An Epistemic Approach to Coercion-Resistance for Electronic Voting Protocols

Ralf Küsters and Tomasz Truderung
University of Trier
Germany
Email: {kuesters,truderun}@uni-trier.de

***Abstract*—Coercion resistance is an important and one of the most intricate security requirements of electronic voting protocols. Several definitions of coercion resistance have been proposed in the literature, including definitions based on symbolic models. However, existing definitions in such models are rather restricted in their scope and quite complex.**

In this paper, we therefore propose a new definition of coercion resistance in a symbolic setting, based on an epistemic approach. Our definition is relatively simple and intuitive. It allows for a fine-grained formulation of coercion resistance and can be stated independently of a specific, symbolic protocol and adversary model. As a proof of concept, we apply our definition to three voting protocols. In particular, we carry out the first rigorous analysis of the recently proposed Civitas system. We precisely identify those conditions under which this system guarantees coercion resistance or fails to be coercion resistant. We also analyze protocols proposed by Lee et al. and Okamoto.

I. Introduction

Coercion resistance is one of the most important and intricate security requirements of voting protocols [23], [33]. Intuitively, a voting protocol is coercion resistant if it prevents voter coercion and vote buying. In other words, a coercer should not be able to influence the behavior of a voter. A notion closely related to coercion resistance, but somewhat weaker is receipt freeness, first proposed in [8].

Most voting schemes and systems that aim to achieve coercion resistance or receipt freeness come without a rigorous security proof. Maybe not surprisingly, some of these protocols have been found to be flawed (see, e.g., discussions in [32] and [19]). The lack of proofs is partly due to the fact that only recently first formal definitions of coercion resistance and receipt freeness have been proposed in the literature, both based on cryptographic and symbolic models [4], [13], [17], [21]–[23], [31]. With "cryptographic models" we mean models in which messages are modeled as bit strings and adversaries are probabilistic polynomial time Turing machines. In contrast, symbolic models take a more abstract view on cryptography. In this paper, our focus will be on symbolic models. While security guarantees in cryptographic models are typically stronger than in symbolic models, security proofs in cryptographic models are usually very involved, and as a result, often omitted or only sketched. For electronic voting protocols, which are among the most complex security protocols, this is even more so (see, e.g., [8], [14], [19], [23], [28], [33], [34]). Conversely, security proofs in symbolic models are easier to carry out and they are more amenable to tool support. Research on security protocol analysis has demonstrated that, while not all, but many attacks on security protocols can be uncovered and prevented by means of symbolic protocol analysis (see, e.g., [3], [7], [9], [10], [12], [26], [30]). In some cases, security guarantees established in symbolic models even imply security in cryptographic models (see, e.g., [2], [15], [29]). Hence, symbolic models certainly have their merits for security protocol analysis, including the analysis of voting protocols.

However, the definitions of coercion resistance in symbolic models proposed in the literature thus far are rather restricted in scope, yet quite complex and not always intuitive (see Section VIII for a detailed discussion).

Contribution of this paper. One of the main contributions of this paper is to provide a general, yet intuitive and simple definition of coercion resistance. Our definition follows an epistemic approach. It is formulated in a model-independent way. In particular, it can be instantiated by different symbolic models. While the focus of this work is on voting protocols, our definition may be applicable beyond this domain.

In order to analyze concrete voting protocols, we instantiate our framework by a rather standard symbolic model. Within our model, we prove several general statements, which underline the adequacy of our model and which have not been proven in other symbolic models. Among others, we show that coercion resistance w.r.t. a single coerced voter implies coercion resistance w.r.t. multiple coerced voters.

As a proof of concept, we analyze coercion resistance of three voting protocols: the recently proposed voting system Civitas [14], a voting protocol by Lee et al. [28] and one by Okamoto [33]. As to the best of our knowledge, Civitas and the scheme by Okamoto have not been rigorously analyzed before. Our modeling, in particular of Civitas, is quite detailed and goes beyond the level of detail considered in other works. For example, for Civitas we model dishonest authorities and the zero-knowledge proofs authorities have to provide to prove their compliance with the protocol. We precisely identify those conditions under which coercion resistance is guaranteed and point out situations in which the protocols do not provide coercion resistance, thereby relativizing previous claims and providing new insights into and improvements of the protocols. The analyzes of the example protocols illustrate that our definition of coercion resistance allows to specify various

degrees of coercion resistance in a fine-grained way. Without this flexibility of our definition, no reasonable statement about the coercion resistance of voting protocols would be possible as every protocol builds on its own assumptions and provides specific security guarantees.

Structure of this paper. In the following section, we present our definition of coercion resistance. A concrete instantiation of this definition is provided in Section III, with general properties given in Section IV. The analyzes of the three mentioned voting protocols are then presented in Sections V, VI, and VII. Related work is discussed in Section VIII. We conclude in Section IX. More details and proofs can be found in the appendix. We point the reader to [27] for a full version of this paper.

II. Defining Coercion Resistance

In this section, we present our definition of coercion resistance in an epistemic framework, independent of a specific, symbolic protocol or adversary model. A concrete instantiation will be considered in Section III.

Our definition of coercion resistance is based on what we call a coercion system. A coercion system will be induced by a voting protocol (see Section III). It emphasizes in an abstract way those parts relevant for defining coercion resistance, without the need to consider details of a protocol and adversary model. More intuition is provided following the next definition.

Definition 1. A *coercion system* is a tuple $S = (R, V, C, E, r, \sim)$, where R is a set of runs, V, C, and E are sets of possible programs of coerced voters, the coercer, and the environment, respectively, r is a mapping which assigns a set $r(v,c,e) \subseteq R$ of *runs induced by* (v,c,e) to each tuple $(v,c,e) \in V \times C \times E$, and $\sim$ is an equivalence relation on the set R, which determines the view of a coercer on a run.

A coercion system determines the possible behaviors of coerced voters, the coercer, and the environment. The environment is the part of the system controlled neither by the coercer nor by the coerced voter. The environment typically describes the possible behaviors of honest entities, such as honest voters and authorities; dishonest voters and authorities will be subsumed by the coercer. The programs carried out by these honest entities will be determined by the voting protocol under consideration. However, the environment typically does not fix up front how and if certain honest voters vote. It may also leave open the number of voters as well as how many of them and which voters are honest or dishonest. The set $r(v,c,e)$ describes the possible runs obtained when the programs v, c, and e of the coerced voter, the coercer, and the environment, respectively, run together. A run is typically a sequence of configurations induced by the interaction of v, c, and e. However, for a general definition of coercion resistance it is not necessary to fix such details at this point. The reason that we do not define $r(v,c,e)$ to be a single run is that a run of v, c, and e might involve some non-deterministic choices, e.g., non-deterministic scheduling of messages. The equivalence relation $\sim$ defines the view of the coercer. The intuition is that if two runs ρ and ρ' are equivalent w.r.t. $\sim$, i.e., $\rho \sim \rho'$, then the coercer has the same view in both runs. In other words, these runs look the same from the coercer's point of view.

We can now turn to the definition of coercion resistance. For the following discussion, we concentrate on the case that only a single voter is coerced. The case of multi-voter coercion resistance is discussed later.

Given a coercion system $S = (R, V, C, E, r, \sim)$, the idea behind our definition of coercion resistance is as follows:

Our definition assumes that the coerced voter has a certain goal γ that he/she would try to achieve in absence of coercion. Formally, γ is a subset of R, the set of runs of S. If, for example, γ is supposed to express that the coerced voter wants to vote for a certain candidate, then γ would contain all runs in which the coerced voter voted for this candidate and this vote is in fact counted. Jumping ahead, as we will see in the analysis of concrete protocols, often such a goal cannot be achieved. This is, for example, the case if ballots are sent over an unreliable channel or an election authority misbehaves in an observable way and as a result the election process is stopped. A more realistic goal γ would then be that the coerced voter successfully votes for a certain candidate, provided the voters ballot is delivered in time and the election authority did not misbehave in an observable way.

Now, in the definition of coercion resistance we imagine that the coercer provides the coerced voter with a program $v \in V$ (the *coercion strategy*), which the coercer wants the coerced voter to run, instead of the program the coerced voter would carry out when following the voting protocol. The program v might determine the candidate for which the coercer wants the coerced voter to vote for or might dictate the coerced voter not to vote (abstention attack). The choice of the candidate or whether or not the coerced voter should abstain from voting might even depend on the course of the election process and the information that the coercer has gathered thus far. Such information can be gathered by the program v or might be given to the program by the coercer; in the most general setting, one assumes that the coercer can freely communicate with the program v, and by this, further influence and control the behavior of the coerced voter. Rather than directly manipulating the outcome of the election, the purpose of v might as well be to merely test whether the coerced voter follows the prescribed program v; for example, to find out whether this voter is "reliable", and hence, is a good candidate for coercion in later elections. This illustrates that the intentions of the coercer are manifold and hard to predict. The set V should therefore contain all programs that a coercer could possibly give to a coerced voter. However, as shown in Section IV-A, in a concrete communication model, it often suffices to consider just one program that simply forwards all messages from/to the coercer. Nevertheless, taking the set V into account only makes our definition more flexible since different classes of coercion strategies can be specified.

Our definition of coercion resistance requires that for all

$v \in V$, there exists a program $v' \in V$, the *counter strategy*, that the coerced voter can run instead of v, such that (i) the voter always achieves his/her own goal γ by running v' and (ii) the coercer does not know whether the coerced voter run v or v'. In other words, in every run in which the coerced voter run v, the coercer thinks, given his/her view of the run, that it is possible that the coerced voter run v'. Conversely, in every run in which the coerced voter run v', the coercer thinks that it is possible that the coerced voter run v. So, the coercer cannot know whether the coerced voter followed the coercer's instructions (i.e., run v) or just tried to achieve his/her own goal (by running v'). If in some situations the coercer knew that the coerced voter run either v or v', then the voter could be influenced: The coercer could give positive and/or negative incentives for running v/v', e.g., by offering money and/or threatening the coerced voter.

The above leads to the following definition. The meaning of α is explained below.

Definition 2. Let $S = (R,V,C,E,r,\sim)$ be a coercion system and $\alpha,\gamma \subseteq R$. The system S is *coercion resistant in α w.r.t. γ*, if for each $v \in V$ there exists $v' \in V$ such that the following conditions are satisfied.

(i) For every $c \in C$, $e \in E$, and $\rho \in r(v,c,e) \cap \alpha$, there exists $e' \in E$ and $\rho' \in r(v',c,e')$ such that $\rho \sim \rho'$.

(ii) For every $c \in C$, $e \in E$, and $\rho \in r(v',c,e) \cap \alpha$, there exists $e' \in E$ and $\rho' \in r(v,c,e')$ such that $\rho \sim \rho'$.

(iii) For every $c \in C$ and $e \in E$, we have $r(v',c,e) \subseteq \gamma$.

Condition (iii) in the above definition directly captures that if the coerced voter runs the counter strategy v', then independently of the actions of the coercer c and the environment e, the coerced voter achieves his/her goal. To explain the conditions (i) and (ii), let us first ignore the set α. Then (i) says that, for every run ρ in which the coerced voter carries out v, there exists another run ρ' in which the coerced voter carries out v' such that the view of the coercer, who runs c in both runs, is the same. In other words, even though the coerced voter carried out v, from the coercer's point of view it is possible that the coerced voter carried out v'. The programs e and e' in (i) might, for example, differ in the way honest voters voted. So even though the coerced voter might not have voted in the way intended by the coercer, the coercer can not tell from the outcome of the election, as the coercer does not have complete knowledge about how everybody voted. Analogously, condition (ii) says that in every run in which the coerced voter run v', the coercer thinks that it is possible that the coerced voter run v. Altogether (i) and (ii) say that the coercer never knows whether the coerced voter run v or v'.

