University of Zurich[UZH]

MASTER BASIS MODULE – Communication Systems Group, Prof. Dr. Burkhard Stiller

Privacy, Verifiability and Auditability in Blockchain-based e-Voting

Christian Killer
Zurich, Switzerland
Student ID: 12-739-118

Supervisor: Dr. Thomas Bocek, Bruno Rodrigues, Daniel Gasteiger
Date of Submission: September 12th, 2017

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Master Basis Module
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

Digitalization of electoral processes counts on confident systems that produce enough verifiable evidence. Thus, the design and implementation of voting systems has been widely studied in prior research, bringing together expertise in many fields. Even though there is a large amount of theoretical research on complex and secure secret-ballot schemes, practically implementing those schemes in nation-wide electoral processes proves to be very difficult. This work aims to provide a brief and non-conclusive overview over the complex cryptographic concepts that are necessary to implement a practical e-Voting scheme, sufficing well-defined properties of privacy and verifiability.

Keywords — privacy, end-to-end verifiability, voting systems, secret-ballot voting

Contents

Chapter 1

Introduction

The design and implementation of voting systems has been widely studied in prior research. An optimal voting scheme assures privacy, verifiability and auditability. Conceivably one of the most important property of elections is the secrecy of the ballot, preventing voter coercion and vote selling. Even though there is a large amount of theoretical research on cryptographic schemes, practically implementing those schemes in nation-wide electoral processes proves to be very difficult. Especially since some of those protocols presume trusted election authorities in their assumptions. Nevertheless, voting schemes are very actively researched field, hopefully converging complex cryptographic voting schemes closer to practical implementations. This work aims to provide a brief and non-conclusive overview over the different cryptographic concepts that are necessary to implement a practical e-Voting scheme.

1.1 Motivation and Description of Work

The goal of this work is to get an overview of current possibilities and the feasibility to implement a secret-ballot voting scheme to achieve privacy and verifiability. In order to do that, the most crucial properties and building blocks are evaluated and summarized, followed by a short summary of the existing cryptographic schemes. The following work will not evaluate cryptographic schemes in detail but rather outline feasible encryption schemes and analyse the general practicality, especially evaluated in connection with an blockchain-based approach.

1.2 Outline

In chapter 2, related work is denoted. Then, in chapter 3, the necessary properties for secret-ballot voting schemes are evaluated and described. Subsequently in chapter 4, the building blocks for voting schemes are outlined. Chapter 5 shortly evaluates the existing secret-ballot voting systems. Finally, this work concludes in Chapter 6.

Chapter 2

Related Work

The amount of research and literature on secret-ballot voting schemes and electronic voting is rapidly growing. In order to get an understanding of the various voting and encryption schemes, it is crucial to use surveys as a starting point [37] [12] [16] [28].

Prior work focusing on verifiability minted the term end-to-end verifiability [13]. Further research focused on user-friendly verifiability mechanisms, such as the Markpledge system [55], continuous auditing [65], and cast-or-audit [10] [11].

Ever since Chaum's paper in 1981 on anonymizing mixes[19], more and better cryptographic voting schemes and techniques emerge. These techniques include homomorphic encryption [1] [27], blind signatures [17], commitment schemes [50] [52] or designated verifier proofs [5] along with others.

The development of secret-ballot voting schemes started with Fujioka, Okamoto and Ohta [30] (FOO92). Benaloh and Tuinstra refined the notion of voter-coercion, ballot-secrecy and receipt-freeness in [14] (BT94).

Around the same time, Niemi and Renvall identified the threat of voter-coercion as well [56] (NR94). Sako and Killian noted the reliance on strong physical assumptions (voting booths) and tried to use a mixnet-based approach [66] (SK95). The landmark work of Cramer, Gennaro and Schoemakers [23] (CGS97) optimized a multi-authority setting, reducing the concentration of trust.

Okamoto observed progress in research and proposed adaptions of FOO92 in [58] and [58]. Finally, Hirt and Sako critically evaluated prior work in [35] (HS00) by pointing out the existence of receipts in BT94 and the immense computational processing load of SK95. Besides, HS00 introduce a generic construction to provide receipt-freeness and illustrate it on the scheme of CGS97.

Similar to HS00, Lee and Kim also introduce receipt-freeness on CGS97 in [44] (LK00). Baudron et al. focus on scalability by using several layers of vote aggregation in [7] (BFP01). Magkos, Burmester and Chrissikopolous [46] (MBC01) propose tamper-free smart cards to transform CGS97 into a receipt-free voting system, using the same basic approach as LK00.

In subsequent works, Lee et al. proposing new approaches to improve on LK00 in [45] (LK02), [43] (LBD03) and in [3] (ALBD04). Introducing the notion of coercion-resistance, Juels, Catalano and Jakobsson refine privacy concerns in [40] (JCJ05). Trying out new ways of ensuring everlasting, unconditional privacy, Moran and Noar used commitment schemes in [50] (MN06) and [52] (MN07).

Chapter 3

Properties of secret-ballot voting protocols

One of the most far-reaching applications of cryptographic schemes are secret-ballot voting protocols. There are basically three types of different categories of cryptographic approaches. One approach is to use mix-nets, another one implements homomorphic encryption. The third strategy is to leverage blind signatures. As usual, the suitability of those three schemes have advantages and disadvantages and they depend on the conditions under which they are applied [16].

Research in voting brings together knowledge in countless fields to address critical real-world problems. However, the adoption of electronic voting in real-world elections is still hindered by the complexity of the underlying cryptographic processes [37]. This chapter provides an overview of the desirable properties of a secure electronic voting system.

Designing voting protocols is challenging, especially since there are often conflicting properties [16]. Additionally, a huge amount of properties have been proposed so far, to further refine the notion of an optimal secret-ballot voting protocol. The expectations towards electronic voting schemes are exceedingly high, especially compared to the expectations towards classic paper-ballot voting schemes [21]. Secret-ballot voting schemes aim to achieve as many properties as possible, making as few assumptions as possible [21]. Still, the question remains: is it practically achievable?

The following list is non-conclusive and serves as an overview of the challenges related to electronic voting.

3.1 Privacy

Privacy of the vote is widely recognized as a fundamental human right and is persisted in Article 21 of the Universal Declaration of Human Rights [70]. Prior work has progressively defined the following three key properties of Privacy: *Ballot-secrecy*, *Receipt-freeness* and *Coercion-resistance* [4].

Ballot Secrecy or Anonymity guarantees that the vote cannot be traced back to the voter. In certain voting schemes, anonymity is provided using blind signatures. Other schemes rely on a trusted verifier achieving *conditional* anonymity. This conditional anonymity allows the trusted third party to assign a unique ballot token. However, if the ballot token is generated randomly by the voter, there is a probability that two voters could generate a collision by choosing the same random token. Traditionally anonymity implies that anonymous parties can not reveal their identities even if they choose to do so. However, in electronic voting, the voter may be able to provide his or her identity and thus loose his or her anonymity[16].

Receipt-freeness is another crucial property. Receipt-freeness should reduce coercion, hence the user should not be able persist any information about the vote. Thus, one cannot prove the content of the vote to anyone. Even though newer schemes reintroduce the usage of receipts, they assure that the ballot content is cryptographically masked [18]. The theoretical concept of receipt-freeness was first introduced by Benaloh and Tuinstra, proposing single and multiauthority voting schemes [14]. Unfortunately, their scheme was proven not to be receipt-free by Hirt and Sako [35], who at the same time proposed new receipt-free voting based on homomorphic encryptio

According to [59] receipt-freeness can be defined as follows:

> A voting system is receipt-free, if there exists a voter, V_i, such that, for any adversary C, V_i can cast v_i ($v_i \neq v_i^*$) which is accepted by T[1] , under the condition that $View_C(X : V_i)$ is accepted by C.

Coercion-Resistance is the third property and should enable the voter to cast a vote for his or her intended choice even while appearing to cooperate with a coercer.

Each property effectively contains the previous one. Coercion-resistance entails receipt-freeness which ultimately implies ballot-secrecy and anonymity [4].

Going further, cryptographic voting schemes should aim for everlasting privacy. Everlasting privacy promises that even a computationally unbounded party does not gain any information about individual votes (other than what can be inferred from the final tally) [51]. Hence, everlasting privacy does explicitly not depend on assumptions of cryptographic hardness.

