCCA(MV(E,P,Ve,C),A,S),HRO.ERO,Left).main() @ &m : res]
       -  Pr[Ind1CCA(E,BCCA(MV(E,P,Ve,C),A,S),HRO.ERO,Right).main() @ &m : res]|
```

FIGURE 6.10: EASYCRYPT lemma establishing that MiniVoting satisfies mb – BPRIV. HRO.ERO and G are independent random oracles. S is the ZK simulator.

particularly when the verification occurs. This marks the first machine-verified proof of mb – BPRIV. The relevant EASYCRYPT lemma `mb_bpriv` is located in the `MiniVotingSecurity_omb.ec` file.

Theorem 1. *Let* $\mathscr{V} = \mathsf{MiniVoting}(\mathsf{E}, \Sigma, \mathsf{Flabel}, \mathsf{ValidInd}, \rho, \mathsf{Publish})$ *be defined as in Definition 13. Then, for any* mb – BPRIV *adversary* $\mathscr{A}$*, there exists a simulator* Sim *and three adversaries* $\mathscr{B}, \mathscr{C}$ *and* $\mathscr{D}$*, such that*

$$\mathsf{Adv}^{\mathsf{mb-bpriv}}_{\mathscr{A},\mathscr{V},\mathsf{Sim}}(\lambda) \leq 2 \cdot \Pr\left[\mathsf{Exp}^{\mathsf{vfr}}_{\mathscr{D},\mathscr{V},\Sigma}(\lambda) = 1\right] + \mathsf{Adv}^{\mathsf{zk}}_{\mathscr{B},\mathsf{P},\mathsf{Sim},\mathscr{R}}(\lambda) + \mathsf{Adv}^{\mathsf{poly-ind1cca}}_{\mathscr{C},\mathsf{E},n}(\lambda).$$

Theorem 1 corresponds to lemma `mb_bpriv` in `MiniVotingSecurity_omb.ec`.

Figure 6.10 displays the EASYCRYPT formulation of Theorem 1. `P` and `Ve` denote the NIZK's prover and verifier, `C` denotes the ValidInd algorithm, `CP` denotes the commitment protocol. `A` is the adversary. The zero-knowledge simulator `S` and the random oracles for tracker encryption (`HRO.ERO`) and for the NIZK proof system (`G`) are defined concretely in the code.

The VFR, zero knowledge and poly–IND–1–CCA security experiments are denoted by `VFRS`, `ZK_L`, `ZK_R` and `Ind1CCA`, respectively, while the respective reductions are denoted by `BVFR`, `BZK` and `BCCA`. These are also given concrete definitions. Also note that the `Ind1CCA` module is parameterized by a left-or-right module, representing the case where $\beta = 0$ and the case where $\beta = 1$, respectively.

The mb – BPRIV security experiment is parameterized by a recovery algorithm, and we use the concrete recovery algorithm described in Section 6.3.1.

We now sketch the proof of Theorem 1. The EASYCRYPT formalization of the full proof is found in the file `MiniVotingSecurity_omb.ec`. Unless explicitly stated, all the modules and lemmas we refer to in the following are also found in this file.

In our EASYCRYPT formalization, we split the security experiment $\mathsf{Exp}^{\mathsf{mb-BPRIV},\mathsf{Recover},\beta}_{\mathscr{A},\mathscr{V},\mathsf{Sim}}$ into two different games, one for $\beta = 0$ and one for $\beta = 1$. The difference between the two games is, as described earlier, what the tally algorithm does. These games are modeled in the modules `OMB_BPRIV_L` and `OMB_BPRIV_R`, respectively, which are found in the file `VotingSecurity_mb.ec`.

The proof is established through the creation of a sequence of hybrid games. The complete collection of games, along with brief descriptions, can be found in Figures 6.11 and 6.12.

- `G0L'`: Refactoring the main procedure by extracting certain steps for reuse.
- `G0L''`: Refactor the 'valid' statement to be outside the nested 'if' block.
- `G0L`: Move the 'tally' calculation to occur before the 'if' statements.
- `G1L`: Simulate the verification of the correct 'tally' value within the current context (left world).
- `G2L`: Stop decrypting legitimately generated ciphertexts and instead select the correct vote from the 'vmap' mapping.

FIGURE 6.11: Left Side Games

- `G0R'`: Refactoring the main procedure by extracting certain steps for reuse.
- `G0R`: Reorganizing code by separating the 'tally' and 'valid' checks from the 'if-else' statements.
- `G1R`: Modifying code to stop decrypting honest ciphertexts and use plain votes from 'vmap' instead. Ballots added by adversaries will still be decrypted as usual.
- `G2R`: Updating code to halt the recovery process.
- `G3R`: Adjusting `G2R` to utilize a single board instead of $\mathsf{BP.bb}_0$ and $\mathsf{BP.bb}_1$, similar to the structure of `G2L`. b_1 will be added to the board, but the tally will use v_0 from 'vmap' instead of the original value.

FIGURE 6.12: Right Side Games

Starting out from the left-side security experiment, the first step is to replace the tally proof produced by the prover in the proof system, with a simulated proof produced by the simulator Sim. This change is modeled in the game `G1L`. Provided that the proof system is zero-knowledge, the adversary cannot distinguish between the original game and the game where we simulated the proof, except with negligible probability.

We then define a new game, `G2L`, where we stop decrypting honestly created ciphertexts, and instead use the ciphertexts stored in V (as described in Section 6.3). Ciphertexts submitted by the adversary are decrypted as usual. We also remove one of the bulletin boards from the vote oracle, so that only the left-side votes are stored. We prove that `G1L` and `G2L` are equivalent. The equivalence follows from the correctness property of the encryption scheme used to encrypt the votes and the fact that the adversary only gets to see BB_0 in the left side security experiment.

Starting out from the right side security experiment `OMB_BPRIV_R`, we first stop decrypting honest ciphertexts and prove that the resulting game `G1R` is equivalent to `OMB_BPRIV_R`. The intuition is the same as for the equivalence between `G1L` and `G2L`.

We then define a game `G2R` where we stop performing recovery on the adversarially created ballot box and simply perform the tally on the ballot box the adversary outputs. We prove that `G1R` and `G2R` are equivalent. Intuitively, this holds because the honest votes no longer come from the adversary's board, but from V, and the ballots submitted by the adversary are present both on the adversary's board BB and on the recovered board, by definition of our recovery algorithm.

In the final game, `G3R`, we remove one of the bulletin boards in the vote oracle. This is similar to what we did in `G2L`, but now only the right-side votes are stored. We prove that `G2R` and `G3R` are equivalent. This also shows that the final game on the right side, `G3R`, is completely equivalent to `OMB_BPRIV_R`.

Finally, we show that the probability of distinguishing between the games `G2L` and `G3R` is equivalent to the probability of winning the $\mathsf{IND\text{–}1\text{–}CCA}$ game.

Setup(1^λ) for set $\mathscr{I}$ of voters

1: $\mathsf{trL}, \mathsf{tpTr}, \Pi_c \leftarrow [\,]$
2: $\mathsf{pTr}, \mathsf{pPk}, \mathsf{pCo}, \mathsf{pOp} \leftarrow$ empty
3: $(\mathsf{vpk}, \mathsf{vsk}) \leftarrow \mathsf{kgen}_\mathsf{v}(1^\lambda)$
4: $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathsf{kgen}_\mathsf{t}(1^\lambda)$
5: **for** $i \leq |\mathscr{I}|$ **do**
6: $\quad \mathsf{tr}_i \leftarrow\!\!\$\, Group$
7: $\quad \mathsf{tpTr} \leftarrow \mathsf{tpTr} \| \mathsf{enc}_\mathsf{t}(\mathsf{tpk}, \mathsf{tr}_i)$
8: $(\mathsf{pTr}, \Pi_t) \leftarrow \mathsf{ReencryptionShuffle}(\mathsf{tpTr})$
9: **for** $i \leq |\mathscr{I}|$ **do**
10: $\quad \mathsf{id} \leftarrow \mathscr{I}[i]$
11: $\quad (upk, usk) \leftarrow \mathsf{gen}(1^\lambda)$
12: $\quad \mathsf{pPk}.[\mathsf{id}] \leftarrow upk$
13: **for** $i \leq |\mathscr{I}|$ **do**
14: $\quad \mathsf{id} \leftarrow \mathscr{I}[i]$
15: $\quad t \leftarrow\!\!\$\, Field$
16: $\quad et1 \leftarrow\!\!\$ \mathsf{enc}_\mathsf{t}(\mathsf{tpk}, \mathsf{pPk}[\mathsf{id}]^t)$
17: $\quad et2 \leftarrow\!\!\$ \mathsf{enc}_\mathsf{t}(\mathsf{tpk}, g^t)$
18: $\quad \mathsf{pCo}[\mathsf{id}] \leftarrow \mathsf{dec}_\mathsf{t}(\mathsf{tsk}, et1 \cdot pTr.[\mathsf{id}])$
19: $\Pi_c \leftarrow \mathsf{P}_{co}((\mathsf{pCo}, \mathsf{pPk}, \mathsf{pTr}, \mathsf{tpk}), \mathsf{tsk}))$
20: $\mathsf{pd} \leftarrow (\mathsf{vpk}, \mathsf{tpk}, \mathsf{tpTr}, \mathsf{pTr}, \mathsf{pPk}, \mathsf{pCo}, \Pi_t, \Pi_c)$
21: $\mathsf{sd} \leftarrow (\mathsf{pOp}, \mathsf{vsk}, \mathsf{tsk})$
22: **return** $(\mathsf{pd}, \mathsf{sd})$

Register(id, pd, sd)

1: $d \leftarrow \mathsf{pOp}.[\mathsf{id}]$
2: $upk \leftarrow \mathsf{pPk}.[\mathsf{id}]$
3: $ct \leftarrow \mathsf{pCo}.[\mathsf{id}]$
4: **return** $((\mathsf{id}, upk, ct), d)$

Vote(pd, id, pc, sc, v)

1: $ev \leftarrow \mathsf{enc}_\mathsf{v}(\mathsf{vpk}, \mathsf{id}, v)$
2: $b \leftarrow (pc, ev)$
3: **return** b

ValidBoard(BB, pd)

1: **for** $((\mathsf{id}, upk, ct), ev)$ **in** BB :
2: $\quad e_1 \leftarrow \neg(\exists \mathsf{id}', \mathsf{id}' \neq \mathsf{id}$
3: $\quad\quad \wedge ((\mathsf{id}', upk, ct), ev) \in \mathsf{BB})$
4: $\quad e_2 \leftarrow \mathsf{ValidInd}((\mathsf{id}, upk, ev), \mathsf{vpk})$
5: $\quad e_3 \leftarrow (\mathsf{PU}.[\mathsf{id}] = (\mathsf{id}, upk, ct))$
6: $\quad$ **return** $(e_1 \wedge e_2 \wedge e_3)$

VerifyTally($(\mathsf{pk}, pbb, r), \Pi$)

1: **return** $\mathsf{V}((\mathsf{pk}, pbb, r), \Pi)$

Tally(BB, pd, sd)

1: $rL = [\,]$
2: **for** i **in** 1..|BB| **do**
3: $\quad (\mathsf{vpk}, \mathsf{tpk}, \mathsf{PTr}) \leftarrow \mathsf{pd}$
4: $\quad (\mathsf{trL}, \pi, \mathsf{vsk}, \mathsf{tsk}) \leftarrow \mathsf{sd}$
5: $\quad ((\mathsf{id}, upk, ct), ev)) \leftarrow \mathsf{BB}[i]$
6: $\quad v \leftarrow \mathsf{dec}_\mathsf{v}(\mathsf{vsk}, \mathsf{id}, ev))$
7: $\quad \mathsf{tr} \leftarrow \mathsf{dec}_\mathsf{t}(\mathsf{tsk}, \mathsf{PTr}.[\mathsf{id}])$
8: $\quad rL[i] \leftarrow (v, \mathsf{tr})$
9: $r \leftarrow \mathsf{Multiset}(rL)$
10: $pbb \leftarrow \mathsf{Publish}(\mathsf{BB})$
11: $\Pi \leftarrow \mathsf{P}((\mathsf{pd}, pbb, r), (\mathsf{sd}, \mathsf{BB}))$
12: **return** (r, Π)

VerifyVote(id, v, r, pc, sc)

1: $(\mathsf{id}, upk, ct) \leftarrow pc$
2: $(usk, d) \leftarrow sc$
3: $\mathsf{tr} \leftarrow \mathsf{open}(upk, ct, d)$
4: **return** $(v, \mathsf{tr}) \in r$

Publish(BB)

1: **return** BB

FIGURE 6.13: Algorithms defining the $[\mathsf{E_v}, \mathsf{E_t}, \mathsf{C}, \Sigma, \mathsf{ValidInd}]$ voting scheme, given an IND–1–CCA secure labeled PKE scheme $\mathsf{E_v} = (\mathsf{kgen_v}, \mathsf{enc_v}, \mathsf{dec_v})$, an IND–CPA secure homomorphic encryption scheme $\mathsf{E_t} = (\mathsf{kgen_t}, \mathsf{enc_t}, \mathsf{dec_t})$, zero-knowledge proof systems $\Sigma_{ta} = (\mathsf{P}_{ta}, \mathsf{V}_{ta})$, $\Sigma_{tsh} = (\mathsf{P}_{tsh}, \mathsf{V}_{tsh})$ and $\Sigma_{co} = (\mathsf{P}_{co}, \mathsf{V}_{co})$ for the tally proof, the tracker shuffle proof and the proof that commitments are correctly formed, respectively, a commitment scheme $\mathsf{CP} = (\mathsf{gen}, \mathsf{commit}, \mathsf{open})$, and the abstract operator ValidInd.

6.5 Selene

For completeness, we here also present the results from [47] that Selene satisfies du-mb-BPRIV.

In this section, the focus is on formalizing Selene's ballot privacy properties. Verifiable shuffles and the ElGamal + PoK construction are abstracted away, and an abstract IND–1–CCA-secure labeled PKE is used instead [19, 18]. The verifiable shuffle is replaced with a parametric Multiset distribution for the tally phase. The formalization of proofs for interactive protocols remains a complex task, and the focus on this challenge is considered orthogonal. In the EasyCrypt formalization, the encryption of votes is abstracted, with an abstract labeled PKE that is assumed to be IND–1–CCA secure. The ElGamal with proof of knowledge construction used in Selene has been proven to be IND–1–CCA secure, though this remains out of the reach of machine-checking. Since privacy is emphasized, the signatures that Selene includes on-cast ballots are also removed, as they cannot compromise privacy. Additional proofs are made available at https://github.com/mortensol/du-mb-bpriv.

Since the focus of this paper is on privacy, the signatures Selene includes on-cast ballots are removed, as they cannot compromise privacy. These simplifications yield the following model for Selene[2].

Definition 14. *Let* $\mathsf{E_v}$ *be a* poly–IND–1–CCA *secure labeled PKE, let* $\mathsf{E_t}$ *be an* IND–CPA *secure PKE, let* $\Sigma_{\mathscr{R}} = (\mathsf{P}, \mathsf{V})$ *be a proof system for a relation* $\mathscr{R}$ *and let* $\mathsf{CP} = (\mathsf{gen}, \mathsf{commit}, \mathsf{open})$ *be a commitment protocol.*

[2] An EasyCrypt proof for Selene with signatures is available at https://github.com/mortensol/du-mb-bpriv.

```
lemma du_mb_bpriv &m : BP.setidents{m} = BP.setH{m} `|` BP.setD{m} =>
(** The du-mb-BPRIV advantage of some adversary A upper bounded by the sum of: **)
`| Pr[DU_MB_BPRIV_L(Selene(Et,Ev,P,Ve,C,CP),A,HRO.ERO,G).main() @ &m: res]
  - Pr[DU_MB_BPRIV_R(Selene(Et,Ev,P,Ve,C,CP),A,HRO.ERO,G,S,Recover').main() @ &m: res]|
(** - the advantages of BVFR(G) and BVFR(S) in breaking the Voting Friendly Relation; **)
<= Pr[VFRS(Et,Ev,BVFR(Selene(Et,Ev,P,Ve,C,CP),A,CP),R,HRO.ERO,G).main() @ &m: res]
 + Pr[VFRS(Et,Ev,BVFR(Selene(Et,Ev,P,Ve,C,CP),A,CP),R,HRO.ERO,S).main() @ &m: res]
(** - the ZK advantage of adversary BZK against the underlying NIZK; and **)
 + `| Pr[ZK_L(R(Et,Ev,HRO.ERO),P,BZK(Et,Ev,P,C,Ve,A,CP,HRO.ERO),G).main() @ &m: res]
    - Pr[ZK_R(R(Et,Ev,HRO.ERO),S,BZK(Et,Ev,P,C,Ve,A,CP,HRO.ERO)).main() @ &m: res]|
(** - the IND1-CCA advantage of adversary BCCA against the ballot-encryption scheme **)
 + `| Pr[Ind1CCA(Ev,BCCA(Selene(Et,Ev,P,Ve,C,CP),Et,CP,A,S),HRO.ERO,Left).main() @ &m: res]
    - Pr[Ind1CCA(Ev,BCCA(Selene(Et,Ev,P,Ve,C,CP),Et,CP,A,S),HRO.ERO,Right).main() @ &m: res]|
```

FIGURE 6.14: EASYCRYPT lemma establishing that Selene satisfies du–mb–BPRIV. HRO.ERO and G are independent random oracles. S is the ZK simulator.

We define

$$(\mathsf{E_v}, \mathsf{E_t}, \Sigma_{\mathscr{R}}, \mathsf{CP}, \mathsf{ValidInd})$$

as the voting system built upon the algorithms given in Figure 6.13, which we informally describe below.

Setup: Takes as input a security parameter λ and returns a key pair $(\mathsf{vpk}, \mathsf{vsk})$ used to encrypt and decrypt votes, a key pair $(\mathsf{tpk}, \mathsf{tsk})$ used to encrypt and decrypt trackers, a list tpTr of encrypted trackers, finite maps pTr, pPk, pCo and pOp from voter identities to encrypted trackers, public commitment keys, tracker commitments, and openings, respectively, as well as a proof Π_t of the correct shuffle of the trackers and a proof Π_c that the tracker commitments are correctly formed. The public data is

$$\mathsf{pd} = (\mathsf{vpk}, \mathsf{tpk}, \mathsf{tpTr}, \mathsf{pTr}, \mathsf{pPk}, \mathsf{pCo}, \Pi_t, \Pi_c)$$

and the secret data is

$$\mathsf{sd} = (\mathsf{pOp}, \mathsf{vsk}, \mathsf{tsk}).$$

Register: Takes as input a voter identity and a pair $(\mathsf{pd}, \mathsf{sd})$ of public and secret data, and returns the voter's public commitment key, the commitment to her tracker and an opening to the commitment.