Now, let us discuss the purpose of α. The intuition is that α describes a property of the environment (which, as mentioned, includes the honest voters) in terms of a set of runs that satisfy this property. The set α typically includes almost all runs of the system, except for those that are unlikely to happen and would reveal to the coercer that the coerced voter is following v or v'. For example, α would typically not contain a run, say ρ, in which a certain candidate, say a, does not get any vote from the honest voters. Indeed, to obtain a successful counter strategy, it is necessary to exclude such a run: Assume that the coercer wants the coerced voter to vote for a (hence, an appropriate v is given by the coercer to the coerced voter). Also assume that the goal γ of the coerced voter is to vote for a different candidate, say b. Then in the run ρ from above, if the coerced voter ran the counter strategy v', the coercer would easily detect this fact: If after the election the coercer observes that there is no vote for a, the coercer can be sure that the coerced voter was not following the coercion strategy v. In other words, in Definition 2, if v' satisfies (iii), then (ii) cannot be satisfied, unless by α runs such as ρ are excluded. This example shows that without taking an appropriate α into account, Definition 2 would be too strong in almost all realistic settings.

The example protocols analyzed in Sections V, VI, and VII will further illustrate the usefulness and necessity of the parameters α and γ of our definition of coercion resistance. These parameters allow to precisely capture under what conditions a protocol is coercion resistant, making for a quite fine-grained and general notion of coercion resistance.

Definition 2 only stipulates the existence of a counter strategy v', given a coercion strategy v. However, it might in general not be easy to come up with v' given v. Fortunately, as already mentioned above, we can show that it is often suffices to come up with a counter strategy only for what we call a dummy coercion strategy, which merely forwards messages to/from the coercer. Given such a counter strategy, one can, in a generic way, construct a counter strategy for any given coercion strategy (see Section IV-A). We believe that the construction of a counter strategy from a (dummy) coercion strategy should be part of the protocol specification, so that a voter knows how to defend against coercion (see also [31]).

We note that Definition 2 captures coercion resistance in a possibilistic way. We do not consider probabilities. While Definition 2 requires that from the coercer's point of view it is always possible that the coerced voter run v, say, the definition does not talk about the probability for this to be the case. If this probability were low, the coercer could tend to believe that the coerced voter run v'. We leave a probabilistic/cryptographic version of our definition as future work. The analysis carried out in this work for the three voting protocols shows that already in a possibilistic setting non-trivial security guarantees can be proved and subtle vulnerabilities can be uncovered.

While in Definition 2 only one goal of the coerced voter is considered, a protocol should of course be coercion resistant no matter what goal the coerced voter would like to achieve; for example, no matter which candidate the coerced voter would like to vote for. This is captured by the following generalization of Definition 2.

Definition 3. Let $S = (R,V,C,E,r,\sim)$ be a coercion system and Γ be a set of goals, i.e. Γ is a set of subsets of R. Then S is *coercion resistant in α w.r.t. Γ*, if S is coercion resistant in α w.r.t. γ, for each $\gamma \in \Gamma$.

Multi-voter coercion. So far, we had in mind that v and v' stand for programs carried out by a single coerced voter. Nevertheless, we can just as well think of v and v' as tuples of programs carried out by multiple coerced voters, where the tuples may be of varying length, depending on how many voters are coerced. In other words, our definition of coercion resistance directly carries over to the case of *multi-voter coercion resistance*, where multiple voters are coerced at the same time. However, the requirement "for all v there exists a v' such that ..." in the definition of coercion resistance then only means that a coerced voter can pick a counter strategy depending on all the programs in v. This is too weak. A coerced voter should be able to pick his/her counter strategy independently of other coerced voters; a coerced voter may in general not know who else is coerced and with whom he/she can (safely) collaborate. Therefore, for multi-voter coercion resistance, we replace the requirement "for all v there exists a v' such that ..." by "there exists a function f which maps a coercion strategy for one voter to a counter strategy for one voter such that, for every tuple v of programs, $v' = f(v)$ is a counter strategy such that ...", where $f(v)$ means that f is applied to every single program in the tuple v.

In Section IV-C we show that (a slight extension of) coercion resistance w.r.t. a single coerced voter implies multi-voter coercion resistance. So, to obtain multi-voter coercion resistance it suffices to consider the case of a single coerced voter.

III. A Concrete Protocol and Adversary Model

In this section, we instantiate the framework presented in the previous section by a concrete protocol and adversary model. Several instantiations are possible, including, for example, one based on I/O automata or process calculus. For the sake of brevity, we pick a quite abstract one, in which computations are described by certain functions, called atomic processes. However, the results presented in the subsequent sections also carry over to other models. We note that these sections should be intelligible without the concrete protocol and adversary model presented in this section. So, some readers may want to skip this section when first reading the paper.

Terms and messages. Let Σ be some signature for cryptographic primitives (including a possibly infinite set of constants for representing participant names, etc.), $X = \{x_1, x_2, \dots\}$ be a set of variables, and $\mathcal{N}$ be an infinite set of *nonces*, where the sets Σ, X, and $\mathcal{N}$ are pairwise disjoint. For $N \subseteq \mathcal{N}$, the set T_N of *terms* over $\Sigma \cup N$ and X is defined as usual. Ground terms, i.e., terms without variables, represent messages. We assume some fixed equational theory associated with Σ and denote by $\equiv$ the congruence relation on terms induced by this theory. The exact definition of Σ and the equational theory will depend on the cryptographic primitives used in the voting protocol under consideration. For the voting protocols we analyze in Section V, VI, and VII quite involved signatures and equational theories will be considered, which, among others, allow to model homomorphic encryption and various kinds of zero knowledge proofs (designated-verifier reencryption proofs, distributed plaintext equivalence tests, etc.). A simple example of a signature Σ_{ex} and its associated equational theory is provided in Figure 1. A term of the form $\mathsf{sig}_k\{m\}$ represents a message m signed using the (private) key k. Checking validity of such a signature is modeled by equation (1). The fact that signatures do not necessarily hide the signed message is taken care of by equation (2). A term of the form $\{x\}^r_{\mathrm{pub}(k)}$ represents the ciphertext obtained by encrypting x under the public key $\mathrm{pub}(k)$ using randomness r. Decryption of such a term using the corresponding private key k is modeled by equation (3). A term of the form $\langle x, y\rangle$ models the pairing of terms x and y. The components x and y of $\langle x, y\rangle$ can be extracted by applying the operators $\mathsf{first}(\cdot)$ and $\mathsf{sec}(\cdot)$, respectively, as modeled by the equations (4). Let $\equiv_{ex}$ denote the congruence relation induced by the equational theory in Figure 1, then we have that $\mathsf{dec}(\{a\}^r_{\mathrm{pub}(k)}, \mathsf{first}(\langle k, b\rangle)) \equiv_{ex} a$.

$$\mathsf{checksig}(\mathsf{sig}_k\{m\}, \mathrm{pub}(k)) = \mathsf{T} \quad (1)$$

$$\mathsf{extractmsg}(\mathsf{sig}_k\{m\}) = m \quad (2)$$

$$\mathsf{dec}(\{x\}^r_{\mathrm{pub}(k)}, k) = x \quad (3)$$

$$\mathsf{first}(\langle x, y\rangle) = x, \qquad \mathsf{sec}(\langle x, y\rangle) = y \quad (4)$$

Fig. 1. The equational theory associated with the signature $\Sigma_{ex} = \{\mathsf{sig}.\{\cdot\}, \langle\cdot,\cdot\rangle, \{\cdot\}_:, \mathsf{T}, \mathsf{checksig}(\cdot,\cdot), \mathsf{extractmsg}(\cdot), \mathsf{first}(\cdot), \mathsf{sec}(\cdot)\}$..

Event sequences and views. Let Ch be a set of *channels* (*channel names*). An *input/output event* is of the form $(c : m)$ and $(\bar{c} : m)$, respectively, for $c \in \mathsf{Ch}$ and a message m (note that $\bar{c} \notin \mathsf{Ch}$). A finite or infinite sequence of events is called an *event sequence*. For an event sequence $\rho = (c_1 : m_1), (c_2 : m_2), \dots$ of input events, we denote by $\mathrm{chan}(\rho)$ the sequence $c_1, c_2, \dots$ of channels. For $C \subseteq \mathsf{Ch}$, we denote by $\rho_{|C}$ the subsequence of ρ containing only the events $(c : m)$ with $c \in C$.

Let $\tau \in T_N$ be a term. Then, with ρ as above, we denote by $\tau[\rho]$ the message $\tau[m_1/x_1, m_2/x_2, \dots]$, where x_i is replaced by m_i. (Recall that the set of variables is $X = \{x_1, x_2, \dots\}$.) For example, assume that $\tau_{ex} = \mathsf{dec}(x_1, \mathsf{first}(x_2))$ and $\rho_{ex} = (c_1 : \{a\}^r_{\mathrm{pub}(k)}), (c_2 : \langle k, b\rangle)$. Then $\tau_{ex}[\rho_{ex}] = \mathsf{dec}(\{a\}^r_{\mathrm{pub}(k)}, \mathsf{first}(\langle k, b\rangle)) \equiv_{ex} a$.

Borrowing the notion of static equivalence from [1], we call two event sequences ρ and ρ' of input events *statically equivalent w.r.t. a set $C \subseteq \mathsf{Ch}$ of channels and a set $N \subseteq \mathcal{N}$ of nonces*, written $\rho \equiv^C_N \rho'$, if (i) $\mathrm{chan}(\rho_{|C}) = \mathrm{chan}(\rho'_{|C})$ and (ii) for every $\tau_1, \tau_2 \in T_N$ we have that $\tau_1[\rho_{|C}] \equiv \tau_2[\rho_{|C}]$ iff $\tau_1[\rho'_{|C}] \equiv \tau_2[\rho'_{|C}]$. Intuitively, a party listening on channels C and a priori knowing the nonces in N, cannot distinguish between the inputs received according to ρ and those received according to ρ'. We call the equivalence class of ρ w.r.t. $\equiv^C_N$, the (C,N)-*view* on ρ. For example, if k, k', a, and b are different constants, r and r' are nonces, $C = \{c_1, c_2\}$, and $N = \emptyset$, then it is easy to see that $\rho^1_{ex} = (c_1 : \{a\}^r_{\mathrm{pub}(k)}), (c_2 : \langle k', b\rangle), (c_3 : k)$ and $\rho^2_{ex} = (c_1 : \{b\}^{r'}_{\mathrm{pub}(k)}), (c_2 : \langle k', b\rangle)$ yield the same (C,N)-view w.r.t. $\equiv_{ex}$.

Processes. Processes are built from atomic processes. An atomic process is basically a function that given a sequence of input events (representing the history so far) produces a sequences of output events. We require that an atomic process behaves the same on inputs on which it has the same view. More precisely, an *atomic process* is a tuple $p = (I, O, N, f)$ where

(i) $I, O \subseteq \mathsf{Ch}$ are finite sets of *input* and *output* channels, respectively,

(ii) $N \subseteq \mathcal{N}$ is a set of *nonces used by* p,

(iii) f is a mapping which assigns a sequence $f(U) = (c_1 : \tau_1) \cdots (c_n : \tau_n)$ with $c_i \in O$ and $\tau_i \in T_N$ to each (I,N)-view U.