If voters encrypt their votes, the holders of decryption keys can view these encrypted votes. This basic premise places unwanted trust in the holder of decryption keys. An alternative to this classic solution are commitment schemes, further described in 4.6.

3.2 Verifiability

Cryptography could provide verifiability in order to reduce or eliminate trust in applications, voting machines or staff. Generally, Verifiability can be divided into universal

[1] In [59], the voting commission consists of a single administrator, A, and a single timeliness commission member, T.

and individual verifiability. After casting a vote, the user might ask for proof that his or her vote was counted in the set of all votes. Finally, after the final tallies are released, an observer can verify that the tally has been correctly computed from the set of all votes [4].

Thus, End-to-End-Verifiability (E2E-V) is a crucial security property. Voters may check significant ingredients of the election results themselves instead of placing trust in other parties (*e.g.* news media, political parties) involved. E2E-V systems conclude to three core steps: Cast-as-Intended, Recorded-as-Cast and Tallied-as-Recorded [13].

In order to preserve voter privacy, many systems encrypt the ballots. In combination with cryptographic proofs, the encryption is proven to contain a valid vote [18]. These proofs can take various forms, *e.g.* zero-knowledge proofs, designated-verifier proofs or visual cryptography proofs [18]). However, these proofs are not very user-friendly. To address this issue, literature proposed three approaches: verification by means of short strings [55], continuous auditing [65], and cast-or-audit ([10][11] Benaloh challenges) which are further outlined in section 4.1

3.3 Auditability

Post-election audits can provide assurance of a correct outcome by examining an audit trail of tamper-evident and voter-verifiable records, normally consisting of paper ballots.

Prior work defined two types of Risk Limiting Audits (RLA): ballot polling and comparison [13]. Both types examine random samples of ballots until there is strong statistical evidence that the outcome is correct or until the manual tally is complete.

The purpose and art in designing RLAs is defined by maintaining the risk limit, while performing less work than a full hand count, when the outcome is correct [13]. In section 4.1, different auditing and verification methods are discussed.

3.4 Other properties

Integrity presumes that only eligible voters can cast their votes only once and can not sell their votes. Also, ignoring votes should not be possible.

Eligibility verifiability assures verifiability that each vote in the set of all cast votes was indeed cast by an eligible voter [4].

Accountability of a voter should be possible without compromising the secrecy of the voters ballot. In case vote verification fails at some stage, possibly due to error or fraud, the voter should be able to conclusively prove that it failed to the relevant authorities, without compromising the secrecy of her ballot [4].

Robustness of the system should provide resistance against a certain degree of malfunction or corruption and still deliver correct results. A small number of misbehaving voters or system failures should not disrupt the election[4].

Usability of the system ensures that voters can easily and efficiently cast their votes through a convenient and accessible interface, providing equal opportunity for access and participation [4].

3.5 Conflicts and challenges

Several of the introduced properties conflict with each other. For instance, Verifiability conflicts with privacy due to Receipt-freeness In conclusion, if a voter could prove the vote to a third party, vote selling or coercion would not be mitigated [37].

Prior work even claims that absolute privacy without any leaking information can simply not coexist with verifiability [21].

Additionally, there is a conflict between Usability and Verifiability. When voters need to verify their vote, additional steps are required and thus impact usability negatively.

The large number of diverging requirements in privacy and verifiability alone, evidently lead to the prioritization of certain requirements. However, the goal should be to design a system that concurrently enables as much privacy and verifiability as possible [21].

Chapter 4

Building blocks of secret-ballot voting protocols

In order to satisfy the desirable properties of a secret-ballot voting scheme, many cryptographic primitives are necessary. The following section describes the majority of the necessary building blocks to ensure privacy, verifiability and auditability.

4.1 Auditing and verification methods

Many secret-ballot voting systems encrypt ballots to obfuscate the content. Hence, there should be mechanisms in place to verify the content of the encryption [18]. Achieving such proof is possible in a variety of ways, *e.g.* zero-knowledge proofs (Section 4.4) or designated verifier proofs (Section 4.5)

In order to preserve voter privacy, many systems encrypt the ballots. In combination with cryptographic proofs, the encryption is proven to contain a valid vote [18]. These proofs can take various forms, *e.g.* zero-knowledge proofs (Section 4.4), designated-verifier proofs (Section 4.5) or visual cryptography proofs [18]. However, these proofs are not very user-friendly. To address this issue, literature proposed three approaches: verification by means of short strings [55], continuous auditing [65], and cast-or-audit (Benaloh challenges [10][11]) which are further outlined in section 4.1.

4.1.1 Verification using short strings

The so-called Markpledge system uses short strings to enable voters to verify that the publicly posted ballot equals the one they saw in the voting booth [55]. In short, after voting for candidate A, the voting system prints a commitment to the encryption of the chosen candidate A and also prints a cryptographic proof that the receipt contains exactly the vote for A. The challenge string is short and easy to remember. After that, the voter can verify the cryptographic details in collaboration with a helper organisation.

4.1.2 Continuous auditing

Another suggestion are continuous audits of the ballot encryptions in the Prét à Voter system [65]. To give some context, in Prét à Voter, authorities prepare ballots in which the candidate order is encrypted and randomised [37]. To rest assured that the claimed order, is indeed the order that is encrypted, voters can submit a blank ballot and have it decrypted to test the system. Repeating this testing, the voter can improve confidence in the system and once ready, mark a choice and cast the vote.

4.1.3 Cast-and-audit

The cast-and-audit approach was proposed by Benaloh [10][11]. After marking his or her choice, the system prepares the ballot. Then, the voter is prompted to either cast or decrypt, thus audit, the ballot. The system can not predict whether the voter is going to audit or cast the encrypted ballot. Therefore, any cheating by the voting system would yield a 50% chance of being audited or detected, respectively. Auditing can be repeated as many times as desired and increases confidence in the correctness of the system.

4.2 Homomorphic Encryption

Encryption is an essential mechanism to preserve confidentiality of sensitive data and information. Yet, ordinary encryption schemes cannot operate on encrypted data without first decrypting it.

The term homomorphic encryption (HE) can be defined as follows

> *Homomorphic encryption is a form of encryption which allows specific types of computations to be carried out on ciphertexts and generate an encrypted result which, when decrypted, matches the result of operations performed on the plaintexts* [72].

In mathematical terms, the most common definition for homomorphic encryption is the following [27]:

> Let M) denote the set of the plaintexts (resp., ciphertexts). An encryption scheme is said to be homomorphic if for any given encryption key k the encryption function E satisfies.
>
> $$\forall m_1, m_2 \in M, \quad E(m_1 \odot_M m_2) \leftarrow E(m_1) \odot_C E(m_2)$$
>
> for some operators $\odot_M$ in M and $\odot_C$ in C, where $\leftarrow$ means "can be directly computed from," that is, without any intermediate decryption. If $(M, \odot_M)$ and $(C, \odot_C)$ are groups, we have a *group homomorphism*. We say a scheme is *additively homomorphic* if we consider addition operators, and *multiplicatively homomorphic* if we consider multiplication operators.

The adoption of cloud services and huge data-centers led to issues regarding the business data and privacy of said data. A practical HE scheme could solve those issues by allowing the efficient storage and operation on encrypted data.

By now, there are three categories of different HE schemes, which differentiate the number of possible and allowed operations on the encrypted data. The first category are Partially Homomorphic Encryption (PHE) schemes allowing only one type of operation for an unlimited number of times. The second category are Somewhat Homomorphic Encryption (SWHE) allowing some operations for a restricted number of times. Third, Fully Homomorphic Encryption (FHE) allows the unlimited number of operations for an unlimited number of times [72].

Historically, RSA [64] was the basis for HE by formulating the first public-key encryption scheme with a homomorphic property. In order to achieve semantic security, RSA had to add padding to messages with random bits before the encryption. Nevertheless, it led to attempts to design a homomorphic scheme with a large set of possible operations on encrypted data.

Ever since 1978, a lot of improvements have been made in the field of homomorphic encryption. In contrast to RSA, even threshold variants of ElGamal and the Paillier cryptosystems prove to be *indistinguishable under chosen-plaintext attack* (IND-CPA), thus offering the best security level that is possible for HE schemes [29].

