Coercion-Resistance in Electronic Voting

Susanna Lassila

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2023

Abstract

Coercion-resistance is one of the most important security properties in an electronic voting protocol. In this report, we analyse and find problems in a definition of coercion-resistance and two protocols claiming to be coercion-resistant. The problems in the definition of coercion-resistance defined in [2] are related to a value in the definition that is not clearly specified. These problems mean that the definition cannot be used in its current state to analyse the incoercibility of electronic voting protocols. The first attack is against the protocol defined in [34], the attack shows that the paper makes incorrect assumptions which result in the protocol not being coercion-resistant. The second attack is against the protocol defined in [13]. The attack shows that the adversary can gain a non-negligible advantage and even confirm whether the voter followed the coercion in some cases. These attacks minimise the contributions of the protocol definitions.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Susanna Lassila*)

Acknowledgements

Firstly, I would like to thank my supervisor Myrto Arapinis for her guidance and support throughout the project. I would also like to thank Thomas Zacharias for offering his insights and feedback for the project.

Finally, I would like to thank the other participants of the reading group in coercion-resistance, Liam Medley, Nikolaos Lamprou, and Pavlos Georgiou, for helping me with tackling conceptual problems and building my understanding in electronic voting and coercion-resistance.

Table of Contents

Chapter 1

Introduction

1.1 Electronic Voting and Coercion-Resistance

Voting is an important method used in a democratic group, allowing the members of the group to have an impact on the decisions the group makes. These groups can be small scale like university societies or large scale nationwide groups. Electronic voting, often referred to as e-voting, is a more modern approach to voting. E-voting can be roughly divided to two types: one requiring a form of in-person involvement and one done completely remotely. In this paper, we are focusing on remote e-voting.

Electronic voting protocols and their properties are usually defined and proved in one of two frameworks: game-based security and universal composability (UC). In the game-based framework, the adversary is trying to distinguish the ideal protocol from the real in a game situation. In this paper, we will focus on the UC framework as it has not been discussed and analysed to the same level that the game-based security has. The universal composability framework aims to define different functionalities as black-boxes which can be combined with other functionalities. The basic idea is that the environment cannot tell apart the behaviour of the ideal functionality interacting with a simulator from the behaviour of the introduced protocol interacting with an adversary. The framework will be introduced in detail in the Preliminaries section of Chapter 2.

Coercion-resistance in electronic voting is a complicated problem many have tried to solve. It is difficult to formally define all aspects of coercion and adversarial behaviour, which can range from observing the election to corruption of the election authorities. Since defining the concept of coercion-resistance is difficult, providing a protocol that would satisfy such definitions is at least as challenging. This difficulty is explored in this report by analysing an introduced definition of coercion-resistance and two protocols that have attempted to solve the problem and claim to be coercion-resistant. The definition introduced in the paper [2] has some problems that prevent it from being usable in analysing e-voting protocols. These problems will be discussed in detail in Chapter 3. The two protocols [34] and [13] fail to satisfy the property of coercion-resistance as shown in Chapters 4 and 5, respectively. The attacks which break the defences in the protocols are discussed in detail.

1.2 Completion Criteria

The original objectives of this report were to look at coercion-resistance and self-tallying and whether the two properties are compatible with the following methodology:

1. Review existing definitions of coercion resistance in voting
2. Formalise the class of self-tallying protocols
3. Establish if self-tallying and coercion resistance are compatible properties

However, the problem of coercion-resistance and how to define it turned out to be more complex than expected. As will be shown in this paper, the existing definitions and protocols often contain omissions or other problems. This meant that trying to find or formalise a definition which sufficiently captures this is very difficult. In addition to the field not being at a suitable state, the project involved learning a lot of concepts and frameworks outside the undergraduate syllabus, such as the universally composable framework and in general formalising electronic voting protocols and attacks against them, which meant that completing the first step of the original criteria was already a lot of work. Thus, the objectives were adjusted to focus on coercion-resistance as follows:

1. Explore definitions formalised in the universally composable framework
2. Analyse protocols that claim to be coercion-resistant

The first modified objective was chosen as universally composable coercion-resistance has not been discussed to the same level as the definitions in the game-based security model, as we already mentioned. The second modified objective was chosen since there are protocols that have been defined and claim coercion-resistance but have not been properly analysed to confirm these claims. These two objectives were completed to an acceptable level by completing the following steps:

- Pointing out problems in a definition of coercion-resistance in the paper [2] in Chapter 3
- Introducing an attack against a protocol defined in [34] in Chapter 4
- Introducing an attack against a protocol defined in [13] in Chapter 5

Chapter 2

Background

This chapter discusses the previous work done in electronic voting and how this fits together with the work done in this report. We will look at different security properties and how those have been informally and formally defined with the focus being on different forms of coercion-resistance. In the preliminaries section, a simple example voting protocol will be introduced to help with discussing different formal definitions and frameworks throughout this paper. We will go into more detail about the universally composable framework since the definition discussed later in the paper will be in this framework. Additionally, we will briefly discuss the JCJ protocol as it is an important coercion-resistant protocol and will be mentioned throughout this report.

2.1 Related Work

Whether electronic or not, and whether a small or a large group the basic expectations for the voting system are the same. Some of these expected properties are correctness, fairness, verifiability, democracy, and eligibility. Additionally, there are the three levels of coercion-resistance: privacy, receipt-freeness, and coercion-resistance. Examples of informal definitions for all of these are listed below.

- Correctness: ”Every honestly cast vote will be included in the tally set which is the same for all honest voters.”[3]
- Verifiability: ”Voters and possibly external auditors should be able to check whether the votes were actually counted, and whether the published election result is correct.”[19]
- Fairness: ”During the Cast phase, no party can learn some partial result.”[3]
- Democracy or one voter – one vote: ”Only one vote per (eligible) voter can be included in the tally set of each honest voter.”[3]
- Eligibility: ”Only authorized voters should be able to vote.”[32]
- Privacy or Input/Output Coercion: ”The (honest) voters’ identities cannot be linked to their votes.”[3]

- Receipt-freeness: ”The inability of a voter to prove to an attacker that she voted in a particular manner, even if the voter wishes to do so.”[26]
- Coercion-resistance: ”An honest voter can cast her vote even if she is, during some time, fully under the control of an attacker.”[15]

There have been many attempts to formally define these properties and many of the properties have been defined in both the universal composability framework and the game-based framework. Some of these definitions are more successful than others as there are many aspects to consider and making one property stronger might make another one weaker. For example, improving privacy by making the votes completely disconnected from the voter might mean that the voter cannot verify whether their vote was tallied correctly or not. Thus, finding a balance between all these properties is important and often depends on what is valued in the specific type of election in question.

We will briefly discuss some of the formal definitions of correctness and verifiability which are often consider as the main properties with privacy. Additionally, some papers consider that for example eligibility and democracy fall under correctness [26, 35] and on the other hand that the privacy implies fairness [27, 35]. Firstly, correctness has been defined and formally proved in [3, 35] and more trivially proved or assumed true in [21, 23, 26]. Verifiability has also been formally defined in a large group of papers, such as [3, 15, 16, 26, 35, 36]. Again some papers [3, 28, 35] prove verifiability more formally than others [15, 16, 26]. [19] discusses the definitions of verifiability and protocols that satisfy these definitions. Additionally, they discuss problems with some of these definitions like [5, 18, 33].

2.1.1 Privacy

Privacy can be considered as the first level of coercion-resistance.[25] Without privacy the coercer can link the votes to the voters and thus know immediately whether the coerced voter followed the coercion or tried to evade it.[32] Another name for privacy is thus input/output coercion which is used by [2]. Privacy is often a core requirement for a voting protocol since without it, anyone can access how any voter voted.

There are a lot of different ways to discuss privacy. Firstly, there is the perfect ballot secrecy also known as full vote confidentiality as discussed in [35]. Perfect-ballot secrecy is satisfied when ”knowledge about the partial tally of the ballots of any set of voters is only computable by the coalition of all the remaining voters” [27]. Another notion of privacy is the ideal privacy, which is defined in the universally composable framework like in [21, 23, 31]. Following this, we have the ballot privacy in the game-based framework like in [1, 6]. The entropy based privacy, like in [7, 17], focuses on the information that can be learned by an adversary and whether this depends on the actions of the voter.

In the end, some of these definitions attempting to formalise privacy end up having problems in them, some of these problems are shown in [8]. For example [6, 9] define notions of privacy that are too weak. This means that they can be satisfied by protocols which do not actually preserve privacy. Alternatively, other definitions like [10] end up too strong. This means that they prevent the voter from verifying that their vote has been correctly tallied. Another problem is that some definitions are too limited or restrictive such as [7, 4, 5]. These definitions do not necessarily have flaws but the limit the class of protocols or privacy breaches that can be considered.

2.1.2 Receipt-Freeness

The second level of coercion-resistance is usually called receipt-freeness. Where privacy mostly concerns honest voters, receipt-freeness extends to dishonest ones. Mainly, it focuses on whether the voter can provide a verifiable receipt to the adversary that proves how they voted. Therefore, in some cases, receipt-freeness is also discussed as vote-selling resistance since with this receipt the voter could easily sell their vote if they wanted to.[26, 34] A receipt-free protocol must be vote-selling resistant, since if a voter can sell their vote using a receipt, the adversary can also force an honest voter to produce a similar receipt. The paper [34] attempts to provide a vote-selling resistant protocol which still provides a receipt. However, as we will show in Chapter 4, the protocol fails to be vote-selling resistant.

Different protocols differentiate between the three levels of coercion-resistance differently. For example, [28] discusses privacy and receipt-freeness more interchangeably than for example [15]. Additionally, [15] extends receipt-freeness to a stronger definition which includes cases where the adversary interacts with the voter more than in traditional receipt-freeness. There are quite a few protocols that satisfy receipt-freeness such as [15, 26, 31] to list a few. This is likely due to the definition of receipt-freeness being more restricted and thus easier to formalise when compared with the next level of coercion-resistance.[2, 35]

2.1.3 Coercion-Resistance

The third level of coercion-resistance is the active coercion which includes all the stronger adversaries. For example, the JCJ protocol [26] defines an adversary that will ask the voters for their credentials and then vote using these. Alternatively, the definition in [2] allows the adversary to ask the voters to complete arbitrary steps and provide receipts or proofs for these. Overall, coercion-resistance includes a large range of adversaries and as will be showed in the later chapters this makes it difficult to formalise. There are many ways to define these adversaries. For example, [1] introduces a protocol that provides protection in a case where coercion is not a serious concern and another protocol [11] provides receipts for voters even for candidates they did not vote for. Alternatively, the paper [22] goes about the problem differently by allowing coercion but instead recording it and publishing these records.