We note that (iii) guarantees that p performs the same computation on event sequences that are equivalent according to $\equiv^I_N$, and hence, on which p has the same view. This is why f is defined on (I,N)-views rather than on sequences of input events.

For an event sequence ρ, we write $p(\rho)$ for the output produced by p on input ρ. This output is $(c_1 : \tau_1[\rho']) \cdots (c_n : \tau_n[\rho'])$, where $\rho' = \rho_{|I}$ and $(c_1 : \tau_1) \cdots (c_n : \tau_n) = f(U)$ for the equivalence class U of ρ' w.r.t. $\equiv^I_N$. For example, let $I = \{c_1, c_2\}$, $N = \emptyset$, U be the equivalence class of ρ^1_{ex}, and assume that $f(U) = (c_4 : \langle x_1, \mathsf{first}(x_2)\rangle)$. Then, $p(\rho^1_{ex}) = (c_4 : \langle \{a\}^r_{\mathsf{pub}(k)}, \mathsf{first}(\langle k', b\rangle)\rangle)$, which modulo $\equiv_{ex}$ can be equivalently written as $(c_4 : \langle \{a\}^r_{\mathsf{pub}(k)}, k'\rangle)$ and $p(\rho^2_{ex}) = (c_4 : \langle \{b\}^{r'}_{\mathsf{pub}(k)}, \mathsf{first}(\langle k', b\rangle)\rangle)$, which modulo $\equiv_{ex}$ can be equivalently written as $(c_4 : \langle \{b\}^{r'}_{\mathsf{pub}(k)}, k'\rangle)$. Note that since ρ^1_{ex} and ρ^2_{ex} yield the same (I,N)-view w.r.t. $\equiv_{ex}$, p performs the same transformation on ρ^1_{ex} and ρ^2_{ex}.

We refer to I, O and N by I_p, O_p, and N_p, respectively. We note that the sets I_p and O_p do not have to be disjoint, i.e., p can send messages to itself.

For atomic processes p and p', we write $p \simeq p'$, if p and p' perform the same computation up to renaming of nonces. This is extended to processes (see below) in the obvious way.

A *process* P is a finite set of atomic processes with disjoint sets of input channels and sets of nonces, i.e., $I_p \cap I_{p'} = \emptyset$ and $N_p \cap N_{p'} = \emptyset$, for distinct $p, p' \in P$. The set of input/output channels and the set of nonces of P is $I_P = \bigcup_{p \in P} I_p$, $O_P = \bigcup_{p \in P} O_p$, and $N_P = \bigcup_{p \in P} N_p$, respectively. We say that P is a *process over* (I,O,N), if $I_P \subseteq I$, $O_P \subseteq O$, and $N_P \subseteq N$. By $\Pi(I,O)$ we denote the set of all processes over (I,O,N), for some $N \subseteq \mathcal{N}$.

Given a process P and a finite sequence s_0 of output events over O_P, a *run ρ of a process P initiated by s_0* is a finite or infinite sequence of input and output events which evolves from s_0 in a natural way: An output event is chosen non-deterministically (initial from s_0). Once an output event has been chosen, it will not be chosen anymore later on. By definition of processes, there exists at most one atomic process, say p, in P with an input channel corresponding to the output event. Now, p (if any) is given the input event corresponding to the chosen output event, along with all previous input events on channels of p. Then, p produces a sequence of output events as described above. Now, from these or older output events an output event is chosen non-deterministically, and the computation continues as before. A run is finite if all output events were chosen at some point and there is no new output event left that has not yet been chosen; otherwise a run is infinite. We emphasize that s_0 can induce many runs, due to the non-deterministic delivery of messages. We typically assume *fair runs*, i.e., every output event in a run will eventually be chosen.

We call two processes P and P' *non-conflicting* if $I_P \cap I_{P'} = \emptyset$ and $N_P \cap N_{P'} = \emptyset$. In this case, we will write $P_1 \parallel P_2$ instead of $P_1 \cup P_2$.

If $P \subseteq P'$, we call process P a *subprocess* of process P'. For such a P, we define an equivalence relation $\equiv_P$ on runs induced by P' as follows: $\rho_1 \equiv_P \rho_2$ iff $\rho_1 \equiv^{I_P}_{N_P} \rho_2$. Hence, $\rho_1 \equiv_P \rho_2$ means that from the point of view of P, the runs ρ_1 and ρ_2 look the same. In particular, P behaves the same on these runs.

Protocols. A *protocol* is a tuple $S = (A, in, out, s_0, P)$, where (i) A is a finite set of *agent names*, with access to input and output channels $in(a), out(a) \subseteq \mathsf{Ch}$, respectively, such that $in(a) \cap in(a') = \emptyset$ for $a \neq a'$, (ii) s_0 is a finite sequence of output events, the *initial output sequence*, for initializing parties, (iii) for every $a \in A$, $P(a) \subseteq \Pi(in(a), out(a))$ is the set of *programs* or *processes of a*; this set is assumed to be closed under $\simeq$. For example, if a is an honest voter, then $P(a)$ would typically contain a program for each way a could vote, possibly including abstention of voting. We note that the set A typically contains the coercer and coerced parties, i.e., these entities are part of the protocol specification.

If $A = \{a_1, \ldots, a_n\}$ and $p_i \in P(a_i)$, then $(p_1 \parallel \cdots \parallel p_n)$ is an *instance of S*, where the $p_1, \ldots, p_n$ are non-conflicting. A *run of S* is a fair run of the process $p_1 \parallel \cdots \parallel p_n$ initiated by s_0, where $p_1 \parallel \cdots \parallel p_n$ is some instance of S.

The induced coercion system. Now, let $S = (A, in, out, s_0, P)$ be a protocol with $A = \{\mathsf{v}, \mathsf{c}, \mathsf{e}\}$. Typically, e subsumes all honest principals and processes in $P(\mathsf{e})$ are of the form $p_1 \parallel \cdots \parallel p_n$, where p_i are programs of honest voters and authorities. Dishonest voters and authorities are subsumed by the coercer c and coerced voters by v. The protocol S *induces the coercion system* $(R, V, C, E, r, \sim)$, where:

(i) $V = P(\mathsf{v})$, $C = P(\mathsf{c})$, and $E = P(\mathsf{e})$,

(ii) R is a set of tuples of the form (v, c, e, π), with non-conflicting $v \in V$, $c \in C$, $e \in E$ and π is a run induced by $(v \parallel c \parallel e)$.

(iii) for every $v \in V$, $c \in C$, $e \in E$, $r(v,c,e) = \{(\hat{v}, \hat{c}, \hat{e}, \pi) \mid \hat{v} \simeq v, \hat{c} \simeq c, \hat{e} \simeq e$, and π is a run of S induced by $(\hat{v} \parallel \hat{c} \parallel \hat{e})\}$ is the set of runs of the process formed by v, c, and e, closed under renaming of nonces,

(iv) for all $(v,c,e,\pi), (v',c',e',\pi') \in R$, we have $(v,c,e,\pi) \sim (v',c',e',\pi')$ iff $c = c'$ and $\pi \equiv_c \pi'$. Hence, the relation $\sim$ models the view of the coercer c on runs of S.

IV. General Properties

In this section, we state general properties of coercion systems induced by protocols, as introduced in the previous section. On the one hand, these properties facilitate proofs of coercion resistance of voting protocols. On the other hand, they demonstrate the adequacy of our model. In Section IV-A, we show that, under reasonable assumptions, to prove coercion resistance it is not necessary to consider all coercion strategies, i.e., all programs $v \in V$. It rather suffices to consider a single coercion strategy, the dummy coercion strategy. In Section IV-B, we briefly discuss the notion of receipt freeness and show that it is implied by our notion of coercion resistance. We also show, in Section IV-C, that multi-voter coercion resistance, where multiple voters are coerced, is implied by a slight extension of single-voter coercion resistance, where only one voter is coerced. Except for the second statement, the other statements have not been proven in other works on the symbolic analysis of voting protocols.

A. Dummy Theorem

The theorem that we want to prove, requires *normal* protocols. In these protocols the coerced voter and the coercer can freely communicate (there are input and output channels in both directions) and the set of programs of both entities contains all processes, with appropriate input and output channels. For general coercion resistance, protocols are typically defined in this way.

We define the *dummy coercion strategy* v_0 to be the process which simply forwards to the coercer all the messages the coerced voter receives from the environment and, conversely, forwards to the environment all the messages the coerced voter receives from the coercer.

Now, we call a coercion system for a protocol *dummy coercion resistant* if it is coercion resistant in case a counter strategy is demanded only for the dummy coercion strategy.

To state our dummy theorem, we need to define a relation $\doteq$ on runs which defines a certain view of the environment. Let $S = (A, in, out, s_0, P)$ be a protocol with $A = \{\mathsf{v}, \mathsf{c}, \mathsf{e}\}$ and let π be a run of P. Then, by $\mathrm{env}(\pi)$ we denote the subsequence of π which only contains input and outputs events for channels of e, i.e., events of the form $(c : m)$ and $(\bar{c}, m)$ with $c \in in(\mathsf{e}) \cup out(\mathsf{e})$. Now, for runs π and π' we write $\pi \doteq \pi'$ iff $\mathrm{env}(\pi) = \mathrm{env}(\pi')$. We extend this relation to the set of runs of the coercion system of S: $(v, c, e, \pi) \doteq (v', c', e', \pi')$ iff $e = e'$ and $\pi \doteq \pi'$. We say that a set H of runs is *closed under* $\doteq$, if $\pi \in H$ and $\pi \doteq \pi'$ implies $\pi' \in H$.

In the following theorem, we assume that α and γ are closed under $\doteq$. As α and γ are typically defined based on the view of the environment, the assumption is satisfied in most applications, including the protocols that we analyzed.

Theorem 1. *Let $S = (V, C, E, r, \sim)$ be a coercion system for a normal protocol and α, γ be sets of runs of S closed under $\doteq$. Then dummy coercion resistance implies (full) coercion resistance.*

Proof sketch: Assume that v_0' is the counter strategy for the dummy strategy v_0. Let $v \in V$ be any coercion strategy. Then we show that the parallel composition of v_0' and v, i.e., the process $v_0' \parallel v$, with a proper renaming of channels, is a counter strategy for v. ∎

We note that theorems of a similar flavor as the one above are also considered in cryptographic, simulation-based settings (see, e.g., [11], [24], [25]).

B. Receipt Freeness

We define receipt freeness similarly to coercion resistance, but with the assumption that the coercer cannot send any messages directly to the coerced voter. Hence, only the coerced voter can send messages to the coercer. These messages can be considered to be receipts. This intuition is shared with many other works. One could further weaken the following definition by fixing a certain class of coercion strategies, where, for example, the coerced voter basically follows the protocol but provides the coercer with all the information obtained during the run of the protocol.

Definition 4. A coercion system $S = (V, C, E, r, \sim)$ is *receipt-free* in α w.r.t. γ, if the system $S' = (V, C', E, r, \sim)$, where C' consists of all the programs in C which do not directly send messages to the coerced voter, is coercion resistant in α w.r.t. γ.

Alternatively to restricting the coercer, one could require the coerced voter not to accept messages from the coercer. As an immediate consequence of the above definition, we obtain the following theorem.