The aforementioned HE schemes that allow simple computations, such as adding or multiplaying ciphertext, have been discovered for a long time. Allowing both operations at the same time has not been possible until the breakthrough discovery be Gentry [31], proposing the first fully homomorphic encryption scheme. However, the performance is not yet sufficient for practical implementations. Nevertheless, multiple applications have been proposed, even though most of them are only of theoretical nature [53]. An active community of researchers and constant improvements of FHE might lead to the development of a more performant scheme in the future.

4.2.1 Paillier Threshold Encryption Toolbox

This section focuses on the popular Paillier encryption scheme invented in 1999[60]. The probabilistic public-key algorithm is named after its inventor Pascal Paillier. Especially because of its additive property, this scheme is suitable for electronic voting. Also, the encryption scheme has been actively improved in the years after its inception [24][25]. Besides that, open source code contributions can be found as well [71].

A popular technique to construct secret-ballot voting schemes is based on homomorphic threshold encryption. Authorities publish a public key and voters send in encrypted votes. Using digital signatures or other means of authentication assure the eligibility of the voters. With threshold schemes, the authorities have to cooperate and are then able to jointly decrypt the final tally [33].

The following example outlines a minimum working example using the Paillier Threshold Encryption Toolbox [71]. The test simply provides an overview of the possibilities of the Paillier cryptosystem.

```
import org.junit.Assert;
import org.junit.Test;
import paillierp.PaillierThreshold;
import paillierp.key.KeyGen;
import paillierp.key.PaillierPrivateThresholdKey;
import paillierp.zkp.DecryptionZKP;

import java.math.BigInteger;
import java.security.SecureRandom;
import java.util.concurrent.ExecutionException;
import java.util.logging.Logger;

public class ThresholdKeyTest {

    // w is the threshold minimum number of decryption servers
    // needed to decrypt a message; w must be less than or equal to half of l
    public static final int THRESHOLD = 3;

    // l is the total number of decryption servers
    public static final int DECRYPTION_SERVERS = 5;
    public static final int KEY_SIZE = 128;

    // number of additive iterations, testing the additive property
    public static final int ADDITIVE_ITERATIONS = 10;

    SecureRandom secureRandom = new SecureRandom();

    // Creates three threshold encryption schemes, needed for the threshold variant
    PaillierThreshold paillierThreshold = new PaillierThreshold();
    PaillierThreshold paillierThresholdTwo = new PaillierThreshold();
    PaillierThreshold paillierThresholdThree = new PaillierThreshold();

    private static Logger logger = Logger.getLogger(ThresholdKeyTest.class.toString());

    @Test
    public void testThresholdKeyEncryptionDecryption() throws ExecutionException, InterruptedException {

        // Generate an array of Paillier Private keys, using a small key size of only 128-bit key size.
        // In practice, according to https://www.keylength.com/en/8/, the keysize should be at least 2048-bits
        PaillierPrivateThresholdKey[] paillierPrivateThresholdKeys =
                KeyGen.PaillierThresholdKey(KEY_SIZE,DECRYPTION_SERVERS,THRESHOLD,secureRandom.nextLong());

        // Assert that all keys can effectively encrypt
        for(int i=0; i<DECRYPTION_SERVERS;i++){
            Assert.assertTrue(paillierPrivateThresholdKeys[i].canEncrypt());
        }

        // Assign the generated keys to the paillier threshold schemes instantiated above
        paillierThreshold.setDecryptEncrypt(paillierPrivateThresholdKeys[0]);
        paillierThresholdTwo.setDecryptEncrypt(paillierPrivateThresholdKeys[1]);
        paillierThresholdThree.setDecryptEncrypt(paillierPrivateThresholdKeys[2]);

        // encrypt a zero for starters
        BigInteger ciphertext = paillierThreshold.encrypt(BigInteger.ZERO);

        // Add BigInteger.ONE iteratively to the existing ciphertext
        for (int i = 0; i < ADDITIVE_ITERATIONS; i++) {
            ciphertext = paillierThreshold.add(ciphertext, paillierThreshold.encrypt(BigInteger.ONE));
        }

        // Create a ZKP in order to decrypt the ciphertext
        DecryptionZKP decryptionZKP = new DecryptionZKP(paillierThreshold.getPrivateKey(),ciphertext);
        DecryptionZKP decryptionZKPTwo = new DecryptionZKP(paillierThresholdTwo.getPrivateKey(),ciphertext);
        DecryptionZKP decryptionZKPThree = new DecryptionZKP(paillierThresholdThree.getPrivateKey(),ciphertext);
```

```
        // Assert that all ZKPs are valid and working
        Assert.assertTrue(decryptionZKP.verify());
        Assert.assertTrue(decryptionZKP.verify(ciphertext));
        Assert.assertTrue(decryptionZKPTwo.verify());
        Assert.assertTrue(decryptionZKPTwo.verify(ciphertext));
        Assert.assertTrue(decryptionZKPThree.verify());
        Assert.assertTrue(decryptionZKPThree.verify(ciphertext));

        // Decrypt the sum by using all available ZKPs
        BigInteger decrypted = paillierThreshold.combineShares(decryptionZKP,decryptionZKPTwo,decryptionZKPThree);

        // Final assertion of sucessful decryption
        Assert.assertTrue(decrypted.intValue() == ADDITIVE_ITERATIONS);
    }
}
```

4.2.2 Paillier Encryption Performance and Cost

Recent benchmarks evaluated by a private company called snips [69] yield a rather optimistic result for different programming languages implementing the Paillier Encryption scheme. Unfortunately, Java performs worse than any other language.

Still, scalability might be impractical for large scale elections using the Paillier encryption scheme. Salamonsen tried to adapt the Helios[2] voting protocol according to the Norwegian county elections, which is one of the easiest election variants in Norway. The effort to compute ciphertexts and the necessary zero knowledge proofs resulted in approximately 2 to 5 hours of work for the voter [67].Nevertheless, the Paillier scheme is steadily improved upon [39], yielding continuing better results.

4.3 Blind Signatures

Blind signatures were introduced in 1983 by David Chaum [17]. In the context of electronic voting, blind signatures enable verifiers to digitally sign a secret held by a voter without knowing the secret. This construct is explicitly useful for verifiers to validate a ballot or token without knowing the exact content [16].

An example usage for blind signatures could be designed as follows: after generating a random ballot token, a voter sends a proof of identity and a request for a blind signature for the ballot token to a verifier. After successful verification of the proof of identity, the verifier blind-signs the request and returns it to the voter. To be applicable in practice, a blind signature must be unforgeable [16].

4.4 Zero-Knowledge Proofs

A zero-knowledge proof (ZKP) is a proof of a statement s which reveals nothing about statement s, except that it is true [32]. With the use of a ZKP system, a prover can

convince a verifier V via an interaction that some statement is true. ZKPs have found numerous applications in practice and are a very useful construct.

According to Blum et al. [15], three main *ingredients* are necessary to differentiate a zero-knowledge system from a traditional proof

- **Interaction**: The prover and the verifier talk back and forth.
- **Hidden Randomization**: The verifier tosses coins that are hidden from the prover and thus unpredictable to him
- **Computational Difficulty**: The prover embeds in his proofs the computational difficulty of some other problem.

Furthermore, three main *features* of ZKP system include completeness, soundness, and zero knowledge.

- **Completeness** If the statement is correct, then the verifier will always accept.
- **Soundness**: . If the statement is incorrect, then the verifier will always reject.
- **Zero-knowledge**: No (malicious) verifier can get any extra information from the proof procedure, except the correctness of the statement.

One possible ZKP system are Σ-protocols or so called three-move protocols. A prover wants to prove statement x and he knows a witness w. The prover sends an initial message a, receives a random challenge e and responds with the answer z. Consequently, the verifier w is now in possession of (x, a, e, z) and can choose to accept or not.

Nevertheless, it is important to minimize the required interaction between the voters and talliers when casting votes. Normally, this is achieved by finding an applicable non-interactive ZKP (NI-ZKP) using the Fiat-Shamir heuristic that was proposed in 1987 [26].

The basic notion is to rely on the prover generating an unpredictable challenge based on his commitment and public knowledge (*e.g.* the hash of the commitment). Because this is unpredictable, the prover can not tamper the commitment to give him an advantage.