There are currently three main frameworks in which coercion-resistance has been worked in and analysed. Firstly, we will discuss the game-based security framework. The JCJ protocol introduced in 2002 [26] is considered in the words of [20] ”the reference paradigm” for coercion-resistance in the game-based security framework. For example, [16, 30] build on the JCJ protocol and aim to improve aspects of it. The paper [25] shows how the definition of JCJ is flawed in that it could never be realised.

Furthermore, another paper in 2022 [20] found that JCJ leaks too much information when voters are allowed to revote, which is often necessary in receipt-freeness and coercion-resistance. We will discuss the JCJ protocol and the fix offered in [20] in the preliminaries section and in Chapter 6.

The second framework used with coercion-resistance is the universal composability framework which we discuss in detail in the preliminaries section. Additionally, we analyse a definition of coercion-resistance introduced in the UC framework by [2] in Chapter 3. An example of a receipt-free protocol in the UC framework is offered by [31]. The third framework is defined by [29]. It focuses on the probability that an adversary can distinguish whether the voter followed the coercion or a deception strategy. Thus, the framework is able to quantitatively analyse whether a protocol is coercion-resistant, rather than the property only being satisfied or not. This framework is more complex than the two other frameworks and therefore, it is used a lot less in literature.

2.1.4 Concepts

Some concepts or terms that will used throughout the paper are defined here. Firstly, we will define the difference between the uses of the words corrupt and coerce. With a corrupted party, we mean that the party is fully under the influence of the adversary. Whereas, a coerced party can either follow the coercion, thus acting similarly to a corrupted party, or attempt to evade the coercion by somehow deceiving the adversary.

We will use the phrase a trivial attack to indicate an attack against coercion-resistance where the adversary can break coercion-resistance when the tally is published. Such an attack is possible if the adversary for example has corrupted all but a single voter that they have coerced, in this case they know how all the other parties have voted and can immediately infer how the coerced party voted after the tally is published. Usually in proofs and definitions it is somehow indicated as to how the trivial attack should be dealt with. For example, [34] states that such cases are not considered.

Besides attempting to coerce a party to vote for a certain candidate, the adversaries can also launch so called forced absence attacks. In such attacks, the adversaries require the coerced voters to not participate in the elections.[25] These are often considered under the coercion-resistant definitions, like in [26]. However, some weaker definitions of coercion-resistance do not consider these.[25]

Sometimes called the trivial protocol is one where all the security properties rely on a single trusted authority to be honest.[24] While a protocol with this trust assumption is easy to define theoretically, the assumption is too strong to be realistic in real-life. Thus, most introduced protocols aim to provide improvements in one way or another, for example by offering distributed trust.

When we discuss coercion-resistance in electronic voting, we usually do not consider scenarios where the adversary is physically in the same space as the voter.[35] This is acceptable since such attacks are not scalable to a majority of the voters in large scale elections.

2.2 Preliminaries

2.2.1 Example Protocol

To help with discussing the definitions and concepts introduced in the paper, we will use the following example protocol. The protocol is a simple voting protocol where each of the voters can choose between two options α_1 and α_2. In this protocol, the eligible voters have credentials that they have received through some process that we will not include as this is only an example protocol focused on the voting aspect. Each eligible voter v_i can then use their credentials to securely and privately receive a ballot b_i from the trusted authority T which is either

$$b_i = \begin{vmatrix} C_1^i & \alpha_1 \\ C_2^i & \alpha_2 \end{vmatrix} \text{ or } b_i = \begin{vmatrix} C_2^i & \alpha_1 \\ C_1^i & \alpha_2 \end{vmatrix}$$

where C_1^i and C_2^i are secure random values specific to the voter v_i. The voter will then vote by sending the random value, i.e. C_1^i or C_2^i, that corresponds to their chosen vote to the bulletin board. The bulletin board then shows either

$$\begin{vmatrix} C_1^i & Voted \\ C_2^i & \end{vmatrix} \text{ or } \begin{vmatrix} C_1^i & \\ C_2^i & Voted \end{vmatrix}$$

depending on how the voter voted. Now since the random value codes are personal to each voter, the authority can verify that the votes on the bulletin board were sent by eligible voters. At the end of the election, the authority will remove the incorrect votes from the bulletin board. Based on the votes on the bulletin board, the authority can then tally and publish the result of the election. Basic-level of privacy is preserved as an adversary would not be able to tell based on the votes on the board, how each individual voter voted, as long as the authority is honest. The trust assumption here is quite strong since there is just one authority that needs to be fully trusted. The trust assumption could also be improved by creating a group of authorities that jointly create the ballots for the voters and tally the result of the election. A method of this kind was used in the JCJ* protocol, which will be introduced in the subsection 2.2.3.1. However, as this is only an example protocol such considerations are not necessary at this stage.

In order to deceive against an adversary asking for a receipt of how the voter voted, the voter should with some probability d flip their ballot, i.e. swap the places of C_1 and C_2, and send this to the adversary with the code they used to vote. The voter can also just flip the ballot in a way that confirms their story to the adversary. In other words, this would mean that a deceiving voter would always flip ($d = 1$) and a coerced voter would never flip ($d = 0$). Since the two voting codes are equal from the perspective of the adversary, it can be informally seen that the protocol should be receipt-free. Essentially, the adversary has no way of confirming whether the voter is lying or not.

On the other hand, this protocol is not coercion-resistant since in that case the adversary can ask the voter to perform arbitrary steps and provide them with more information. For example, the adversary could just ask the voter to send over their credentials used to verify the voter's eligibility to the authority. Using this credential, the adversary could receive the ballot directly from the authorities and vote on behalf of the voter. An even stronger adversary could corrupt the authority and thus be able to control the whole election. Alternatively, if the adversary can corrupt all the voters except one then they would be able to just tell based on the result whether this one voter followed the coercion or not. This shows, that under coercion there are many different types of adversaries and as we will show in this paper, this makes it difficult to define coercion-resistance and what it should include.

2.2.2 Universally Composable Security

Canetti introduces universally composable security in [14]. The universal composability framework is used ”for describing cryptographic protocols and analysing their security”[14]. In this section we will look at this definition in detail as it is necessary background knowledge in order to understand the next chapters. To help with understanding the main idea, we will discuss the framework with the example protocol introduced in the last subsection.

For a property to be universally composable under the framework, it needs to satisfy the definition we will formally introduce in the next section. The parts in the framework are the environment Z, the voters $v_i \in V$ for $i \in [n]$, the adversary A and the corresponding simulator S, the ideal functionality F, and the real-life protocol π. The environment Z is interacting with the ideal process and the real-life model and is trying to distinguish between the two executions. In the ideal process, the ideal functionality F of the property is interacting with the voters V and a simulator S. In the real-life model, the voters V are running the introduced protocol π and interacting with an adversary A. The environment can freely interact with the voters as well as the simulator and the adversary. Thus, the information that the simulator and the adversary have access to has an important meaning, mainly that the ideal functionality and the protocol must leak the same information. In the end, if the environment cannot distinguish between the executions happening in the two processes, the property is securely realised by the protocol.

2.2.2.1 Formal definition

In this section, we will look at the formal details of the framework. Before introducing the definition for universally composable properties, we need to look at the definition 1 in [14] which defines indistinguishability for binary distributions as follows

”**Definition 1** *Two binary distribution ensembles X and Y are **indistinguishable** (written $X \approx Y$) if for any $c \in N$ there exists $k_0 \in N$ such that for all $k > k_0$ and for all a we have*

$$|Pr(X(k,a) = 1) - Pr(Y(k,a) = 1)| < k^{-c}.\text{”}$$

This means that X and Y are indistinguishable if the advantage of being able to tell them apart is negligible. Now, the definition 2 from [14] defines how protocols can be proved to satisfy different universally composable properties.

”**Definition 2** *Let $n \in N$. Let F be an ideal functionality and let π be an n-party protocol. We say that π **securely realizes** F if for any adversary A there exists an ideal-process adversary S such that for any environment Z we have*

$$IDEAL_{F,S,Z} \approx REAL_{\pi,A,Z}.\text{''}$$

Essentially, this means that the protocol π securely realises the ideal functionality F if the ideal and real executions are indistinguishable to the environment. In this paper, we will use the following notation to indicate the same thing:

$$EXEC_{F,S,Z} \approx EXEC_{\pi,A,Z}$$

Notice that the environment is the only participant that is shared between the two sides of the equation. This is because the environment is trying to distinguish whether it is interacting with the real or the ideal process. Putting these two together and using a slightly different notation we get that π securely realizes F if for any adversary A there exists an ideal-process adversary S such that for any environment Z we have

$$|Pr(EXEC_{F,S,Z}(1^{\lambda}) = 1) - Pr(EXEC_{\pi,A,Z}(1^{\lambda}) = 1)| < negl.$$

where 1^{λ} notates the randomness in the execution and $negl$ indicates a negligible function. An illustration of how the framework works is shown in Figure 2.1. It shows the interactions between the different participants as well as the differences between the two processes. The biggest difference is that in the real-life model, the participants of the protocol, i.e. the voters, run the protocol π, shown by having the voters inside the protocol box, whereas in the ideal process the functionality is a participant on its own.

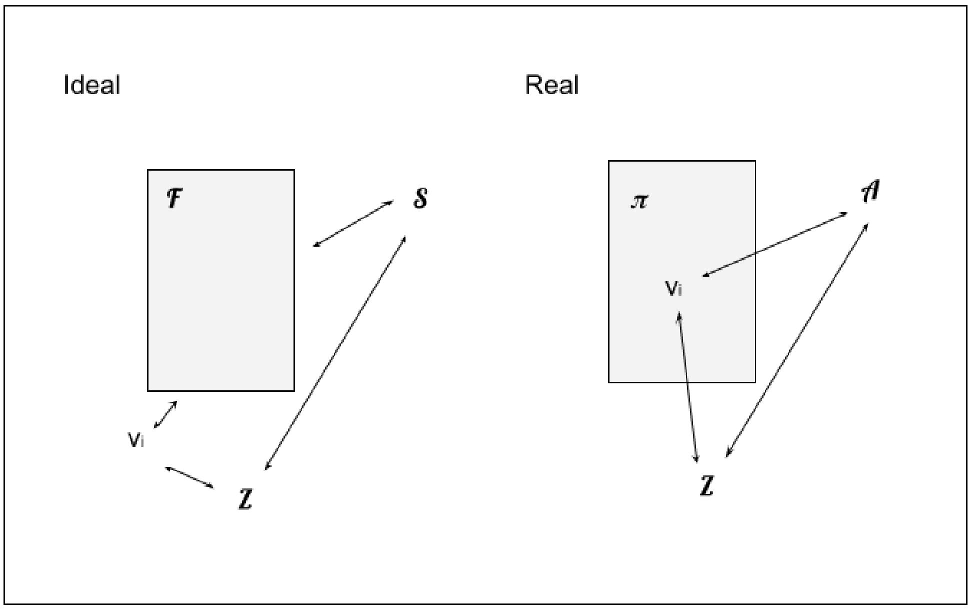

Figure 2.1: Illustration of universally composable framework

2.2.2.2 Example protocol and the UC framework

We can now look at the definition with an example property and the example protocol introduced in the previous section. We will choose the property to be privacy, since that is related to coercion-resistance but is easier to define. Now, the ideal functionality F will define the wanted level of privacy. The protocol π will be the protocol that was just defined in the section 2.2.1. The environment Z is the input provider and it would also be the coercer but as we are discussing privacy, this is not necessary. In privacy, the adversary is trying to find out how the voter voted based on the public information, but in this protocol they should not be able to. The environment tries to distinguish the two processes by communicating with the adversary A and the simulator S.