Theorem 2. *If a coercion system is coercion-resistant, then it is receipt-free.*

C. Multi-voter Coercion Resistance

In this section, we show that multi-voter coercion resistance is implied by a slight extension of single-voter coercion resistance. The main idea is that in case of multiple coerced voters, all coerced voters, except for one, can be considered to be dishonest, and hence, their behavior can be subsumed by the coercer, leaving the case of a single coerced voter.

In what follows, let $S = (A, in, out, s_0, P)$ be a protocol with $A = \{\mathsf{v}, \mathsf{c}, \mathsf{e}\}$. According to the definition of multi-voter coercion resistance (see Section II), we assume that the programs of v are processes of the form $(p_1 \parallel \cdots \parallel p_n)$, where p_i represents a process of the coerced voter v_i, with its own set I_i and O_i of input and output channels, respectively. We have that $in(\mathsf{v}) = I_1 \cup \cdots \cup I_n$ and $out(\mathsf{v}) = O_1 \cup \cdots \cup O_n$.

Given S, we define for every coerced voter v_i a new protocol S_i, where v_i is the only coerced voter and every other coerced voter is considered to be dishonest, and hence, subsumed by the coercer. The environment e in S_i is the same as in S.

We let T denote the coercion system for S and $T_1, \ldots, T_n$ the coercion systems for $S_1, \ldots, S_n$, respectively.

Now, we slightly extend the notion of (single-voter) coercion resistance, as mentioned before. An explanation follows the definition.

Definition 5. A system $S = (R,V,C,E,r,\sim)$ is *coercion resistant for* $(\alpha_0,\ldots,\alpha_n)$ *w.r.t.* γ, where $\alpha_0,\ldots,\alpha_n,\gamma \subseteq R$, if for each $v \in V$ there exists $v' \in V$ such that the following conditions are satisfied.

(i) For every $k \in \{1,\ldots,n\}$, $c \in C$, $e \in E$, and $\rho \in r(v,c,e) \cap \alpha_k$, there exists $e' \in E$ and $\rho' \in r(v',c,e') \cap \alpha_{k-1}$ such that $\rho \sim \rho'$.

(ii) For every $k \in \{1,\ldots,n\}$, $c \in C$, $e \in E$, and $\rho \in r(v',c,e) \cap \alpha_k$, there exists $e' \in E$ and $\rho' \in r(v,c,e') \cap \alpha_{k-1}$ such that $\rho \sim \rho'$.

(iii) For every $c \in C$ and $e \in E$, we have that $r(v',c,e) \subseteq \gamma$.

First note that condition (iii) of the definition is the same as the corresponding condition in Definition 2. Also, for $n = 1$ and $(\alpha_0,\alpha_1) = (R,\alpha)$ the rest of the conditions coincide with Definition 2 as well. A property α_i contains, for example, all runs in which there are at least i votes for all candidates by honest voters. Now, when going from a run where the coerced voter carries out v to a run where he/she carries out v', then in the latter runs honest voters might have to vote in different ways in order to balance the behavior of v'. The above definition requires that in the run with v' still α_{i-1} is satisfied, and hence, in the example, there are still at least $i-1$ votes for all candidates by honest voters.

We obtain the following theorem, which says that to prove multi-voter coercion resistance, it suffices to show single-voter coercion resistance in the sense of Definition 5. Despite the quantification over i in the following theorem, it typically suffices to prove (single-voter) coercion resistance for one T_i, due to symmetry.

Theorem 3. *Let $S,S_1,\ldots,S_n$ and $T,T_1,\ldots,T_n$ be defined as above. Let $\alpha_0,\ldots,\alpha_n$ and $\gamma_1,\ldots,\gamma_n$ be properties of T, i.e., sets of runs of T. If, for each $i \in \{1,\ldots,n\}$, we have that T_i is coercion-resistant for $(\alpha_0,\ldots,\alpha_n)$ w.r.t. γ_i, then T is multi-voter coercion resistant in α_n w.r.t. $\gamma_1 \cap \cdots \cap \gamma_n$.*

The proof of this theorem is postponed to Appendix A. As mentioned before, the proof of this theorem relies on the fact that coerced voters, except for one, can be considered to be dishonest voters, and hence, can be subsumed by the coercer. Our analysis on the protocol by Okamoto [33] shows that if dishonest voters are not considered, then single-voter coercion does in fact not imply multi-voter coercion: One can show that the Okamoto protocol is coercion resistant in the case of a single coerced voter without any dishonest voters. But the protocol is not coercion resistant with two coerced voters and still no dishonest voters (see Section VII).

V. Civitas

In this section, we briefly recall the Civitas system [14], discuss how this system is modeled in our framework, and present positive and negative results of our analysis of Civitas, i.e., we state conditions under which Civitas does not guarantee coercion resistance and conditions under which coercion resistance is achieved. This is the first rigorous analysis of Civitas and our analysis brings out subtleties that have not been observed before.

A. Protocol Description

We now briefly describe the Civitas system. A more detailed specification of this system in our framework is provided in the appendix. We start with a short description of the various cryptographic primitives employed in Civitas.

Cryptographic primitives. Civitas uses, among others, encryption schemes that allow for homomorphic encryption, random reencryption, and/or distributed decryption. In an encryption scheme with distributed decryption, a public key is generated by multiple parties. This public key can be used for encryption as usual. However, the participation of all parties involved in generating the public key is necessary to decrypt a message encrypted under the public key. Civitas also uses a *distributed plaintext equivalence test* (PET), where multiple parties participate in determining whether two different ciphertexts contain the same plaintext. Finally, Civitas employs a number of zero-knowledge proofs and a mix network.

Protocol participants. The Civitas system assumes the following protocol participants: the supervisor S, voters $\mathsf{v}_0,\ldots,\mathsf{v}_m$, the bulletin board B (which is a kind of write-only, publicly accessible memory), registration tellers $\mathsf{R}_0,\ldots,\mathsf{R}_k$, ballot boxes $\mathsf{X}_0,\ldots,\mathsf{X}_k$, and tabulation tellers $\mathsf{T}_0,\ldots,\mathsf{T}_k$. As in [14], we make the following assumptions: S, B, R_0, X_0, and T_0 are honest, the remaining voting authorities may be dishonest. An arbitrary number of voters are dishonest, they are subsumed by the coercer. The channel between the coerced voter and the honest registration teller is untappable. Channels from voters to the ballot boxes are anonymous, but not untappable (the coercer can see whether ballots are sent to a ballot box).

For now, we consider one coerced voter, say v_0. We note that in [14], it is assumed that v_0 knows which one of the registration tellers is honest. It is in fact easy to see that Civitas is not coercion resistant otherwise. We discuss the case of multi-voter coercion at the end of this section.

Phases of the protocol. The protocol has three phases: the setup, voting, and tabulation phase.

In the *setup phase* the following steps are performed. The tabulation tellers collectively generate a public key K_{T} and post it on the bulletin board; messages encrypted under K_{T} are decrypted in a distributed manner by the tabulation tellers. Next, each registration teller R_j randomly generates, for each voter v_i, a *private credential share* s_{ij} and posts the corresponding public share $S_{ij} = \{s_{ij}\}_{K_{\mathsf{T}}}^{r_{ij}}$ on the bulletin board, where r_{ij} represents the random coins used in the encryption of s_{ij}. The *public credential* S_i *of* v_i is publicly computable as $S_i = (S_{i0} \times \cdots \times S_{ik})$. Now, a voter v_i *registers* at *each* R_j to acquire his/her private credential shares s_{ij}, which comes with a designated verifier reencryption proof (DVRP) that s_{ij} corresponds to the public share S_{ij} posted on the bulletin board (such a proof is built using the public key of the voter; a voter, or any party who knows the corresponding private key, is able to forge such a proof, which is crucial for coercion

resistance). The voter then computes his/her private credential $s_i = s_{i1} \times \cdots \times s_{ik}$.

In the *voting phase*, a voter v_i posts his *ballot* b_i on all the ballot boxes (it is enough, if the ballot is published on only one such a box to be taken into account in the tabulation phase). A ballot consists of an encrypted vote $\{v\}^r_{K_T}$, the encrypted credential $\{s_i\}^{r'}_{K_T}$, a zero-knowledge proof showing that v is a valid vote, and a zero knowledge-proof showing that the submitter simultaneously knows s_i and v_i.

In the *tabulation phase*, tabulation tellers collectively tally the election by performing the following steps: (1) They retrieve the ballots from ballot boxes and the public credentials from the bulletin board. (2) They check the proofs of the ballots, eliminating those ballots with invalid proofs. (3) Using PETs, duplicate ballots, i.e., ballots with the same encrypted credential, are eliminated according to some fixed policy. (4) First the ballots and then the credentials are mixed by each tabulation teller, by applying a permutation and using reencryption. (5) Ballots without valid credentials are eliminated, again using PETs. (6) The votes of the remaining ballots are decrypted in a distributed manner by the tabulation tellers and published. In steps (3)-(6) zero-knowledge proofs are posted to ensure that these steps are performed correctly.

B. Negative Results

Clarkson et al. [14] claim that under the assumptions mentioned before, Civitas is coercion resistant. Just as in the protocol by Juels et al. [23], the idea behind the counter strategy of the coerced voter is to provide the coercer with a fake credential, which prevents the coercer from voting. Clarkson et al. briefly mention that a voter might not be able to vote if a registration teller refuses to provide a credential share to the voter and propose to use an additional voting authority, which attest the misbehavior of the registration teller. However, in the course of trying to prove that Civitas is coercion resistant, we found further problems that make clear that, under the mentioned conditions, Civitas does not provide coercion resistance, if the goal of the coerced voter is to vote for a specific candidate, say z.

The first problem is the following. We may well assume that all dishonest registration tellers provide credential shares to all voters. But they might in addition inform the coercer who has registered. Now, if the coercion strategy dictates the coerced voter not to register, there is no way that the coerced voter can register, as the coercer would be informed. In particular, there is no counter strategy that would allow the coerced voter to vote for z, as the coerced voter cannot register in the first place, and hence, does not know all credential shares required for casting a valid ballot.

There is also another more subtle coercion strategy, which instructs the coerced voter to reveal his/her private key to the coercer before the registration phase. Now, a dishonest registration teller collaborating with the coercer, can use this private key to forge the DVRP. As a result, the coerced voter cannot be sure to have obtained a valid credential share. Hence, even if this voter obtained a credential share from every registration teller, he/she might still not be able to vote.

C. Positive Results

We found that Civitas is coercion resistant in all of the following three settings:

1) All registration tellers are honest and the goal of the coerced voter is to successfully vote for the candidate of his/her choice.
2) The goal of the coerced voter is only to prevent the coercer from casting a valid ballot, where otherwise the assumptions about channels and honest and dishonest authorities are as in [14] and discussed above.
3) The goal of the coerced voter is to successfully vote for the candidate of his/her choice, but the coercion strategies are restricted in that they first dictate the coerced voter to register as prescribed by the protocol and only then follow some arbitrary coercion strategy. Otherwise, the assumptions are as in [14] and discussed before.

The assumptions in the first setting appear to be too strong, given that the main difference of Civitas compared to the Juels et al. protocol, on which Civitas is based, was to replace a single trusted registration teller by a group of possibly dishonest registration tellers. The second setting does not provide the coerced voter with much guarantees. The last setting, which we refer to by *Civitas with restricted coercion strategies*, seems to be the most interesting and certainly the most challenging to prove. We will therefore concentrate on this setting in the rest of the section. One can imagine that the registration is performed long before the election and that in this phase the coercer does not yet try to influence the voter.