In other words, the prover computes the challenge e as a hash function of x and a, generating the argument (a, e, z) where $e = hash(x, a)$.

In order to secure such protocol, prior research suggests the application of the random oracle model [33]. The model was proposed by Bellare and Rogaway [8]. The random oracle model is a heuristic argument to give rigorous " proofs of security " of cryptographic protocols. Typically, the random oracle model is designed as a random hash function that pairs inputs (x, a) with a random output e.

Besides that, the random oracle can be programmed. The simulator can choose inputs (x, a) and corresponding outputs e and the random oracle will return output to the input provided.

Furthermore, this means that the hash function H is designed as a black box that responds to a query for the hash value of a bitstring M by returning a random value. For each query the random oracle returns an independent random value, except that it keeps a record of its responses and repeats the same responses should M be queried again [41].

The privacy and authentication properties of ZKP systems make it widely used in the design of cryptographic schemes. Specifically in voting schemes based on homomorphic threshold encryption, voters encrypt their votes and send it to the tallier to compute the final tally. In order to only allow properly formatted votes and omit cheating, it is important that an argument of knowledge (of the plaintext being a properly formed vote) is attached [33]. In other words, zero-knowledge proofs can be used in voting to prove that a vote contains a valid vote, without being able to leak what the ciphertext definitely contains.

4.5 Designated Verifier Proofs

Regular non-interactive proofs are transferable [37]. Especially in the case of electronic voting, this is not desirable.

While preserving privacy of the prover P, *designated verifier signatures* make the ballot authenticated to the designated verifier V [5]. P can now prove the following: "I am V OR s". Concluding, since the verifier V knows P can not be V, the statement s must be true. Hence, V is unable to re-use this non-transferable proof. Such proof is called a *designated verifier proof* [37].

4.6 Commitment schemes

First committing to a statement (*e.g.* the candidate order) and later revealing that commitment, is often a necessity for proofs [37]. Given a commitment key c, which is normally a random string chosen by the prover, the main property of a commitment scheme can be defined as:

$$decommit_c(commit_c(m)) = m$$

Another favourable property of commitment schemes is that the prover can not deny the commitment at a later point in time. This can be achieved using regular encryption to commit and revealing the decryption key afterwards [37]. However, in order to guarantee everlasting privacy, commitment schemes should be perfectly hiding [61], *i.e.* no amount of computation can compute the decryption key. This can be achieved, for example, because the commitment can be opened in any way:

$$\forall m' \;\; \exists c' \;\; decommit_{c'}(commit_c(m) = m'$$

However, to find a pair of m', c' is computationally hard. Thus, while the prover cannot find such an alternative pair in a short period of time to lie about his commitment, anyone trying to break the commitment later will find that the commitment could have been to any message [37].

There are quite several voting schemes leveraging such commitment schemes instead of encrypted votes [22] [50] [52] . To summarize, even if the commitments are made public and linked to the casting voter, no one can ever find out how a voter voted.

4.6.1 Trapdoor commitments

A special type of commitment schemes are so-called trapdoor commitment schemes. In trapdoor commitment schemes, the knowledge of a trapdoor allows the committing party to open the commit in any way. To assign the vote to a specific candidate, the voter sends the decommit value c privately to the election authorities. By using this decommit value c', the commit will open to the chosen candidate. For every other candidate m', the voter can compute an appropriate c' such that the commit can be opened as if it is a vote for m'. In this way, the commits to candidates can be made public, without the voter being able to prove which candidate she committed to. x This approach was used in [58] and [59]

4.7 Mixnets

The idea of mixes has been introduced by Chaum in 1981 [19]. Mixes form a simple way to ensure privacy by shuffling one's vote with a number of other votes. To assure that there is no connection between the input and output list, a mix needs to change the appearance of every message without changing the its content. Generally, there are two approaches how shuflfing can be implemented: either by using re-encryption mixes or decryption mixes.

Re-encryption mixes receive encrypted elements, re-encrypt them and randomises the order in which the elements are output. Furthermore, multiple mixes can be chained together to form a mixnet. Output can only be linked to input if all mixes collude [37].

Decryption mixes are used when the initial input consists of several layers of encryption. Each layer of encryption targets exactly one mix. By peeling off one layer per mix, the message changes its appearance through the mixnet. The same approach is used in anonymous onion routing [62].

To assure that outputs are indeed a permutation of the input, the mix can use zero-knowledge proofs (section 4.4) or designated verifier proofs (section 4.5) In voting systems, mixnets can provide anonimty in various ways [37]. The first approach is to use mixnets to anonymize traffic and obfuscate the origin of a ballot, thus providing an anonymous communication channel. Second, the voting system shuffles the list of candidates. The order

of the candidate is only revealed to the voter using a designated verifier proof. Additionally, a zero-knowledge proof on the correctness of the shuffle is proven publicly. The third option is to mix together voter credentials in order to disable individual identification [55].

4.8 Public Bulletin Boards

Public Bulletin Boards (PBB) serve as a public source of information during and after the voting phase. In most voting schemes, published information takes the the form of ballots, either in plaintext or in encrypted ciphertext. The goal is to provide a verifiable log of communication [37].

According to [38], the basic properties of a secure public bulletin board are:

- information cannot be removed or modified, only added to the board
- anyone may read the board
- the board provides a consistent view to everyone

Publishing on a PBB is used to assure individual verifiability during the voting phase and universal verifiability for the final tally result in the end. In this work, the focus lies on a blockchain-based PBB. However, P2P-based PBBs and centralized PBBs are also possible solutions.

4.8.1 Blockchain-based voting

The blockchain emerged as the backbone of the Bitcoin cryptocurrency [54]. Leaving away the currency aspect, numerous application areas open up for the blockchain technology. For instance, blockchains can be used to implement a distributed PBB.

Ever since the emergence of blockchains, the public distributed ledgers have been evaluated as a viable basis for voting schemes [63], *e.g.* prior work focused on building working proofs of concept using the Ethereum blockchain [47].

Inevitably the question arises, whether privacy and secrecy of the ballot can be assured in a public distributed ledger. Prior research has shown that it is possible to provide anonymous Bitcoin transactions [36]. Even practical implementations show the practicality of a fully anonymous distributed ledger, namely Zerocash [68], assure untraceable transactions by using the privacy-preserving properties of zk-SNARKs [9]. Nevertheless, there is active commercial and academic interest in deanonymising blockchain-based transactions [20], posing a threat to voter privacy.

Chapter 5

Secret-ballot voting protocols

Jonker et al. [37] provided a consistent and broad survey for secret-ballot voting schemes. Table 5.1 shows an overview of the different approaches. The systems are discussed in order of publication year.

Privacy claims are ballot-privacy (BP), receipt-freeness (RF), coercion-resistance (CR) and everlasting ballot-privacy (EBP). Verifiability claims are classified as universal (UV) and individual verifiability (IV). Since the main classification mechanism for voting systems is the underlying technique to achieve privacy, the table also includes those classifications: mixnets (MN), homomorphic encryption (HE), multiparty computations (MC), blind signatures (BS) and commitment schemes (CS).

In order to provide more context, the following subsections summarize the systems in the table. The descriptions are largely based on prior work by Jonker et al. [37] and the respective papers on the individual voting scheme.

5.0.1 FOO92 – Fujioka, Okamoto and Ohta

This system aims to combine privacy and verifiability. The scheme can be outlined as follows: First, the voter encrypts the vote using a commitment scheme, blinds that encryption and sends the blinded, encrypted vote to the registrar. Then, the registrar verifies the voters eligibility and if the vote has already been casted. If this is successful, the registrar signs the blinded, encrypted vote and sends the signed version to the voter. After that, The voter unblinds the message and retrieves the encrypted vote, previously signed by the registrar. Via an anonymous channel, the vote is now cast to the counter. After the voting period, the counter publishes all signed, encrypted votes. After the publication, the decryption keys are sent to the counter to allow decrypting followed by counting [30].

Two systems have been implemented using this scheme, though derivating from the specific requirements. Problems that have been avoided in later systems are for example: all voters must interact twice (cast and send encryption key), the system is not receipt-free (a voter could decrypt his individual vote). Concluding, the FOO92 system is not considered secure even though it may satisfy its original security objectives [37].