The Functionality $\mathcal{F}_{Privacy}$

- Upon receiving input $(Init, sid, C^i_j)$ from party P_i, send $(sid, C^i_j, Voted)$ to the simulator.
- Upon discarding ineligible votes $\{C^i_j\}$, send $(sid, \{C^i_j\}, Discarded)$ to the simulator.
- Upon finding the tally T, send $(sid, T, Tally)$ to the simulator.

Figure 2.2: The Privacy Functionality used with the example protocol.

To talk about whether the protocol securely realises the property, we have to define the ideal functionality. In order to do this, we need to decide what is the level of privacy we wish to achieve. We know that the protocol will reveal some values to the adversary so we must consider what values can be revealed to the simulator while still preserving the wanted level of privacy. The behaviour of the functionality is shown in Figure 2.2. Upon receiving a vote the protocol will take the vote and put it on the bulletin board. As this is public information, this needs to be revealed to the simulator by the functionality since the environment can freely communicate with the adversary and the simulator and it will be able to distinguish between the two processes if the simulator is not sent this information. Therefore, after receiving a vote from a voter the functionality will send this information to the simulator. This is also shown in Figure 2.2. In the end of the election, the removal of any incorrect votes and the result of the tally are shown on the board. Thus, as these are public information, the simulator must be informed of them. Leaking all this information to the simulator is acceptable as privacy is still preserved because with all the information given to the simulator one cannot find out how any one voter voted.

We can now discuss the information leaked by the protocol π to the adversary. Firstly, the adversary receives no information about the credentials of the voter or the initial ballot that the voter receives. After the voter posts their vote on the bulletin board, the adversary can see what random value the voter voted for. However, this reveals no information about the actual contents of the vote to the adversary. At the end of the election, the adversary can see which votes get discarded but receives no additional information about these votes. Finally, the adversary can see the result of the election when it is published to the board. Again the adversary receives no additional information at this stage.

Now, this means that the functionality $F_{Privacy}$ and the protocol π leak the same information to the simulator S and the adversary A. Thus, the environment will not be able to tell the simulator and the adversary apart from just interacting with them, meaning that the two situations appear indistinguishable to the environment. Based on this discussion it would seem that the example protocol securely realizes the defined privacy functionality $F_{Privacy}$. A formal proof would be needed if we wanted to be certain of this. However, as we are only illustrating the framework, this is not necessary.

2.2.3 JCJ Protocol

In this subsection, we will briefly discuss the protocol called JCJ introduced by [26]. This protocol has an important role in the area of coercion-resistant electronic voting, which is why it is important to understand the basic idea of it for the following chapters.

The main idea behind the coercion-resistance in the JCJ protocol is that the voters can produce fake credentials and give these to the adversaries attempting to coerce them. The adversaries can then vote with these fake credentials and will not be able to verify whether or not the vote was counted in the final tally. The downside of JCJ is that it requires a large overhead, i.e. quadratic in the number of voters in the tally phase.

The paper [26] uses the game-based security framework and so do many other coercion-resistant papers that follow the main idea used in JCJ. The definition of coercion-resistance in JCJ ”centers on a kind of game between the adversary A and a voter targeted by the adversary for coercive attack” [26]. This game is defined in terms of the ideal and real behaviour, where there is a coin flipped to decide whether the voter will try to evade the coercion or just submits to it. The adversary will then have to guess whether the coin was flipped or not.

2.2.3.1 JCJ*

The paper [20] shows that the JCJ protocol is vulnerable to an attack if the voters are allowed to revote. This is because the protocol will reveal not only the number of revotes but also the distribution of them to the adversary. This allows the adversary to gain a non-negligible advantage and [20] even demonstrates that in certain cases the adversary is able to verify how the coerced voter voted. The paper [20] then provides an improved of version JCJ, from now on referred to as JCJ*, which aims to solve this problem by hiding the distributions for the removed votes and instead just allowing the adversary to receive the total number of removed votes. This is done by changing the ballot cleaning phase into one which discards all the ballots, that failed any of the checks, at the same time after all the checks have been completed.

JCJ* also improves the definition of coercion-resistance in JCJ by better modelling revoting and addition of fake votes. It acknowledges that revoting and the addition of fake votes is often done by the authorities rather than the honest voters, since usually honest voters will not revote randomly at random times which would be required. We will discuss some aspects regarding the deception strategies that [20] leaves underspecified in Chapter 6.

Chapter 3

Definition of Coercion-Resistance

Alwen, Ostrovsky, Zhou and Zikas introduce a way to show if a universally composable protocol is incoercible.[2] However, this definition leaves some aspects unclear and in other parts is too broad to capture the wanted functionality; it captures the trivial attack in active coercion and the definition does not properly capture the distinguishability between the coercion and deception scenarios, i.e. the value of Δ. This chapter discusses the definition and the ambiguities in it. The discussion will be done with respect to two of the types of coercion defined by the paper, active coercion and receipt-freeness.

3.1 Definition

To understand the definition of UC incoercibility, it is necessary to define the parts it consists of. There are two universal composability scenarios used in the definition, these are the coercion and the deception. In the coercion scenario, the ideal and real-life processes capture the behaviour where the coerced parties follow the instructions given by the coercer. Similarly, the ideal and real deception scenarios model the behaviour where the coerced parties do not wish to follow these instructions.

Before moving onto the definition of incoercibility, we will discuss the notation and participants used in the definition. We will use the example protocol introduced in section 2.2.1 to illustrate the different parts of the definition. First, we have the set of coerced parties $\mathcal{J}$, in particular $\mathcal{J} \subseteq [n]$. This set is chosen by $\mathcal{Z}$ in the beginning of the protocol execution. In the example protocol, this could be any set of voters. In the ideal coercion situation, the coerced parties will behave as instructed. This behaviour is described using the dummy protocol $dum_{\mathcal{J}}$. This protocol ensures that the simulator handles all the communication not intended for the functionality. The dummy protocol in the example protocol could be for example providing a receipt for how they voted or providing the credentials to the adversary. In the corresponding real-life situation, this is defined as $C_{\mathcal{J}}$. This means that for a coerced party $\pi_i = \pi(p_i)$, the $\mathrm{C}(\pi_i)$ has the same set of communication tapes as π_i and it models the behavior the coercer is attempting to enforce upon a party p_i running protocol π. In the example protocol, this would be similar to the dummy protocol so giving over the receipts or credentials. However, here the party is doing the work and whereas in the ideal process the simulator is. For the deception scenario there is a deception strategy $D_i(\pi_i) = (D_i^1, \pi_i, D_i^2)$. This triple does not change the messages between π_i and the adversary but D_i^1 is placed between π_i and $\mathcal{Z}$ and D_i^2 is placed between π_i and its hybrids. From this it follows that we can write the ideal and real deceptions as follows: $DI_i = D_i(dum_i)$ and $DR_i = D_i'(C_i(\pi))$. In the example protocol, these could be for example flipping the ballot or lying about the credentials.

”**Definition 1 (UC Incoercibility).** *Let π be an n-party protocol and for an n-party functionality $\mathcal{F}$ let ϕ denote the dummy $\mathcal{F}$-hybrid protocol, and let $\mathcal{C}$ be a coercion. We say that π $\mathcal{C}$-IUC realizes $\mathcal{F}$ if for every $i \in [n]$ and every ideal deception strategy DI_i there exists an real deception strategy DR_i with the following property. For every adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that for any set $DI_{\mathcal{J}} = \{DI_i : i \in \mathcal{J}\}$ and every environments $\mathcal{Z}$:*

$$EXEC_{\phi,dum_{\mathcal{J}},\mathcal{S},\mathcal{Z}} \overset{c}{\approx} EXEC_{\pi,C_{\mathcal{J}},\mathcal{A},\mathcal{Z}} \tag{1}$$

$$EXEC_{\phi,DI_J,\mathcal{S},\mathcal{Z}} \overset{c}{\approx} EXEC_{\pi,DR_J,\mathcal{A},\mathcal{Z}} \tag{2}$$

where dum denotes the dummy coercer described above.” [2]

While the definition captures the two UC scenarios separately in (1) and (2), it fails to include the relationship between these two scenarios. This is vaguely discussed in the paper outside the definition as follows: ”if the advantage of the best environment (i.e., the one that maximizes its advantage) in distinguishing the top-left world from the bottom-left world is $0 \leq \Delta \leq 1$, then the advantage of the best environment in distinguishing the top-right world from the bottom-right world is also $\Delta' = \Delta$ (plus/minus some negligible quantity)”. Here the top-left and bottom-left worlds are the ideal coercion and the ideal deception scenarios, respectively. The right side is equally the real coercion and deception. From this description, it would seem that $\Delta = 1$ is bad and $\Delta = 0$ is good since Δ is said to be the advantage in distinguishing between the UC scenarios (1) and (2).

The interpretation of the meaning of Δ can be further emphasised by looking at a figure given in the paper [2]. This figure has four individual figures representing the four situations, however for Figure 3.1 these figures were changed to the equations defined in Definition 1 in [2] to make it easier to understand. The top two boxes in the figure show equation (1) from the definition and the bottom two show equation (2). The figure also includes Δ and aims to illustrate its meaning. However, while the figure does not necessarily further clarify the meaning of Δ, it does point towards the interpretation that Δ describes the distance between the two situations within the ideal world. Thus, it seems that the only mentions of Δ imply that 0 is good and 1 is bad. However, we will show that $\Delta = 0$ can be true in a scenario where the protocol is not incoercible and also that Δ can always be made 1 in active coercion, meaning that no protocol could satisfy it. Thus, the meaning of the value Δ is left unclear.