We note that in case of Civitas with restricted coercion strategies, the coercer can still ask the voter to reveal his/her private key, but only after the registration of the voter. Hence, the voter can check whether he/she has obtained a valid credential share. Also note that registration tellers might be dishonest.

The main theorem of this section states that Civitas with restricted coercion strategies is coercion resistant in α w.r.t. γ_z for any candidate z, in the sense of Definition 2. We now formulate α and γ_z.

We first introduce some terminology. We say that a ballot posted by a voter is *posted successfully*, if this ballot is delivered to the honest ballot box before the voting phase ends. A run ρ is *fair* w.r.t. the coerced voter v_0, if, in this run, (1) all the registration and tabulation tellers follow the protocol, i.e. post all messages and correct zero-knowledge proofs, as required, (2) v_0 obtains his credentials before the voting phase ends, and (3) if v_0 posts a valid ballot before the voting phase ends, then this ballot is posted successfully.

The properties γ_z and α defined next, will be discussed below.

For every candidate (or valid vote) z, the goal γ_z of the coerced voter v_0 is defined to be the set of all runs satisfying the following conditions: If a run is fair w.r.t. v_0, then the coerced voter successfully votes for z.

The set α of runs contains all runs satisfying the following conditions: (1) For each possible candidate (or valid vote), there is at least one honest voter who successfully casts this vote. (2) There is at least one honest voter who obtains his credential before v_0 finishes registration and abstains from voting. (3) There is at least one honest voter who obtains his credential, but posts successfully a ballot with an *invalid* credential. (4) There is at least one honest voter who posts a ballot after v_0 finishes registration.

Let us first discuss γ_z. By Definition 2, (iii) γ_z means that the counter strategy of v_0 must be such that v_0 votes successfully for z in every fair run. In runs that are not fair w.r.t. v_0 it is clear that the vote of the coerced voter will not be counted, either because a tabulation teller misbehaved in an observable way, making the election invalid, or the ballot did not reach any ballot box in time, and as a result is not decrypted and published by a tabulation teller. The latter can happen if messages on the network are delayed for too long, possibly caused by the coercer. These are obvious reasons why a vote might not be counted. Hence, γ_z is a very strong goal.

Now, consider the conditions (1) to (4) for α: Condition (1) was already motivated in Section II. Condition (2) is needed because if no honest voter abstains from voting, the coercer could tell that the coerced voter does not abstain from voting, even though he/she was supposed to abstain, just by counting the published votes. Moreover, if v_0 completed registration before everybody else (the coercer can even force this to happen when cooperating with a dishonest registration teller), then if some ballot is posted, the coercer knows that this must have been v_0. (We assume that honest voters do not post ballots without completing registration.) In this way, the coercer could again force v_0 to abstain from voting. Condition (3) is also necessary. If the coercer posts a ballot with the fake credential provided by v_0, and if all honest voters only post valid credentials, then the coercer can tell that he/she was fooled, and hence, the counter strategy of the coerced voter fails. Finally, condition (4) is needed for similar reasons as condition (2).

Conditions (1) and (4) arguably exclude runs that are unlikely to happen anyway. However, this is debatable for condition (3) (maybe also for (2)). There is no reason to assume that an honest voter would use an invalid credential, even if he/she has a valid one (such a voter would have to deviate from the protocol). To avoid condition (3), we suggest that Civitas contains some authority which randomly casts some ballots with invalid credentials. Similar "noise" can also help to avoid condition (2).

Theorem 4. *The coercion system induced by Civitas with restricted coercion strategies is coercion resistant in α w.r.t. γ_z, for any valid vote z.*

Before sketching the proof, let us note that the theorem holds for any number of honest and dishonest voters and authorities. We also note that the proof of this theorem does not depend on the policy used to remove duplicates. In particular, it does not matter whether re-voting is allowed or not. Below we show how Theorem 4 can be generalized to the case of multi-voter coercion resistance, using Theorem 3.

Proof sketch: First note that we assume that the registration phase is honestly performed by the coerced voter v_0. Only after this phase coercion starts. Therefore, it is convenient to consider the registration for the coerced voter to be performed within the environment. Once the registration phase is over, all messages the coerced voter received within this phase are handed to v/v', the coercion strategy and counter strategy performed by v_0, respectively. These strategies may be arbitrary processes. By pushing the registration of the coerced voter into the environment as just described, we obtain a normal protocol (see Section IV-A). Otherwise, we would have to restrict the set of possible coercion strategies to those that first honestly follow the registration phase and then start the actual coercion strategy. This would not be a normal protocol. In fact, the dummy coercion strategy would not be part of this set of coercion strategies. Note that the messages given to v/v' include the messages received from the registration tellers, i.e., the private credentials along with the DVRPs, plus private information of the coerced voter, such as private keys.

It is not hard to check that α and γ_z are closed under $\doteq$. Now, since we have a normal protocol, by Theorem 1 it suffices to show that there exists a counter strategy v' for the dummy coercion strategy v. The counter strategy v' is defined as follows. Just as v, it simply forwards all messages between the coercer and the environment, with one exception: The message from the honest registration teller R_0, which contains the private credential share s_{00}, is not forwarded by v'. Instead, v' generates a new (fake) credential share $\tilde{s}_{00}$ and, using its private key, produces a fake DVRP for $\tilde{s}_{00}$ and the public share S_{00}. Then, $\tilde{s}_{00}$ and the DVRP are sent to the coercer. In addition, when v' has the complete collection of private credential shares from the registration tellers, v' computes the private credential s_0 and posts the valid ballot with this credential and a vote for candidate z. We will now show that v' is indeed a counter strategy for v, i.e., we show that the conditions (i), (ii), and (iii) of Definition 2 are satisfied, starting with those conditions that are easier to prove.

Condition (iii) of Definition 2. Let ρ be a run of the process $(v' \parallel c \parallel e)$, for some $c \in C, e \in E$. We have to show that $\rho \in \gamma_z$. If ρ is not fair, then clearly $\rho \in \gamma_z$. So, suppose that ρ is fair. First, note that, by the fairness assumption, all the registration tellers post all messages and zero-knowledge proofs as required. Since, by assumption, coercion starts only after the registration is completed, v_0 does not send out his/her private key until then. Therefore, the DVRPs v_0 gets cannot be faked, and thus the private credential shares v_0 obtains are valid.

Second, again by the fairness assumption, v_0 obtains his registration information before the voting phase ends. Since, by construction, v' posts a valid ballot right away (still before the voting phase ends), by the fairness assumption, this ballot is posted successfully. Now, it is easy to show that this ballot will be successfully processed by tabulation tellers, using the

fact that v' never reveals his private credential (so it is not used in any other ballot) and the assumption that the run is fair. In particular, the tabulation tellers correctly perform all steps, because otherwise they would not be able to construct valid zero-knowledge proofs.

Condition (ii) of Definition 2. Let $\rho \in \alpha$ be a run induced by $(v' \parallel c \parallel e)$, for some $c \in C$ and $e \in E$. We have to construct $e' \in E$ and a run ρ' induced by $(v \parallel c \parallel e')$ such that $\rho \sim \rho'$.

We define the vote z_c as follows: If the coercer, in ρ, successfully posts at least one ballot with $\tilde{s}_0$ (where $\tilde{s}_0$ is computed like s_0, but using the fake credential share $\tilde{s}_{00}$ instead of s_{00}), then let z_c be the vote in the ballot containing $\tilde{s}_0$ which is left after the duplicate elimination phase; otherwise let z_c be any vote. Note that, since a valid ballot has to contain a proof that the vote contained in it is valid, z_c must be a valid vote.

Since ρ is in α, there is some honest voter, say v_1, who obtains his/her credential shares before v_0 finishes registration, but abstains from voting, and some honest voter, say v_2, who successfully votes for z_c.

We define $e' \in E$ to coincide with e with the following exceptions: v_1 votes for z and, moreover, if, in ρ, at least one ballot with credential $\tilde{s}_0$ is successfully posted, then v_2 posts an invalid ballot (i.e. a ballot with an invalid credential). We also need to slightly change the permutations used by the honest tabulation teller in the mixing phase.

The run ρ' of $(v \parallel c \parallel e')$ is constructed from ρ in the following way. The messages in ρ' are delivered in the same order as the corresponding messages in ρ with the following exceptions: The ballot sent by v_1 is delivered in the same step when the ballot sent by v_0 was delivered in ρ. (This is possible because v_1 received his/her credential shares before v_0 finishes registration.)

Now, one can show that $\rho \sim \rho'$. The rough idea is as follows: v_2 is used to hide the fact that the ballots involving the credential given to the coerced voter and possibly posted by the coercer are invalid in ρ but valid in ρ'. Voter v_1 is used to hide the fact that v_0 posts his/her ballot in ρ, but not in ρ'. Moreover, due to the fact that the coercer cannot distinguish an original DVRP and the faked one, and the fact that the messages posted on the bulletin board are mixed and reencrypted, the runs are indistinguishable from the point of view of the coercer.

Condition (i) of Definition 2. Let $\rho \in \alpha$ be a run induced by $(v \parallel c \parallel e)$, for some $c \in C$ and $e \in E$. We have to construct $e' \in E$ and a run ρ' induced by $(v' \parallel c \parallel e')$ such that $\rho \sim \rho'$.

Since ρ is in α, there is some honest voter, say v_1, who successfully posts a ballot with a vote for candidate z, some honest voter, say v_2, who obtains his/her credential and successfully posts a ballot with an invalid credential, and some honest voter, say v_3, who posts his/her ballot after v_0 finishes registration.

We define $e' \in E$ to coincide with e with the following exceptions: (a) v_3 abstains from voting, (b) if v_3 in ρ posted his/her ballot successfully, then v_1 votes like v_3 voted in ρ, (c) if at least one proper ballot with s_0 is successfully posted in ρ and z_c is the vote in the ballot with s_0 that is kept after duplicate elimination (note that z_c must be a valid vote), then v_2 posts a valid ballot with z_c instead of the invalid one. Also, slightly different permutation are used by the honest tabulation teller in the mixing phase.

The run ρ' of $(v' \parallel c \parallel e')$ is constructed from ρ in the following way. The messages in ρ' are delivered in the same order like the corresponding messages in ρ with the following exceptions: the ballot sent by v_0 in ρ' is delivered at the same step, when the ballot sent by v_3 is delivered in ρ (because v_0 sends his ballot, just when he gets the registration data, and v_3 posts his ballot after it, the ballot of v_0 is ready to be delivered at the mentioned step).

Now, one can show that $\rho \sim \rho'$. The rough idea is as follows: v_2 is used to hide the fact that valid ballots possibly posted by the coercer in ρ become invalid in ρ', v_3 is used to hide the fact that v_0 posts his ballot in ρ', but abstains from voting in ρ, and finally v_1 is used to balance the outcome of the voting. Due to the fact that the coercer cannot tell any difference between an original DVRP and a faked one and the fact that the messages posted on the bulletin board are mixed and reencrypted before decryption, the runs are indistinguishable from the point of view of the coercer. ■

Multi-voter coercion. Theorem 4 can easily be generalized to multi-voter coercion resistance. Suppose that a number k of voters is being coerced. Suppose that the goal of voter v_k is γ_k. By Theorem 3, to prove multi-voter coercion resistance in α w.r.t. $\gamma = (\gamma_1 \cap \cdots \cap \gamma_n)$, it is enough to prove (*): a system with only one coerced voter v_i is coercion resistant for $(\alpha_0, \ldots, \alpha_n)$ w.r.t. γ_i, with $\alpha_n = \alpha$.