Table 5.1: Table from [37], *Claims, subclassification, analyses of voting systems.*

scheme	privacy	verifiability	attacks	based on
FOO92 [30]	BP	UV	KR05 [42]	BS
CGS97 [23]	BP	UV	–	HE
BT94 [14]	RF	–	HS00 [35]	HE
NR94 [56]	RF	–	–	MC
SK95 [66]	RF	UV	MH96 [48]	MN
Oka96 [58]	RF	–	Oka97 [59]	BS
Oka97 [59]	RF	IV, UV	–	BS
HS00 [35]	RF	IV,UV	–	MN
LK00 [44]	RF	UV	LBD04 [43]	HE
BFP01 [7]	RF	UV	Hir01 [34],JCJ05 [40]	HE
MBC01 [46]	RF	–	LK02 [45], JCJ05 [40]	HE
LK02 [45]	RF	UV	–	HE
LBD03 [43]	RF	UV	–	MN
ALBD04 [3]	RF	IV, UV	–	MN
JCJ05 [40]	RF, CR	UV	–	MN
MN06 [50]	EBP*, RF	UV	–	CS
MN07 [52]	EBP*, RF	UV	–	CS, HE

5.0.2 BT94 – Benaloh and Tuinstra

This paper outlined a severe attack on vote privacy, apparently used in Italy . Certain election systems allowed voters to list their votes in any order. The number of possible permutations is rather large, even greater than the number of voters, thus an attacker can assign specific permutations to voters in advance. The assigned permutation serves as an identifying signature. If a voter does not follow the assigned strategy, the attacker is able to identify it easily [14].

This example outlines that if the voter can produce a receipt or prove that a certain vote is his, there is a risk of coercion. The solution provided in this work is based on a "doubly" homomorphic randomised encryption scheme, allowing ciphertext operations $\oslash$ and $\otimes$, equalling plaintext operations $+$ and $-$. Also, voting is constrained to a binary choice, either 0 or 1. Besides that, the cryptosystem has to be able to generate proofs for decryption [14].

The genuine idea is to make use of an interactive challenge-response protocol. Only the voter knows the commitment and tus is convinced. The voting authority publishes a series of pairs containing encrypted 0 and encrypted 1. Then, the authority privately commits to the decryption of the published pairs.

Next, an independent and trustworthy random source generates challenge c. The authority respons to the challenge, after which the voter knows which encryption contains the voters preference. After that, the voter publicly announces which encryption is preferred.

To reach the final result, the authority sums up all encrypted votes by using the $\oplus$ homomorphic encryption and then decrypts the result.

Besides being only able to handle two candidates, this system relies on a cryptographic system with very specific properties, the existence a voting booth and a private channel [37].

BT94 claims receipt-freeness (RF) but does not explicitly claim a verifiability property. However, Hirt and Sako [35] were able to prove how to generate a receipt in the multi-candidate version of BT94.

5.0.3 NR94 – Niemi and Renvall

Around the same time as Benaloh and Tuinstra, Niemi and Renvall identified the threat of voter-coercion as well [57].

However, their solution is to adapt the voters tokens in a way that the voter can not prove that it is his or hers anymore. Unlike BT94, NR94 relies on multiparty computation. The voter is required to go to a private voting booth in order to execute the computation. The computation anonymizes the voting tokens but still ensures individual and universal verifiability. However, the computations do not scale well with respect to false votes, already noted by Niemi and Renvall. Also, the system requires a voting booth [57].

Same as BT94, NR94 claims receipt-freeness but does not explicitly claim a verifiability property [37].

5.0.4 SK95 – Sako and Kilian

Sako and Kilian [66] noted the reliance on strong physical assumptions. In detail, the voting booths in both BT94 and NR94.

Their approach is based on mixnets and one of the first to use proofs of correctness for a mixnet to achieve universal verifiability. Comparable to BT94, their system lets the voting authorities mix an encrypted 0 and 1, providing a proof to the voter on how they mixed.

The last mixer generates a list of pairs of an encrypted zero and an encrypted one. The plaintexts are encrypted using a probabilistic encryption scheme assuring that no two ciphertexts are equal. The mixer publishes the list of pairs and a proof for each pair. Additionally, the mixer sends a designated-verified proof to the voter of the order of each pair.

This proof, sent over an untappable channel, is such that the voter can lie about the order. After that, the mixer shuffles the list, after which he publishes the shuffled list and a proof that he shuffled correctly. Moreover, he sends a proof of how he reordered, again over an untappable channel, to the voter. Again, this proof can be used by the voter to

lie. This process is repeated by all mixers in the mixnet. After the last mix, the voter sends one element of one pair of the final list over an anonymous channel to the counter.

SK95 effectively managed to relax the strict physical assumptions of previous systems which relied on a voting booth. Instead the system requires an anonymous channel and a private channel.

However, their approach does not scale well to multi-candidate elections their cryptographic approach does not offer full security. SK95 claims receipt-freeness and universal verifiability. Yet, Michaels and Horster [49] identified a flaw in the mixing scheme.

5.0.5 CGS97 – Cramer, Gennaro and Schoenmakers

The CGS97 [23] system claims to be optimally efficient for a multi-authority setting. Multi-authority settings are desirable, as it reduces the concentration of trust.

In CGS97, a voter publishes a single encrypted vote plus a proof that the published message contains a valid vote. The system is an additively homomorphic voting system based on the ElGamal encryption scheme, i.e. the system sums all encrypted votes and then uses threshold (joint) decryption to decrypt the sum.

Cramer et al. outlined that proofs can be made non-interactive by using the Fiat-Shamir technique [26] as described in section 4.4. They also note that the proof should be voter-specific to avoid duplications of challenges. All the communication in this system runs via the public bulletin board. Especially the joint decryption of the sum is run via the bulletin board, giving any observer the chance to verify the correctness of the completed decryption.

Unfortunately, the system is not receipt-free but claims universal verifiability and ballot-privacy. Additionally, it claims robustness and the prevention of vote duplication. The motive of CGS97 was to create an optimally efficient scheme [37]. Thus it served as a basis for various theoretical systems such as HS00 [35], LK00 [44] and MBC01 [46].

5.0.6 Oka96 and Oka97 – Okamoto

After the observable progress in research, Okamoto proposed an adaption of FOO92 [30] to achieve receipt-freeness [58].

The main diferences from FOO92 were: direct and anonymous publishing of voters to a bulletin board, the use of untappable and anonymous channels to the authorities and the use of a trapdoor commitment scheme to encrypt the votes [37].

Knowledge of the trapdoor allows an encryption to be opened in any way, not forming a receipt. Because the trapdoor is voter-generated, the system is not entirely coercion-resistant. An intruder could force a voter to use an intruder-generated trapdoor, effectively allowing the voter to open the encryption only in one way. Thus, in Oka97 [59], Okamoto made three propositions to mitigate this flaw, further optimizing the scheme.

Both schemes, Oka96 and Oka97, claim receipt-freeness. While Oka96 does not explicitly claim verifiability properties, Oka97 claims that blind signature schemes combined with a public bulletin board satisfy both individual and universal verifiability.

5.0.7 HS00 – Hirt and Sako

Hirt and Sako [35] critically evlauated previous work, *e.g.* by proving the existence receipts in BT94 [14] and by pointing out the immense processing load of SK95 [57].

To further optimize existing protocols and lessen the strict physical assumptions, they propose a generic system to introduce receipt-freeness in systems based on homomorphic cryptography [37]. Even though strong assumptions are reduced in their work, it still requires an untappable channel from the voter to the voting authorities.

Just like BT94 [14] and SK95 [57], they propose to randomize the candidate list. HS00 uses a mixnet to randomise the list. Then, a zero knowledge proof of correctness of the mix is published, along with a designated verifier proof (indicating how the candidate list was mixed) is sent to the voter. Using this information, the voter identifies which entry in the shuffled list is the choice he wants to make.

Rather than sending the vote through an anonymous channel, the voter then publishes the vote on the bulletin board. According to HS00, all communications occur via the public bulletin board (section 4.8).

Next, the counters take all these published votes, homomorphically summing them together to jointly decrypt the result, without fully reconstructing the decryption key. Just like SK95, the used designated verifier proofs require possession of a private key. However, in HS00 a protocol is provided in which a verifier proves the possession of the private key, thus assuring non-transferability of the proof.