3.2 Active Coercion and the Value of Δ

From the way the value Δ is defined, it is possible to make the value of it always 0 and 1 for any protocol in an active coercion situation. First, we can always have $\Delta = 0$ by

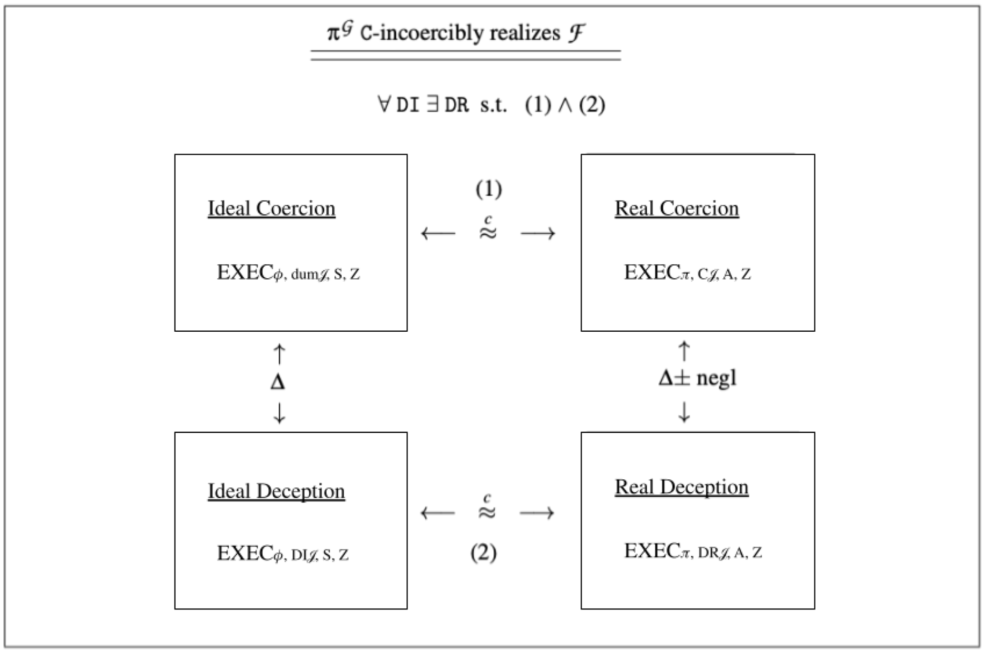
$\pi^{\mathcal{G}}$ C-incoercibly realizes $\mathcal{F}$
$\forall$ DI $\exists$ DR s.t. (1) $\wedge$ (2)
Ideal Coercion
$\mathrm{EXEC}_{\phi,\,\mathrm{dum}_{\mathcal{I}},\,S,\,Z}$
(1)
$\overset{c}{\approx}$
Real Coercion
$\mathrm{EXEC}_{\pi,\,C_{\mathcal{I}},\,A,\,Z}$
Δ
$\Delta \pm$ negl
Ideal Deception
$\mathrm{EXEC}_{\phi,\,\mathrm{DI}_{\mathcal{I}},\,S,\,Z}$
$\overset{c}{\approx}$
(2)
Real Deception
$\mathrm{EXEC}_{\pi,\,\mathrm{DR}_{\mathcal{I}},\,A,\,Z}$

Figure 3.1: Modified version of a figure in [2], the four small figures in the boxes were changed to the equations from Definition 1 [2].

having a deception strategy that makes the top and the bottom world the same. This means that the ideal deception strategy is equal to the dummy protocol. In the example protocol from section 2.2.1, this would mean that when asked for the credentials the voter would send the credentials to the adversary, even if they are trying to evade the coercion. This would be included in the definition as it requires ”for every ideal deception strategy” and does not provide restrictions for what counts as a deception strategy. Using such a deception strategy would make the advantage in distinguishing between the worlds zero. However, this strategy is the same as no deception strategy in the sense that the coerced voters are just instructed to follow the coercion, thus it goes against the intentions of the definition.

On the other hand, if the tally function is non-constant in active coercion, there exists an adversary for every scenario that can make $\Delta = 1$ meaning that the environment can always tell from the tally whether the participants followed the coercion or not. This is the trivial attack which was discussed in the background chapter. In the general case, the tally function can be defined as

$$f \text{ s.t. } \exists < x_1,...,x_n >, < x'_1,...,x'_n >: f(x_1,...,x_n) \neq f(x'_1,...,x'_n)$$

meaning that for different inputs the tally will be different. Now, the environment Z is the input provider and the coercer. As an input provider, Z gives the voters the inputs $< x_1,...,x_n >$ and as a coercer it gives the voters the inputs $< x'_1,...,x'_n >$. Now, in ideal coercion the voters send $< x'_1,...,x'_n >$ to the functionality and in ideal deception they send $< x_1,...,x_n >$ to the functionality. The functionality sends the results of the tally function to the environment when the result of the election is announced. Now as these two sets of inputs will tally to different results, the environment can always tell whether the voters followed the coercion or not. Therefore, the value Δ becomes 1. This is true for any non-constant tally function without even knowing more details about the system, meaning that this does not depend on the simulator, the protocol, or the deception strategy. As discussed this is the trivial attack and usually this is not captured in definitions. However, in this definition, this will be captured as shown above.

We can further illustrate this by using the example protocol defined in section 2.2.1. We consider a situation where there are two voters being coerced and the environment provides the first voter with the input to vote for candidate α_1 and the coercion to vote for candidate α_2 and the second voter with the input and coercion to vote for candidate α_2. The deception strategy here is to vote for the input rather the coercion. Now, if the first voter follows the coercion, there will be no vote for candidate α_1 when the tally is announced, but if they evade the coercion there will be a vote for candidate α_1. Therefore, the adversary will always be able to tell from the result of the tally whether the first voter followed the coercion or not. Thus, the environment can distinguish between the adversary and the simulator and the advantage of the environment in being able to distinguish the ideal deception from the ideal coercion is always 1.

3.3 Receipt-Freeness and the Example Protocol

In the last section, we saw that the meaning of Δ is unclear in a situation with active coercion. As the paper [2] also discusses receipt-freeness, we will see how the Δ behaves in such a situation. We will do this by using the example protocol from section 2.2.1. We have already defined the strategy in the protocol to be the following: the voter must flip the ballot with some probability d and send this ballot to the adversary with the code they used to vote. Alternatively, we also discussed the option that the voter could just flip the ballot in a way that confirms their story to the adversary. In other words, this would mean that a deceiving voter would always flip ($d = 1$) and coerced voter would never flip ($d = 0$).

We can construct the following maths to find a value for Δ in this receipt-free situation where the voters will always flip their ballot with probability d, no matter if they are following the coercion or attempting to deceive the adversary. For convenience, we indicate that $u = 0$ if we do not flip the ballot and $u = 1$ if we do flip the ballot. First, we can write the Δ as follows based on the definition of the UC framework introduced in the section 2.2.2

$$\Delta = \frac{1}{2}\sum_{u\in 0,1}\left|Pr[EXEC_{\phi,dum_J,\mathcal{S},\mathcal{Z}}(1^\lambda)=1]-Pr[EXEC_{\phi,DI_J,\mathcal{S},\mathcal{Z}}(1^\lambda)=1]\right| \tag{3}$$

This means that Δ is the sum of the advantages in the two situations where in the first we do not flip the ballot and in the second we do. For the first probability in the equation (3) we can construct the following

$$\begin{aligned}
&\sum_{u\in 0,1} Pr\left[EXEC_{\phi,dum_J,\mathcal{S},\mathcal{Z}}(1^\lambda)=1\right]\\
&\quad = Pr\left[u=0 \textit{ in } EXEC_{\phi,dum_J,\mathcal{S},\mathcal{Z}}(1^\lambda)\right] Pr\left[EXEC_{\phi,dum_J,\mathcal{S},\mathcal{Z}}(1^\lambda)=1|u=0\right]\\
&\qquad + Pr\left[u=1 \textit{ in } EXEC_{\phi,dum_J,\mathcal{S},\mathcal{Z}}(1^\lambda)\right] Pr\left[EXEC_{\phi,dum_J,\mathcal{S},\mathcal{Z}}(1^\lambda)=1|u=1\right]
\end{aligned} \tag{4}$$

Now since the probability that the ballot is flipped in this ideal coercion scenario is d, we can rewrite this as

$$\begin{aligned}\sum_{u\in 0,1} Pr\left[EXEC_{\phi,dum_J,\mathcal{S},\mathcal{Z}}(1^\lambda)=1\right]&\\ =&(1-d)Pr\left[EXEC_{\phi,dum_J,\mathcal{S},\mathcal{Z}}(1^\lambda)=1|u=0\right]\\ &+dPr\left[EXEC_{\phi,dum_J,\mathcal{S},\mathcal{Z}}(1^\lambda)=1|u=1\right]\end{aligned} \tag{5}$$

We can indicate the two probabilities left as q and p. Now we have that

$$\sum_{u\in 0,1} Pr\left[EXEC_{\phi,dum_J,\mathcal{S},\mathcal{Z}}(1^\lambda)=1\right]=(1-d)q+dp \tag{6}$$

The second part of the equation in (3), is nearly identical with (4), as we have that

$$\begin{aligned}\sum_{u\in 0,1} Pr\left[EXEC_{\phi,DI_J,\mathcal{S},\mathcal{Z}}(1^\lambda)=1\right]&\\ =Pr\left[u=0\ in\ EXEC_{\phi,DI_J,\mathcal{S},\mathcal{Z}}(1^\lambda)\right]&Pr\left[EXEC_{\phi,DI_J,\mathcal{S},\mathcal{Z}}(1^\lambda)=1|u=0\right]\\ +Pr\left[u=1\ in\ EXEC_{\phi,DI_J,\mathcal{S},\mathcal{Z}}(1^\lambda)\right]&Pr\left[EXEC_{\phi,DI_J,\mathcal{S},\mathcal{Z}}(1^\lambda)=1|u=1\right]\end{aligned} \tag{7}$$

Now again the ballot is flipped with probability d and the equation becomes

$$\begin{aligned}\sum_{u\in 0,1} Pr\left[EXEC_{\phi,DI_J,\mathcal{S},\mathcal{Z}}(1^\lambda)=1\right]&\\ =&(1-d)Pr\left[EXEC_{\phi,DI_J,\mathcal{S},\mathcal{Z}}(1^\lambda)=1|u=0\right]\\ &+dPr\left[EXEC_{\phi,DI_J,\mathcal{S},\mathcal{Z}}(1^\lambda)=1|u=1\right]\end{aligned} \tag{8}$$

Due to the simulator being the same for both scenarios, the probabilities for this equation are also q and p. This is because from the environment's point of view the two scenarios are equal, and it is equally likely to guess correctly given that the input was flipped or not. Thus, the equation becomes

$$\sum_{u\in 0,1} Pr\left[EXEC_{\phi,DI_J,\mathcal{S},\mathcal{Z}}(1^\lambda)=1\right]=(1-d)q+dp \tag{9}$$

Now putting the equations (6) and (9) together in (3), we get that

$$\Delta=\frac{1}{2}\left|(1-d)q+dp-((1-d)q+dp)\right|=0 \tag{10}$$

This means that no matter what the value of d is, we will always have $\Delta=0$ if the all the voters flip their ballot with this probability d. Now, as we discussed before $\Delta=0$ is supposed to be good. However, even if we chose $d=0$ or $d=1$ meaning that we would always flip or never flip, we would get $\Delta = 0$. Now, if we were to never flip then in the coercion scenario the voter could be verified to follow the coercion and the deceiving party could be caught in trying to evade the coercion. On the other hand, always flipping would result in a similar situation, where it would seem like the voter who actually followed the coercion tried to evade and the deceiving party followed the coercion. Thus, these situations would not actually lead to a good incoercible protocol execution even though $\Delta = 0$. We know that for example $d = 1/2$ would be okay since in that case all the voters might be telling the truth or lying and thus the situation is more difficult for the adversary. However, it is not simply enough to find one value for d that makes sense, because the definition requires the properties ”for every ideal deception strategy”.