Publish: Outputs the public part of the ballot box.

Vote: Encrypts a vote v and outputs the ciphertext together with the voter's public credential.

VerifyTally: Run the V algorithm of the proof-system to check if the tally proof is valid.

ValidBoard: For each element in the ballot box BB, we perform three checks: every ballot contains a unique voter identity, every ballot is well-formed, and every public credential corresponds to the correct identity.

Tally: Decrypts every encrypted vote in the ballot box BB and the tracker for each voter. Returns the multiset of all vote/tracker pairs (v, tr).

VerifyVote: To verify, a voter opens her tracker commitment and checks if her vote v and tracker tr is in the list of vote/tracker pairs returned by Tally.

Morten in the following theorem proved that Selene satisfies du–mb–BPRIV.

Theorem 2. *Let* $\mathscr{V} = (\mathsf{E_v}, \mathsf{E_t}, \Sigma_{\mathscr{R}}, \mathsf{CP}, \mathsf{ValidInd})$, *where* $\mathsf{ValidInd}(\mathsf{pc}, \mathsf{vpk}) = \top$ *for* $c \leftarrow \mathsf{enc_v}(\mathsf{vpk}, id, v)$ *and any public encryption key* vpk, *identity* id, *public credential* pc *and vote* v. *For any* du–mb–BPRIV *adversary*

$\mathscr{A}$, there exists a simulator Sim *and four adversaries $\mathscr{B}, \mathscr{C}, \mathscr{D}_S$ and $\mathscr{D}_G$, such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{du\text{–}mb\text{–}bpriv}}_{\mathscr{A},\mathscr{V},\mathsf{Sim}}(\lambda) \leq\ & \Pr\left[\mathsf{Exp}^{\mathsf{vfr}}_{\mathscr{D}_G,\mathscr{V},\Sigma_{\mathscr{R}}}(\lambda) = 1\right] \\ & + \Pr\left[\mathsf{Exp}^{\mathsf{vfr}}_{\mathscr{D}_S,\mathscr{V},\Sigma_{\mathscr{R}}}(\lambda) = 1\right] \\ & + \mathsf{Adv}^{\mathsf{zk}}_{\mathscr{B},\mathsf{P},\mathsf{Sim},\mathscr{R}}(\lambda) + \mathsf{Adv}^{\mathsf{poly\text{–}ind1cca}}_{\mathscr{C},\mathsf{E_v},n}(\lambda).\end{aligned}$$

Theorem 2 corresponds to lemma `du_mb_bpriv` in `SeleneBpriv.ec`. Figure 6.14 displays the EASYCRYPT formulation of Theorem 2. The lemma itself is inside a section which quantifies over all core components: `Et` and `Ev` denote the encryption schemes used for the trackers and the votes, respectively, `P` and `Ve` denote the NIZK's prover and verifier, `C` denotes the ValidInd algorithm, `CP` denotes the commitment protocol. `A` is the adversary. The zero-knowledge simulator `S` and the random oracles for tracker encryption (`HRO.ERO`) and for the NIZK proof system (`G`) are defined concretely in the code, but not fully displayed in this paper. The VFR, zero knowledge and poly–IND–1–CCA security experiments are denoted by `VFRS`, `ZK_L`, `ZK_R` and `Ind1CCA`, respectively, while the respective reductions are denoted by `BVFR`, `BZK` and `BCCA`. These are also given concrete definitions. Also note that the `Ind1CCA` module is parameterized by a left-or-right module, representing the case where $\beta = 0$ and the case where $\beta = 1$, respectively. The du–mb–BPRIV security experiment is parameterized by a recovery algorithm, and we use the concrete recovery algorithm described in Section 6.3.1.

We now sketch the proof of Theorem 2. The EASYCRYPT formalization of the full proof is found in the file `SeleneBpriv.ec`. Unless explicitly stated, all the modules and lemmas we refer to in the following are also found in this file.

6.6 Concluding Remarks and Future Work

In this work, we presented a refined version of the mb – BPRIV privacy definition which we call delay-use malicious-ballot box ballot privacy (du–mb–BPRIV). Our new definition allows the modeling of schemes (such as Selene) where verification occurs after tallying. The security claim is also more explicit. We formalized our new definition in the interactive theorem prover EASYCRYPT and showed that labeled MiniVoting, Belenios, and Selene all satisfy the definition. We also proved that MiniVoting and Belenios satisfy the original mb – BPRIV privacy definition. Furthermore, we have discussed the relation between du–mb–BPRIV and individual verifiability (and the corresponding relation for mb – BPRIV) and proved that a voting system satisfying du–mb–BPRIV and a property called strong consistency, also satisfies a form of individual verifiability which depends on the choice of recovery function.

While we have encoded Selene correctly in what we firmly believe is the most appropriate privacy definition in literature, our work highlights certain defiances in privacy definitions that future work should address. The defiances are fairly deep and addressing them is far out of the scope of this work. We will, however, briefly mention two principal defiances of mb – BPRIV and related definitions. First, the definition, while handling a malicious bulletin board, assumes an honest setup. Secondly, du–mb–BPRIV and related definitions are highly calibrated to schemes where the auxiliary data produced by the tally to be used in verification are zero-knowledge proofs. In particular, schemes like Selene which output trackers to use for verification are difficult to express in these definitions.

Further, the BPRIV style of the definition implies certain restrictions. As an example, the adversary can only see the result and verification from the left side board which constrains the attacker model. In particular, this means that we cannot detect privacy attacks relying on inducing candidate-specific errors for an observed voter while giving the adversary access to whether the corresponding voter verification

fails or not, see e.g. [82] for such a style of attack. Of course, such attacks can be ruled out by considering recovery functions preventing any changes to honestly cast votes as in [41], but it is not the case in general. An important line of future research is thus to find alternative definitions capturing both more general and more transparent attacker models, e.g. by decreasing the generality of the definition or to consider simulation-based security.

CHAPTER 7

ANALYSIS OF ESTONIAN ELECTION MARGIN

A cornerstone in the study of electoral auditing is understanding the election margin. This key concept delineates the volume of votes necessary to effect an outcome shift, serving as a barometer of the robustness and competitiveness of an election. It functions as a critical control measure for determining the reliability of electoral results, offering insights into the extent of potential vote manipulation or irregularities.

The focus of this chapter is an in-depth marginal analysis of the Riigikogu election in Estonia. This case study is particularly interesting due to the unique characteristics of the Estonian electoral process, which includes a complex candidate selection process that is unlike more familiar models found in other parliamentary systems.

In order to account for this complexity, we have segmented our analysis into various stages. Each stage of the candidate selection process is treated as a discrete event, which allows us to identify the critical points at which the election margin can significantly influence the overall outcome.

In the following sections, we will provide a comprehensive overview of our methodological approach, present the results of our marginal analysis, and discuss the implications of our findings for the design and auditing of future elections. Through this rigorous and granular analysis, we aim to contribute to a more nuanced understanding of the relationship between election margins and electoral outcomes, particularly in the context of the Riigikogu elections.

7.1 Introduction

The margin of victory [127] is an important measure of the strength of an election result. It is the smallest number of votes that, if changed, would alter the outcome of the election given a specific voting rule and collection of votes. Therefore, a big margin of victory is seen as a more decisive and resilient result than a narrow margin of victory.

The margin of victory is an important factor in the effectiveness of post-election audits. In the United States, these audits are commonly used to detect errors in electronic voting systems caused by software or hardware issues with voting machines [94]. When voters use these machines, their votes may not be counted correctly due to bugs, programming mistakes, or other machine output errors.

The margin of victory has been tailored to work effectively with a diverse range of voting rules[127], not just elections employing the plurality rule where each voter casts a vote for a single choice, and the option with the highest number of votes is deemed the winner. In such cases, the margin of victory is straightforward to compute, calculated by halving the difference between the highest and second-highest scores. Nevertheless, there's a continued emphasis on broadening this research to further adapt and enhance auditing procedures for a multitude of voting rules.

Estonia, a digitally advanced nation, stands as an intriguing case study for our research due to its unique electoral system and sophisticated integration of technology into governance. The country's pioneering use of electronic voting, or i-Voting, along with the meticulously crafted structure of its parliamentary elections, provides a rich canvas for exploring the dynamics of digital democracy. Additionally, the fact that 101 candidates are elected to serve in the Riigikogu through free and proportional elections, held every four years on the first Sunday of March, adds further complexity and dimension to our investigation. This multi-layered electoral system, which includes personal, district, and compensation mandates, allows us to examine the democratic process through multiple lenses. It is these qualities that render Estonia a compelling subject for our research.

Electronic voting (i-Voting or internet voting) is one of the 14 available methods in Estonia[1].

7.2 Ascertaining the Election Results

In order to estimate the margins in the Estonian National Elections, we first need to understand and implement the election counting function in order to reproduce true and assess counterfactual, election tally results.

In the Riigikogu elections, both political parties and independent candidates are eligible to participate. Each political party has the privilege to nominate candidates across the twelve electoral districts. Moreover, parties also compile a national list, which is essentially a ranking of all their nominated candidates across these districts.

The process of determining the election results commences at the district level. Here, personal and district mandates are allocated based on the votes received. Following this, the remaining compensation mandates are distributed at the national level. In total, there are 101 mandates to be allocated across the twelve districts.

7.2.1 Allocation of Personal Mandates within an Electoral District

The process of distributing personal mandates within an electoral district begins by dividing the total number of voters in the district by the number of mandates available (which is predetermined before the election). This calculation yields what is referred to as the 'simple quota' for the district.

$$\text{SimpleQuota} = \frac{\#\text{Voters}}{\#\text{Mandates}}$$

[1] For more information you can see https://www.riigikogu.ee/en/introduction-and-history/riigikogu-tasks-organisation-work/elections-riigikogu/

To illustrate, let's consider a district where 10,000 voters have cast their votes and there are 5 mandates to be allocated. The simple quota for this district would be 10,000 / 5 = 2,000 votes. Therefore, any candidate who garners at least 2,000 votes is elected, that is, they receive a personal mandate.

The criterion for election is straightforward: any candidate (whether affiliated with a party or running independently) who secures votes equal to or more than the simple quota is elected. There are no additional prerequisites for election.

7.2.2 Allocation of District Mandates within an Electoral District

The distribution of district mandates is restricted to parties that have garnered at least 5% of the valid votes at the national level, a criterion known as the electoral threshold.

The process of distribution involves summing up the votes cast for the candidates of a party at the district level and comparing this total to the simple quota. The party is entitled to as many mandates as the number of times the total votes exceed the simple quota. However, from this number, the already allocated personal mandates, if any, are deducted. These mandates are referred to as district mandates.

$$\#\text{DistrictMandatesForParty} = \frac{\text{TotalVotesForParty}}{\text{SimpleQuota}}$$

For instance, if the simple quota is 2000 votes and a party has received 5000 votes, the party is entitled to 2.5, or effectively 2, district mandates (5000 / 2000 = 2.5). However, if the party has already secured one personal mandate, it is entitled to only one additional district mandate.

In the 2003 Riigikogu elections, an additional requirement was introduced to the Election Act to increase the share of district mandates. Accordingly, if a party receives votes amounting to at least 75% of the simple quota, or if the remaining votes for a party exceed 75% of the simple quota after the distribution of district mandates, the party is awarded one district mandate.

For example, if the quota of Electoral District A is 2000 votes and a party receives 1800 votes (which is 90% of the quota), the party is awarded one district mandate. Similarly, in Electoral District B, if the quota is 2000 votes and a party receives 3500 votes, the party is awarded one district mandate. If, after this mandate is accorded, the party still has 1500 votes left (which is 75% of the district quota), the party is awarded one more mandate, making it 2 mandates in total.

The candidates are elected based on their ranking in the district list of the party, which is determined by the number of votes they receive. However, to be elected under the district mandate, a candidate must receive at least 10% of the simple quota.

For instance, if the simple quota is 2000 votes, a candidate must receive at least 200 votes to be elected under the district mandate.

7.2.3 National Distribution of Compensation Mandates

Following the allocation of mandates at the district level, the remaining undistributed mandates are divided nationally using the modified d'Hondt distribution method. Named after the Belgian mathematician Victor d'Hondt, this method allows for a proportional distribution of mandates among the lists, enabling even parties with fewer votes to secure a mandate.

The d'Hondt distribution series typically runs as 1, 2, 3, 4, etc. However, in Estonia, a slightly modified series is used, where the numbers from 2 onwards are raised to the power of 0.9. This results in a series of values such as 1,000, 1,866, 2,688, 3,482, etc. This modification slightly favors parties with a higher vote count.

Only parties that have received at least 5% of the valid votes at the national level are eligible to participate in the distribution of compensation mandates.

The distribution process involves calculating comparative figures for each party. The number of comparative figures corresponds to the number of candidates on the party list. These figures are obtained by dividing the number of votes for a party by each element of the distribution series.

Subsequently, the comparative figures of the parties are compared. A mandate is awarded to the party with the highest comparative figure, the next mandate goes to the party with the second highest comparative figure, and so on, until all undistributed mandates are allocated.

For instance, consider the distribution of 3 mandates between two parties, A and B, using the classic d'Hondt distribution series for simplicity:

Party A has received 3000 votes, resulting in comparative figures of 3000/1=3000, 3000/2=1500, 3000/3=1000, etc.

Party B has received 2700 votes, resulting in comparative figures of 2700/1=2700, 2700/2=1350, 2700/3=900, etc.

The three mandates are thus allocated as follows:

- **1st mandate** - Party A (Comparative figure: 3000)
- **2nd mandate** - Party B (Comparative figure: 2700)
- **3rd mandate** - Party A (Comparative figure: 1500)

If a party has already secured a personal or a district mandate, the calculation of comparative figures for that party starts from the corresponding element in the series.

For example, if Party A has already secured one district mandate, the calculation of comparative figures for Party A starts from the second element of the series:

Party A's comparative figures are now 3000/2=1500, 3000/3=1000, 3000/4=750, etc.

The three mandates are now allocated as follows:

- **1st mandate** - Party B (Comparative figure: 2700)
- **2nd mandate** - Party A (Comparative figure: 1500)
- **3rd mandate** - Party B (Comparative figure: 1350)

The compensation mandates are allocated according to the party's national list of candidates. A compensation mandate is awarded to the candidate who is higher on the list and has received votes amounting to at least 5% of the simple quota of the district. Candidates who have already secured a personal or a district mandate are bypassed during this distribution.

For example, if the simple quota is 2000 votes, a candidate must receive at least 5% of this quota, i.e., 100 votes, to be elected under the compensation mandate.

We utilized open data provided in a machine-readable XML format [121] to analyze the election results. The data, available for free download, is subject to the Creative Commons Attribution License (CC BY 4.0). The tally function, which accepts an XML file of the election and returns the 101 elected candidates, has been implemented using Python [50]. Additionally, we have developed several functions to validate our results, ensuring they accurately reproduce the outcomes of the election.

7.3 Margin Analysis

In this section, we analyze the margins in the Estonian elections. Such an analysis is not only relevant for our work, but precise margins are important for Risk-Limiting Audits. Further, the margins are related to Election Competitiveness.

This latter concept has been extensively studied and is found to be intricately linked to various aspects of electoral dynamics [66, 31, 89]. One approach to quantify this competitiveness is by assessing the susceptibility of election outcomes to vote alterations. In this context, the focus is not on favoring a specific candidate or party but on understanding the overall impact of vote changes on the election results. This approach provides a measure of the election's competitiveness, which can serve as an incentive for voter turnout in subsequent elections, given the potential influence of each vote on the election outcome [23].

When we discuss altering the outcome of an election, we could have different concepts in mind. For this paper, we are specifically referring to a modification in the final list of candidates, which we conceptualize as a set, rather than a list. In other words, our primary focus is on the identity of the candidates who emerge as winners, not on how they are ranked or the type of political mandates they receive. Any shifts in the order of candidates or transformations in the nature of their mandates are not viewed as alterations to the election results in this context.

For example, if the voting process or some other factor were to result in a rearrangement of candidates without impacting who actually wins the election, we would not deem this as a change in the election outcome. Similarly, if the type of mandate given to the elected candidates changes, this too is not considered a change in the election results, as long as the winning candidates remain the same.

In a more differentiated approach to altering election results, we consider specific strategies and objectives. This leads to different types of margins based on the constraints of our objectives. We have explored several objectives, each corresponding to a unique capability of vote alteration. These include $\text{margin}^{\text{del}}$, $\text{margin}^{\text{add}}$, $\text{margin}^{\text{coalition+del}}$, and $\text{margin}^{\text{coalition+add}}$.

The $\text{margin}^{\text{del}}$ represents the minimum number of votes that need to be deleted from a candidate to remove them from the election outcome. The $\text{margin}^{\text{add}}$ denotes the minimum number of votes that need to be added to a candidate for them to be included in the election result. The $\text{margin}^{\text{coalition+del}}$ signifies the minimum number of votes that need to be deleted to break a coalition. Similarly, the $\text{margin}^{\text{coalition+add}}$ denotes the minimum number of votes that need to be added to form a new coalition. A coalition, in the context of parliamentary politics, refers to an alliance of two or more political parties that come together to form a majority in a legislative assembly, such as the Riigikogu in Estonia. This majority, typically controlling at least 51 mandates, allows the coalition to pass legislation and make executive decisions. In our study, we utilize the script we developed to explore various scenarios pertaining to different types of margins. Our objective is to establish some bounds for these margins. The results of these explorations are presented in Table 7.1.

The distribution of seats within a coalition is a critical factor that determines the power dynamics and stability of the government. Each party's seat count directly influences its bargaining power and role in shaping the coalition's policies. Parties with a higher number of seats typically have a stronger say in key governmental roles, including the Prime Minister, cabinet positions, and committee chairs.

In the context of our study, it's important to clarify the distinction between the types of attacks associated with these margins. When an adversary aims to add a vote, they would need to compromise the voter's vote-casting application to alter the vote in favor of the desired candidate. This is a more complex operation, requiring a level of intrusion into the voter's device.

Conversely, for vote deletion, the adversary's task is somewhat simpler. If they can ascertain that a particular voter intends to vote for a specific candidate, they can simply block the transmission of the

ballot. This doesn't necessitate tampering with the vote-casting application itself but could simply involve interfering with the communication process.