Hirt and Sako applied their construction to CGS97 [23]. HS00 claims receipt-freeness and a threshold variant of ballot privacy, meaning that a group with less than (threshold) t colluding authorities can not break privacy. Additionally, HS00 claims individual verifiability, universal verifiability and correctness of the tally.

5.0.8 LK00 – Lee and Kim

Similar to HS00, LK00 [44] also introduce receipt-freeness in CGS97. Instead of providing a generic construction like HS00, Lee and Kim tried to integrate receipt-freeness into CGS97.

Their proposal is to assure that the voter's vote present in CGS97 can no longer act as a receipt. By introducing a third party, an honest verifier, that needs to cooperate with the voter to construct the encrypted vote.

Because neither the voter, nor the honest verifier, is fully aware of the randomness that is used to encrypt the vote, neither can prove the link between ciphertext and plaintext.

Again, all communication is ocurring through a public bulletin board. As usual for homomorphic tallying, it is necessary to prove that the resulting ciphertext is indeed a valid vote. To prove the validity, the voter and honest verifier must cooperate. The vote encryption process involves two partly overlapping challenge-response protocols.

First, the voter publicly commits to the encrypted vote. The verifier responds with a challenge to the encrypted vote and a commitment to the encrypted randomness. The voter encrypts the appropriate response for the received challenge. Then, the voter sends the computed response accompanied by another challenge of the verifiers randomness. If the voter's response is correct, the verifier computes the correct response to the voters challenge and additionally computes and publishes a proof of the validity for the re-encryption of the voter's encrypted vote. Then again, the voter re-encrypts the vote using the encrypted randomness of the verifier and posts this on the bulletin board.

LK00 claims receipt-freeness and universal verifiability. Besides that, it claims accuracy, eligibility, fairness and robustness [44]. Hirt [34] pointed out two issues. First, that the verifier can help the voter in casting a vote that falsifies the result and second that the voter is able to generate a receipt similarly to the multi-candidate version of BT94.

5.0.9 BFP01 – Baudron, Fouque and Pointcheval

Prior work does not mention scalability of their approaches. However, for actual elections, scalability is a critical property. For instance, mixing all votes becomes immensely time-consuming for large-scale elections. Baudron et al. address the scalability issue in BFP01 [7], aiming ot design a voting system sufficiently scalable, while satisfying privacy and verifiability requirements

In order to achieve this, BFP01 proposes a system that distributes the computational load of the result over several aggregation levels. First, the system computes the result at the local level. Second, the system aggregates these at the regional level. Third and finally, the regional results are aggregated at the national level.

The voter casts three encrypted votes, one for each aggregation level, along with a proof that the three votes encrypt the same candidate. In order to facilitate this process, an ancoding scheme was introduced.

Because the votes are made public, the system is not receipt-free. Baudron et al. adressed this issue by the suggestion to use a hardware device to re-encrypt the votes, proving correct re-encryption with interactive zero-knowledge proofs or a non-interactive designated verifier proofs. Thus, they claimed that the interactive zero-knowledge proof would not not harm receipt-freeness as the actual transcript of the interaction is indistinguishable from a simulated or fake transcript.

BFP01 claims receipt-freeness, universal verifiability and robustness against faulty authorities. Still, Hirt [34] pointed out that the interactiviy in a zero knowledge protocol can be abused to construct a receipt – effectively revoking the claim of receipt-freeness made by BFP01 [37].

5.0.10 MBC01 – Magkos, Burmester and Chrissikopoulos

MBC01 [46] introduces smart card re-encrypted votes to transform CGS97 into a receipt-free voting system. Independent of LK00, they followed the same approach of trying to ensure that the obvious receipt is not a proof. Same as LK00, randomness from a third party is used to encrypt the voter's vote. In detail, they implemented a tamper-resistant smart card.

Basically, the voter sends a random order of two possible votes ($+1$ and -1) to the smart card. The smart card re-encrypts these values and returns them. Then, the smart card proves the correctness of the re-encryption using an interactive zero-knowledge proof. Thus, the voter knows which of the re-encrypted values is an encryption of the voter's choice. After that, MBC01 [46] follows the same steps as the CGS97 system.

Magkos et al. identified that in order to ensure vote-privacy, randomness is required. There are three possible origins of randomness: the voter, authorities or an external source. Using the voter as a source of randomness opens the door for abuse, potentially abusing randomness as a receipt. Using authorities implies the need for an untappable channel. Finally, using an external source would also need such an untappable channel.

Thus the solution of a tamper-proof smart-card. As long as the voter unobserved while interacting with the smart card, receipt-freeness can be achieved.

MBC01 claims receipt-freeness (if using a tamper-free smart card) but does not make any verifiability claims. Unfortunately, further studies [45] [40] were able to generate a receipt in the MBC01 system, independent of using a tamper-resistant smart card.

5.0.11 LK02, LBD03, ALBD04 – Lee et al.

In order to repair the LK00 system, Lee made inquiries into external re-encryption of votes by using three closely related systems. The first system, LK02 uses dedicated hardware in a homomorphic tallying scheme. The second system LBD03 is using an adoption of a mixnet instead. Third, ALBD04 is tweaking the mixnet setting to use an optimistic mixnet system.

LK02 – Lee and Kim

As Hirt [34] pointed out, LK00 was not receipt-free and the verifier could help the voter to falsify the results. To further research, Lee and Kim remark in LK02 [45] that the flaws uncovered in LK00 were also present in MBC01, because the voter may abuse the interactivity to construct a receipt. LK02 also claims that this holds true in general for blind signatures – thus the used blinding factor could act as a receipt.

Lee and Kim now based their LK02 system on the system proposed in by Hirt [34]. Although LK02 replaced the assumption of a trusted third party randomiser with a tamper-resistant randomiser possessed by the voter, hence greatly resembling MBC01.

The interactive proof is replaced with a designated verifier proof. However, Lee and Kim note that the costs of such dedicated hardware for every eligible voter may make LK02 financially infeasible. Furthermore, LK02 also points out that one of the proof-techniques they use may leak more information than intended [45].

In addition to the above remarks, the conclusions for MBC01 also apply to LK02: relaxing constraints, introducing a new object but the security of this new object is not extensively tested [37]. Still, LK02 claims receipt-freeness, universal verifiability, soundness, completeness, unreusability, eligibility and fairness.

LBD03 – Lee, Boyd and Dawson

In LBD03 [43], Lee et al. took the LK02 system and used mixnets instead of homomorphic tallying. LBD03 does not propose a specific mixnet. Proofs for individual votes are not necessary anymore since LBD03 decrypts each individual vote. However, Lee et al. note that malleability of the cryptographic system used in LBD03 enables a voter to copy another voter's encrypted vote [43]. However, they imply that this is not a security risk, even though the signed and encrypted votes of each voter is cast by making it public [37]. LBD03 claims receipt-freeness, universal verifiability, prevention of double voting and fairness.

ALBD04 – Aditya, Lee, Boyd and Dawson

In ALBD04 [3], Lee et al. are joined by Aditya and propose an optimistic mixnet for LBD03. Tamper-resistant hardware is removed and replaced by an authority to fulfill the re-encryption. The mixnet is optimistic in the sense that it only proves that the product of the input is the product of the output, hence not that there exist a strict correspondence between the two. As pointed out be Aditya et al., the security of an optimistic mixnet is not yet beyond doubt. However, a dishonest mix could possibly

According to Jonker et al. [37]

> Thus, a dishonest mix could mix votes
> c_1, c_2 as $\frac{1}{2}c_1$ and $2c_2$ since $\frac{1}{2}c_1 \cdot 2c_2 = c_1 \cdot c_2$.

In order to mitigate such situations, ALDB04 proposes a trace-back protocol. The protocl traces a vote back through the mixnet in case the decrypted vote was invalid. The protocol effectively has each mix undoing its mixing for that vote, to find the faulty mix in the mixnet. ALDB04 claims receipt-freeness, individual and universal verifiability.

5.0.12 JCJ05 – Juels, Catalano and Jakobsson

Juels, Catalano and Jakobsson [40] noticed that there are remaining privacy attacks even though a system is provably receipt-free. In detail, they identified three issues: forced

abstention (a voter is force to abstain), simulation (the voter is forced to give the credentials to the attacker who votes instead) and randomised voting (the voter is forced to vote randomly).