3.4 Conclusion

The points made in the discussion highlight the issues with the meaning of Δ. We have shown that the value Δ can be made 0 by removing the deception strategy, thus resulting in a situation which clearly is not coercion-resistant. We have also looked at how the trivial attack is captured in the definition, meaning that an active coercer could find out whether the voters followed the coercion based on the result of the tally. We showed that this is true for all protocols no matter what the specifics of the simulator or the deception strategy are. We also discussed a receipt-free protocol that cannot be properly analysed using this definition due to the unclear meaning and definition of Δ. These examples indicate that the intended meaning of Δ and the ideal values for it do not translate meaningfully to the actual protocols. They also show that the meaning of Δ needs to be clarified before the definition in [2] can be used properly to analyse the incoercibility of voting protocols.

Besides needing clarification on the meaning of Δ, the definition also should not capture the trivial attack. Usually different coercion-resistant papers either state that it is not included or somehow otherwise include it in the definition. This is because as shown it is not possible for the protocol to be coercion-resistant if the adversary can determine whether they followed the coercion just by looking at the result of the election.

Another way to attempt to fix the problem with the definition, is to look at limiting the scope of it. As shown in this chapter even bad deception strategies are included in the definition, as it requires ”for every ideal deception strategy”. This is likely due to the fact that it is very difficult to formalise what is a good deception strategy. A good deception strategy would in general need to allow the voter to vote for their choice of a candidate rather than the following the coercion. This would remove the option of choosing a deception strategy that is equal to the coercion. Additionally, a good deception strategy would need hide from the adversary that the voter did not follow the coercion. This would remove the option of flipping the ballot in a way that still makes it noticeable from the perspective of the adversary. Even though stating such requirements for a good deception strategy is fairly easy, formalising them is difficult. A formal definition would need to be applicable to a large range of different protocols and deception strategies. Due to all of this, it is unclear how to best fix the definition introduced by [2].

Chapter 4

Blockchain-based protocol

An attempt at creating a coercion-resistant protocol was made by Spadafora, Longo, and Sala in [34]. The paper defines a coercion-resistant blockchain-based e-voting protocol that has receipts. We will focus on the two-candidate version of the protocol but the paper also discusses the possibility to generalise it to multiple candidates. The paper claims that the protocol is coercion- and vote-selling -resistant. However, in this chapter we will show that the voter can sell their vote to the adversary with a proof that it is correct. This means that the protocol is not vote-selling resistant which in turn means that the protocol is not coercion-resistant since the adversary can force everyone to provide such proof of their credentials or of the way they voted.

4.1 Attack Overview

Firstly, let us introduce the protocol. The participants of the protocol are the voters $V = \{v_1, ..., v_n\}$, the candidates C_1 and C_2, and the trusted authorities T_1 and T_2. The protocol consists of four stages: setup, registrar, voting, and tallying phases. In the setup stage, each of the authorities generate different random values that are required for the running of the protocol. These values are only known to the respective authorities. In the registrar stage, the authorities create a fake and a valid voting token and give the voter proofs that these tokens are correct. All of this is done via a private and untappable channel. In the voting phase, the voter must spend the two tokens together and they have to go to distinct candidates. The voter receives a receipt of spending the tokens. It is assumed that each candidate receives at least one legitimate vote to avoid the trivial attack. In the tallying phase, the votes are processed and the number of valid and fake votes received by each candidate is revealed. The authorities also publish proofs for these results.

We will show that the voter can sell their vote with the proofs provided to them in the registrar stage. These proofs are interactive zero-knowledge proofs (ZKPs) which are secure against an adversary that is trying to fool others by pretending to be the creator of the proofs and to know the secrets. However, in this scenario the voter is not trying to fool the adversary, instead the voter is just trying to show that the information he shares with the adversary is correct. Thus, the voter can just provide the adversary with all of the information and transcripts he has. Due to the construction of ZKPs used, these transcripts are enough for the adversary to verify that the information provided is correct. It also is not possible for the voter to create fake proofs due to the way the ZKPs are constructed. In the end, this means that the attacker can force the coerced party to send over these transcripts as well as all the receipts and since the coerced party cannot fake these it means that they cannot evade the coercion. Therefore, the protocol is not vote-selling resistant nor coercion-resistant.

4.2 Zero-Knowledge Proofs

Zero-knowledge proofs (ZKPs) are an important part of the protocol definition in [34]. However, as the paper makes incorrect assumptions about the strength of the ZKPs used, it is possible to construct as adversary that breaks the coercion-resistance in the protocol. The paper claims that the transcripts of the ZKPs cannot be used to provide proof to a third party. However, in this section we will show that the ZKPs used allow more information to be shared by the voter to the adversary than is assumed in the paper. This section discusses the ZKPs used in the paper as well as introduces a way the voter can share these proofs with the adversary.

The paper defines its ZKPs as follows

"**Protocol 1.** *Let $\mathbb{G}$ be a cyclic group of prime order p, let u, $\bar{u}$ be generators of $\mathbb{G}$, and let z, $\bar{z} \in \mathbb{G}$, $\omega \in \mathbb{Z}_p$. The prover knows ω and wants to convince the verifier that:*

$$u^{\omega} = z \text{ and } \bar{u}^{\omega} = \bar{z},$$

without disclosing ω. The values of u, z, $\bar{u}$ and $\bar{z}$ are publicly known.

1. *The prover generates a random r and computes $t = u^r$ and $\bar{t} = \bar{u}^r$, then sends (t, $\bar{t}$) to the verifier.*
2. *The verifier computes a random $c \in \{0,1\}$ and sends it to the prover.*
3. *The prover creates a response $s = r + c \cdot \omega$ and sends s to the verifier.*
4. *The verifier checks that $u^s = z^c \cdot t$, $\bar{u}^s = \bar{z}^c \cdot \bar{t}$. If the check fails the proof fails and the protocols aborts.*
5. *The previous steps are repeated $\tau = poly(log_2(p))$ times, i.e. the number of repetitions is polynomial in the length of p (the security parameter).*" [34]

We will use the following short notation to indicate this zero-knowledge proof protocol: $ZKP(\omega, z, \bar{z}) = ZKP(\omega, u^{\omega}, \bar{u}^{\omega})$.

Now in this protocol in all the ZKPs used, the voter is the verifier and one of the authorities is the prover. This means that the voter will have all of the following information for each of the ZKPs they participate in: u, z, $\bar{u}$, $\bar{z}$, and t, $\bar{t}$, c, and s for each of the runs of a single ZKP. Due to this, the voter can reproduce a proof to the adversary, not to fool them that they know the secret but to prove that the information is correct. More formally after the verifier V has finished the interaction described in Protocol 1 with the prover P, he can perform the following Protocol 1b with another party A:

Protocol 1b. *Again the values of u, z, $\bar{u}$ and $\bar{z}$ are publicly known.*

1. *V sends $(t, \bar{t})$ to A.*
2. *V sends c to A.*
3. *V sends s to A.*
4. *A checks that $u^s = z^c \cdot t$, $\bar{u}^s = \bar{z}^c \cdot \bar{t}$. If this passes, then A will be convinced that $u^\omega = z$ and $\bar{u}^\omega = \bar{z}$.*
5. *The previous steps are repeated necessary number of times depending on the initial interaction between V and P.*

This interaction would not convince A that V knows the secret ω since in step 2 V provides A with the random value instead of A creating such a value. However, this interaction is enough to convince A that $u^\omega = z$ and $\bar{u}^\omega = \bar{z}$ which is what the initial interaction between V and P was used to show. Thus, such a transcript can be used by the voter to send verifiable receipts to the adversary. We will use the short notation $FZKP(\omega, z, \bar{z}) = FZKP(\omega, u^\omega, \bar{u}^\omega)$ for forged zero-knowledge proof to indicate the Protocol 1b.

Now in order to fake Protocol 1b the verifier V could state that c is always 0. This would means that A cannot actually verify z and $\bar{z}$ meaning that the voter could lie about the values of these. Such a proof would allow the voter to for example swap the places of the valid and fake tokens, thus being able to fake a vote for the opposite candidate. The specifics of this are in the appendix A since this is not a very strong evasion strategy against an adversary. We can easily construct an adversary that can spot such an evasion strategy. This is because such an evasion strategy would require faking, i.e. making the c equal 0 for all the repetitions of the ZKP, for two specific ZKPs. Thus, the adversary could easily notice this evasion strategy and we can construct the adversary to not accept ZKPs that have $c = 0$ for all repetitions in a single ZKP. It is fine to construct the adversary in this way since the voter would not initially be convinced by a proof with just $c = 0$ for all the repetitions. This is because when $c = 0$ the proof does not require the prover to know the secret. It would also be very unlikely for the two specific ZKPs to both have all c chosen randomly to equal 0, i.e. if we have 20 repetitions per ZKP we have the probability $(1/2)^{20}$ per ZKP that all the c are 0, thus for it to happen to both of the ZKPs we have the probability $(1/2)^{40}$. Another way to fake a ZKP would be for the voter to construct a ZKP that suitably matches with the initial ZKP, however this is not possible due to the discrete logarithm assumption.