These differences in attack strategies underscore the varying levels of complexity and risk associated with different types of vote manipulation. Understanding these nuances is crucial for developing robust defenses against potential election interference. Each of these margins provides a distinct perspective on the sensitivity of election results and offers valuable insights into potential vulnerabilities in the voting process.

Our analysis encompassed the election results from the year 2019. After calculating various margins using a Python script [50], we present the results in Table 7.1 [2]

The significant difference in the margins for the addition and deletion methods can be attributed to the inherent complexity and multi-stage nature of the election process.

In the context of the election process, the allocation of seats is not a straightforward procedure. It involves multiple rounds of computation and seat allocation, where the outcome of each stage can significantly influence the subsequent stages. This iterative nature of the process adds a layer of complexity to the election results, making them highly sensitive to even minor changes in vote counts.

When we consider the addition method, we are essentially increasing the vote count for a candidate. This increase can potentially tip the balance in a close race, leading to a change in the allocation of seats in the current round and, by extension, in the subsequent rounds. This cascading effect can result in a significant alteration in the final election results, even if the initial increase in vote count was relatively small.

On the other hand, the deletion method involves decreasing the vote count for a candidate. While this can also lead to changes in seat allocation, the effect is not necessarily the same as that of the addition method. This is because the deletion of votes can potentially move a candidate below the threshold required for a seat, which can have a different impact on the overall seat allocation compared to adding votes.

Furthermore, during the seat allocation process, there are instances where we deal with floating numbers and only their integer parts are considered. The impact of adding or deleting votes can therefore depend on the decimal distance to the closest integer part.[3].

TABLE 7.1: Computed Results for Different Margins

Margin Type	Computed Value	What is Removed	What is Added
$\text{margin}^{\text{del}}$	9	Kalev KALLO	Erki SAVISAAR
$\text{margin}^{\text{add}}$	3	Anneli OTT	Jaan MÄNNIK
$\text{margin}^{\text{coalition+del}}$	1500	REF + EKRE	—
$\text{margin}^{\text{coalition+add}}$	1898	—	SDE+KESK+IE

Another objective of our study is to identify the minimal number of modifications required to change the election results in a way that triggers a shift in the number of seats allocated to a party. This adjustment substantially increases the significance of the margin. In our study, we performed an upper-bound examination on the recalculated margin. The marginal analysis indicated that if *Urmas REITELMANN* received more than 1041 votes, the *Eesti Reformierakond* party would lose one seat, while the *Eesti Konservatiivne Rahvaerakond* party would gain an additional seat. Similarly, for the deletion scenario, a change in the distribution of seats would require a margin of 1282 votes. It is crucial to underscore, however, that these figures may not precisely correspond to the actual margins.

[2] When we remove 9 votes from Kalev KALLO, he is removed from the final list and instead Erki SAVISAAR will be added(note that Erki SAVISAAR will be selected without any change on his vote).

[3] Especially in terms of the d'Hondt method errors in rounding errors can easily impact the election result [111]

We have also calculated an upper bound for the margin of the parties' coalition. The alliance between parties *Eesti Reformierakond* and *Eesti Konservatiivne Rahvaerakond* emerges as the most precarious one in our analysis. The coalition holds a slim advantage, with just three seats more than the threshold. This minimal edge indicates that the loss of merely three seats would be sufficient to destabilize this partnership. Consequently, under the scenario of Personal Mandates being leveraged to change the seat allocation, 1500 votes would be necessary to shatter this coalition. We also investigated a scenario involving the formation of a coalition between parties *Sotsiaaldemokraatlik Erakond*, *Eesti Keskerakond*, and *Isamaa Erakond*. Our findings suggest that the addition of 1898 votes in total to these parties would result in them gaining three additional seats. This increase would provide the coalition with a majority of the seats in the parliament.

Remark 1. *The strategy for computing the number of required vote alterations to form or break a coalition can vary significantly depending on the specific objectives and constraints of the adversary. One approach could be to focus solely on one candidate, altering their votes to either boost their chances or undermine their position. This strategy could be effective if the adversary's objective is to influence the outcome for a specific candidate. However, it may also draw more attention and increase the risk of detection, especially if the number of altered votes is large.*

Alternatively, the adversary (the adversarial scenario will be discussed in detail in the next chapter) could opt for a more distributed approach, simultaneously altering the votes for several candidates. This strategy could be more subtle and potentially harder to detect, as the alterations are spread across multiple candidates. However, it may also be more complex to execute, as it requires a more sophisticated understanding of the interplay between different candidates and their voters.

In either case, the adversary would need to carefully consider the trade-offs between risk and reward. A strategy that alters a large number of votes for a single candidate may have a higher impact but also a higher risk of detection. On the other hand, a strategy that subtly alters votes for multiple candidates may be less risky but also less impactful.

Ultimately, the choice of strategy would depend on the adversary's specific objectives, their risk tolerance, and their understanding of the election dynamics. This highlights the importance of robust election security measures that can detect and prevent a wide range of potential attack strategies. Finally we stress that an adversary attacking an election cannot know the margins beforehand but only can estimate these based on earlier elections and poll data and hence needs to include a buffer resulting in a higher number of necessary ballots to compromise.

CHAPTER 8

IDENTIFYING VOTER CHARACTERISTICS THROUGH ML-ASSISTED MODELS

Chapter 7 of the thesis focuses on identifying voter characteristics through ML-assisted models. The chapter starts with an introduction, followed by a detailed description of the process of data set creation, data exploration, and data balancing techniques. Feature engineering techniques are then discussed, including feature selection and extraction. The chapter also covers the evaluation of different classification models, such as logistic regression, decision trees, random forest, naive Bayes, support vector machines, gradient boosting algorithms, k-nearest neighbors, and simple and random classifiers. An ensemble method and association rules are discussed, and an adversarial viewpoint is presented. The chapter concludes with the results and discussions.

8.1 Introduction

Estonia, a pioneer in Internet voting, conducted its first feasibility studies in the early 2000s and implemented the first legally binding country-wide Internet voting in 2005. This voting method has been used in all 8 elections up until 2015, with over 30% votes cast over the Internet in the 2014 European Parliament and 2015 Parliamentary elections [122] and more than 50% in 2023 [98]. The basic protocol, in effect until 2011, mirrored double envelope postal voting, using server encryption and a national eID signing device [68].

However, potential attacks were observed, including proof-of-concept malware that could alter or block votes without detection. To defend against this, an individual verification mechanism was developed for the 2013 elections. This mechanism uses an independent mobile device that downloads the vote cryptogram from the server and lets the voter check the contained vote using the randomness from vote

casting which in turn is obtained for the vote-casting computer using a QR code. This allows voters to verify their votes [70] and detect malware.

The implementation of the verification method has not been perfect, as discussed in [119], and has problems with vote updates [100](Note that the implementation of the Estonian i-Voting system is referred to as the IVXV system. The source code for this system is open and readily accessible[1]).

8.1.1 The Effectiveness of Individual Verification

In the Estonian elections, the actual rate of verification has been around 4% [2, 117]. This low rate is not a peculiarity of the Estonian but is a common problem in e-voting [97, 96]. It has been argued that even low verification rates are sufficient to provide security guarantees against malware that randomly infects voters [70]. However, it is a natural question, and the main research question in this paper, *whether an attacker can predict which voters are verifying, and hence sidestep detection.*

The main obstacle to answering this question is what information the attacker has access to and how to reproduce this. It is our thesis that the best approach here is to use the rich data provided by the log files collected during the real electronic elections in Estonia, which were studied in other contexts in [117]. The log files contain data about features in 8.1.

This is data a strong adversary can get his hands on, especially if compromising voter's computers covertly, or even the actual log file. For a deeper understanding of the log analysis software, data storing process, and web front-end see [69] which explains this in great detail.

However, the identification of voters who are more likely to verify their votes is a complex task. Numerous factors such as age, education level, political affiliation, and personal beliefs can influence the probability of verification, see also [117].

Artificial intelligence (AI) and machine learning algorithms can be utilized to detect patterns for verifying voters. By analyzing vast amounts of data, we can discern trends among voters who verify. The insights derived from this analysis can inform predictions about voters likely to verify their votes. This in turn aids in enhancing outreach and voter education efforts, leading to improved accuracy and integrity in the electoral process.

We do remark that an actual attacker might have access to even more information that makes the classification of verifying voters even more precise. Especially, for ethical reasons, as mentioned above, we do not have access to the actual vote choice of the voters which might be correlated with the verification rate. Likewise, other factors are quite likely to help an attacker, especially if the attacker can access social media-related data for the voter.

The act of recognizing voters who verify their votes in an election is of the utmost importance, for reasons too numerous to mention. Chief among them is the role verification plays in upholding the credibility and precision of the electoral process. The act of allowing voters to confirm the accuracy of their recorded votes guarantees the accountability of every vote and the reflection of the people's will in the election outcome.

Verification also serves to cultivate trust in the electoral process, as voters are more likely to have faith in the results when they see that their votes are being accurately counted and their voices heard. This renewed trust helps promote electoral participation since voters are more likely to partake in the process when they know their votes will be accurately tallied. Verification also contributes to the identification and prevention of potential electoral fraud or errors, providing an opportunity to rectify these issues before they significantly impact the election outcome. Ultimately, this reinforces the fairness, transparency, and reflection of the people's will in the electoral process.

[1]The GitHub repo link is https://github.com/vvk-ehk/ivxv

Identifying the specific voters more likely to verify their votes is challenging as several factors such as age, education level, political affiliation, and personal beliefs influence the likelihood. Studies reveal that younger voters tend to verify their votes more than older voters, while highly educated voters are more likely to do so as well. Furthermore, voters with a high level of engagement in the political process and a keen interest in the election outcome are also more inclined to verify their votes. To establish patterns among verified voters, artificial intelligence (AI) and machine learning algorithms can scrutinize copious amounts of data, including demographic data and voting patterns such as in-person, mail, and early voting.

By analyzing this data, the AI model can identify trends and patterns among verified voters, such as younger voters or highly educated voters. These findings can then be utilized to make predictions about which voters are likely to verify their votes in future elections, allowing for improved outreach and voter education efforts, thereby improving the accuracy and integrity of the electoral process.

8.2 The Process of Data Set Creation

8.2.1 Introduction and Background

The RK2011 elections were a watershed moment in Estonian i-voting. The proportion of votes cast over the Internet reached a critical high of 24.3 percent. This led to the expansion of the i-voting protocol and initiatives for in-depth analysis to detect system malfunctions, attacks, and study voter behavior.

8.2.2 Mechanisms of i-Voting in Estonia

The primary mechanism for Internet voting in Estonia is based on a double-envelope postal voting system, modified to include encryption and digital signatures. Voters can authenticate using smart card-based eID tools or SIM card-based Mobile-ID. They then use the official i-voting client application to cast their vote.

8.2.3 Vote Validation and Resubmission

Validation of votes using mobile devices has been feasible since the 2013 elections. During the advanced voting period, voters have the option to re-submit their votes.

8.2.4 Analysis Software Overview

The analysis software consists of a log processing engine, a database engine, and a web front-end. These components are essential for database updating, information storage, and presenting results to election officials.

8.2.5 Log Processing Details

The log processing engine is a Python program that matches log entries to a list of predefined patterns using regular expressions. Unmatched log entries are recorded as incidents requiring manual review by an election official.

8.2.6 Database Infrastructure

MySQL is used for data storage, with tables for voting sessions, vote verifications, and incidents. The incidents are linked to responses from election officials.

8.2.7 Healthiness of the System

System Monitoring The health of the i-voting system is continuously monitored through VFS and VSS. These components collect key metrics such as system load, memory usage, and free disk space on a minute-to-minute basis. A web interface displays this information and triggers an alarm if the most recent data record exceeds a one-minute threshold. Additionally, a watchdog monitors the responsiveness of the Estonian National Certificate Authority's (NCA) OCSP server.

Log Processor and Incident Management An incident management interface provides an overview of incidents logged by the processor. Each issue can be traced back to a specific voting session and the corresponding log entry. Upon review by an election official, the status of an incident is updated to "handled."

Investigation Features The web front-end equips investigators with useful background information about each incident. Features are included that enable linking of any i-voting session to another session sharing common parameters such as personal code, IP address, or phone number. This assists in identifying if a particular IP address or voter is involved in multiple incidents.

Geographical Information The MaxMind GeoLite City database is utilized for adding geographic context to the IP addresses. All relevant entries in Apache log files are extracted and displayed in the IP view section.

Voting Session Details Finally, the voting session view pulls all log entries related to a particular voting session from the database. This view enables a more in-depth analysis and assists in ascertaining the validity of each voting session.

8.3 Data Exploration

Data exploration serves as a pivotal phase in our data analysis project. Its primary objective is to deepen our understanding of the dataset at hand. This encompasses the scrutiny of relationships between variables, data distribution, as well as identifying any discernible patterns or trends.

Through meticulous exploration, we set the stage for hypothesis formation and insight generation. These insights inform the types of factors that could be influencing the phenomena under study. This step is indispensable for the entire data analysis process as it shapes the methodology for our subsequent analysis.

In pursuit of these goals, we will deploy a variety of visualization techniques coupled with statistical methods. These tools aid in unearthing insights and providing a multifaceted understanding of the data.

As we navigate through this exploratory phase, we will continually document our findings. This practice contributes to the reliability and meaningfulness of the end results, thus enriching the overall analytical framework.

8.3.1 Structure Of Data Set

To better comprehend the overall structure of the data set, we will examine it from multiple dimensions. These dimensions include the number of observations and variables, the data types of these variables, and the presence or absence of missing or incomplete data. Such an understanding is critical to inform our analysis techniques and ensure reliable results.

The dataset consists of 12 features and 158,076 observations. The data types for these features vary and include datetime64, object, and int64. It is noteworthy that the dataset does not contain any null values, thus enhancing its reliability.

In terms of memory usage, each column ranges from 1 to 11 bytes, summing up to a total memory usage of 73 MegaBytes for the entire dataset. This is relatively moderate, making the dataset manageable for most computational setups.

As for the features, they are diverse and include a variety of data types. For example, 'aeg' is a datetime column that represents a timestamp. 'Autentimisvahend' is an object column that denotes an authentication tool, while 'cert-issuer' represents the issuer of a certificate. 'Isikukood' is an integer-based column for personal identification codes. Additional columns represent gender ('sugu'), age ('vanus'), operating systems ('os' and 'v-os'), and Internet Protocol addresses ('ip' and 'geoip'). 'Haaletamise_aeg' is another datetime column that signifies the time of voting, and 'target' is an integer column serving as the target variable.

Overall, the dataset is fairly straightforward, featuring a mix of numerical and categorical variables. While the memory usage is moderate and generally feasible for most applications, optimizations might be required for specific scenarios. This could involve data type conversions, feature reductions, or using memory-efficient techniques such as memory mapping.

Column	Type
aeg	datetime64[ns]
autentimisvahend	object
cert-issuer	object
isikukood	int64
sugu	object
vanus	int64
os	object
ip	object
geoip	object
haaletamise_aeg	datetime64[ns]
target	int64
v-os	object

TABLE 8.1: List of features in log files

8.4 Adversary Model

In our study, we categorize adversaries based on their knowledge level and the tools at their disposal, and we label them appropriately:

1. **No-Knowledge Adversary (Type A)**: This adversary has no specific knowledge or data about the election. They operate primarily on guesswork. We further subdivide this category into two distinct models:

- **Uniform Random Model (No-Knowledge Adversary A1)**: This model represents a Naive Adversary who guesses uniformly. They operate under the assumption that every voter has an equal chance of verifying their vote, essentially treating the verification process as a coin flip with a 50% chance of verification.

- **Verification Rate-Informed Random Model (No-Knowledge Adversary A2)**: This model represents a slightly more informed Naive Adversary. While they still guess randomly, their guesses are informed by the overall verification rate of the election. They adjust their guessing strategy based on this rate, which may slightly increase their chances of correctly guessing who will verify their vote.

2. **Scholar Adversary (Type B)**: This adversary has a basic understanding of the election process and verification rates. In particular, we assume they have the knowledge of the academic paper [117] containing a log file analysis of Estonian electronic elections. This gives insights into correlations between verification rates and various factors such as gender, age, and voter operating system [117].

For the second adversary type (Scholar Adversary), we introduce three specific models based on the following features identified in [117] as being strongly correlated to verifying voters: "Age", "Sex", "Voter's Operating System", "Certificate Issuer" of the voter, and "Authentication Tool" of the voter. These models use different logical operators to make predictions.

- **Scholar Adversary Logical OR Model (Type B1)**: In this model, the Scholar Adversary predicts that a voter will verify their vote if any one of the five features (Age, Sex, Voter's Operating System, Certificate-Issuer, or Authentication Tool) is true hence indicating a higher likelihood of verification.

- **Scholar Adversary Logical AND Model (Type B2)**: In this model, the Scholar Adversary predicts that a voter will verify their vote only if all five features (Age, Sex, Voter's Operating System, Certificate-Issuer, and Authentication Tool) are true hence indicating a high likelihood of verification. It is a more risk-adverse model compared to the Logical OR Model.

3. **Machine Learning Adversary (Type C)**: This adversary has full access to the data equivalent to what is available in the log files and plans to train machine learning classifiers using the following types of algorithms: Naive Bayes, Logistic Regression, K-Nearest Neighborhood, Decision Tree, and Random Forest. For a comprehensive understanding of the machine learning concepts and techniques used in this research, the reader is referred to [57].

The adversary employs a variety of techniques, including imbalanced recovery methods, feature selection methods, feature extraction methods, and other parameters, to train a total of 3960 models specifically for type C. Additionally, we have 20 models tailored for other types of adversaries, namely types A and B. Therefore, in total, we have 3980 models encompassing various types of adversaries.

The Type C adversaries represent the most informed and potentially dangerous adversaries due to their sophisticated use of machine learning techniques. These adversaries are expected to be highly adaptive and capable of exploiting patterns and trends within the data to predict voters' behavior accurately.

However, it's important to note that their strength is also their weakness. The complexity of machine learning models requires them to undergo extensive training and testing phases before they can be deployed for an attack. This process can be time-consuming and resource-intensive, potentially slowing down their operations and providing opportunities for detection and countermeasures."