In order to prevent the three attacks and being receipt-free, JCJ05 proposed a refined notion of privacy called coercion-resistance. A scheme is defined coercion-resitant if it is infeasible for the adversary to determine whether a coerced voter complies with the demands [40].

The system of JCJ05 uses zero-knowledge proofs and an anonymous public channel. The channel between the registrar and the voters is assumed to be untappable.

The voters receive a credential from the reigstrar in the registration phase. Also, the registrar publishes the credential encrypted by the tallying authority's public key.

During the voting phase, the voters cast their vote on the anonymous channel by sending their vote, their encrypted credential and a zero-knowledge proof of the credential and of validity of the vote. In the end, the tallying authorities first check the proofs, eliminate duplicates, mix the ballots and then check the credentials and then publish the set of valid votes.

Testing plain-text equivalence between the encrypted credential (received from the voter) and the published encrypted credential allows the tallying authorities to check the validity of the credentials without decryption, thus preserving secrecy.

This scheme is mainly coercion-resistant because a voter can generate fake credentials from his real credential and use these fake credentials as the coercer enforces. Votes that were cast using fake credentials are removed in the tallying phase. Also, credentials are only validated after mixing, thus the link between a removed vote and a published vote is not possible to retrieve [37].

The scheme has an immense computational workload, thus prior work suggested adaptions to improve upon this. JCJ05 claims receipt-freeness, coercion-resistance and universal verifitability.

5.0.13 MN06, MN07 – Moran and Noar

Moran and Noar argued that regular encryption is only computationally hiding, thus only providing computational privacy [50][52]. Therefore, Moran and Noar argue the urgency of unconditional, everlasting information-theoretic privacy.

Instead of using encryption, MN06 and MN07 is based on a commitment scheme. A commitment scheme has the converse property that although commitments are computationally binding, they are unconditionally hiding [61].

By applying the Markpledge method, Moran and Noar ensured privacy and verifiability. To assure verifiability, Moran and Noar employed a cut-and-choose zero-knowledge proof. The counter produces k new lists of commitments, each list being a masked permutation

of the list of all voters' commitments. Then, a public random source outputs bits 1,...,k. If bit $i = 0$, list i is opened (proving the result of list i is the announced result) If bit $i = 1$, the permutation is revealed, proving that i is a reordering of list 0. This system relies on a trusted authority, who learns how each voter votes [37].

In MN07 this issue was addressed by adding a way to secret-share commitments, removing the need for the MarkPledge method. Basically, the commitment is privately shared between two authorities, communicating about the randomness each used and thus enabling reconstruction of the voter's choice.

MN06 and MN07 claim receipt-freeness, universal verifiability and everlasting ballot-privacy.

Chapter 6

Evaluation

The general trend in research can be summarized as follows [37]: First, properties were identified and initially defined. Second, privacy and verifiability was further refined and classified. Third, these definitions were followed by theoretical attempts to satisfy the refined concepts. Fourth, the process restarts with new notions learned from the previous steps or critical findings by related research.

As table 5.1 indicates, more and more work points out problems in prior schemes. This underlines the need for a uniform approach to test and evaluate privacy claims by using frameworks [6]. Regarding verifiability, the same trend was observable. After refined notions of end-to-end verifiability, systems were developed to satisfy those properties.

Since some systems have gained practical exposure, future research will broaden in scope (*e.g.* regarding the practical and social aspects of voting systems), thus becoming increasingly interdisciplinary in nature.

Chapter 7

Summary and Conclusions

Electronic voting is a very active field of research, bringing together expertise in many fields (*e.g.* cryptography, systems security, statistics, law and policy [12]). Digitalization of electoral processes counts on confident systems that produce enough verifiable evidence. Many of the properties described in this work are still being refined and frameworks to prove and verify certain attributes are being developed.

The different building blocks are also large seperate fields of research while each block tries to satisfy one or many properties, effectively optimizing privacy, verifiability and also auditability of voting schemes. As already pointed out by prior work [21], there is a conflict between privacy and verifiability. Further research will show if this conflict can be minimized or even entirely eliminated.

Bibliography

[1] Acar, A., Aksu, H., Uluagac, A. S., and Conti, M. (2017). A survey on homomorphic encryption schemes: Theory and implementation. *CoRR*, abs/1704.03578.

[2] Adida, B. (2012). Helios voting: Technical documentation. `http://documentation.heliosvoting.org/`.

[3] Aditya, R., Lee, B., Boyd, C., and Dawson, E. (2004). *An Efficient Mixnet-Based Voting Scheme Providing Receipt-Freeness*, pages 152–161. Springer Berlin Heidelberg, Berlin, Heidelberg.

[4] Ali, S. T. and Murray, J. (2016). An overview of end-to-end verifiable voting systems. *CoRR*, abs/1605.08554.

[5] Asaar, M. R. (2016). Code-based strong designated verifier signatures: Security analysis and a new construction. Cryptology ePrint Archive, Report 2016/779. `http://eprint.iacr.org/2016/779`.

[6] Backes, M., Hritcu, C., and Maffei, M. (2008). Automated verification of remote electronic voting protocols in the applied pi-calculus. In *Proceedings of the 2008 21st IEEE Computer Security Foundations Symposium*, CSF '08, pages 195–209, Washington, DC, USA. IEEE Computer Society.

[7] Baudron, O., Fouque, P.-A., Pointcheval, D., Poupard, G., and Stern, J. (2001). Practical multi-candidate election system. In *In PODC*, pages 274–283. ACM Press.

[8] Bellare, M. and Rogaway, P. (1993). Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 62–73, New York, NY, USA. ACM.

[9] Ben-Sasson, E., Chiesa, A., Tromer, E., and Virza, M. (2014). Succinct non-interactive zero knowledge for a von neumann architecture. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 781–796, San Diego, CA. USENIX Association.

[10] Benaloh, J. (2006). Simple verifiable elections. In *Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop*, EVT'06, pages 5–5, Berkeley, CA, USA. USENIX Association.

[11] Benaloh, J. (2007). Ballot casting assurance via voter-initiated poll station auditing. In *Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology*, EVT'07, pages 14–14, Berkeley, CA, USA. USENIX Association.

[12] Benaloh, J., Bernhard, M., Halderman, J. A., Rivest, R. L., Ryan, P. Y. A., Stark, P. B., Teague, V., Vora, P. L., and Wallach, D. S. (2017). Public evidence from secret ballots. *CoRR*, abs/1707.08619.

[13] Benaloh, J., Rivest, R. L., Ryan, P. Y. A., Stark, P. B., Teague, V., and Vora, P. L. (2015). End-to-end verifiability. *CoRR*, abs/1504.03778.

[14] Benaloh, J. and Tuinstra, D. (1994). Receipt-free secret-ballot elections. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 544–553. ACM.

[15] Blum, M., De Santis, A., Micali, S., and Persiano, G. (1991). Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118.

[16] Brandon Carter, Ken Leidal, D. N. Z. N. (2016). Survey of fully verifiable voting cryptoschemes.

[17] Chaum, D. (1983). *Blind Signatures for Untraceable Payments*, pages 199–203. Springer US, Boston, MA.

[18] Chaum, D. (2004). Secret-ballot receipts: True voter-verifiable elections. *IEEE Security Privacy*, 2(1):38–47.

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

[20] Chen, L., Xu, L., Shah, N., Diallo, N., Gao, Z., Lu, Y., and Shi, W. (2017). Unraveling blockchain based crypto-currency system supporting oblivious transactions: a formalized approach.

[21] Chevallier-Mames, B., Fouque, P.-A., Pointcheval, D., Stern, J., and Traoré, J. (2010). *On Some Incompatible Properties of Voting Schemes*, pages 191–199. Springer Berlin Heidelberg, Berlin, Heidelberg.

[22] Cramer, R., Franklin, M., Schoenmakers, B., and Yung, M. (1996). *Multi-Authority Secret-Ballot Elections with Linear Work*, pages 72–83. Springer Berlin Heidelberg, Berlin, Heidelberg.