4.3 Formal Attack

We will first discuss the formal definition of the protocol, we will focus on the registrar stage since the attack described will happen at that stage. In the setup stage, the authority T_1 will create α_1' and α_2' one for each of candidates, the pair x_i' and y_i' for each of the voters and k for valid v-token and λ for fake. Similarly, the authority T_2 will create α_1'' and α_2'' one for each of candidates and the pair x_i'' and y_i'' for each of the voters. The authorities T_1 and T_2 will also commit to these values in a safe way to ensure that

The registrar phase

1. The voter v_i creates a wallet in a safe and controlled way.
2. The voter v_i registers this wallet with the authorities T_1 and T_2.
3. T_1 creates the preliminary ballot: $\bar{b}_i = (g^{y'_i(x'_i+k)}, g^{y'_i(x'_i+\lambda)})$ and sends it to the voter v_i.
4. T_1 will now send v_i the values $g^{x'_i y'_i}$, $g^{y'_i k}$, and $g^{y'_i \lambda}$ with the following ZKPs to prove their correctness: $ZKP(x'_i, g^{x'_i}, g^{x'_i y'_i})$, $ZKP(y'_i, g^{y'_i}, g^{x'_i y'_i})$, $ZKP(k, g^k, g^{y'_i k})$, and $ZKP(\lambda, g^\lambda, g^{y'_i \lambda})$.
5. T_2 sends v_i the value $g^{x''_i y''_i}$ and has the interactions $ZKP(x''_i, g^{x''_i}, g^{x''_i y''_i})$ and $ZKP(y''_i, g^{y''_i}, g^{x''_i y''_i})$.
6. T_2 sends $g^{x''_i y''_i}$ to T_1. T_1 computes $(g^{x''_i y''_i})^{y'_i}$, sends it to v_i and completes the interaction $ZKP(y'_i, g^{y'_i}, (g^{x''_i y''_i})^{y'_i})$ with v_i
7. The voter v_i flips a random coin $c_i \in \{0,1\}$ and defines $\tilde{b}_i = \bar{b}_i$ if $c_i = 0$, $\tilde{b}_i = (\bar{b}_{i,2}, \bar{b}_{i,1})$ if $c_i = 1$. Then v_i sends $\tilde{b}_i$ and $(g^{x''_i y''_i})^{y'_i}$ to T_2.
8. T_2 confirms with T_1 that these values are correct and then computes $(\tilde{b}_{i,l})^{y''_i}$ for $l \in \{0,1\}$. T_2 sends this to v_i and completes $ZKP(y''_i, g^{y''_i}, (\tilde{b}_{i,l})^{y''_i})$ with v_i.
9. Finally, T_2 flips a random coin $c'_i \in \{0,1\}$ and sends the final ballot b_i to the voter v_i. This will be $b_i = ((\tilde{b}_{i,1})^{y''_i} \cdot (g^{x''_i y''_i})^{y'_i}, (\tilde{b}_{i,2})^{y''_i} \cdot (g^{x''_i y''_i})^{y'_i})$ if $c'_i = 0$ and $b_i = ((\tilde{b}_{i,2})^{y''_i} \cdot (g^{x''_i y''_i})^{y'_i}, (\tilde{b}_{i,1})^{y''_i} \cdot (g^{x''_i y''_i})^{y'_i})$ if $c'_i = 1$. The voter v_i knows which token is valid due to the proofs and intermediate values.

Figure 4.1: The registrar phase for the voting protocol in [34].

the values stay secret but the authorities cannot forge different values for them. This commitment means that the values $g^{\alpha'_1}$, $g^{\alpha'_2}$, $g^{\alpha''_1}$, $g^{\alpha''_2}$, g^k, and g^λ as well as the pairs (v_i, $g^{x'_i}$), (v_i, $g^{y'_i}$),(v_i, $g^{x''_i}$), and (v_i, $g^{y''_i}$) for every voter v_i are made public. The exact details of the setup stage and the commitment scheme used are not necessary as they do not impact the attack in any way.

The formal definition of the registrar stage is shown in Figure 4.1. Since this is the stage where the attack can be conducted, it is shown in detail. First, the voter will create a wallet and then register it with the authorities. This is required since the protocol is run in a blockchain environment, however the details of it are not necessary in the attack we introduce. After this, the authority T_1 creates the preliminary ballot which is made using the values for the fake and valid token k and λ as well as the values (x'_i, y'_i) which are specific to the voter v_i. T_1 then sends this ballot to the voter v_i. Following this there are some interactions between the different parties with ZKPs to prove the correctness of these steps. In the end, the voter has the final ballot and the authorities can not distinguish the valid and fake token in the ballot.

Now the protocol the voter can follow to sell their vote is described in Figure 4.2. This shows that all the information sent to the voter is sent to the adversary with the corresponding FZKPs. The adversary is able to verify all the information shared with them by the voter, thus they can be convinced that the ballot is the real one. The values

The vote-selling protocol on the registrar phase

1. Upon receiving the preliminary ballot from T_1, the voter v_i sends it to the adversary A.
2. After finishing the initial ZKP interactions with the authority T_1, the voter v_i will do the following with the adversary A: $FZKP(x'_i, g^{x'_i}, g^{x'_i y'_i})$, $FZKP(y'_i, g^{y'_i}, g^{x'_i y'_i})$, $FZKP(k, g^k, g^{y'_i k})$, and $FZKP(\lambda, g^{\lambda}, g^{y'_i \lambda})$.
3. Upon receiving $g^{x''_i y''_i}$ from T_2, the voter v_i sends it to the adversary A. The voter v_i will also have the interactions $FZKP(x''_i, g^{x''_i}, g^{x''_i y''_i})$ and $FZKP(y''_i, g^{y''_i}, g^{x''_i y''_i})$ with A after finishing the corresponding interactions with T_2.
4. Upon receiving $(g^{x''_i y''_i})^{y'_i}$ from T_1, the voter v_i sends it to the adversary A. After completing the ZKP interaction with T_1, the voter v_i completes $FZKP(y'_i, g^{y'_i}, (g^{x''_i y''_i})^{y'_i})$ with A.
5. The voter v_i sends $\tilde{b}_i$ to the adversary A. Since A received the initial ballot, he can confirm that this is a correct permutation of it and he will know which token is valid and which is not.
6. Upon receiving $\tilde{b}_{i,l})^{y''_i}$ from T_2, the voter v_i sends it to A and completes $FZKP(y''_i, g^{y''_i}, (\tilde{b}_{i,l})^{y''_i})$ with A.
7. Upon receiving the final ballot from T_2, the voter v_i sends it to the adversary A. Now, since the adversary A knows everything the voter does, he will know which token is valid.

Figure 4.2: The attack against the voting protocol introduced in [34].

used in the proofs need to be unforgeable in order for the protocol to be safe against a malicious authority, which means that the voter cannot lie to the adversary.

The attack works by the voter specifically sending all of the information received from the authorities to the adversary and then repeating all the ZKP completed with the authorities as the respective FZKP with the adversary. The steps are shown in Figure 4.1 for the real protocol and in Figure 4.2 for the attack. Because the first two steps in the voting protocol are only to do with the wallet, the steps in the two figures match so that the third step in the real protocol is followed by the first step in the attack protocol, fourth is followed by the second and so on. At the end of the two protocols the voter has the final ballot and the authorities cannot distinguish between the valid and fake tokens but the adversary can. Thus, the adversary can then vote on the behalf of the voter or verify how the voter voted using these tokens.

4.4 Fixes and Conclusion

There is a possible way in which the voter could try to lie to the adversary, which we already briefly discussed in the previous section. In this method, the voter is faking a ZKP in step two of the attack protocol. Specifically, instead of doing $FZKP(k, g^k, g^{y'_i k})$, and $FZKP(\lambda, g^{\lambda}, g^{y'_i \lambda})$ they would swap them and always have $c = 0$. Then, the voter would lie to the adversary after step 7 in the real protocol, in step 5 in the attack protocol about the value of the coin flip c_i and $\tilde{b}_i$. These two steps would result in the the adversary believing that the initial ballot was the opposite of what it really was. However, like we already stated, the adversary constructed requires that in the FZKP the c are not 0 for all the repetitions. The detailed description is included in the Appendix A.

Now, since there is not a working evasion strategy the voter could execute, this means that the voter can verifiably sell their vote to the adversary. In turn, this results in the protocol not being coercion-resistant as the adversary can require all the coerced parties to provide this verifiable proof of their ballot. As this problem with the protocol is in the key contributions of the paper [34], it would not be possible to fix it without changing the whole idea tried to convey.

Chapter 5

Minimal Anti-Collusion Infrastructure

Vitalik Buterin introduces a voting protocol called minimal anti-collusion infrastructure (MACI) in [13]. This protocol attempts to solve issues related to collusion which is very similar to coercion and receipt-freeness. However, the contribution offered by the protocol is unclear since it does not seem to improve the known coercion-resistant protocols in any way. In this chapter, we will first discuss collusion and coercion, then the MACI protocol introduced in [13] and finally a way we can construct an adversary that breaks coercion-resistance in the protocol.

5.1 Collusion and Coercion

Collusion as described by Vitalik Buterin on his website in [12] comes down to a situation where a user wants to prove that they took some action in order to collude with other users or sell their vote in order to impact the result of the election. The collusion between voters is not a concern in an election if there is no coercion involved. This is since without coercion the participants of the collusion can only change their own vote which they can use how they prefer anyway. For example, being part of a political party can be seen as a collusion, but this is acceptable and often done by voters. However, having a verifiable receipt the voters can use to prove how they voted is not something that can be allowed in elections due to it making vote-selling and coercion possible.

Thus, when coercion is involved in collusion, it becomes very similar to coercion as we have discussed it and specifically receipt-freeness. It receipt-freeness, the voters are required to provide a receipt to their coercer whereas in collusion, the voters wish to or are asked to have a receipt they can provide to other participants of the collusion or the person they want to sell their vote to. Thus, in order to fight collusion and coercion from a receipt asking adversary, the measures are very similar, the protocol should not allow the voters to create or have a verifiable proof of how they voted in the election. Meaning that in the case where a receipt is provided it should be easy to forge a receipt and thus, the coercer cannot verify and be certain that the proof is valid.

5.2 Protocol

The MACI protocol consists of two stages the setup stage and the execution stage. During the setup stage the different participants are initiated and the protocol state is prepared for the execution stage. In the execution stage, the voters are allowed to cast their votes. In the end of the execution stage, the election result is tallied and published.

The protocol has the following parties: n voters each of which has a private and a public key, registry R, operator O, and mechanism M. The registry R acts as a public bulletin board since it keeps track of the list of eligible voters as well as the messages posted on the board. In the end of the election, the operator O processes the actions performed by the voters and the mechanism M is used to tally the result of the election.