8.5 Attack Model

In our attack model, we consider an adversary who has a fixed model and a fixed margin number, denoted as m. This margin number represents the number of vote alterations the adversary aims to achieve. The adversary's goal is to alter m votes, but they can tolerate a certain number of alerts, denoted as a. The number of allowed alerts depends on the adversary's risk appetite and models that not all verifying votes

```
def Attack(classifier = clf, margin = m, alert = a, V = [X,y]):
    # Where X the database of voters information
    # from the log file without verification statue
    # y the column vector of verification states of voters
    actual = y  #voter's label for verification
    predicted = clf.predict(X)
    success = 0
    alert = 0
    i = 0
    while i < len(actual):
        if (predicted[i] == 0) and (actual[i] == 0):
            success += 1
        if (predicted[i] == 0) and (actual[i] == 1):
            alert += 1
        if success >= Nmargin and alert <= A:
            return 1
        if alert > A:
            return 0
        i += 1
    return 0
```

FIGURE 8.1: In the context of our attack game algorithm, if the Attack function returns 1, it signifies that the adversary has successfully managed to alter m votes while triggering fewer than a alerts. Conversely, any other return value indicates that the adversary has failed to meet these conditions and, therefore, has not succeeded in the game.

will actually detect an altered vote or report it. And even if reported, the alert might not lead to an investigation, the malware will not be detected or has been removed beforehand without a trace. It can even happen that an adversary might not even care about a few confirmed malware samples since this will just cast some doubt on the election, but not a general halt of the election.

The adversary selects a voter and, according to the type of adversary above, checks if this voter is classified as a verifying voter. If the voter is classified as verifying, the adversary skips them to prevent detection and selects another voter. This process continues until the adversary finds a voter classified as a non-verifying voter. If the adversary knows that the voter is going to vote in favor of their objective, they do not change the vote. However, if the voter is going to vote against the adversary's objective, they change the vote.

In our abstract modeling, we have the opportunity to check our model with real true values. This is not the case in real-world scenarios, but we can use the data from our abstract model to provide a baseline estimation for real-world experiments. After each attack, if the attack was successful, we increment a variable, success. If the attack was not successful, we increment a variable, alert. The adversary is determined as successful if they reach the number of successes equal to the margin m before running out of available voters to attack and provided that the number of alerts is less than or equal to a. The more formal view of the attack model is given in Fig 8.1.

Predicting a voter's choice is a complex task that depends on various factors. Several scientific approaches and methodologies have been explored to gain insights into voter behavior. These include opinion polls and surveys [71, 8], election forecasting models [53, 81], social media analysis [27], psychological profiling [59, 56], machine learning and data mining [120, 101], neuroscience and behavioral economics [86], and big data analytics [88]. These methods provide valuable insights into voting behavior and can be used by the adversary to predict voter choices.

The concept of 'margin value' is critical in this context. An adversary must select a margin value that is likely to exceed the actual required margin, as the exact margin is not known beforehand. Our analysis

is further complicated by the fact that we do not know the actual vote choices of the voters in our log file data. Therefore, we cannot determine whether a voter has already voted for a candidate preferred by the adversary. Instead, we consider an attack successful if the voter is categorized as non-verifying and does not verify their vote. Consequently, the margin must be increased based on the expected number of voters who are already voting in line with the adversary's preference.

8.6 Evaluating Classifiers

In order to evaluate the different classifiers, it is not enough to simply check if an adversary (or classifier) would be successful or not, as this provides no information regarding the quality of the classifier. For instance, a poorly performing classifier might accidentally be successful or vice versa. Instead, we create a model that allows us to quantitatively measure a probability, giving the attack probability which allows us to compare adversarial models (classifiers).

In the subsequent phase of our study, following the establishment of the attack model, we focus on calculating the probability of an adversary's success. Formally, we aim to compute the following expression:

$$\Pr[\text{Attack}(\text{classifier} = \text{clf}, \text{margin} = \text{m}, \text{alert} = \text{a}, \text{V} = [\text{X}, \text{y}]) = 1]$$

To achieve this, we adopt a two-pronged approach. Firstly, we derive an analytical formula that encapsulates the variables and conditions of our model. This formula serves as a theoretical framework for understanding the success probability of an adversary under given conditions and allows fast evaluation of the classifiers.

Secondly, we employ a Monte Carlo experiment to estimate this probability empirically. This simulation-based approach allows us to account for the inherent randomness and variability in the real-world voting process, thereby providing a more robust and practical estimate of the success probability.

We then cross-validate these two approaches to ensure their consistency and mutual support. This dual validation strengthens the reliability of our findings and provides a comprehensive understanding of the adversary's success probability. This function serves a dual purpose in our study. Firstly, it enables us to evaluate the performance of different types of adversary attacks. By comparing the success probabilities across different adversary types, we can gain insights into their relative strengths and weaknesses.

Secondly, this function is instrumental in the hyperparameter tuning of our machine learning classifier. By using the success probability as a score function, we can systematically evaluate different parameter combinations and optimize our classifier for maximum performance.

To derive a quantitative measure, we propose randomizing the order of voters, while retaining all data, including voting time and other relevant parameters, followed by a rerun of the attack. By repeating this process a significant number of times, we can estimate the attack probability. For Type C adversaries, we suggest a separation into training, validation, and attacking sets. The adversary could apply this methodology to the validation set to identify the most effective algorithm for the planned attack. However, the validation set might not be the same size as the actual set deployed for the attack, thus the attack probability derived from the validation set might not be directly applicable to the actual attack scenario.

Nonetheless, the validation set might not be the same size as the actual set deployed for the attack, thus the attack probability derived from the validation set might not be directly applicable to the actual attack scenario. Furthermore, performing iterations and classifications multiple times to obtain a reliable

estimate of the probability might be time-consuming. In light of this, we introduce a method to approximate the attack probability grounded in the classification of the validation set. For comparative purposes, the attack set could comprise the entire set of remaining voters post-training.

Suppose a classifier has been validated on a set comprised of N_v voters, of which N_ver are verifying. We obtain an evaluation matrix of the classification, denoted as N_{ij}, where $i, j = 0, 1$. In this context, '0' represents non-verifying, and '1' signifies verifying voters. i pertains to the actual behavior of the voters, while j corresponds to the classification. Thus, N_{00} and N_{11} correspond to non-verifying and verifying voters respectively, which have been accurately classified. Conversely, N_{10} corresponds to false negatives, i.e., voters who verify but were erroneously classified as non-verifiers. These individuals would trigger an alarm when attacked due to the incorrect classification. On the other hand, N_{01} represents false negatives that are classified as verifiers but do not actually verify. While these pose less of a problem in that they do not raise the alarm, they nevertheless reduce the effectiveness of the attack. Note that $N_{11} + N_{10} = N_\text{ver}$ and $N_{01} + N_{00} = N_\text{v} - N_\text{ver}$.

Now we want to estimate the probability p_att of the attack being successful against N voters where each has a probability p_ver of verifying. Note that this is only a good estimation if p_ver is close to the verification rate in the original set which is close to N_ver/N_v. We could have used a fixed number of verifiers out of the N voters, however, this makes the probability formula more complicated and thus unnecessarily slower to compute, and in real life, we would anyway need to estimate the verification rate and distribution. We number the voter in order of how they are attacked as $V_i, i = 1, \ldots, N$. Let VE_i be the stochastic variable for whether V_i is verifying which is then Bernoulli distributed with probability p_ver (and we assume they are independent). Further, we denote by C a stochastic variable giving the classification of V_i. Again we will assume these are independent, and we will assume that their conditional probabilities given VE are Bernoulli distributed with

$$\begin{aligned}\Pr(\mathsf{C} = b|\mathsf{VE} = 1) &= \tfrac{N_{1b}}{N_\text{ver}} := p_{1b}\\ \Pr(\mathsf{C} = b|\mathsf{VE} = 0) &= \tfrac{N_{0b}}{N_\text{v}-N_\text{ver}} := p_{0b}\end{aligned}$$

and correspondingly for $\mathsf{C} = 0$.

We can then compute the attack probability as the adversary manages to change N_mar votes without raising more than A alarms. We assume that the adversary considers voter $V_1, V_2, \ldots$ and for each decides to attack if $\mathsf{C}_i = 0$ i.e. the classifier reports that the voter will not verify. The attacker stops immediately if N_mar votes have been changed. Also, the attacker loses if less than N_mar votes have been changed when applying the attack to all N voters.[2]

$$p_\text{att} = \Pr\left(\exists j \le N.\left(\sum_{i=1}^{j} \neg\mathsf{C}_i \wedge \neg\mathsf{VE} = N_\text{mar}\right) \bigwedge \left(\sum_{i=1}^{j} \neg\mathsf{C}_i \wedge \mathsf{VE}_i \le \mathsf{A}\right)\right)$$

If we consider $\mathsf{A} = 0$ then we simply split into non-intersecting event sets according to j denoting how long the attack had to run to be successful. Note, that if the overall attack is successful, we know that the last voter was attacked successfully, however, since we assume each voter is independent and identically distributed we get a binomial factor from choosing the $N_\text{mar} - 1$ remaining successful vote changes out of the $j - 1$ first voters

$$p_\text{att} = (p_{00}(1-p_\text{ver}))^{N_\text{mar}} \sum_{j=N_\text{mar}}^{N} \tbinom{j-1}{N_\text{mar}-1} (p_{01}(1-p_\text{ver}) + p_{11}\,p_\text{ver})^{j-N_\text{mar}}$$

[2] For those raising an alarm, we cannot count the vote to modified as they might be revoting. We here make the mild assumption that the adversary knows this and thus continues the attack to reach N_mar.

and with the estimates of the probabilities above we get

$$p_{\text{att}} = \left(\frac{N_{00}}{N_{\text{v}}}\right)^{N_{\text{mar}}} \sum_{j=N_{\text{mar}}}^{N} \binom{j-1}{N_{\text{mar}}-1} \left(\frac{N_{01}+N_{11}}{N_{\text{v}}}\right)^{j-N_{\text{mar}}}.$$

Likewise for $\mathsf{A} \geq 0$ we split into non-intersecting event sets according to the attack length j and the actual number of alarms $a \leq A$

$$p_{\text{att}} = (p_{00}(1-p_{\text{ver}}))^{N_{\text{mar}}} \sum_{j=N_{\text{mar}}}^{N} \binom{j-1}{N_{\text{mar}}-1} \times \sum_{a=0}^{\min(A,j-N_{\text{mar}})} \binom{j-N_{\text{mar}}}{a} (p_{\text{ver}}p_{10})^{a} (p_{01}(1-p_{\text{ver}}) + p_{11}p_{\text{ver}})^{j-N_{\text{mar}}-a}$$

Again with the estimates of the probabilities:

$$p_{\text{att}} = \left(\frac{N_{00}}{N_{\text{v}}}\right)^{N_{\text{mar}}} \sum_{j=N_{\text{mar}}}^{N} \binom{j-1}{N_{\text{mar}}-1} \sum_{a=0}^{\min(A,j-N_{\text{mar}})} \binom{j-N_{\text{mar}}}{a} \left(\frac{N_{10}}{N_{\text{v}}}\right)^{a} \left(\frac{N_{01}+N_{11}}{N_{\text{v}}}\right)^{j-N_{\text{mar}}-a}$$

8.6.1 Monte Carlo Experiment for Success Probability Estimation

Monte Carlo experiments are a class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying principle is to use randomness to solve problems that might be deterministic in principle. In the context of our study, we employ a Monte Carlo experiment to estimate the probability of a successful attack by an adversary.

The Monte Carlo experiment for our study is conducted as follows:

1. **Random Sampling:** A multitude of random permutations are generated from the voter list, providing a diverse range of potential voter orderings for the simulation.

2. **Attack Simulation:** Each unique voter ordering is subjected to a simulated attack by the adversary, utilizing the attack model previously defined. The result of each attack, whether successful or unsuccessful, is meticulously recorded for subsequent analysis.

3. **Probability Estimation:** Upon the completion of a substantial number of simulations, the ratio of successful attacks is computed. This ratio serves as an empirical estimate of the adversary's probability of successfully executing an attack.

This Monte Carlo experiment provides a practical and robust method for estimating the success probability of an adversary. By accounting for the inherent randomness and variability in the voting process, it offers a more direct estimate than a purely analytical approach.

Furthermore, by repeating the experiment with different adversary types and attack strategies, we can gain valuable insights into their relative effectiveness.

We have implemented the Monte Carlo experiment in a Jupyter notebook, which is available in the GitHub repository associated with this project [50]. The notebook illustrates how the results of the Monte Carlo experiment converge to the values predicted by our analytical formula after 10,000 iterations. This convergence provides empirical validation of our analytical approach and reinforces the robustness of our methodology. We refer the reader to this notebook for a detailed walkthrough of the Monte Carlo experiment and its results.

8.7 Attack Simulations

We can now perform the attack simulations for all three categories of adversaries, each with its unique models and varying parameters. The total number of models under consideration amounts to 3980, each representing a distinct attack strategy.

The models can be broadly classified into three types. Type A and Type B models are designed to be straightforward and efficient, requiring no training phase. They are capable of receiving data inputs directly and making predictions accordingly. These models are particularly useful for scenarios where rapid response is required, and there is limited time or resources for extensive training.

On the other hand, Type C models are more complex and resource-intensive. They necessitate a comprehensive training phase to fit them with preprocessed data. While this process may require additional time and computational power, the payoff is a more nuanced and potentially more effective attack strategy.

It's important to note that the training and validation phases inherent to Type C models consume some portion of data on their training and validating phases, leaving fewer voters for the actual attack. This is a significant consideration as it means that the number of attacked voters N is larger for Type A and B models than for Type C models. This point warrants further discussion as it impacts the overall effectiveness and applicability of the different adversary types.

To evaluate the effectiveness of each model, we instantiated them for four different margins: 10, 20, 50, and 100. The margin represents the number of vote alterations the adversary aims to achieve. Furthermore, we assessed the success chance of each model for varying numbers of alerts, ranging from 0 to 10. The alert number represents the maximum number of alerts the adversary is willing to trigger without jeopardizing their operation.

Another crucial parameter is the verification rate, which represents the proportion of voters who verify their votes. This rate can significantly impact the success of the attack, as a higher verification rate makes it more challenging for the adversary to manipulate votes undetected.

The number of voters is also a significant factor, as it determines the size of the pool from which the adversary can attempt to manipulate votes. A larger number of voters may provide more opportunities for vote manipulation but also increases the complexity of the attack.

The results of these simulations are meticulously recorded and stored in a Google Sheets document [48] for readability. This comprehensive dataset enables us to compare the effectiveness of different models and identify the most promising strategies for successful election interference.

8.8 Performance Analysis of Attack Models

In this section, we delve into the performance evaluation of the various models employed in our study. The objective is to discern which attack strategies are more effective in predicting voter behavior and consequently, more successful in manipulating election outcomes. This analysis is crucial as it not only provides insights into the strengths and weaknesses of different adversary types but also informs strategies for improving election security. Note that we limit our comparison of models based on different margins.

We emphasize that our comparison of models is limited to those operating under the same margin. This is because the margin, which represents the number of votes an adversary aims to alter, significantly influences the difficulty of the task and the potential effectiveness of the adversary. Therefore, comparing models across different margins could lead to misleading conclusions. By maintaining a consistent margin in our comparisons, we ensure a fair and meaningful evaluation of the model's performance.

The chart depicted in Figure 8.2 illustrates the probability of a successful attack by different adversary models across varying margins. The margins represent the number of vote alterations the adversary aims to achieve, and the probabilities are calculated for a scenario where no alerts are triggered (Alerts: 0). The chart is parameterized by the model type, including Type A1, A2, B1, B2, and C, allowing for a comparative analysis of their performance.

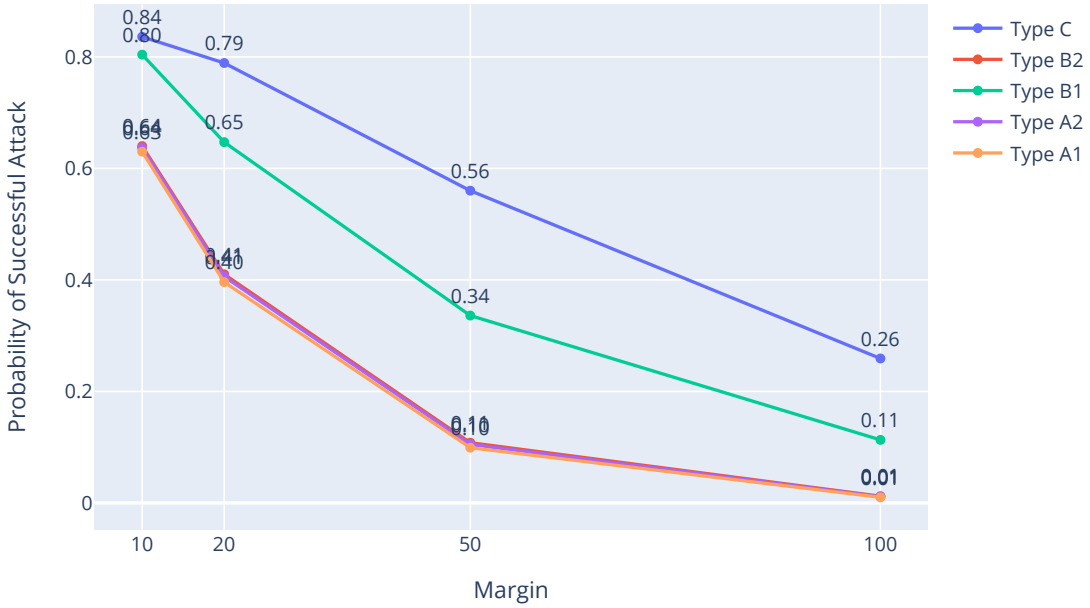


FIGURE 8.2: Probability of Successful Attack by Margin

The chart in Figure 8.3 represents a similar analysis as the previous figure, but with specific consideration for scenarios where 5 alerts are allowed. This means that the adversary thinks that he can trigger up to 5 alerts without jeopardizing their operation.

The choice of adversary strategy is highly contingent on the specific margin of votes they aim to alter and the number of voters available for the attack. It's important to note that Type A and Type B adversaries can attack all voters as they do not require a training phase, unlike Type C adversaries. This distinction should be taken into account when comparing the effectiveness of different adversary types.

Type C adversaries, equipped with machine learning classifiers trained on log files, consistently outperform other types across all margins. Their ability to leverage detailed voter data to make informed predictions about voter behavior gives them a distinct advantage in scenarios ranging from precision targeting to large-scale vote alteration.