We define α_k as the set of runs where (1) for each possible vote, there are at least k honest voters who successfully cast this vote, (2) there are at least k honest voters who obtain their credentials, before any of the coerced voters finishes registration, and abstain from voting, (3) there are at least k honest voters who obtain their credential, but post ballots with invalid credentials, (4) there are at least k honest voters who post a ballot after the coerced voters finish registration.

The proof of (*) is very similar to the one for Theorem 4. Hence, multi-voter coercion resistance follows.

VI. Lee et al. Protocol

In this section, we analyze a protocol proposed by Lee et al. [28] within our framework. We show that the protocol is not coercion resistant in general, but propose an extension of the protocol for which we can show coercion resistance.

A. Protocol Description.

The Lee et al. protocol assumes that every voter owns a tamper-resistant device, called a *randomizer*.

In the *setup phase*, the tallying tellers $\mathsf{T}_1, \ldots, \mathsf{T}_k$ generate and publish their common public key K_T for threshold decryption.

In the *voting phase*, a voter prepares his/her ballot, containing a vote encrypted under K_T, and gives it to his/her

randomizer which reencrypts the ballot and signs it, and sends the result s back to the voter along with a designated verifier reencryption proof (DVRP) (such a DVRP can be forged by anyone who knows the private key of the voter). This part of the communication is assumed to be entirely private. Then the voter checks the proof, computes his/her own signature on s and posts it on the bulletin board.

In the *tallying phase*, the following is done: (1) the double signatures of voters and their randomizers on the posted ballots are verified and invalid ballots are eliminated, (2) the remaining ballots are shuffled and reencrypted and the result is posted on the bulletin board, (3) talliers jointly decrypt shuffled ballots and publish the tally result. Correctness of all these steps is assured by posting appropriate non-interactive zero-knowledge proofs.

B. Negative Results.

Assuming that the goal of the coerced voter is to vote for a particular candidate, it is easy to see that this protocol is not coercion resistant: There is a simple abstention attack where the coercer disallows the coerced voter to put a ballot signed by this voter on the bulletin board. So, one can at most hope to prove that if a ballot signed by the coerced voter and his/her randomizer has been put on the bulletin board, then the vote of the coerced voter is counted. However, even this weaker form of coercion resistance cannot be shown: A coercer could prepare a ballot with some invalid vote which is unlikely to occur otherwise and then ask the coerced voter to give this ballot to his/her randomizer, sign the result and put it on the bulletin board. The coercer can check whether his/her vote is decrypted, assuming a dishonest tallying teller collaborating with the coercer. (The Lee et al. protocol is designed to deal with dishonest tallying tellers.) Therefore, a counter strategy is forced to use the ballot prepared by the coercer, and hence, the goal of the coerced voter cannot be achieved.

C. Positive Results.

To prove coercion resistance, one could assume that all tallying tellers are honest, but this is not the point of the Lee et al. protocol. We instead propose a slight extension of the protocol, where the randomizer expects in addition to the ballot a zero-knowledge proof which shows that the vote in the ballot is well-formed (just as in Civitas). The randomizer then checks the proof before replying. With this extension of the protocol, we obtain coercion resistance for a natural γ_z and α: γ_z contains all runs where the coerced voter successfully votes for z, if some ballot signed by this voter and his/her randomizer appears on the bulletin board (within the voting phase) and all zero-knowledge proofs that have to be provided by the authorities are valid. Note that this goal does not exclude abstention attacks. For the same reason explained above, these attacks are still possible in the extended version of the Lee et al. protocol. The set α is simply the set of runs where for each possible vote there is at least one honest voter who successfully casts this vote.

Theorem 5. *The coercion system induced by the extended version of the Lee et al. protocol is coercion resistant in α w.r.t. γ_z, for any valid vote z.*

Proof sketch: First, one can show that the protocol is normal and both α and γ_z are closed under $\doteq$. Hence, we can use Theorem 1. So, it is enough to provide a counter strategy v' for the dummy coercion strategy v, which simply forwards all messages between the coercer and the environment.

Let v' be the process which behaves like v with the following exception. If v is instructed to send to the randomizer a ballot m_c along with a valid DVRP P_c, then v' sends instead a ballot m_0 with the vote z along with a valid DVRP P_0. The answer of the randomizer is then forwarded by v' to the coercer, but with a faked DVRP, instead of the original one. This faked DVRP shows that the message signed by the randomizer is a reencryption of m_c (while actually it is a reencryption of m_0). We will show that v' is a counter strategy for v. Condition (iii) of Definition 2 is easy to prove.

Condition (i) of Definition 2. Let ρ be a run of the process $(v \parallel c \parallel e)$, for some $c \in C, e \in E$. We have to construct $e' \in E$ and a run ρ' induced by $(v' \parallel c \parallel e')$ such that $\rho \sim \rho'$.

Since ρ is in α, there is some honest voter, say v_1, who successfully votes for z.

We define $e' \in E$ to coincide with e with the following exceptions: If, in ρ, there is a ballot, signed by v_0 and v_0's randomizer, on the bulletin board and this ballot contains some vote z_c (note that z_c has to be a valid vote), then v_1, in e', votes for z_c instead of z.

The run ρ' of $(v' \parallel c \parallel e')$ is obtained from ρ in a straightforward way: The messages in ρ' are delivered in the same order as the corresponding messages in ρ. One can show that $\rho \sim \rho'$. The rough idea is as follows: v_1 is used to balance the different behaviors of v_0 in ρ and ρ'. Due to the fact that the coercer cannot tell any difference between a real DVRP and a faked one, and the fact that the messages posted on the bulletin board are mixed and reencrypted before decryption, the runs are indistinguishable from the coercer's point of view.

Condition (ii) of Definition 2. Let ρ be a run of the process $(v' \parallel c \parallel e)$, for some $c \in C, e \in E$. We have to construct $e' \in E$ and a run ρ' induced by $(v \parallel c \parallel e')$ such that $\rho \sim \rho'$. Let us define z_c in the following way: If the coerced voter in ρ is instructed to use his/her randomizer to reencrypt some ballot m_c which is accompanied with a valid proof that m_c contains a valid vote, then we denote this vote by z_c; otherwise let z_c be any vote. Since ρ is in α, there is some honest voter, say v_1, who successfully votes for z_c.

We define $e' \in E$ to coincide with e with the following exceptions: If, in ρ, a message signed by v_0 and v_0's randomizer is posted on the bulletin board, then v_1, in e', votes for z instead of z_c.

The run ρ' of $(v \parallel c \parallel e')$ is obtained from ρ in a straightforward way: The messages in ρ' are delivered in the same order as the corresponding messages in ρ. Analogously to the case of condition (i), one can show that $\rho \sim \rho'$. ■

VII. Okamoto Protocol

In this section, we briefly discuss our results on the analysis of the Okamoto protocol [33].

A. Protocol Description.

The protocol is based on blind signatures and trapdoor commitment. Moreover, for some of the authorities involved, anonymous untappable channels with voters are assumed; a very strong, rather unrealistic assumption.

A voter constructs his ballot using some randomly generated secret s, which we will call a *private credential*. Now, roughly speaking, the private credential is split into a number of shares $s_1, \ldots, s_N$ (*private credential shares*). Also, a *public credential* S and *public credential shares* $S_1, \ldots, S_n$ are computed from s and $s_1, \ldots, s_n$, respectively. A ballot contains S, $S_1, \ldots, S_n$ and a trapdoor commitment, computed using s, of the vote z. Such a ballot is blindly signed by a voting authority during the registration phase. The authority makes sure that this is done at most once for every voter.

In the voting phase, the ballot, as described above, is posted anonymously along with the vote z itself and some additional information for opening the commitment. Moreover, the voter provides private credential shares s_i to certain authorities, proving that he/she knows s. Now, for coercion resistance, the point is that, knowing the private credential s, a voter is able to open a commitment to any vote z'. This is true, even if the ballot was prepared by the coercer.

B. Results.

In short, the Okamoto protocol does not provide coercion resistance even under strong assumptions. However, the protocol is interesting in that it highlights the difference between single-voter coercion and multi-voter coercion, in absence of dishonest voters:

If we assume that the coercer is an entitled voter or there is some dishonest voter, the protocol is clearly not coercion-resistance: The coercer prepares a ballot, ask the coerced voter to obtain a blind signature on this ballot and then completes the process by himself, using the anonymous untappable channels he has access to.

Even if we assume that the coercer is not an entitled voter and there is no dishonest voter, then still the protocol is not coercion-resistant, provided that more than one voter is coerced at the same time. In this case the coercion strategy is as follows. All the coerced voters are supposed to obtain a blind signature of the appropriate voting authority on messages provided by the coercer. Then, the coercer distributes the private credential shares to the voters in such a way that no coerced voter has a complete collection of private credential shares, i.e., the shares for one vote are distributed among different coerced voters. As a result, no coerced voter can open his/her commitment in an arbitrary way. This suffices for the ballots of the coercer to be accepted. To the best of our knowledge, this attack has not been observed before.

While the above attack allows the coercer to vote as he wishes, an abstention attack is possible even if only one voter is coerced, the coercer is not entitled to vote and there are no dishonest voters.

The only setting in which we could prove coercion resistance of the Okamoto protocol is in the setting just described where α is defined similarly to the Lee et al. protocol and the goal γ is merely that if the coerced voter posts his/her (part of) ballot on the bulletin board, then his/her vote is counted. Note that for this result to hold it is essential that only one voter is coerced.

VIII. Related Work

Coercion resistance in a symbolic model was first formulated by Delaune et al. [16]–[18]. This work was then further developed by Backes et al. [4]. Both the work by Delaune et al. and Backes et al. were motivated by the desire to use ProVerif [10], a tool for security protocol analysis, for the automatic analysis of voting protocols. Due to the focus on automation, the notions of coercion resistance studied in these works are more restricted than the one considered here. For example, the notion of coercion resistance introduced by Delaune et al. does not apply to Civitas or the protocol by Juels et al. [23], as the class of coercion strategies and counter strategies they consider are too restricted. To show coercion resistance of the Lee et al. protocol, Delaune et al. study a variant of this protocol which is different to the one studied here. One of the abstention attacks that we point out still works for their variant. However, this attack is out of the scope of their notion of coercion resistance. Conversely, the notion of coercion resistance by Backes et al. is inspired by the one of Juels et al., which in turn is especially tailored to the specific protocol structure of the protocol by Juels et al. and the specific forms of coercion strategies. In order to facilitate automation, the protocol models that Delaune et al. and Backes et al. consider are much coarser than ours. For example, the way votes are tallied is simplified and mix networks and proofs of compliance are not modeled.