[23] Cramer, R., Gennaro, R., and Schoenmakers, B. (1997). *A Secure and Optimally Efficient Multi-Authority Election Scheme*, pages 103–118. Springer Berlin Heidelberg, Berlin, Heidelberg.

[24] Damgård, I. and Jurik, M. (2001). *A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System*, pages 119–136. Springer Berlin Heidelberg, Berlin, Heidelberg.

[25] Damgård, I., Jurik, M., and Nielsen, J. B. (2010). A generalization of paillier's public-key system with applications to electronic voting. *International Journal of Information Security*, 9(6):371–385.

[26] Fiat, A. and Shamir, A. (1987). *How To Prove Yourself: Practical Solutions to Identification and Signature Problems*, pages 186–194. Springer Berlin Heidelberg, Berlin, Heidelberg.

[27] Fontaine, C. and Galand, F. (2007). A survey of homomorphic encryption for nonspecialists. *EURASIP J. Inf. Secur.*, 2007:15:1–15:15.

[28] Fouard, L., Duclos, M., and Lafourcade, P. (2010). Survey on electronic voting schemes.

[29] Fouque, P.-A. and Pointcheval, D. (2001). *Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks*, pages 351–368. Springer Berlin Heidelberg, Berlin, Heidelberg.

[30] Fujioka, A., Okamoto, T., and Ohta, K. (1993). *A practical secret voting scheme for large scale elections*, pages 244–251. Springer Berlin Heidelberg, Berlin, Heidelberg.

[31] Gentry, C. (2009). *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford, CA, USA. AAI3382729.

[32] Goldwasser, S., Micali, S., and Rackoff, C. (1985). The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 291–304, New York, NY, USA. ACM.

[33] Groth, J. (2005). *Non-interactive Zero-Knowledge Arguments for Voting*, pages 467–482. Springer Berlin Heidelberg, Berlin, Heidelberg.

[34] Hirt, M. (2001). *Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting*. PhD thesis.

[35] Hirt, M. and Sako, K. (2000). *Efficient Receipt-Free Voting Based on Homomorphic Encryption*, pages 539–556. Springer Berlin Heidelberg, Berlin, Heidelberg.

[36] Ibrahim, M. (2017). Securecoin: A robust secure and efficient protocol for anonymous bitcoin ecosystem. 19:295–312.

[37] Jonker, H., Mauw, S., and Pang, J. (2013). Survey. *Computer Science Review*, 10(Complete):1–30.

[38] Jonker, H. and Pang, J. (2011). Bulletin boards in voting systems: Modelling and measuring privacy. In *2011 Sixth International Conference on Availability, Reliability and Security*, pages 294–300.

[39] Jost, C., Lam, H., Maximov, A., and Smeets, B. J. M. (2015). Encryption performance improvements of the paillier cryptosystem. *IACR Cryptology ePrint Archive*, 2015:864.

[40] Juels, A., Catalano, D., and Jakobsson, M. (2010). *Coercion-Resistant Electronic Elections*, pages 37–63. Springer Berlin Heidelberg, Berlin, Heidelberg.

[41] Koblitz, N. and Menezes, A. J. (2015). The random oracle model: a twenty-year retrospective. *Designs, Codes and Cryptography*, 77(2):587–610.

[42] Kremer, S. and Ryan, M. (2005). *Analysis of an Electronic Voting Protocol in the Applied Pi Calculus*, pages 186–200. Springer Berlin Heidelberg, Berlin, Heidelberg.

[43] Lee, B., Boyd, C., Dawson, E., Kim, K., Yang, J., and Yoo, S. (2004). *Providing Receipt-Freeness in Mixnet-Based Voting Protocols*, pages 245–258. Springer Berlin Heidelberg, Berlin, Heidelberg.

[44] Lee, B. and Kim, K. (2000). Receipt-free electronic voting through collaboration of voter and honest verifier. In *Proceeding of JW-ISC2000*, pages 101–108.

[45] Lee, B. and Kim, K. (2003). *Receipt-Free Electronic Voting Scheme with a Tamper-Resistant Randomizer*, pages 389–406. Springer Berlin Heidelberg, Berlin, Heidelberg.

[46] Magkos, E., Burmester, M., and Chrissikopoulos, V. (2001). *Receipt-freeness in Large-scale Elections without Untappable Channels*, pages 683–693. Springer US, Boston, MA.

[47] McCorry, P., Shahandashti, S. F., and Hao, F. (2017). A smart contract for board-room voting with maximum voter privacy. Cryptology ePrint Archive, Report 2017/110. `http://eprint.iacr.org/2017/110`.

[48] Michels, M. and Horster, P. (1996a). *Some remarks on a receipt-free and universally verifiable Mix-type voting scheme*, pages 125–132. Springer Berlin Heidelberg, Berlin, Heidelberg.

[49] Michels, M. and Horster, P. (1996b). *Some remarks on a receipt-free and universally verifiable Mix-type voting scheme*, pages 125–132. Springer Berlin Heidelberg, Berlin, Heidelberg.

[50] Moran, T. and Naor, M. (2006a). *Receipt-Free Universally-Verifiable Voting with Everlasting Privacy*, pages 373–392. Springer Berlin Heidelberg, Berlin, Heidelberg.

[51] Moran, T. and Naor, M. (2006b). Receipt-free universally-verifiable voting with everlasting privacy. In *Proceedings of the 26th Annual International Conference on Advances in Cryptology*, CRYPTO'06, pages 373–392, Berlin, Heidelberg. Springer-Verlag.

[52] Moran, T. and Naor, M. (2007). Split-ballot voting: Everlasting privacy with distributed trust. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 246–255, New York, NY, USA. ACM.

[53] Naehrig, M., Lauter, K., and Vaikuntanathan, V. (2011). Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 113–124, New York, NY, USA. ACM.

[54] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf.

[55] Neff, C. A. (2001). A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, CCS '01, pages 116–125, New York, NY, USA. ACM.

[56] Niemi, V. and Renvall, A. (1995a). *How to prevent buying of votes in computer elections*, pages 164–170. Springer Berlin Heidelberg, Berlin, Heidelberg.

[57] Niemi, V. and Renvall, A. (1995b). *How to prevent buying of votes in computer elections*, pages 164–170. Springer Berlin Heidelberg, Berlin, Heidelberg.

[58] Okamoto, T. (1996). *An electronic voting scheme*, pages 21–30. Springer US, Boston, MA.

[59] Okamoto, T. (1998). *Receipt-free electronic voting schemes for large scale elections*, pages 25–35. Springer Berlin Heidelberg, Berlin, Heidelberg.

[60] Paillier, P. (1999). *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, pages 223–238. Springer Berlin Heidelberg, Berlin, Heidelberg.

[61] Pedersen, T. P. (1992). *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing*, pages 129–140. Springer Berlin Heidelberg, Berlin, Heidelberg.

[62] Reed, M. G., Syverson, P. F., and Goldschlag, D. M. (1998). Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494.

[63] Riemann, R. and Grumbach, S. (2017). Distributed protocols at the rescue for trustworthy online voting. *CoRR*, abs/1705.04480.

[64] Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126.

[65] Ryan, P. Y. A. (2005). A variant of the chaum voter-verifiable scheme. In *Proceedings of the 2005 Workshop on Issues in the Theory of Security*, WITS '05, pages 81–88, New York, NY, USA. ACM.

[66] Sako, K. and Kilian, J. (1995). *Receipt-Free Mix-Type Voting Scheme*, pages 393–403. Springer Berlin Heidelberg, Berlin, Heidelberg.

[67] Salamonsen, K. (2014). A security analysis of the helios voting protocol and application to the norwegian county election. Master's thesis, Norwegian University of Science and Technology.

[68] Sasson, E. B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., and Virza, M. (2014). Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474.

[69] Snips (2017). Paillier benchmarking. `https://github.com/snipsco/paillier-libraries-benchmarks`.

[70] UN General Assembly (1948). Universal declaration of human rights: Resolution adopted by the general assembly 10(12).

[71] University of Texas at Dallas (2010). Paillier threshold encryption toolbox. `http://cs.utdallas.edu/dspl/cgi-bin/pailliertoolbox/javadoc/paillierp/package-summary.html`.

[72] Yi, X., Paulet, R., and Bertino, E. (2014). *Homomorphic Encryption*, pages 27–46. Springer International Publishing, Cham.

List of Tables