5.2.1 The Setup Stage

First, we have the setup stage as shown in Figure 5.1. This stage starts with every voter registering with the registry R. After this, the operator O, mechanism M, and the timings are initialised. Meaning that at the end of the setup stage the participants have the following states. Each of the voters $i \in 1...n$ has a private key k_i and a public key K_i. The registry contains a list with the public keys of the eligible voters, i.e. $K_1, ..., K_n$. The operator has an initial internal state $S_{start} = \{i : (key = K_i, action = \emptyset)\}$ for $i \in 1...n$ which is a list of keys and actions for each of the voters. Since no actions have been taken yet, the action parameter equals an empty set for each of the voters. The mechanism is initialised as the tally function which depends on the requirements for the elections. In a simple case it takes all the actions given to it by the operator and then outputs the action that has been taken most frequently, i.e. the candidate or decision that received the largest amount of votes. Lastly, the start and end times for the execution stage are initialised and set.

<u>The setup stage</u>

- Each eligible voter sends their public key to the registry. In the beginning of the execution stage, the registry contains a list of public keys $K_1, ..., K_n$.
- The operator has a private key k_w and a public key K_w. The operator has the initial internal state $S_{start} = \{i : (key = K_i, action = \emptyset)\}$ for $i \in 1...n$.
- The mechanism is defined as a function $action^n \rightarrow Outputs$. The input is the actions taken by the participants and the output is the action that appear the most, meaning the winner of the election.
- The times T_{start} and T_{end} are initialised. These define the timing for the execution stage.

Figure 5.1: The setup stage for the MACI protocol.

5.2.2 The Execution Stage

Next we have the execution stage as shown in Figure 5.2. The execution stage starts by making sure that the timings are correct before moving onto accepting messages

The execution stage

- Between times T_{start} and T_{end} anyone can publish messages into an on-chain registry.
- The message can be of two types:
  1. Actions of form $enc(msg = (i, sign(msg = action, key = k_i)), pubkey = K_w)$, where k_i is the user's private key and i is the user's index in R.
  2. Key changes of form $enc(msg = (i, sign(msg = NewK_i, key = k_i)), pubkey = K_w)$, where $NewK_i$ is the user's desired new public key, k_i is the user's private key and i is the user's index in R.
- The operator will process these messages as follows:
  1. Decrypt the message. If the decryption does not result in a message of one the defined types, the operator will skip over the message.
  2. Verify the internal signature using $state[i].key$.
  3. If the message is an action, set $state[i].action = action$. If the message is a new-key, set $state[i].key = NewK_i$.
- After time T_{end} the operator must publish $M(state[1].action...state[n].action)$ and the ZK-SNARK proving that this is the correct result of processing the messages.

Figure 5.2: The execution stage for the MACI protocol.

from anyone. Now, the two accepted types of messages are an action and a key change message. In an election, the action messages would be votes for either a candidate or a decision depending on the type of election in question. These messages are encrypted using the operators public key and signed using the voters private key. The key change messages allow the voters to change their key for whatever reason. These messages differ from the action messages in that they include the new public key rather than an action to be taken.

The operator will process the messages in the order that they have been published. First, the operator will decrypt the message and try to match it to one of the message types, otherwise it will skip over the message. Next, the operator will verify the signature used in the message using the key stored in the operator's internal state. Finally, the operator will update its internal state S according to the message, i.e. for an action message it updates the action taken by the voter and for the key change message it updates the voter's key. Now something to consider, is that it is unclear from the protocol description in [13] whether this processing is done after the end of the execution phase or during the execution phase. This has different implications on security as processing during the execution stage might allow an adversary to receive additional information from the number or type of messages that have been disregarded and stored, similar to how JCJ was attacked in [20]. Therefore, we will use the interpretation that the message processing happens after the end of the execution stage.

Finally, the operator must publish the result of the election with a proof of its correctness. After the processing there will be a list of actions representing the votes of the eligible voters, the result of the election is then found using the mechanism M on this list. To prove the correctness of this result, the operator will have to provide a zero-knowledge proof, specifically a ZK-SNARK, to show that all the processing and tallying was done accurately. The argument for the collusion-resistance of the protocol is that the voter could point to their actions in the registry and provide the adversary with a zero-knowledge proof that the action corresponds to what was required. However, as the voter could have changed their key before doing the mentioned action, the adversary cannot be certain that the given vote is valid.

Problem with MACI

- If the voter with the index i has a way to add the first two actions to the registry, then he can add the following two messages
  1. Vote according to the coercion: $enc(msg = (i, sign(msg = action, key = k_i)), pubkey = K_w)$
  2. Invalidate their key: $enc(msg = (i, sign(msg = 0x0, key = k_i)), pubkey = K_w)$
- Now the voter can point to these actions on the registry and the adversary can verify their vote and be certain that they will not be able to change it afterwards.

Figure 5.3: The problem in the MACI protocol introduced in [13].

5.3 Correction to Initial Definition

After the initial description of the protocol, an attack described in Figure 5.3 was introduced by another writer in [13]. The attack is based on the idea that if someone is able to publish the first vote on the public chain, then this first voter could sell their vote. However, since they could just revote afterwards, they would also have to make their key invalid, by changing it to $0x0$. This could also be extended to other early voters as long as all of them give their receipts to the adversary since the adversary would have access to all the messages on the chain. For example, if the adversary receives the first four messages from two voters in this way, then he can verify these two votes to be as shown by the receipts.

In order to avoid this attack, a correction was offered in [13]. This correction requires each of the voters to change their key before they are allowed vote. This means that it is more difficult for any voter to be able to send all those first 3 messages, i.e. key change, vote, and key invalidating, without honest voters appearing in between. Since all the voters can change their key at any point as long as it is the first message they post on the chain, this fix is different from requiring all the voters to change their key before anyone is allowed to vote, which would just move the attack to a later point in the registry.

5.4 Attack Against the protocol

We will use a similar attack to what was used in [20] to attack the JCJ protocol. After the fix discussed in the last section, each voter is required to send at least two messages to the registry. For an honest voter the total amount of messages sent will be exactly two as they have no reason to send more messages, i.e. they do not have a reason to revote or change their key after the initial key change. Of course an honest voter could change their mind and want to revote due to that but in general most voters will not do this.

For this attack, we will assume that the adversary knows the number of honest voters n and the adversary knows that the number of messages posted by these voters should be $2n$ as none of the voters are revoting. Thus, if the adversary coerces the voter v_i, then he will expect the total number of published messages to be $2n+2$ if the voter follows the coercion. If the total number of message is not $2n+2$, then the adversary will detect that the voter has not followed the coercion. The advantage the adversary will gain in such a scenario is more than is allowed and thus this will break coercion-resistance in the protocol.

Additionally, since we already saw before that the voter can provide the adversary with receipts, i.e. pointing to a message on the chain and proving this with a zero-knowledge proof. This means that after the adversary sees that there are additional messages on the registry, he can request the voter to send him these as well. Now, the voter could attempt to lie and claim that these are not theirs. However, as discussed the honest and non-coerced voters do not really have a reason to revote and thus the adversary would not believe that the additional messages are not the coerced voter's. All of this, means that coercion-resistance and receipt-freeness in the protocol are broken.

5.5 Fixes and Conclusion

One way to attempt to fix the protocol, could be the addition of a random number of fake votes or revotes to the registry. However, in general the voters cannot be trusted to do this, since a random number of votes posted at random times is required and not something an average voter will be able to properly complete. Therefore, this would need to be done by the authorities. However, modelling such behaviour is quite difficult. An attempt in such modelling was made by VoteAgain [30] where the authorities added dummy ballots to hide the revoting done by the voters. However, another paper [24] showed that this resulted in VoteAgain needing a trusted authority for all of the security properties. As discussed in the beginning, this is not wanted as it means that the protocol is not improving a simple voting protocol in a meaningful way.

Another way to fix the protocol would be to limit the voters' access to their randomness and keys. This could be done by having a trusted hardware for each voter which stores these values and the voter cannot access them. This would prevent the voter from sharing their messages with the authority. However, as this would not allow key changing, it would totally change the idea tried to convey by the protocol MACI [13]. Thus, it would not really be possible to fix the protocol without completely changing it, similar to the result we got in the previous chapter.

Chapter 6

Other Work

There are a lot of protocols that try to improve the existing class of coercion-resistant e-voting protocols. However, as we have shown this task is not an easy one and there are often some problems that get overlooked. We already showed attacks against two introduced protocols [34] and MACI from [13]. In this chapter, a few other protocols and points that were looked at throughout working on the project, will be discussed more informally.

6.1 Other Protocols

We already mentioned VoteAgain [30] in the previous chapter. It aims to improve the cost of tallying when compared with JCJ, however it was proved to rely on a single trusted authority by [24]. This means that the protocol has a lot stronger trust assumptions than JCJ even if the cost is better in the cleaning stage. Additionally, as discussed due to these trust assumptions it does not meaningfully improve a simple coercion-resistant protocol. Another paper that was looked at is E-cclesia [3]. However, it is not even receipt-free and as it is a self-tallying protocol, trying to make it coercion-resistant is even more difficult than with distributed trust assumptions.

6.2 Other Problems

The two protocols MACI [13] and the blockchain-based protocol [34] were compared with JCJ [26] and JCJ* [20] in order to see whether they were performing better than the JCJ protocol in any aspect. While this comparison is not necessary after the attacks provided against the protocols in the previous chapters, it will still be briefly discussed here. Specifically, this is not necessary since the attacks introduced break or remove the need for the main contributions in each paper: key changing in MACI and providing a receipt in the blockchain-based protocol.

While JCJ and JCJ* can have distributed trust with a threshold for the number of corrupted authorities, MACI needs a single trusted third party in order to satisfy the security properties. On the other hand, the protocol in [34] claims to be decentralised, but in reality has two trusted authorities where, according to their proofs, at least one of them needs to be honest. Additionally, this protocol requires a private, untappable channel between the voter and each of the authorities, whereas JCJ only requires an anonymous channel at some point in the election. According to our fix, MACI would also need a trusted hardware which is not something required by JCJ.

6.3 JCJ* Underspecification

We have discussed JCJ* already and while the protocol is very thoroughly described and proved, it lacks in the description. Mainly the deception strategy used in the protocol in a receipt-free situation is underspecified. The evasion strategy discussed in JCJ* is that the voter casts one vote for the desired party and then provides a fake credential to the adversary. The paper mentions that ”a coercer may ask the voter to cast some specific vote or to perform some specific computations, but this is not considered in the definition as the adversary might as well do it herself”[20]. However, since we want to discuss receipt-freeness, we have to specify a different, more restricted adversary as well as an appropriate evasion strategy. These are included in the definition, they are just not properly specified so we will do that here. This is to show that there is a large variety of adversaries and the deception strategies change a lot in different situations.