There is also a more fundamental difference between the work by Delaune et al. and Backes et al. on the one hand, and our work on the other hand. The symbolic model by Delaune et al. and Backes et al. is the applied pi calculus [1], with its notion of observational equivalence for comparing systems/processes. Observational equivalence is a bisimilarity relation which demands that every step of one system is matched by a similar step of the other system. In particular, in the works by Delaune et al. and Backes et al. the two systems in which the coerced voter runs the coercion strategy and the counter strategy, respectively, are related using the notion of observational equivalence. This is fundamentally different to the approach taken here: In our epistemic approach, we relate *traces* of systems and say that for every trace of one system, there exists a trace of the other system such that the coercer has the same view on both traces. In the two traces, honest voters may vote in different ways. By this, votes (including abstention) can be balanced in case coerced voters vote in different ways in the two systems and this balancing may be based on the traces as a whole. Conversely, observational

equivalence, with its strict stepwise correspondence between systems, prohibits a simple balancing of votes. As a result, the formulations of coercion resistance proposed by Delaune et al. and Backes et al. are very complex and less intuitive. In Delaune et al., the balancing problem is tackled by restricting the set of coercers and coercion strategies. It is assumed that the coercer's goal is to vote for a particular party and that coercion strategies only slightly deviate from the prescribed protocol. Altogether this leads to a rather weak notion of coercion resistance, excluding, for example, abstention attacks and other natural coercion strategies, e.g., those relevant for Civitas. Backes et al. introduce what they call an extractor to solve the balancing problem, which makes the definition of coercion resistance quite complex and hard to understand.

In [21], [22], Jonker et al. also follow an epistemic approach to model properties of voting protocols. However, they do not consider coercion resistance, only receipt freeness. Receipt freeness is modeled w.r.t. a message that a voter could use as a receipt. This is only a very rough approximation of the intuition behind receipt freeness. Also, Jonker et al. do not model any cryptographic operators. A more recent work on receipt freeness by Jonker et al. is [20].

The work by Baskar et al. [6] focuses on the decidability of knowledge-based properties of voting protocols. However, they only study a very simplistic notion of receipt-freeness, which resembles privacy of votes; coercion resistance is not considered.

As already mentioned in the introduction, there also exist several cryptographic definitions of coercion resistance and receipt freeness (see, e.g., [13], [23], [31], [33], [35]). On the one hand, compared to the cryptographic definitions, our symbolic approach abstracts from many cryptographic details, including details of cryptographic primitives and probabilistic aspects. This leads to weaker security guarantees. On the other hand, the simplicity of the symbolic approach in general, and our definition in particular, facilitates the analysis of protocols and is more amenable to automation, which, given the complexity of voting protocols, is a crucial advantage.

IX. Conclusion

In this paper, we presented a general, yet simple and intuitive definition of coercion resistance of voting protocols in an epistemic setting, which does not depend on any specific, symbolic protocol or adversary model. We applied our definition to three different voting protocols, two of which, namely Civitas and the protocol by Okamoto, have not been rigorously analyzed before. For all three protocols, we identified conditions under which these protocols are coercion resistant or fail to be coercion resistant. To obtain these results it was vital that our definition of coercion resistance allows to specify various degrees of coercion resistance in a way more fine-grained than in previous proposals. Our analyzes brought out several insights about the three protocols that have not been observed before and that led us to propose improvements of the protocols.

We believe that our definition of coercion resistance provides a good basis for automated analysis of coercion resistance, in particular since the definition can be instantiated with different protocol and adversary models. However, carrying out tool supported analysis was out of the scope of the present work.

Acknowledgement. We thank the anonymous reviewers for their detailed remarks, which helped to improve the presentation of this work. This work was partially supported by the *Deutsche Forschungsgemeinschaft* (DFG) under Grant KU 1434/5-1, the SNF under Grant 200021-116596, and the Polish Ministry of Science and Education under Grant 3 T11C 042 30.

References

[1] M. Abadi and C. Fournet. Mobile Values, New Names, and Secure Communication. In *Proceedings of the 28th ACM Symposium on Principles of Programming Languages (POPL 2001)*, pages 104–115. ACM Press, 2001.

[2] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In J. van Leeuwen, O. Watanabe, M. Hagiya, P.D. Mosses, and T. Ito, editors, *Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics, International Conference (IFIPTCS 2000)*, volume 1872 of *Lecture Notes in Computer Science*, pages 3–22. Springer-Verlag, 2000.

[3] A. Armando, D.A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P.H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In K. Etessami and S.K. Rajamani, editors, *Computer Aided Verification, 17th International Conference (CAV 2005)*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285. Springer-Verlag, 2005.

[4] M. Backes, C. Hritcu, and M. Maffei. Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-Calculus. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF 2008)*, pages 195–209. IEEE Computer Society, 2008.

[5] M. Backes, M. Maffei, and D. Unruh. Zero-Knowledge in the Applied Pi-calculus and Automated Verification of the Direct Anonymous Attestation Protocol. In *2008 IEEE Symposium on Security and Privacy (S&P 2008)*, pages 202–215. IEEE Computer Society, 2008.

[6] A. Baskar, R. Ramanujam, and S. P. Suresh. Knowledge-based modelling of voting protocols. In Dov Samet, editor, *Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2007)*, pages 62–71, 2007.

[7] G. Bella, F. Massacci, and L.C. Paulson. An overview of the verification of SET. *International Journal of Information Security*, 4:17–28, 2005.

[8] J. C. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing (STOC 1994)*, pages 544–553. ACM Press, 1994.

[9] K. Bhargavan, C. Fournet, A. D. Gordon, and S. Tse. Verified Interoperable Implementations of Security Protocols. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW-19 2006)*, pages 139–152. IEEE Computer Society, 2006.

[10] B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW-14)*, pages 82–96. IEEE Computer Society, 2001.

[11] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. Technical report, Cryptology ePrint Archive, December 2005. Online available at `http://eprint.iacr.org/2000/067.ps`.

[12] I. Cervesato, A.D. Jaggard, A. Scedrov, J.-K. Tsay, and C. Walstad. Breaking and fixing public-key Kerberos. In *International Workshop on Issues in the Theory of Security (WITS 2006)*, 2006.

[13] B. Chevallier-Mames, P.-A. Fouque, D. Pointcheval, J. Stern, and J. Traoré. On Some Incompatible Properties of Voting Schemes. In *IAVoSS Workshop On Trustworthy Elections (WOTE 2006)*, 2006.

[14] M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a Secure Voting System. In *2008 IEEE Symposium on Security and Privacy (S&P 2008)*, pages 354–368. IEEE Computer Society, 2008.

[15] V. Cortier, S. Kremer, R. Küsters, and B. Warinschi. Computationally Sound Symbolic Secrecy in the Presence of Hash Functions. In *Proceedings of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2006)*, volume 4337 of *Lecture Notes in Computer Science*, pages 176–187. Springer, 2006.

[16] S. Delaune, S. Kremer, and M. .D. Ryan. Verifying Privacy-type Properties of Electronic Voting Protocols. *Journal of Computer Security*, 2009. To appear.

[17] S. Delaune, S. Kremer, and M.D. Ryan. Coercion-Resistance and Receipt-Freeness in Electronic Voting. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW'06)*, pages 28–39. IEEE Computer Society Press, 2006.

[18] S. Delaune, S. Kremer, and M.D. Ryan. Verifying properties of electronic voting protocols. In *Proceedings of the IAVoSS Workshop On Trustworthy Elections (WOTE'06)*, pages 45–52, 2006.

[19] M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 539 – 556. Springer, 2000.

[20] H. Jonker, S. Mauw, and J. Pang. Measuring voter-controlled privacy. In *Proceedings of the 4th Conference on Availability, Reliability and Security - ARES'09*. IEEE Computer Society, 2009. To appear.

[21] H.L. Jonker and E.P. de Vink. Formalising Receipt-Freeness. In S.K. Katsikas, J. Lopez, M. Backes, S. Gritzalis, and B.Preneel, editors, *9th Information Security Conference (ISC 2006)*, volume 4176 of *Lecture Notes in Computer Science*, pages 476–488. Springer, 2006.

[22] H.L. Jonker and W. Pieters. Receipt-Freeness as a special case of Anonymity in Epistemic Logic. In *IAVoSS Workshop On Trustworthy Elections (WOTE 2006)*, 2006.

[23] A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *Proceedings of Workshop on Privacy in the Eletronic Society (WPES 2005)*. ACM Press, 2005.

[24] R. Küsters. Simulation-Based Security with Inexhaustible Interactive Turing Machines. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW-19 2006)*, pages 309–320. IEEE Computer Society, 2006.

[25] R. Küsters, A. Datta, J. C. Mitchell, and A. Ramanathan. On the Relationships Between Notions of Simulation-Based Security. *Journal of Cryptology*, 21(4):492–546, 2008.

[26] R. Küsters and T. Truderung. Reducing Protocol Analysis with XOR to the XOR-free Case in the Horn Theory Based Approach. In P. Syverson, S. Jha, and X. Zhang, editors, *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008)*, pages 129–138. ACM Press, 2008.

[27] R. Küsters and T. Truderung. An Epistemic Approach to Coercion-Resistance for Electronic Voting Protocols. Technical Report arXiv:0903.0802, arXiv, 2009. Available at http://arxiv.org/abs/0903.0802.

[28] B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, and S. Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *Proceedings of Information Security and Cryptology (ICISC 2003)*, volume 2971 of *Lecture Notes in Computer Science*, pages 245–258. Springer, 2003.

[29] D. Micciancio and B. Warinschi. Soundness of Formal Encryption in the Presence of Active Adversaries. In M. Naor, editor, *First Theory of Cryptography Conference (TCC 2004)*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151. Springer, 2004.

[30] J.C. Mitchell, V. Shmatikov, and U. Stern. Finite-State Analysis of SSL 3.0. In *Seventh USENIX Security Symposium*, pages 201–216, 1998.

[31] T. Moran and M. Naor. Receipt-Free Universally-Verifiable Voting With Everlasting Privacy. In C. Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 373–392. Springer, 2006.

[32] T. Moran and M. Naor. Split-ballot voting: everlasting privacy with distributed trust. In P. Ning, S. De Capitani di Vimercati, and P. F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007*, pages 246–255. ACM, 2007.

[33] T. Okamoto. Receipt-Free Electronic Voting Schemes for Large Scale Elections. In B. Christianson, B. Crispo, T. M. A. Lomas, and M. Roe, editors, *Proceedings of the 5th International Workshop on Security Protocols*, volume 1361 of *Lecture Notes in Computer Science*, pages 25–35. Springer, 1997.

[34] K. Sako and J. Kilian. Receipt-Free Mix-Type Voting Scheme — A practical solution to the implementation of a voting booth. In *Advances in Cryptology — EUROCRYPT '95, International Conference on the Theory and Application of Cryptographic Techniques*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403. Springer-Verlag, 1995.

[35] V. Teague, K. Ramchen, and L. Naish. Coercion-Resistant tallying for STV voting. In *IAVoSS Workshop On Trustworthy Elections (WOTE 2008)*, 2008.

APPENDIX A
PROOF OF THEOREM 3

Before we prove the theorem, we state some definitions only sketched or omitted in Section IV-C.

Let S be a protocol as in Section IV-C. We define $S_i = (A, in_i, out_i, s_0, P_i)$, where v now represents voter v_i only, e is unchanged, and c gets direct access to the channels of the coerced voters $\mathsf{v}_1, \ldots, \mathsf{v}_{i-1}, \mathsf{v}_{i+1}, \ldots, \mathsf{v}_n$, i.e., $in_i(\mathsf{v}) = I_i$, $out_i(\mathsf{v}) = O_i$, $in_i(\mathsf{c}) = in(\mathsf{c}) \cup \bigcup_{i \in W} I_i$, and $out_i(\mathsf{c}) = out(\mathsf{c}) \cup \bigcup_{i \in W} O_i$, where $W = \{1, \ldots, n\} \setminus \{i\}$. Moreover, $P_i(\mathsf{e}) = P(\mathsf{e})$, $P_i(\mathsf{v}) = \Pi(in_i(\mathsf{v}), out_i(\mathsf{v}))$, and $P_i(\mathsf{c}) = \Pi(in_i(\mathsf{c}), out_i(\mathsf{c}))$.