Specifically, for smaller margins, the superiority of Type C is evident, and as the margin increases, the gap between Type C and other types widens even further. For example, for a margin of 10, the best Type C model, Random Forest, has a higher success chance compared to the best Type B models, and this advantage grows substantially with larger margins.

Unlike Type B adversaries, who rely on knowledge from scholarly papers and log file analysis but lack access to current log files, Type C adversaries utilize real-time data, allowing for more accurate predictions. This makes Type C not only suitable for precise attacks but also highly effective in scenarios where

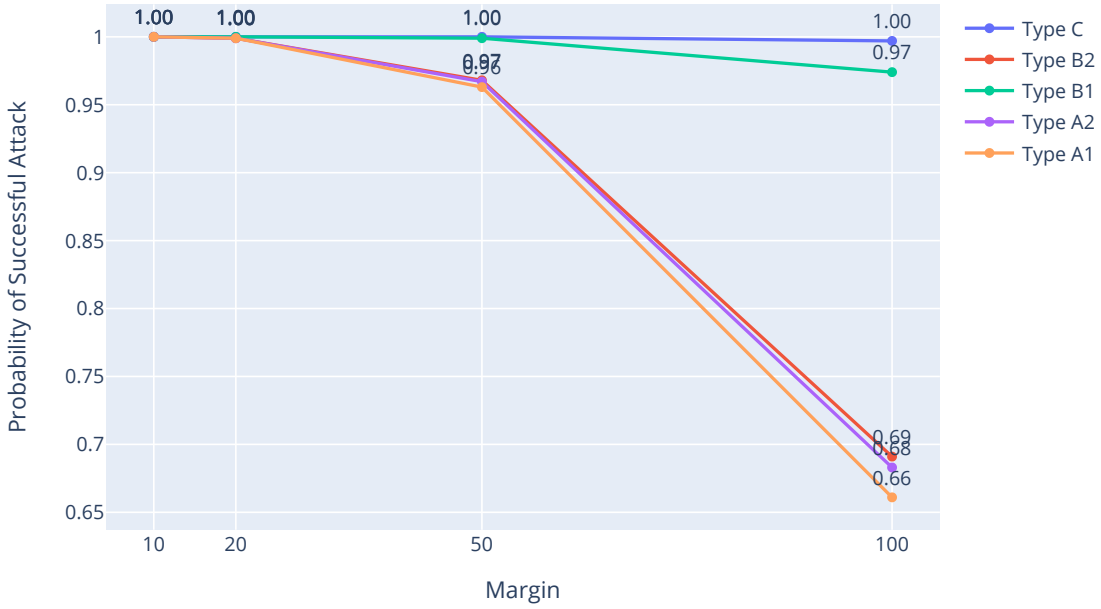


FIGURE 8.3: Probability of Successful Attack With 5 Allowed Alarms by Margin

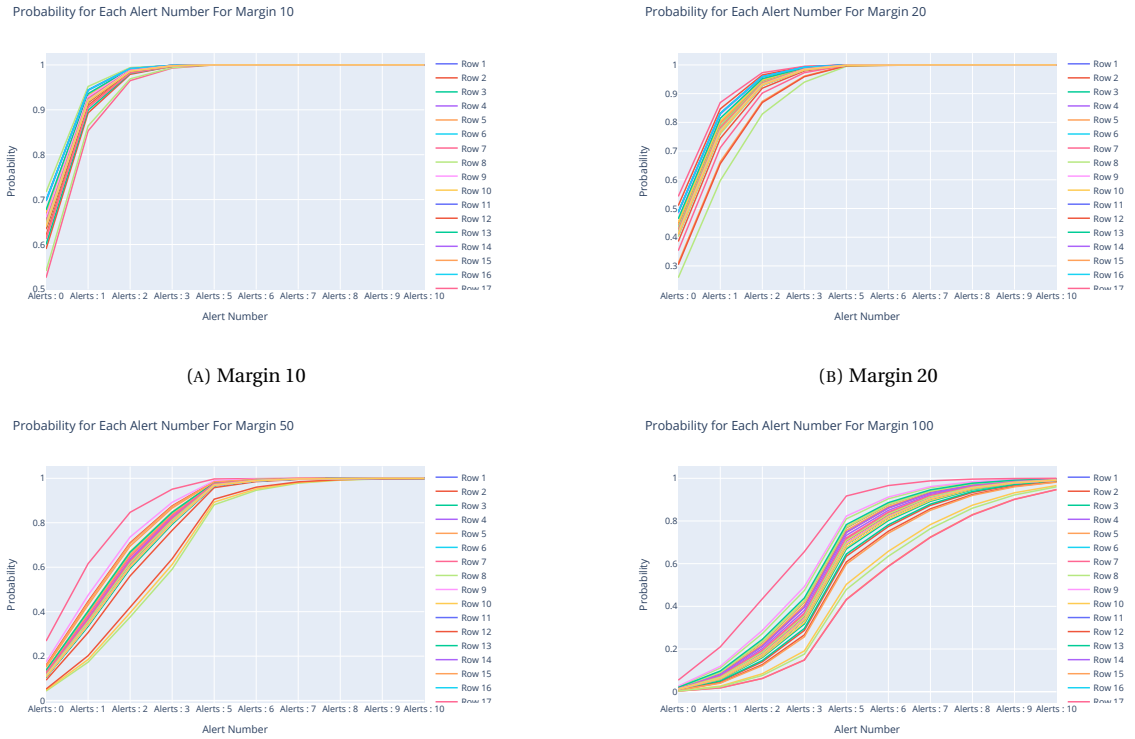
the objective is to alter a larger number of votes.

It's important to clarify that Type B adversaries do not require a training phase. Instead, they create a pivot table of verification rates against different available features based on scholarly papers and research from previous elections. They identify the top k important features that have a strong correlation with verification. Then, they define different variations of these types for their attack strategy.

In conclusion, the choice of adversary model is a strategic decision that depends on the specific objectives and constraints of the adversary. Understanding these dynamics can provide valuable insights for improving election security and developing countermeasures against potential attacks (For more detail about this comparison you can check [50]).

As a future direction, we propose exploring the use of probability outputs from our classifiers. All five types of classifiers, in addition to classifying, also output a probability measure for their predictions. We could define tweaked versions of models by changing the default threshold from 0.5 to higher values to observe the impact on the success rate of the adversaries. This approach could potentially enhance the performance of our models and provide a more nuanced understanding of the adversary's capabilities.

The charts depicted in Figures 8.5, 8.6, and 8.7 provide a comprehensive visual representation of the success chance of an adversary across different types of models for various election margins. These figures specifically pinpoint the first possible alert number that is required to reach specific success probabilities of 0.1, 0.5, and 0.9. Each chart represents one of these thresholds, detailing a layered analysis of potential adversarial activity. We, e.g., see that a type C strategy allowing one alarm and changing 100 votes has a 50% chance of success.



(A) Margin 10

(B) Margin 20

(C) Margin 50

(D) Margin 100

FIGURE 8.4: Probability for Each Alert Number for Different Margins

8.8.1 Descriptive Analysis of Type C Model

The Type C models in our study result from a multi-layered preprocessing pipeline, which includes steps such as imbalanced recovery, feature extraction, feature selection, and the determination of the number of features and components. Each of these steps plays a crucial role in shaping the final model and its performance. In this section, we aim to dissect the impact of each of these steps on the success rate of the adversary.

For each specified margin, we select all models trained with that particular margin in mind. These models are then sorted based on their respective attack scores. From this sorted list, we identify the top 50 models.

Figure 8.4 provides a visual representation of the impact of the number of allowed alarms on the adversary's success probability. Each line in the figure corresponds to one of the top-performing models, illustrating how the success probability of each model changes as the number of allowed alarms increases.

From the figure, it is evident that the success probability of the adversary generally increases with the number of allowed alarms. This is because a higher number of allowed alarms gives the adversary more leeway to alter votes without triggering suspicion. However, the rate of increase varies between models, reflecting the different strategies and capabilities of the adversaries.

Figure 8.8 depicts the distribution of classifier types among the top-performing models. This visualization serves as an exploratory step to identify which classifiers are better suited for our adversarial tasks. It is important to note that each classifier comes with a set of hyperparameters that require careful tuning. Merely selecting a specific classifier does not guarantee superior performance; the hyperparameters must be meticulously adjusted to optimize the model's effectiveness.

Also figure 8.9 presents the distribution of feature selection techniques among the top-performing models. This analysis is crucial in identifying which feature selection methods are more effective for our

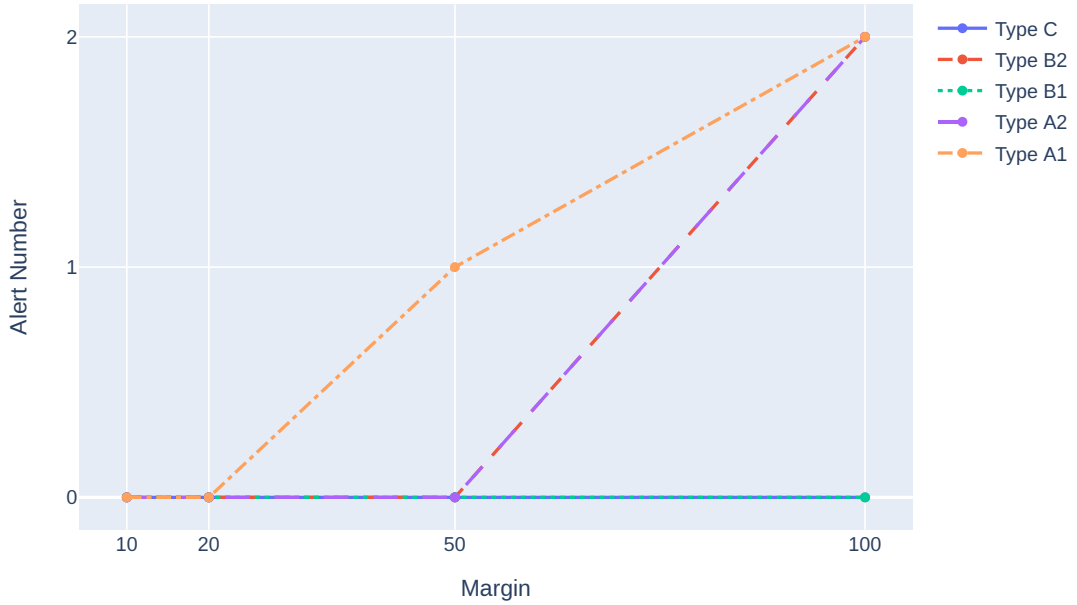


FIGURE 8.5: Margin-Alert Success Rate Of Adversary For 0.1. In this chart, types C and B1 align, as do types B2 and A2.

adversarial tasks. The three feature selection methods under consideration are 'ChiSquare', 'Pearson', and 'MutualInformationGain'. As depicted in the figure, 'ChiSquare' is the most frequently used method in the top models, indicating its relative success in this context.

The comprehensive reports covering various aspects of model performances have been implemented in a Jupyter Notebook file [50].

In our study, we evaluated the performance of our models across two distinct phases: the validation phase and the attack phase. Our objective was to assess the consistency of model performance across these two phases. We found that the degree of consistency between the validation and attack phase rankings varied depending on the margin size. For smaller margins, the top-performing models in the validation phase were largely consistent with those in the attack phase. However, as the margin size increased, the degree of consistency between the two phases diminished.

Several factors could contribute to this observed discrepancy:

- **Overfitting:** Models may perform exceptionally well on the validation set if they have overfitted to the specific characteristics of that set. However, these models may not perform as well when applied to the attack phase, which contains different data.
- **Randomness:** Many machine learning algorithms incorporate a degree of randomnesses, such as random weight initialization or random data splits for training and validation. Consequently, even identical models with the same hyperparameters can yield slightly different results across multiple runs.
- **Data Characteristics:** The validation and attack sets may exhibit different characteristics. For instance, they may contain different types of voters or display different voter behaviors. If a model is

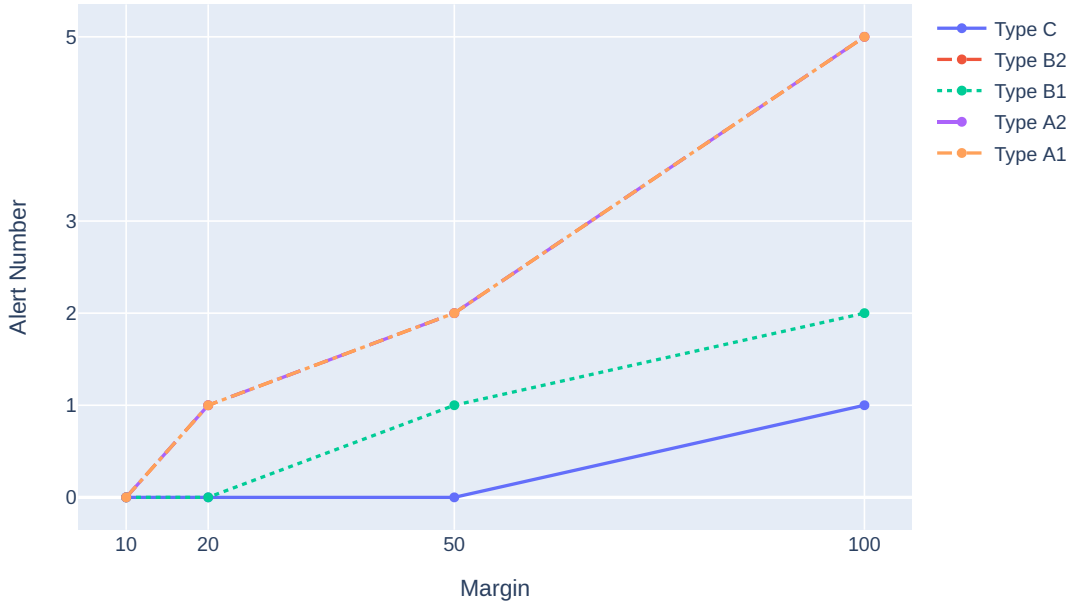


FIGURE 8.6: Margin-Alert Success Rate Of Adversary For 0.5. For this chart, we observe three types A1, A2, and B2 share the same value.

particularly adept at predicting certain types of voters or behaviors, it may perform better on one set than the other.

- **Model Complexity:** More complex models may perform better on the validation set as they can capture more intricate patterns in the data. However, these complex patterns may not generalize well to the attack set, leading to lower performance.

We conducted an analysis to investigate the relationship between the number of attacked voters (N) and the success probability of the adversary. Our findings, as illustrated in Figure 8.10, confirm that the success probability of the adversary is indeed monotonically increasing with respect to N. This holds true across different parameters and is consistent for varying numbers of alerts.

This observation is crucial as it validates our approach of selecting the best algorithm based on its performance with a given N. It assures us that the chosen algorithm will continue to yield the best results even when N is increased. This is because, as the number of attacked voters increases, the success probability of the adversary also increases, given that other parameters are fixed.

This monotonic relationship underscores the importance of voter verification in securing the election process. However, it's important to note that when the total number of voters increases, as in a large-scale national election, even a fixed verification rate could leave a substantial number of voters available for the adversary to attack. This could potentially increase the adversary's success probability. Therefore, it's crucial to not only maintain but also strive to increase the verification rate, especially in large-scale elections. This highlights the need for strategies to encourage voter verification as a means of enhancing election security, particularly in large-scale electoral contexts.

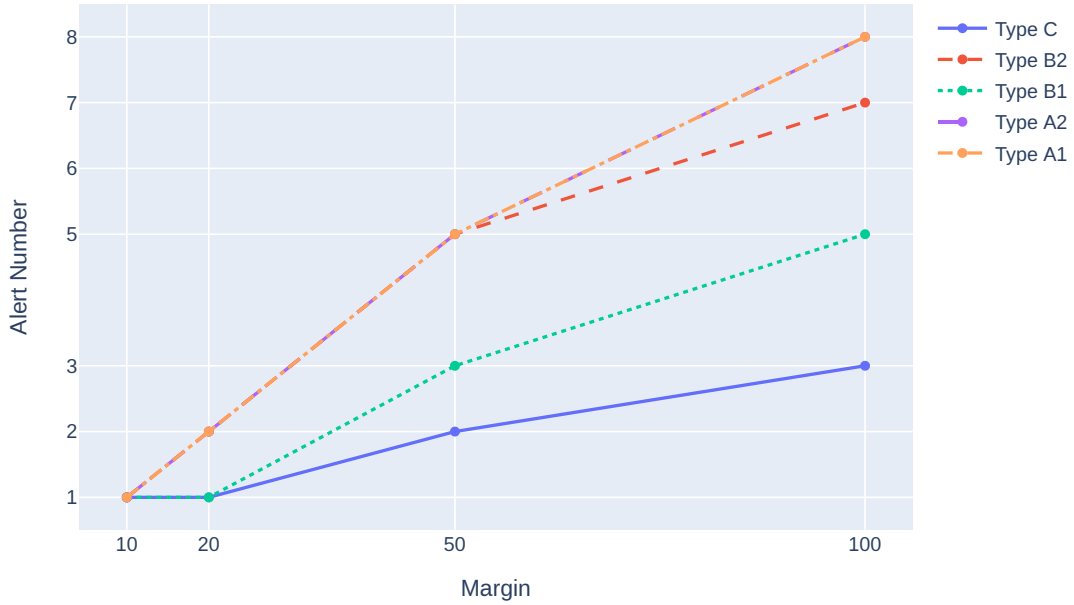


FIGURE 8.7: Margin-Alert Success Rate Of Adversary For 0.9. For this chart, two types of A1 and A2 coincide with each other.

8.9 Lessons Learned

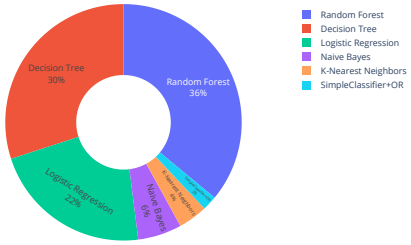
Understanding the behavior of verifying voters is of paramount importance in ensuring the integrity and transparency of an election. Profiling these voters allows us to identify patterns and trends that could be indicative of potential vulnerabilities in the election process. It provides insights into the factors that influence a voter's decision to verify their vote, which can be instrumental in developing strategies to encourage more voters to verify their votes, thereby enhancing the robustness of the election.