The receipt-free adversary we will discuss is weaker in that they cannot request the credential and use this to vote but rather they will ask the voter to vote for a certain candidate and then send all the receipts to the adversary. The adversary then tries to verify how the voter voted based on these receipts. In JCJ*, in order to avoid such an adversary, the voter can create a fake credential and vote with this. Thus, being able to send these receipts to the adversary. Alternatively, the voter could first vote according to the coercion and then vote again according to their own wishes. Again, the voter could send the receipts to the adversary. Now, as we know that the protocol is coercion-resistant, we know that the adversary cannot distinguish between a real and a fake credentials, thus the adversary is not able to verify the receipts sent by the voter in any way.

Now, we know that revoting and addition of fake ballot is difficult to model and create, as described in the previous chapter. The protocol JCJ* in [20] mathematically models this behaviour by including in their vote distribution the factor of revoting and addition of fake votes, whether by the voters or the authorities. The paper does not further discuss how this can be done in a secure way. However, as the protocol hides the reasons why any single vote is dropped from the final tally, this might just be done by the multiple authorities adding random ballots at random times. Thus, the protocol would avoid the issues found by [24] in the VoteAgain protocol [30], while also hiding the evasion strategies from the adversary. However, these details should be specified in the paper [20] in order for it to have a meaningful evasion strategy against a receipt-free adversary.

Chapter 7

Conclusions and Future Work

In this report, we have shown problems in a definition of coercion-resistance and attacks against two electronic voting protocols. The problems in the definition of coercion-resistance defined in [2] were related to the value Δ that was not clearly specified. We discussed how these problems mean that the definition cannot be used in its current state to analyse the incoercibility of e-voting protocols. The first attack we presented was against the protocol defined in [34], the attack showed that the paper assumed the ZKPs to be stronger than they actually were. The second attack presented was against the MACI protocol defined in [13]. The attack showed that the adversary can gain a non-negligible advantage and even confirm whether the voter followed the coercion in some cases. As discussed, these attacks break the coercion-resistance in the protocols and result in the contributions of the protocol definitions to be minimised.

The report provides insight to the problems with formalising and actualising coercion-resistance. The omissions and attacks discussed show that sufficiently considering the relevant aspects of coercion-resistance is very difficult and more work is needed before coercion-resistance can be a property of a scalable e-voting protocol. Scalability is not currently possible since the current coercion-resistant protocols often rely on similar methods as JCJ which has a large overhead, i.e. quadratic in the number of voters. Additionally, in order to use electronic voting protocols in large elections, coercion-resistance is a necessary property since coercion is a real and serious problem in such elections.

Bibliography

[1] Ben Adida. Helios: Web-based open-audit voting. In *USENIX security symposium*, volume 17, pages 335–348, 2008.

[2] Joël Alwen, Rafail Ostrovsky, Hong-Sheng Zhou, and Vassilis Zikas. Incoercible Multi-party Computation and Universally Composable Receipt-Free Voting. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 763–780, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[3] Myrto Arapinis, Nikolaos Lamprou, Lenka Mareková, Thomas Zacharias, Léo Ackermann, and Pavlos Georgiou. E-cclesia: Universally Composable Self-Tallying Elections. Cryptology ePrint Archive, Paper 2020/513, 2020.

[4] 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*, pages 52–62, 1986.

[5] Josh Daniel Cohen Benaloh. *Verifiable secret-ballot elections*. Yale University, 1987.

[6] D. Bernhard, O. Pereira, and B. Warinschi. On necessary and sufficient conditions for private ballot submission. Cryptology ePrint Archive, Paper 2012/236, 2012.

[7] David Bernhard, Véronique Cortier, Olivier Pereira, and Bogdan Warinschi. Measuring Vote Privacy, Revisited. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, page 941–952, New York, NY, USA, 2012. Association for Computing Machinery.

[8] David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. SoK: A Comprehensive Analysis of Game-Based Ballot Privacy Definitions. In *2015 IEEE Symposium on Security and Privacy*, pages 499–516, 2015.

[9] 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*, pages 626–643. Springer, 2012.

[10] David Bernhard and Ben Smyth. Ballot privacy and ballot independence coincide. In *Proceedings of the 18th European Symposium on Research in Computer Security (ESORICS'13)*, 2013.

[11] Jens-Matthias Bohli, Jörn Müller-Quade, and Stefan Röhrich. Bingo voting: Secure and coercion-free voting using a trusted random number generator. In *E-Voting and Identity: First International Conference, VOTE-ID 2007, Bochum, Germany, October 4-5, 2007, Revised Selected Papers 1*, pages 111–124. Springer, 2007.

[12] Vitalik Buterin. On Collusion. Vitalik Buterin's website (2019).

[13] Vitalik Buterin. Minimal anti-collusion infrastructure. *Ethereum Research*, 2019.

[14] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145, 2001.

[15] Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. BeleniosRF: A Non-interactive Receipt-Free Electronic Voting Scheme. Cryptology ePrint Archive, Paper 2015/629, 2015.

[16] 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)*, pages 354–368, 2008.

[17] Lillie Coney, Joseph L Hall, Poorvi L Vora, and David Wagner. Towards a privacy measurement criterion for voting systems. In *Proceedings of the 2005 national conference on Digital government research*, pages 287–288. Citeseer, 2005.

[18] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Election verifiability for helios under weaker trust assumptions. In Mirosław Kutyłowski and Jaideep Vaidya, editors, *Computer Security - ESORICS 2014*, pages 327–344, Cham, 2014. Springer International Publishing.

[19] Véronique Cortier, David Galindo, Ralf Küsters, Johannes Müller, and Tomasz Truderung. SoK: Verifiability Notions for E-Voting Protocols. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 779–798, 2016.

[20] Véronique Cortier, Pierrick Gaudry, and Quentin Yang. Is the JCJ voting system really coercion-resistant? Cryptology ePrint Archive, Paper 2022/430, 2022.

[21] Olivier de Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Simulation-Based Analysis of E2E Voting Systems. In Ammar Alkassar and Melanie Volkamer, editors, *E-Voting and Identity*, pages 137–149, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[22] Gurchetan S Grewal, Mark D Ryan, Sergiu Bursuc, and Peter YA Ryan. Caveat coercitor: Coercion-evidence in electronic voting. In *2013 IEEE Symposium on Security and Privacy*, pages 367–381. IEEE, 2013.

[23] Jens Groth. Evaluating Security of Voting Schemes in the Universal Composability Framework. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *Applied*

Cryptography and Network Security, pages 46–60, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[24] Thomas Haines and Johannes Müller. How not to VoteAgain: Pitfalls of Scalable Coercion-Resistant E-Voting. *Cryptology ePrint Archive*, 2020.

[25] Thomas Haines and Ben Smyth. Surveying definitions of coercion resistance. *Cryptology ePrint Archive*, 2019.

[26] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. Cryptology ePrint Archive, Paper 2002/165, 2002.

[27] Aggelos Kiayias and Moti Yung. Self-tallying elections and perfect ballot secrecy. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography*, pages 141–158, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[28] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. Cryptology ePrint Archive, Paper 2015/346, 2015.

[29] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. A game-based definition of coercion-resistance and its applications. In *2010 23rd IEEE Computer Security Foundations Symposium*, pages 122–136, 2010.

[30] Wouter Lueks, Iñigo Querejeta-Azurmendi, and Carmela Troncoso. VoteAgain: A scalable coercion-resistant voting system. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1553–1570. USENIX Association, August 2020.

[31] Tal Moran and Moni Naor. Split-Ballot Voting: Everlasting Privacy with Distributed Trust. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, page 246–255, New York, NY, USA, 2007. Association for Computing Machinery.

[32] CD Mote Jr. Report of the national workshop on internet voting: issues and research agenda. In *The Internet Policy Institute*, 2001.

[33] Ben Smyth, Steven Frink, and Michael R. Clarkson. Election verifiability: Cryptographic definitions and an analysis of helios, helios-c, and jcj. Cryptology ePrint Archive, Paper 2015/233, 2015.

[34] Chiara Spadafora, Riccardo Longo, and Massimiliano Sala. Coercion-resistant blockchain-based e-voting protocol. *Cryptology ePrint Archive*, 2020.

[35] Alan Szepieniec and Bart Preneel. New Techniques for Electronic Voting. *USENIX Journal of Election Technology and Systems (JETS)*, August 2015.

[36] Pavel Tarasov and Hitesh Tewari. The future of e-voting. *IADIS International Journal on Computer Science & Information Systems*, 12(2), 2017.

Appendix A

Evasion Strategy in the Blockchain-based protocol

An evasion strategy was discussed in the chapter 4. This evasion strategy is not a valid one due to the reasons discussed in the chapter but the details for the evasion strategy are included here.

The evasion strategy starts by the voter v_i swapping the places of the valid and fake token in the ballot, i.e. changing the ballot from $\bar{b}_i = (g^{y'_i(x'_i+k)}, g^{y'_i(x'_i+\lambda)})$ to $\bar{b}_i = (g^{y'_i(x'_i+\lambda)}, g^{y'_i(x'_i+k)})$. Now, since the adversary knows g^k and g^λ from the initial commitments published by the authority T_1, the voter v_i needs to lie about the ZKPs that include these, i.e. $ZKP(k, g^k, g^{y'_i k})$ and $ZKP(\lambda, g^\lambda, g^{y'_i \lambda})$. Lying about these ZKPs means that the voter claims that the random value c is 0 for all the repetitions of each of the proofs and thus they can lie about the value of the secret since it never gets checked. This was discussed in section 4.2.

Now after this the voter v_i can move on with the real and the attack protocols as described until step 7 in Figure 4.1 and step 5 in Figure 4.2. In this step, the voter v_i needs to lie about the value of the coin flip c_i to the adversary and send over the respective ballot. This is because the voter flipped the initial ballot and the adversary will find out the real final ballot from the protocol. Thus, the final ballot the voter sends must equal to that one, meaning that the ballot must be flipped again if the real ballot is not flipped at all and not flipped if the real ballot is flipped at this stage. The voter can then finish the protocols as defined. Since the adversary got all the receipts and proofs, they believe they have verified knowledge of which token in the final ballot is valid and which is fake. However, the ballot is the opposite to what the adversary believes since the voter swapped the places of the tokens in the proofs.

However, as already stated in the main paper, this is not a valid strategy due to the adversary being constructed in a way that does not accept a ZKP that has $c = 0$ for all the repetitions.