Our study presents three types of adversaries, each with their unique strategies and capabilities. While these adversaries were designed with the objective of manipulating election outcomes, the classifiers they employ can be repurposed for more benign objectives. For instance, these classifiers can be used to predict which voters are likely to verify their votes. This information can be used to target voter education campaigns, focusing on those voters who are less likely to verify their votes. By educating these voters about the importance of vote verification, we can potentially increase the overall rate of vote verification, thereby making the election more resistant to manipulation.

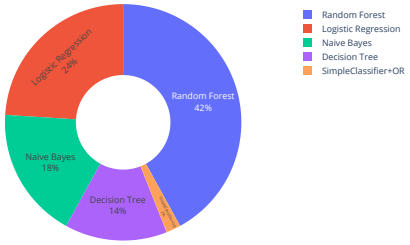
Our initial descriptive analysis examined the correlation between the verification rate and other features. We found that certain features, such as the operating system used by the voter to cast their votes, were more common among voters who verified their votes.

Subsequently, we conducted several layers of feature engineering on our dataset, without directly considering the correlations with verification. Interestingly, we observed that the top models were created based on features that coincided with those identified in our initial correlation analysis.

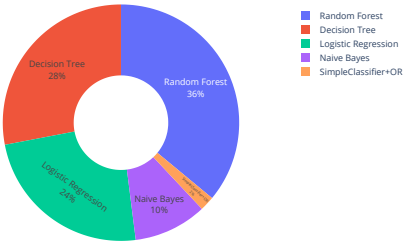
The research findings presented in this chapter offer a compelling insight into the capabilities and effectiveness of different adversary types in manipulating election outcomes. Among the three types of



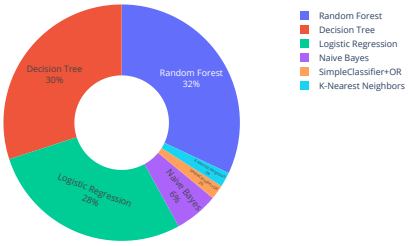
(A) Margin 10



(B) Margin 20



(C) Margin 50



(D) Margin 100

FIGURE 8.8: Distribution of Different Classifier On Top Models

adversaries examined, Type C emerges as a particularly potent threat, consistently outperforming Types A and B across various scenarios.

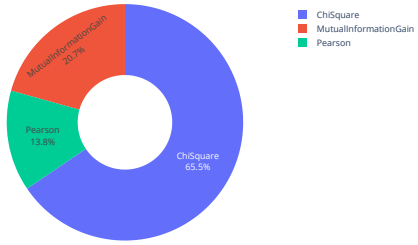
What sets Type C apart is its utilization of machine-learning models that are trained on log files. These models are capable of discerning intricate patterns and insights within the data, enabling the adversary to make highly informed predictions about voter behavior. This ability to learn from the data not only enhances the success rate of Type C but also makes it resilient to challenges that diminish the efficiency of other types.

As the number of vote alterations increases, representing larger-scale attacks, Types A and B experience a significant decline in their effectiveness. In contrast, Type C maintains its superiority, demonstrating impressive adaptability to varying attack scales. For instance, when attempting to alter 100 votes with just one alert, the success probability of Type C exceeds 0.5, a testament to its robust performance.

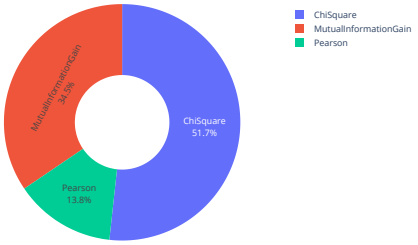
In a more complex scenario, where the adversary aims to change the coalition by altering around 2000 votes, the superiority of Type C becomes even more pronounced. With a permissible threshold of 40 alerts, the Random Forest model under Type C exhibits a remarkable 82 percent success rate. This is a significant achievement, especially when contrasted with other types, most of which have no chance of success in this context. The only exception is Type B2, which still lags considerably behind with a 26 percent chance. This also underscores the importance of taking alerts seriously and investigating them.

This specific example underscores the robustness and adaptability of Type C, particularly when employing sophisticated machine-learning models like Random Forest. It demonstrates that Type C is not only effective in scenarios requiring precision but also in large-scale vote alteration scenarios where the stakes are high, such as changing a coalition.

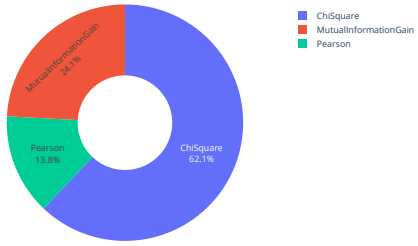
For a more detailed discussion of these findings, including a comprehensive analysis of the models and features, please refer to the final report of our study [48].



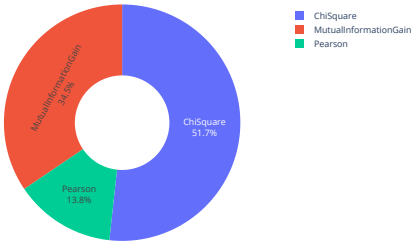
(A) Margin 10



(B) Margin 20



(C) Margin 50



(D) Margin 100

FIGURE 8.9: Feature Selection Distribution of Top Models for Different Margins

8.10 Future Work

Our study opens up several avenues for future research and improvement. Here are some potential directions that could be explored:

1. **Application of Deep Learning:** The use of deep learning could be explored in the context of our scoring metric. Training these networks according to our scoring metric could potentially enhance the accuracy and efficiency of the attack models. Deep learning has the added advantage of performing feature selection and extraction under the hood, providing a more nuanced understanding of the election dynamics and the potential impact of vote alterations.

2. **Candidate Scoring:** One possible enhancement could be to introduce a scoring metric for each candidate after the election results are published. This score could reflect the difficulty of changing the candidate's status from elected to non-elected, or vice versa. Such a metric could provide valuable insights into the robustness of the election results and help identify potential vulnerabilities. It could also serve as a useful tool for election authorities in assessing the effectiveness of their security measures and in devising strategies to enhance election integrity.

3. **Differentiating Candidates Based on Seat Types:** Another potential improvement could be to differentiate between election candidates based on their seat types. In the current system, seats are allocated in three different layers: Personal, District, and Compensation, which we can refer to as P-type, D-type, and C-types, respectively. Differentiating candidates based on these seat types could provide a more nuanced understanding of the election dynamics and the potential impact of vote alterations. It could also help devise more targeted and effective strategies for preventing and detecting election manipulation.

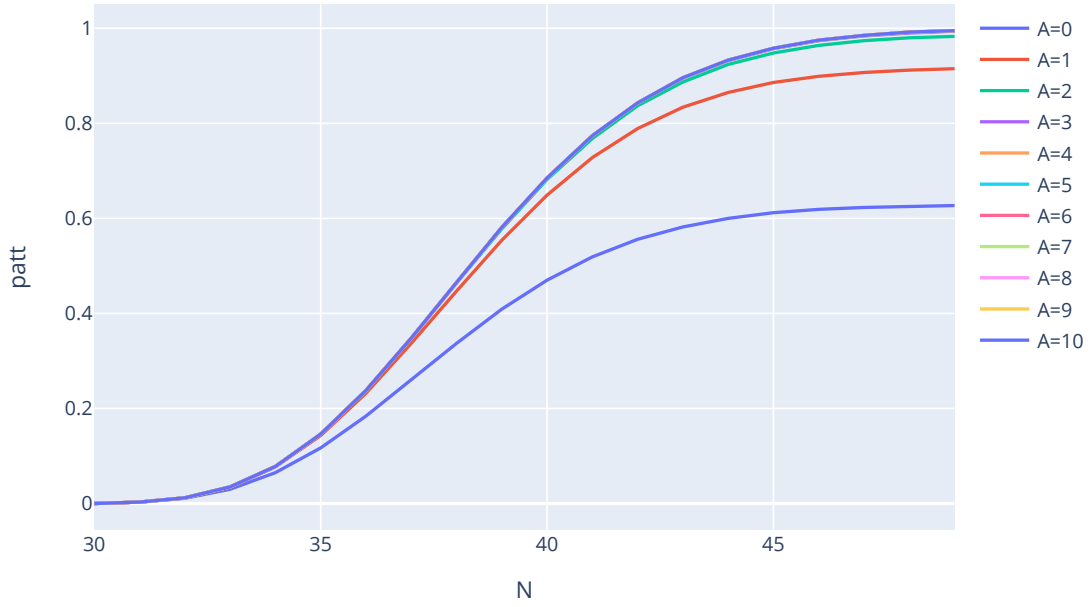


FIGURE 8.10: Success probability of the adversary as a function of the number of attacked voters (N), parameterized by the number of alerts. Each line represents a different number of alerts, ranging from 0 to 10.

These proposed enhancements, along with the findings from our study, could contribute to the ongoing efforts to ensure the integrity and transparency of elections. They highlight the importance of continuous research and innovation in the field of election security, particularly in the face of evolving threats and challenges.

Part III

Conclusion

CHAPTER 9

CONCLUSIONS AND FURTHER WORK

In this concluding chapter, we synthesize the key findings and insights from our multifaceted research journey, which is segmented into three distinct contributions. The first contribution explores the complex landscape of attacks and repairs on the NV12 scheme, with an emphasis on protocols resilient to human errors and the integration of biometric data. The second contribution refines existing privacy definitions, introducing a novel concept of delay-use malicious-ballot box ballot privacy (du–mb–BPRIV), and highlights the challenges and opportunities in privacy definitions. The third contribution delves into the critical aspect of understanding voter behavior, particularly in the context of vote verification, and outlines future directions for enhancing election security.

9.1 Revisiting Practical and Usable Coercion-Resistant Remote E-Voting

In our first contribution, we have embarked on a comprehensive exploration of attacks and repairs on the NV12 scheme, with a particular focus on protocols that are resilient to human errors, such as PIN typos. This work has revealed intriguing possibilities for enhancing security through the integration of biometric data. Specifically, we have posited that the digitally stored key could be augmented or even supplanted by a key derived from biometric information. This opens up an exciting avenue for future research, where the challenge lies in making the error correction so proficient that it can accommodate noisy biometric data without the need for fuzzy extraction.

Furthermore, we have presented a Paillier-based system, and it has become evident that the natural progression would be to incorporate an MPC-based tally system like that from Ordinos [84]. This integration is particularly appealing as it is also founded on Paillier encryption. Ordinos, with their ability to reveal only the winner or the ranking of candidates in an election, can significantly contribute to coercion-resistance, especially in scenarios where certain candidates are anticipated to receive minimal or no votes. Additionally, the risk-limiting tally method described in [75] presents itself as a viable option for both protocols, offering plausible deniability for the voter.

Our analysis of the PIN space has unearthed intriguing connections and may be of broader interest. More precise results are yet to be discovered, but the relationship between one-digit error in k-digit PINs

and Rook-polynomials [5] on a k-dimensional chessboard is a fascinating observation that warrants further investigation.

The socio-technical dimensions of our contribution have also raised compelling questions that extend beyond the technical realm. These include:

1. Understanding the nature of PIN errors made by voters in a voting setting where no feedback on PIN correctness is provided.

2. Determining the optimal PIN policy that strikes a balance between correcting as many PIN typos as possible and maintaining the entropy of the PIN space at a sufficiently high level.

3. Investigating the efficacy of voter training in handling, faking, and concealing secret keys, especially in scenarios where a smart card is not used or is used in conjunction with key storage.

Lastly, a critical aspect that remains to be addressed is the provision of a robust proof of security for our protocols. This constitutes a vital component of our future work, as it will solidify the theoretical foundations of our contributions and provide a more comprehensive understanding of the underlying security mechanisms.

In conclusion, our first contribution has not only advanced the field by presenting novel attacks, repairs, and protocols but has also laid the groundwork for future research in areas ranging from biometric integration to voter behavior analysis. The intersection of technology, mathematics, and human factors presents a rich tapestry of opportunities for continued exploration and innovation.

9.2 Machine-Checked Proofs of Privacy Against Malicious Boards for Selene & Co

In our second contribution, we have presented a refined version of the mb – BPRIV privacy definition, introducing a new concept we term delay-use malicious-ballot box ballot privacy du–mb–BPRIV. This innovative definition facilitates the modeling of schemes such as Selene, where verification takes place subsequent to tallying. Moreover, the security claim associated with this definition is more explicit, providing a clearer understanding of the underlying principles.

We have taken the significant step of formalizing our new definition within the interactive theorem prover EasyCrypt, demonstrating that labeled MiniVoting, Belenios, and Selene all conform to this definition. Additionally, we have established that MiniVoting and Belenios are in compliance with the original mb – BPRIV privacy definition.

While our encoding of Selene aligns with what we firmly believe to be the most suitable privacy definition in existing literature, our work has illuminated certain defiances in privacy definitions that future research must address. These defiances are profound and extend beyond the scope of this work. However, we will briefly touch upon two principal defiances of mb – BPRIV and related definitions:

1. The definition, although adept at handling a malicious bulletin board, operates under the assumption of an honest setup.

2. The du–mb–BPRIV and related definitions are finely tuned to schemes where the auxiliary data generated by the tally for verification purposes are zero-knowledge proofs. This makes it challenging to express schemes like Selene, which utilizes trackers for verification.

Furthermore, the BPRIV style of definition imposes certain limitations. For instance, the adversary's view is restricted to the result and verification from the left side board, constraining the attacker model.

This means that privacy attacks that rely on inducing candidate-specific errors for an observed voter—while granting the adversary access to whether the corresponding voter verification fails or not (see e.g., [82] for such a style of attack)—cannot be detected. While such attacks can be negated by considering recovery functions that prevent alterations to honestly cast votes as in [41], this is not universally applicable.

An essential trajectory for future research, therefore, lies in the discovery of alternative definitions that encapsulate both more general and more transparent attacker models. This may involve reducing the generality of the definition or considering simulation-based security.

In conclusion, our second contribution has not only refined existing privacy definitions but also paved the way for future explorations into more nuanced and comprehensive privacy models. By identifying and addressing the limitations and challenges inherent in current definitions, we have opened up new horizons for research, setting the stage for the development of more robust and adaptable privacy frameworks. The implications of this work extend beyond the immediate context, offering valuable insights and directions for the broader field of privacy and security.

9.3 Evaluating Security Guarantees from Individual Verification in Estonian E-Voting

In our third contribution, we have delved into the critical aspect of understanding voter behavior, particularly in the context of vote verification. This understanding is vital in safeguarding the integrity and transparency of elections. By profiling voters, we have been able to discern patterns and trends that may signal potential vulnerabilities in the election process. Our insights into the factors that influence a voter's decision to verify their vote have laid the groundwork for strategies to motivate more voters to engage in verification, thereby fortifying the election process.

Our study has introduced three distinct types of adversaries, each equipped with unique strategies and capabilities. While initially conceived for manipulating election outcomes, the classifiers employed by these adversaries can be adapted for more benign purposes, such as predicting voter verification behavior. This predictive capability can be harnessed to target voter education campaigns, focusing on those less likely to verify their votes, with the potential to enhance overall vote verification rates.

Through a series of analyses, we have identified correlations between verification rates and specific features, such as the voter's operating system. Subsequent feature engineering has further corroborated these findings, leading to the creation of top models that align with our initial correlation analysis.

Our study has opened up a plethora of avenues for future research and enhancement, including:

1. **Application of Deep Learning:** Exploring deep learning in the context of our scoring metric could lead to more accurate and efficient attack models. The inherent feature selection and extraction capabilities of deep learning offer a more nuanced understanding of election dynamics and the potential ramifications of vote alterations.

2. **Candidate Scoring:** Introducing a scoring metric for candidates post-election could provide valuable insights into the robustness of election results and uncover potential vulnerabilities. This metric could be instrumental for election authorities in evaluating and bolstering their security measures.

3. **Differentiating Candidates Based on Seat Types:** Differentiating candidates by seat types (P-type, D-type, and C-types) could offer a more refined understanding of election dynamics and the potential impact of vote alterations. This differentiation could aid in crafting more targeted strategies to thwart election manipulation.

Our third contribution has not only shed light on the complexities of voter behavior and verification but has also provided actionable insights and tools to enhance election security. The proposed enhancements and findings underscore the necessity of ongoing research and innovation in election security, especially in an era marked by ever-evolving threats and challenges.

The lessons learned from this study, coupled with the future directions outlined, contribute to a broader effort to ensure the integrity and transparency of elections. By continually pushing the boundaries of understanding and technology, we can build more resilient and trustworthy democratic processes, reinforcing the very foundations of our society.

BIBLIOGRAPHY

[1] Masayuki Abe and Fumitaka Hoshino. 'Remarks on mix-network based on permutation networks'. In: *Public Key Cryptography: 4th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001 Cheju Island, Korea, February 13–15, 2001 Proceedings 4*. Springer. 2001, pp. 317–324.

[2] *Accessing Estonian Election Log Files*. In-person access. 2023.

[3] Ben Adida. 'Helios: Web-based Open-Audit Voting.' In: *USENIX security symposium*. Vol. 17. 2008, pp. 335–348.

[4] Ben Adida et al. 'Electing a university president using open-audit voting: Analysis of real-world use of Helios'. In: *EVT/WOTE* 9.10 (2009).

[5] Reginald BJT Allenby and Alan Slomson. *How to count: An introduction to combinatorics*. Chapman and Hall/CRC, 2010.

[6] Gilles Barthe et al. *EasyCrypt: Computer-Aided Cryptographic Proofs*. 2018.

[7] Stephanie Bayer and Jens Groth. 'Efficient zero-knowledge argument for correctness of a shuffle'. In: *Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31*. Springer. 2012, pp. 263–280.

[8] Nicholas Beauchamp. 'Predicting and interpolating state-level polls using Twitter textual data'. In: *American Journal of Political Science* 61.2 (2017), pp. 490–503.

[9] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. 'Public-key encryption in a multi-user setting: Security proofs and improvements'. In: *Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19*. Springer. 2000, pp. 259–274.

[10] Mihir Bellare and Amit Sahai. 'Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization'. In: *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19*. Springer. 1999, pp. 519–536.

[11] Josh Benaloh. 'Ballot Casting Assurance via Voter-Initiated Poll Station Auditing.' In: *EVT* 7 (2007), pp. 14–14.

[12] Josh Benaloh. *Improving Privacy in Cryptographic Elections*. Tech. rep. MSR-TR-1994-3. Microsoft, 1994. URL: https://www.microsoft.com/en-us/research/publication/improving-privacy-cryptographic-elections/.

[13] Josh Benaloh. 'Rethinking voter coercion: The realities imposed by technology'. In: *2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 13)*. 2013.

[14] Josh Benaloh. 'Simple Verifiable Elections.' In: *EVT* 6 (2006), pp. 5–5.

[15] Josh Benaloh and Dwight Tuinstra. 'Receipt-free secret-ballot elections'. In: *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*. 1994, pp. 544–553.

[16] Josh Benaloh and Dwight Tuinstra. 'Receipt-Free Secret-Ballot Elections (Extended Abstract)'. In: *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*. STOC '94. Montreal, Quebec, Canada: Association for Computing Machinery, 1994, 544–553. ISBN: 0897916638. DOI: 10.1145/195058.195407. URL: https://doi-org.proxy.bnl.lu/10.1145/195058.195407.

[17] Josh C Benaloh and Moti Yung. 'Distributing the power of a government to enhance the privacy of voters'. In: *Proceedings of the fifth annual ACM symposium on Principles of distributed computing*. 1986, pp. 52–62.

[18] David Bernhard. 'Zero-knowledge proofs in theory and practice'. PhD thesis. University of Bristol, 2014.

[19] David Bernhard, Olivier Pereira, and Bogdan Warinschi. 'How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios'. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2012, pp. 626–643.

[20] David Bernhard and Ben Smyth. 'Ballot secrecy with malicious bulletin boards'. In: *Cryptology ePrint Archive* (2014).

[21] David Bernhard et al. 'Adapting Helios for provable ballot privacy'. In: *European Symposium on Research in Computer Security*. Springer. 2011, pp. 335–354.

[22] David Bernhard et al. 'SoK: A comprehensive analysis of game-based ballot privacy definitions'. In: *2015 IEEE Symposium on Security and Privacy*. IEEE. 2015, pp. 499–516.

[23] Daniel R. Biggers et al. 'Can Addressing Integrity Concerns about Mail Balloting Increase Turnout? Results from a Large-Scale Field Experiment in the 2020 Presidential Election'. In: *Journal of Experimental Political Science* (2022). DOI: 10.1017/xps.2022.31.

[24] Dan Boneh and Matthew Franklin. 'Efficient generation of shared RSA keys'. In: *Advances in Cryptology—CRYPTO'97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17*. Springer. 1997, pp. 425–439.

[25] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. 'Evaluating 2-DNF formulas on ciphertexts'. In: *Theory of cryptography conference*. Springer. 2005, pp. 325–341.

[26] Dan Boneh and Philippe Golle. 'Almost entirely correct mixing with applications to voting'. In: *Proceedings of the 9th ACM conference on Computer and communications security*. 2002, pp. 68–77.

[27] Kellyton Dos Santos Brito, Rogério Luiz Cardoso Silva Filho, and Paulo Jorge Leitão Adeodato. 'A systematic review of predicting elections based on social media data: research challenges and future directions'. In: *IEEE Transactions on Computational Social Systems* 8.4 (2021), pp. 819–843.

[28] Jurlind Budurushi, Stephan Neumann, and Melanie Volkamer. 'Smart cards in electronic voting: lessons learned from applications in legally-binding elections and approaches proposed in scientific papers'. In: *5th International Conference on Electronic Voting 2012 (EVOTE2012)*. Gesellschaft für Informatik eV. 2012.

[29] Sergiu Bursuc, Constantin-Catalin Dragan, and Steve Kremer. 'Private votes on untrusted platforms: models, attacks and provable scheme'. In: *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2019, pp. 606–620.

[30] Ran Canetti, Alley Stoughton, and Mayank Varia. 'Easyuc: Using easycrypt to mechanize proofs of universally composable security'. In: *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*. IEEE. 2019, pp. 167–16716.

[31] N. Chan, Joyce H. Nguy, and Natalie Masuoka. 'The Asian American Vote in 2020: Indicators of Turnout and Vote Choice'. In: *Political Behavior* (2022). DOI: 10.1007/s11109-022-09844-9.

[32] David Chaum, Peter YA Ryan, and Steve Schneider. 'A practical voter-verifiable election scheme'. In: *Computer Security–ESORICS 2005: 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005. Proceedings 10*. Springer. 2005, pp. 118–139.

[33] David Chaum et al. 'Scantegrity: End-to-end voter-verifiable optical-scan voting'. In: *IEEE Security & Privacy* 6.3 (2008), pp. 40–46.

[34] David L Chaum. 'Untraceable electronic mail, return addresses, and digital pseudonyms'. In: *Communications of the ACM* 24.2 (1981), pp. 84–90.

[35] Benoît Chevallier-mames et al. 'On Some Incompatible Properties of Voting Schemes'. In: *In IAVoSS Workshop On Trustworthy Elections, WOTE '06*. 2006.

[36] Nikos Chondros et al. 'D-DEMOS: A distributed, end-to-end verifiable, internet voting system'. In: *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2016, pp. 711–720.

[37] Michael R Clarkson, Stephen Chong, and Andrew C Myers. 'Civitas: Toward a secure voting system'. In: *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE. 2008, pp. 354–368.

[38] Cloudflare Blog. 'Using EasyCrypt and Jasmin for post-quantum verification'. In: (2022). https://blog.cloudflare.com/using-easycrypt-and-jasmin-for-post-quantum-verification/.

[39] Josh D. Cohen and Michael J. Fischer. 'A robust and verifiable cryptographically secure election scheme'. In: *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*. 1985, pp. 372–382. DOI: 10.1109/SFCS.1985.2.

[40] Véronique Cortier and Joseph Lallemand. 'Voting: You can't have privacy without individual verifiability'. In: *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 2018, pp. 53–66.

[41] Véronique Cortier, Joseph Lallemand, and Bogdan Warinschi. 'Fifty shades of ballot privacy: Privacy against a malicious board'. In: *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*. IEEE. 2020, pp. 17–32.

[42] Véronique Cortier et al. 'Election verifiability for helios under weaker trust assumptions'. In: *Computer Security-ESORICS 2014: 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part II 19*. Springer. 2014, pp. 327–344.

[43] Véronique Cortier et al. 'Machine-checked proofs of privacy for electronic voting protocols'. In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017, pp. 993–1008.

[44] Michael Radwin December, Michael J. Radwin, and Professor Phil Klein. *An Untraceable, Universally Verifiable Voting Scheme*. 1995.

[45] Stéphanie Delaune, Steve Kremer, and Mark Ryan. 'Verifying privacy-type properties of electronic voting protocols'. In: *Journal of Computer Security* 17.4 (2009), pp. 435–487.

[46] Stéphanie Delaune, Steve Kremer, and Mark D Ryan. 'Receipt-freeness: Formal definition and fault attacks'. In: *Proceedings of the Workshop Frontiers in Electronic Elections (FEE 2005), Milan, Italy*. 2005, pp. 15–16.

[47] Constantin Cătălin Drăgan et al. 'Machine-Checked Proofs of Privacy Against Malicious Boards for Selene & Co'. In: *2022 IEEE 35th Computer Security Foundations Symposium (CSF)*. IEEE. 2022, pp. 335–347.

[48] ESTAJI E. *Estonian Reports*. https://docs.google.com/spreadsheets/d/17IKFvzEDOWvagpqc45TeC edit?usp=sharing. Accessed: 2023-07-28. 2023.

[49] Taher ElGamal. 'A public key cryptosystem and a signature scheme based on discrete logarithms'. In: *IEEE transactions on information theory* 31.4 (1985), pp. 469–472.

[50] Ehsan Estaji. *EstonianElection-PublicVersion*. https://github.com/ehsanestaji/EstonianElection-PublicVersion. 2023.

[51] Ehsan Estaji et al. 'Revisiting practical and usable coercion-resistant remote e-voting'. In: *Electronic Voting: 5th International Joint Conference, E-Vote-ID 2020, Bregenz, Austria, October 6–9, 2020, Proceedings 5*. Springer. 2020, pp. 50–66.

[52] Christian Feier, Stephan Neumann, and Melanie Volkamer. *Coercion-resistant internet voting in practice*. Gesellschaft für Informatik eV, 2014.

[53] Morris P Fiorina. 'Economic retrospective voting in American national elections: A micro-analysis'. In: *American Journal of political science* (1978), pp. 426–443.

[54] Dinei Florencio and Cormac Herley. 'A large-scale study of web password habits'. In: *Proceedings of the 16th international conference on World Wide Web*. 2007, pp. 657–666.

[55] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. 'A Practical Secret Voting Scheme for Large Scale Elections'. In: *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*. ASIACRYPT '92. Berlin, Heidelberg: Springer-Verlag, 1992, 244–251. ISBN: 3540572201.

[56] Alan S Gerber et al. 'The big five personality traits in the political arena'. In: *Annual Review of Political Science* 14 (2011), pp. 265–287.

[57] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[58] Shafi Goldwasser, Silvio Micali, and Chales Rackoff. 'The knowledge complexity of interactive proof-systems'. In: *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*. 2019, pp. 203–225.

[59] Donald Granberg and Sören Holmberg. *The political system matters: Social psychology and voting behavior in Sweden and the United States*. Cambridge University Press, 1988.

[60] Gurchetan S Grewal et al. 'Caveat coercitor: Coercion-evidence in electronic voting'. In: *2013 IEEE Symposium on Security and Privacy*. IEEE. 2013, pp. 367–381.

[61] Panagiotis Grontas et al. 'Towards everlasting privacy and efficient coercion resistance in remote electronic voting'. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2018, pp. 210–231.

[62] Panagiotis Grontas et al. 'Towards everlasting privacy and efficient coercion resistance in remote electronic voting'. In: *Financial Cryptography and Data Security: FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers 22*. Springer. 2019, pp. 210–231.

[63] Jens Groth and Amit Sahai. 'Efficient noninteractive proof systems for bilinear groups'. In: *SIAM Journal on Computing* 41.5 (2012), pp. 1193–1232.

[64] Thomas Haines and Ben Smyth. 'Surveying definitions of coercion resistance'. In: *Cryptology ePrint Archive* (2019).

[65] Ramzi A Haraty, Hadi Otrok, and Abdul Nasser El-Kassar. 'A Comparative Study of Elgamal Based Cryptographic Algorithms.' In: *ICEIS (3)*. 2004, pp. 79–84.

[66] Liran Harsgor and N. Nevitte. 'Do Leader Evaluations (De)Mobilize Voter Turnout? Lessons From Presidential Elections in the United States'. In: *Politics and Governance* 10.4 (2022). DOI: 10.17645/pag.v10i4.5723.

[67] Carmit Hazay et al. 'Efficient RSA key generation and threshold paillier in the two-party setting'. In: *Journal of Cryptology* 32 (2019), pp. 265–323.

[68] Sven Heiberg, Peeter Laud, and Jan Willemson. 'The application of i-voting for Estonian parliamentary elections of 2011'. In: *International Conference on E-Voting and Identity*. Springer. 2011, pp. 208–223.

[69] Sven Heiberg, Arnis Parsovs, and Jan Willemson. 'Log Analysis of Estonian Internet Voting 2013–2015'. In: *Cryptology ePrint Archive* (2015).

[70] Sven Heiberg and Jan Willemson. 'Verifiable internet voting in Estonia'. In: *2014 6th international conference on electronic voting: Verifying the vote (evote)*. IEEE. 2014, pp. 1–8.

[71] D Sunshine Hillygus. 'The evolution of election polling in the United States'. In: *Public opinion quarterly* 75.5 (2011), pp. 962–981.

[72] Martin Hirt and Kazue Sako. 'Efficient receipt-free voting based on homomorphic encryption'. In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2000, pp. 539–556.

[73] Charles Antony Richard Hoare. 'An axiomatic basis for computer programming'. In: *Communications of the ACM* 12.10 (1969), pp. 576–580.

[74] Vincenzo Iovino et al. 'Using selene to verify your vote in JCJ'. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2017, pp. 385–403.

[75] Wojciech Jamroga et al. 'Risk-limiting tallies'. In: *Electronic Voting: 4th International Joint Conference, E-Vote-ID 2019, Bregenz, Austria, October 1–4, 2019, Proceedings 4*. Springer. 2019, pp. 183–199.

[76] Hugo L Jonker and Erik P de Vink. 'Formalising receipt-freeness'. In: *Information Security: 9th International Conference, ISC 2006, Samos Island, Greece, August 30-September 2, 2006. Proceedings 9*. Springer. 2006, pp. 476–488.

[77] Bengt Jonsson, Wang Yi, and Kim G Larsen. 'Probabilistic extensions of process algebras'. In: *Handbook of process algebra*. Elsevier, 2001, pp. 685–710.

[78] Ari Juels, Dario Catalano, and Markus Jakobsson. 'Coercion-resistant electronic elections'. In: *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*. 2005, pp. 61–70.

[79] Backus JW. 'c The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference'. In: *Conference, on,, Inform&on, Processing*. 1959, pp. 125–131.

[80] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2020.

[81] Scott Keeter, Ruth Igielnik, and Rachel Weisel. 'Can likely voter models be improved?' In: *Pew Research Center* (2016), pp. 1–12.

[82] Steve Kremer and Peter B Rønne. 'To du or not to du: A security analysis of du-vote'. In: *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2016, pp. 473–486.

[83] Oksana Kulyk, Vanessa Teague, and Melanie Volkamer. 'Extending helios towards private eligibility verifiability'. In: *International Conference on E-Voting and Identity*. Springer. 2015, pp. 57–73.

[84] Ralf Küsters et al. 'Ordinos: A verifiable tally-hiding e-voting system'. In: *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2020, pp. 216–235.

[85] Byoungcheon Lee et al. 'Providing Receipt-Freeness in Mixnet-Based Voting Protocols'. In: *Information Security and Cryptology - ICISC 2003*. Ed. by Jong-In Lim and Dong-Hoon Lee. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 245–258. ISBN: 978-3-540-24691-6.

[86] I-Ching Lee et al. 'Are we rational or not? The exploration of voter choices during the 2016 presidential and legislative elections in Taiwan'. In: *Frontiers in Psychology* 8 (2017), p. 1762.

[87] Helger Lipmaa and Tomas Toft. 'Secure equality and greater-than tests with sublinear online complexity'. In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2013, pp. 645–656.

[88] Amaryllis Mavragani and Konstantinos P Tsagarakis. 'Predicting referendum results in the Big Data Era'. In: *Journal of Big Data* 6.1 (2019), pp. 1–20.

[89] Tshegofatso Mogaladi and M. Mlambo. 'Demographic and Systematic Factors Affecting Student Voter Turnout: An Empirical Study For Africa's Largest Distance Higher Education Institution'. In: *Journal of Student Affairs in Africa* 10.2 (2022). DOI: 10.24085/jsaa.v10i2.3574.

[90] André Silva Neto et al. 'Usability considerations for coercion-resistant election systems'. In: *Proceedings of the 17th Brazilian Symposium on Human Factors in Computing Systems*. 2018, pp. 1–10.

[91] Stephan Neumann and Melanie Volkamer. 'Civitas and the real world: problems and solutions from a practical point of view'. In: *2012 Seventh International Conference on Availability, Reliability and Security*. IEEE. 2012, pp. 180–185.

[92] Stephan Neumann et al. 'Towards a practical jcj/civitas implementation'. In: (2013).

[93] Takashi Nishide and Kouichi Sakurai. 'Distributed paillier cryptosystem without trusted dealer'. In: *Information Security Applications: 11th International Workshop, WISA 2010, Jeju Island, Korea, August 24-26, 2010, Revised Selected Papers 11*. Springer. 2011, pp. 44–60.

[94] Lawrence D Norden et al. *Post-election audits: Restoring trust in elections*. Brennan Center for Justice, 2007.

[95] Tatsuaki Okamoto. 'Receipt-free electronic voting schemes for large scale elections'. In: *Security Protocols: 5th International Workshop Paris, France, April 7–9, 1997 Proceedings 5*. Springer. 1998, pp. 25–35.

[96] M Maina Olembo et al. 'Voter, what message will motivate you to verify your vote?' In: *Workshop on Usable Security*. 2014.

[97] Maina M Olembo, Steffen Bartsch, and Melanie Volkamer. 'Mental models of verifiability in voting'. In: *E-Voting and Identify: 4th International Conference, Vote-ID 2013, Guildford, UK, July 17-19, 2013. Proceedings 4*. Springer. 2013, pp. 142–155.

[98] *Online votes make up two-thirds of Reform, less than a third of EKRE votes*. https://news.err.ee/1608906014/online-votes-make-up-two-thirds-of-reform-less-than-third-of-ekre-votes. Accessed on [Insert the date you accessed the link here].

[99] Pascal Paillier. 'Public-key cryptosystems based on composite degree residuosity classes'. In: *Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18*. Springer. 1999, pp. 223–238.

[100] Olivier Pereira. 'Individual verifiability and revoting in the Estonian internet voting system'. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2022, pp. 315–324.

[101] Jyoti Ramteke et al. 'Election result prediction using Twitter sentiment analysis'. In: *2016 international conference on inventive computation technologies (ICICT)*. Vol. 1. IEEE. 2016, pp. 1–5.

[102] James Reason. *Human error*. Cambridge university press, 1990.

[103] Ronald L Rivest and Warren D Smith. 'Three voting protocols: ThreeBallot, VAV, and Twin'. In: *USENIX/ACCURATE Electronic Voting Technology (EVT 2007)* (2007).

[104] Zuzana Rjaskova. *Electronic Voting Schemes*. 2002.

[105] Peter Roenne. 'JCJ with improved verifiability guarantees'. In: (2016).

[106] Peter Roenne. *Private Communication*. 2016.

[107] Peter B Rønne et al. 'Short Paper: Coercion-Resistant Voting in Linear Time via Fully Homomorphic Encryption: Towards a Quantum-Safe Scheme'. In: *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*. Springer. 2020, pp. 289–298.

[108] Peter YA Ryan, Peter B Rønne, and Vincenzo Iovino. 'Selene: Voting with transparent verifiability and coercion-mitigation'. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2016, pp. 176–192.

[109] Peter YA Ryan et al. 'The prêt à voter verifiable election system'. In: *IEEE transactions on information forensics and security* 4.4 (2009), pp. 662–673.

[110] Berry Schoenmakers. 'A simple publicly verifiable secret sharing scheme and its application to electronic voting'. In: *Annual International Cryptology Conference*. Springer. 1999, pp. 148–164.

[111] Carsten Schürmann. 'Rounding Considered Harmful'. In: *Electronic Voting: Third International Joint Conference, E-Vote-ID 2018, Bregenz, Austria, October 2-5, 2018, Proceedings 3*. Springer. 2018, pp. 189–202.

[112] Adi Shamir. 'How to Share a Secret'. In: *Commun. ACM* 22.11 (1979), 612–613. ISSN: 0001-0782. DOI: 10.1145/359168.359176. URL: https://doi-org.proxy.bnl.lu/10.1145/359168.359176.

[113] Adi Shamir and Nicko van Someren. 'Playing 'hide and seek'with stored keys'. In: *International conference on financial cryptography*. Springer. 1999, pp. 118–124.

[114] Fateme Shirazi et al. 'Robust electronic voting: Introducing robustness in civitas'. In: *2011 International Workshop on Requirements Engineering for Electronic Voting Systems*. IEEE. 2011, pp. 47–55.

[115] Victor Shoup. 'A proposal for an ISO standard for public key encryption'. In: *Cryptology ePrint Archive* (2001).

[116] Joao Miguel Barros da Silva Mendes. 'Trusted Civitas: Client trust in Civitas electronic voting protocol'. In: *Trabajo de Máster en Ingeniería Informática y de Computadores. Instituto Superior Técnico, Universidade Técnica de Lisboa* (2011).

[117] Mihkel Solvak. 'Does vote verification work: usage and impact of confidence building technology in internet voting'. In: *Electronic Voting: 5th International Joint Conference, E-Vote-ID 2020, Bregenz, Austria, October 6–9, 2020, Proceedings 5*. Springer. 2020, pp. 213–228.

[118] Oliver Spycher et al. 'A new approach towards coercion-resistant remote e-voting in linear time'. In: *Financial Cryptography and Data Security: 15th International Conference, FC 2011, Gros Islet, St. Lucia, February 28-March 4, 2011, Revised Selected Papers 15*. Springer. 2012, pp. 182–189.

[119] Anggrio Sutopo, Thomas Haines, and Peter Roenne. 'On the Auditability of the Estonian IVXV System'. In: ().

[120] Meng-Hsiu Tsai et al. 'A machine learning based strategy for election result prediction'. In: *2019 international conference on computational science and computational intelligence (CSCI)*. IEEE. 2019, pp. 1408–1410.

[121] Valimised.ee. *Election Results for the 2019 Riigikogu Election*. https://www.valimised.ee/sites/default/files/uploads/misc/RK2019_election_result_data.zip. 2019.

[122] Priit Vinkel and Robert Krimmer. 'The how and why to internet voting an attempt to explain E-Stonia'. In: *International joint conference on electronic voting*. Springer. 2016, pp. 178–191.

[123] Giuseppe Vitto. 'Factoring Primes to Factor Moduli: Backdooring and Distributed Generation of Semiprimes'. In: (2021).

[124] Jonathan N Wand et al. 'The butterfly did it: The aberrant vote for Buchanan in Palm Beach County, Florida'. In: *American Political Science Review* 95.4 (2001), pp. 793–810.

[125] Stefan Weber. 'A Coercion-Resistant Cryptographic Voting Protocol - Evaluation and Prototype Implementation'. de. MA thesis. Technische Universität Darmstadt, 2006. URL: http://tubiblio.ulb.tu-darmstadt.de/100719/.

[126] Sarah Wiseman, Paul Cairns, and Anna Cox. 'A taxonomy of number entry error'. In: *Proceedings of HCI 2011 The 25th BCS Conference on Human Computer Interaction 25*. 2011, pp. 187–196.

[127] Lirong Xia. 'Computing the Margin of Victory for Various Voting Rules'. In: *Proceedings of the 13th ACM Conference on Electronic Commerce*. EC '12. Valencia, Spain: Association for Computing Machinery, 2012, 982–999. ISBN: 9781450314152. DOI: 10.1145/2229012.2229086. URL: https://doi-org.proxy.bnl.lu/10.1145/2229012.2229086.

Part IV

Appendix

APPENDIX A: REMOTE DATA ANALYSIS AND COLLABORATION

In the final chapter of the thesis, we focus on the process of analyzing real data obtained from Estonian log files. These log files contain sensitive information, and due to privacy concerns and legal restrictions, it is prohibited to transfer them outside Estonia through email or any other file-sharing methods. Consequently, to effectively work with the data, the following steps were undertaken:

1. **Visit to Estonia:** To access the data securely and in compliance with the relevant regulations, I arranged a visit to the Estonian facility where the log files were stored. This trip facilitated a hands-on approach to working with the data and provided an opportunity to gain insights into the specific requirements of the Estonian context.

2. **In-person data analysis:** During my visit, I conducted a preliminary analysis of the log files to understand the nature of the data and identify potential patterns or trends. This exploration also allowed me to establish the necessary groundwork for developing the main structure of the scripts required for further analysis.

3. **Script development:** Based on the initial findings from the in-person data analysis, I developed a series of scripts tailored to the specific characteristics of the Estonian log files. These scripts were designed to automate the processing and analysis of the data, enabling more efficient and accurate results.

4. **Remote collaboration with Prof. Willemson:** After returning from Estonia, I shared the developed scripts with Prof. Willemson, who then ran the codes on the Estonian log files within the confines of the secure facility. This approach ensured that the sensitive data remained within the legally permitted environment while still enabling me to obtain the necessary output for my thesis.

5. **Output analysis:** Prof. Willemson provided me with the output generated by the scripts, which I then used to conduct further analysis and draw conclusions relevant to the research objectives. This process involved a thorough examination of the results, as well as a comparison with previous research and established methodologies.

Throughout this process, several concerns and challenges were addressed:

- **Data privacy and legal compliance:** By working directly with the data in Estonia and collaborating with Prof. Willemson, we ensured that sensitive information remained within the legally prescribed environment, thus adhering to data privacy regulations.

- **Remote collaboration challenges:** The nature of the project necessitated remote collaboration, which posed challenges in terms of communication, data synchronization, and output sharing. To overcome these challenges, we established a clear communication protocol and utilized secure, approved channels for sharing code and output.
- **Code adaptability and scalability:** The scripts developed for this project were designed to be adaptable to the unique characteristics of the Estonian log files, as well as scalable to accommodate potential increases in data volume or complexity in the future.

By addressing these concerns and implementing the outlined process, we were able to effectively analyze the Estonian log files and generate valuable insights for the thesis while adhering to data privacy and legal requirements.

APPENDIX B: MODEL'S PERFORMANCE REPORTS

The appendix section of this thesis includes several tables that provide detailed information about the performance of different classifiers used in the research. These tables are essential for understanding the model performance and the parameters used for each classifier. Each table is dedicated to a specific model and shows the performance metrics, such as accuracy, recall, and F1 score. The tables also indicate how the imbalanced dataset issue was addressed, the feature selection method used, and the extraction method applied to the dataset. These tables serve as a comprehensive reference for readers who wish to gain a deeper understanding of the classifiers used in the research.

For those interested in acquiring exhaustive details about all models, these can be found in an accompanying Google Doc file[1].

[1] Google Doc Spreadsheet

TABLE 9.1: Top Classifier For Margin 10

Classifier	Parameters	Sampling	Feature Selection	Feature Extraction	Score
Random Forest	{'criterion': 'entropy', 'max_depth': 3, 'n_estimators': 100}	UnderSampling	ChiSquare	PCA	0.836
Random Forest	{'criterion': 'gini', 'max_depth': 9, 'n_estimators': 100}	OverSampling	ChiSquare	PCA	0.830
Random Forest	{'criterion': 'gini', 'max_depth': 5, 'n_estimators': 100}	UnderSampling	MutualInformationGain	PCA	0.827
Decision Tree	{'criterion': 'gini', 'max_depth': 9, 'splitter': 'best'}	OverSampling	ChiSquare	ICA	0.817
Random Forest	{'criterion': 'entropy', 'max_depth': 5, 'n_estimators': 100}	OverSampling	MutualInformationGain	PCA	0.826
Random Forest	{'criterion': 'gini', 'max_depth': 9, 'n_estimators': 100}	UnderSampling	MutualInformationGain	PCA	0.821
Decision Tree	{'criterion': 'entropy', 'max_depth': 9, 'splitter': 'best'}	UnderSampling	ChiSquare	PCA	0.819
Decision Tree	{'criterion': 'gini', 'max_depth': 9, 'splitter': 'best'}	OverSampling	ChiSquare	ICA	0.817
Random Forest	{'criterion': 'entropy', 'max_depth': 3, 'n_estimators': 50}	UnderSampling	ChiSquare	PCA	0.816
Random Forest	{'criterion': 'gini', 'max_depth': 9, 'n_estimators': 100}	OverSampling	ChiSquare	PCA	0.806
Decision Tree	{'criterion': 'entropy', 'max_depth': 9, 'splitter': 'best'}	UnderSampling	ChiSquare	PCA	0.803
Decision Tree	{'criterion': 'gini', 'max_depth': 9, 'splitter': 'best'}	None	ChiSquare	ICA	0.793
Random Forest	{'criterion': 'gini', 'max_depth': 9, 'n_estimators': 50}	OverSampling	ChiSquare	PCA	0.789
Random Forest	{'criterion': 'entropy', 'max_depth': 9, 'n_estimators': 100}	UnderSampling	MutualInformationGain	PCA	0.786
Random Forest	{'criterion': 'gini', 'max_depth': 3, 'n_estimators': 100}	UnderSampling	ChiSquare	PCA	0.782
Naive Bayes	{'var_smoothing': 1e-09}	OverSampling	MutualInformationGain	ICA	0.777
Decision Tree	{'criterion': 'gini', 'max_depth': 7, 'splitter': 'best'}	OverSampling	ChiSquare	PCA	0.776
Logistic Regression	{'C': 0.001, 'penalty': 'l2', 'solver': 'newton-cg'}	UnderSampling	ChiSquare	PCA	0.775
Random Forest	{'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50}	UnderSampling	Pearson	ICA	0.775
Decision Tree	{'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'}	OverSampling	Pearson	ICA	0.775
Random Forest	{'criterion': 'gini', 'max_depth': 9, 'n_estimators': 50}	UnderSampling	Pearson	PCA	0.774
Logistic Regression	{'C': 0.001, 'penalty': 'l2', 'solver': 'newton-cg'}	OverSampling	ChiSquare	PCA	0.774
Naive Bayes	{'var_smoothing': 1e-09}	OverSampling	Pearson	ICA	0.773
Decision Tree	{'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'}	UnderSampling	ChiSquare	ICA	0.771
Logistic Regression	{'C': 0.01, 'penalty': 'l2', 'solver': 'newton-cg'}	UnderSampling	ChiSquare	ICA	0.771

TABLE 9.2: Top Classifier For Margin 20

Classifier	Parameters	Sampling	Feature Selection	Feature Extraction	Score
Decision Tree	{'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'}	UnderSampling	ChiSquare	ICA	0.789
Random Forest	{'criterion': 'gini', 'max_depth': 9, 'n_estimators': 100}	UnderSampling	ChiSquare	PCA	0.725
Random Forest	{'criterion': 'entropy', 'max_depth': 3, 'n_estimators': 50}	UnderSampling	ChiSquare	PCA	0.716
Random Forest	{'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50}	OverSampling	ChiSquare	ICA	0.705
Decision Tree	{'criterion': 'gini', 'max_depth': 9, 'splitter': 'best'}	OverSampling	ChiSquare	PCA	0.705
Decision Tree	{'criterion': 'gini', 'max_depth': 9, 'splitter': 'best'}	UnderSampling	MutualInformationGain	PCA	0.696
Random Forest	{'criterion': 'gini', 'max_depth': 9, 'n_estimators': 100}	OverSampling	ChiSquare	PCA	0.694
Naive Bayes	{'var_smoothing': 1e-09}	UnderSampling	ChiSquare	ICA	0.678
SimpleClassifier+AND	nan	nan	nan	nan	0.670
Random Forest	{'criterion': 'gini', 'max_depth': 5, 'n_estimators': 100}	OverSampling	ChiSquare	PCA	0.670
Random Forest	{'criterion': 'entropy', 'max_depth': 3, 'n_estimators': 100}	UnderSampling	ChiSquare	PCA	0.663
Decision Tree	{'criterion': 'entropy', 'max_depth': 9, 'splitter': 'best'}	UnderSampling	ChiSquare	PCA	0.653
Random Forest	{'criterion': 'gini', 'max_depth': 9, 'n_estimators': 100}	OverSampling	ChiSquare	PCA	0.651
Random Forest	{'criterion': 'gini', 'max_depth': 9, 'n_estimators': 100}	OverSampling	MutualInformationGain	PCA	0.633
Random Forest	{'criterion': 'entropy', 'max_depth': 9, 'n_estimators': 50}	UnderSampling	MutualInformationGain	PCA	0.632
Logistic Regression	{'C': 0.01, 'penalty': 'l2', 'solver': 'newton-cg'}	UnderSampling	MutualInformationGain	ICA	0.615
Random Forest	{'criterion': 'entropy', 'max_depth': 7, 'n_estimators': 100}	UnderSampling	MutualInformationGain	PCA	0.606
Naive Bayes	{'var_smoothing': 1e-09}	OverSampling	Pearson	PCA	0.605
Logistic Regression	{'C': 0.001, 'penalty': 'l2', 'solver': 'newton-cg'}	UnderSampling	MutualInformationGain	PCA	0.604
Logistic Regression	{'C': 0.001, 'penalty': 'l2', 'solver': 'newton-cg'}	UnderSampling	MutualInformationGain	PCA	0.603
Random Forest	{'criterion': 'gini', 'max_depth': 9, 'n_estimators': 100}	UnderSampling	MutualInformationGain	PCA	0.603
Decision Tree	{'criterion': 'gini', 'max_depth': 9, 'splitter': 'best'}	OverSampling	ChiSquare	PCA	0.603
Naive Bayes	{'var_smoothing': 1e-09}	OverSampling	Pearson	ICA	0.601
Decision Tree	{'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'}	OverSampling	Pearson	ICA	0.601
Logistic Regression	{'C': 0.001, 'penalty': 'l2', 'solver': 'newton-cg'}	UnderSampling	MutualInformationGain	PCA	0.600

TABLE 9.3: Top Classifier For Margin 50

Classifier	Parameters	Sampling	Feature Selection	Feature Extraction	Score
Decision Tree	'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'	UnderSampling	ChiSquare	ICA	0.560
Random Forest	'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50	UnderSampling	ChiSquare	ICA	0.552
Random Forest	'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50	OverSampling	ChiSquare	PCA	0.506
Random Forest	'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50	UnderSampling	ChiSquare	PCA	0.493
Random Forest	'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50	OverSampling	ChiSquare	PCA	0.479
Random Forest	'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50	UnderSampling	ChiSquare	PCA	0.458
SimpleClassifier+OR	nan	nan	nan	nan	0.448
Random Forest	'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50	UnderSampling	ChiSquare	PCA	0.441
Random Forest	'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50	OverSampling	ChiSquare	PCA	0.426
Random Forest	'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50	OverSampling	MutualInformationGain	PCA	0.407
Random Forest	'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50	OverSampling	ChiSquare	PCA	0.391
Decision Tree	'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'	OverSampling	ChiSquare	PCA	0.355
Decision Tree	'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'	UnderSampling	ChiSquare	PCA	0.354
Random Forest	'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50	OverSampling	MutualInformationGain	PCA	0.340
Random Forest	'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50	OverSampling	MutualInformationGain	PCA	0.323
Random Forest	'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50	OverSampling	MutualInformationGain	PCA	0.314
Random Forest	'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50	UnderSampling	MutualInformationGain	PCA	0.310
Random Forest	'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50	UnderSampling	MutualInformationGain	PCA	0.304
Decision Tree	'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'	OverSampling	ChiSquare	ICA	0.296
Decision Tree	'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'	OverSampling	ChiSquare	PCA	0.293
Random Forest	'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50	UnderSampling	ChiSquare	ICA	0.293
Naive Bayes	'var_smoothing': 1e-09	OverSampling	ChiSquare	ICA	0.286
Decision Tree	'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'	UnderSampling	ChiSquare	PCA	0.281
Logistic Regression	'C': 0.001, 'penalty': 'l2', 'solver': 'newton-cg'	OverSampling	Pearson	ICA	0.279

TABLE 9.4: Top Classifier For Margin 100

Classifier	Parameters	Sampling	Feature Selection	Feature Extraction	Score
Random Forest	{'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50}	UnderSampling	ChiSquare	PCA	0.259
Random Forest	{'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50}	OverSampling	ChiSquare	PCA	0.255
Random Forest	{'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50}	UnderSampling	ChiSquare	PCA	0.254
Random Forest	{'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50}	OverSampling	MutualInformationGain	PCA	0.194
Random Forest	{'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50}	OverSampling	ChiSquare	PCA	0.190
Decision Tree	{'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'}	UnderSampling	ChiSquare	PCA	0.170
Random Forest	{'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50}	OverSampling	ChiSquare	PCA	0.164
Random Forest	{'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50}	UnderSampling	ChiSquare	PCA	0.157
SimpleClassifier+XOR	nan	nan	nan	nan	0.135
Random Forest	{'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50}	UnderSampling	ChiSquare	PCA	0.129
Decision Tree	{'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'}	OverSampling	ChiSquare	PCA	0.123
Random Forest	{'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50}	OverSampling	MutualInformationGain	PCA	0.116
Decision Tree	{'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'}	UnderSampling	MutualInformationGain	PCA	0.089
Decision Tree	{'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'}	UnderSampling	MutualInformationGain	PCA	0.089
Decision Tree	{'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'}	UnderSampling	ChiSquare	PCA	0.088
Random Forest	{'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50}	UnderSampling	MutualInformationGain	PCA	0.086
Random Forest	{'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50}	OverSampling	ChiSquare	PCA	0.086
Decision Tree	{'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'}	OverSampling	ChiSquare	PCA	0.084
Naive Bayes	{'var_smoothing': 1e-09}	UnderSampling	Pearson	PCA	0.082
Logistic Regression	{'C': 0.001, 'penalty': 'l2', 'solver': 'newton-cg'}	UnderSampling	Pearson	ICA	0.078
Random Forest	{'criterion': 'gini', 'max_depth': 3, 'n_estimators': 50}	UnderSampling	Pearson	ICA	0.078
Decision Tree	{'criterion': 'gini', 'max_depth': 3, 'splitter': 'best'}	UnderSampling	Pearson	ICA	0.078
Logistic Regression	{'C': 0.001, 'penalty': 'l2', 'solver': 'newton-cg'}	OverSampling	ChiSquare	PCA	0.077