For the proof of Theorem 3, we define a mapping from runs ρ of T to runs $\rho^{(i)}$ of T_i and from properties β of T to properties $\beta^{(i)}$ of T_i: Recall that each $v \in P(\mathsf{v})$ is of the form $(v_1 \parallel \cdots \parallel v_n)$ with $v_i \in \Pi(I_i, O_i)$. For a run $\rho = ((v_1 \parallel \cdots \parallel v_n), c, e, \pi)$, we define $\rho^{(i)}$ as $(v_i, (v_1 \parallel \ldots \parallel v_{i-1} \parallel v_{i+1} \parallel \ldots \parallel v_n \parallel c), e, \pi)$. For a property β of T, we define $\beta^{(i)}$ to be $\{\rho^{(i)} : \rho \in \beta\}$. When it is clear from the context, we will write β instead of $\beta^{(i)}$, treating β as a property of T_i.

We can now turn to the proof of Theorem 3. We define a function f which maps a coercion strategy v_i of the i-th voter to a counter strategy $v_i' = f(v_i)$, by defining v_i' as some (arbitrarily chosen) counter strategy for v_i in T_i (such a counter strategy exists, since T_i is coercion resistant).

Now, for any $v \in P(\mathsf{v})$ which, as we know, must be of the form $(v_1 \parallel \ldots \parallel v_n)$ with $v_i \in P_i(\mathsf{v})$, and for $v' = (v_1' \parallel \ldots \parallel v_n')$, where $v_i' = f(v_i)$, we will show that T, along with v and v', meets the conditions of the definition of multi-voter coercion resistance.

First, let us show that condition (iii) holds. Let $c \in P(\mathsf{c})$, $e \in P(\mathsf{e})$ and $\rho \in r(v', c, e)$. So, ρ is of the form $((\hat{v}_1' \parallel \ldots \parallel \hat{v}_n'), \hat{c}, \hat{e}, \pi)$. For each $i \in \{1, \ldots, n\}$ we have that $\rho^{(i)} \in r_i(v_i', c_i, e)$, where $c_i = (v_1 \parallel \ldots \parallel v_{i-1} \parallel v_{i+1} \parallel \ldots \parallel v_n \parallel c)$. Thus, $\rho^{(i)} \in \gamma_i$ and so $\rho \in \gamma_i$. Hence, $\rho \in \gamma_1 \cap \cdots \cap \gamma_n$.

Now, let us show that condition (i) holds. The proof for condition (ii) is very similar. Let $\rho \in r(v, c, e) \cap \alpha_n$, for some c and e. Let u_k denote $(v_1' \parallel \ldots \parallel v_k' \parallel v_{k+1} \parallel \ldots \parallel v_n)$. Note that $u_0 = v$ and $u_n = v'$. We will show, by induction, that for each $k \in \{0, \ldots, n\}$ there exists e_k and ρ_k such that $\rho_k \in r(u_k, c, e_k) \cap \alpha_{n-k}$ and $\rho_k \sim \rho$. Note that, for $k = 0$, we can simply take $e_0 = e$ and $\rho_0 = \rho$. So, let us assume that the above holds for $k-1$. We will show that it also holds for k. So, we have some e_{k-1} and $\rho_{k-1} \sim \rho$ such that $\rho_{k-1} \in r(u_{k-1}, c, e_{k-1}) \cap \alpha_{(n-k+1)}$. It follows that $\rho_{k-1}^{(k)} \in \alpha_{(n-k+1)}$ and $\rho_{k-1}^{(k)} \in r_i(v_k, c^*, e_{k-1})$, where $c^* = (v_1' \parallel \ldots \parallel v_{k-1}' \parallel v_{k+1} \parallel \ldots \parallel v_n \parallel c)$. By coercion resistance of T_k, there exists e_k and $\rho_k' \in r(v_k', c^*, e_k) \cap \alpha_{n-k}$ such that

$$\mathsf{reenc}(\{x\}_k^r, k, r') = \{x\}_k^{r+r'}$$

$$\mathsf{reenc}(\mathsf{reenc}(x,k,r),k,r') = \mathsf{reenc}(x,k,r+r')$$

$$\{m_1\}_k^{r_1} \times \{m_2\}_k^{r_2} = \{m_1 \times m_2\}_k^{r_1+r_2}$$

$$\mathsf{distdec}(p_1,\ldots,p_k) = m \quad \text{where } p_i = \mathsf{dshare}(\{m\}_Y^r, x_i) \text{ with } Y = (\mathrm{pub}(x_1) \times \cdots \times \mathrm{pub}(x_k))$$

$$\mathsf{distpet}(p_1,\ldots,p_k) = \mathsf{T} \quad \text{where } p_i = \mathsf{petshare}(\{m\}_Y^r, \{m\}_Y^{r'}, x_i) \text{ with } Y = (\mathrm{pub}(x_1) \times \cdots \times \mathrm{pub}(x_k))$$

Fig. 2. The equational theory E contains the equational theory depicted in Figure 1, the equations listed above plus associativity and commutativity of $+$ and $\times$ as well as the equations $((x \doteq x) = \mathsf{T})$, $(\mathsf{T} \vee x = \mathsf{T})$, $(\mathsf{T} \wedge \mathsf{T} = \mathsf{T})$, and $(x \vee \mathsf{T} = \mathsf{T})$.

$\rho'_k \sim_k \rho^{(k)}_{k-1}$. Let ρ_k be such that $\rho^{(k)}_k = \rho'_k$. Hence, $\rho_k \sim \rho_{k-1}$ (as the coercer can see more in T_k than in T). By transitivity of $\sim$, we have $\rho_k \sim \rho$. We also have that $\rho_k \in r(u_k, c, e_k)$ and $\rho_k \in \alpha_{n-k}$.

APPENDIX B
CIVITAS

A. Cryptographic Primitives

To model the Civitas protocol, we extend the signature Σ_{ex} from Section III and its equational theory (see Fig. 1) by the following function symbols, with E as defined in Figure 2 being the associated equational theory.

The term $\mathsf{reenc}(c,k,r)$ represents a *reencryption* of a ciphertext c under a public key k with randomness r. Encryption $\{m\}_k^r$ of m under a public key k with randomness r is considered to be homomorphic (see Figure 2).

Distributed decryption is modeled as follows. Suppose that $x_1,\ldots,x_n$ are *private key shares* of some parties $a_1,\ldots,a_n$. Then, $\mathrm{pub}(x_1),\ldots,\mathrm{pub}(x_n)$ denote the corresponding *public key shares*. The *distributed public key* of $a_1,\ldots,a_n$ is computed as $K = \mathrm{pub}(x_1) \times \cdots \times \mathrm{pub}(x_n)$. To decrypt a ciphertext $c = \{m\}_K^r$, the cooperation of all parties $a_1,\ldots,a_n$ is necessary: each a_i posts his/her *public decryption share* $p_i = \mathsf{dshare}(c, x_i)$. Now, the result of decryption can be computed from these shares: $m = \mathsf{distdec}(p_1,\ldots,p_n)$ (see Figure 2).

In a very similar way we model a *distributed plaintext equivalence test* (PET), which can be used to determine, whether, for two ciphertexts c and c', the plaintexts of c and c' are the same, without revealing anything more about these plaintexts. Suppose, again that $x_1,\ldots,x_n$ are private key shares of $a_1,\ldots,a_n$ and $K = \mathrm{pub}(x_1) \times \cdots \times \mathrm{pub}(x_n)$ is their distributed public key. To perform a PET on ciphertexts $c = \{m\}_K^r$ and $c' = \{m'\}_K^{r'}$, each a_i posts his/her *public PET share* $p_i = \mathsf{petshare}(c, c', x_i)$. Now, the result of the PET can be computed from these shares: $\mathsf{distpet}(p_1,\ldots,p_n) = \mathsf{T}$ iff $m = m'$ (see Figure 2).

B. Zero-knowledge Proofs

We model zero-knowledge proofs following the approach of [5]. A zero-knowledge proof is represented by a term $P = \mathsf{ZK}_\varphi^{n,k}(t_1,\ldots,t_n;s_1,\ldots,s_k)$ where $t_1,\ldots,t_n$ is *the private component* (the proof will keep these terms secret), $s_1,\ldots,s_k$ is *the public component*, and φ is a term built upon variables $x_1,\ldots,x_n,y_1,\ldots,y_n$ (no other variables and no nonces can occur in this term; x_i is intended to refer to t_i, while y_i is intended to refer to s_i), called *the formula of P*. The symbols $\doteq$, $\vee$, and $\wedge$ mentioned in Figure 2 may be used in such formulas. We consider the following equational theory for zero-knowledge proofs:

$$\mathsf{public}(\mathsf{ZK}_\varphi^{n,k}(t_1,\ldots,t_n,s_1,\ldots,s_k)) = \langle \varphi, s_1,\ldots,s_k\rangle$$

$$\mathsf{check}(\mathsf{ZK}_\varphi^{n,k}(t_1,\ldots,t_n,s_1,\ldots,s_k)) = \mathsf{T} \quad \text{if } \varphi[t_i/x_i, s_i/y_i] \equiv_E \mathsf{T}.$$

This means that the operation public can be used to reveal the public component of a zero-knowledge proof and check can be used to check validity of a proof. Note that the above equations employ the equational theory E.

For example, assume we want to model a designated-verifier reencryption proof $\mathsf{DVRP}(\alpha, x;\ m, m', k, k_v)$. Intuitively, such a proof shows that m' is a reencryption of m under k; k_v is the public key of the designated verifier who, having the corresponding private key, is able to forge a faked proof; α is additional randomness. The proof is valid if either (a) $m' = \mathsf{reenc}(m,k,x)$ or (b) $k_v = \mathrm{pub}(x)$, i.e. x is a private key associated with the public key k_v of the designated verifier. This can be captured formally by a zero-knowledge proof $\mathsf{ZK}_\varphi^{2,4}$ with $\varphi = \big(y_2 \doteq \mathsf{reenc}(y_1, y_3, x_2) \vee y_4 \doteq \mathrm{pub}(x_2)\big)$.

In addition to DVRPs, Civitas also makes use of the following zero-knowledge proofs. It is straightforward to model these zero-knowledge proofs by terms of the form $\mathsf{ZK}_\varphi^{n,k}(t_1,\ldots,t_n;s_1,\ldots,s_k)$ as illustrated for DVRPs.

- $\mathsf{KnowPriv}(x;\ y)$ represents a proof of knowledge of the private key x associated with the given public key y (i.e. $y = \mathrm{pub}(x)$).
- $\mathsf{ProofDShare}(x;\ p, y, c)$ represents a proof that p is the public share for the distributed decryption of c w.r.t. y, i.e. $p = \mathsf{dshare}(c,x)$ and $y = \mathrm{pub}(x)$.
- $\mathsf{ProofPETShare}(x;\ p, y, c, c')$ represents a proof that p is the public share for distributed PET of ciphertexts c and c' w.r.t. y, i.e. $p = \mathsf{petshare}(c, c', x)$ and $y = \mathrm{pub}(x)$.
- $\mathsf{OneOf}_l(r;\ m, k, \vec{b})$ represents a proof that m is an encryption under k of one of the values in $\vec{b} = \langle b_1,\ldots,b_l\rangle$ ($m = \{b\}_k^r$, where b is an element of $\vec{b}$).
- $\mathsf{MutKnow}(m, m', r, r';\ c, c', k)$ represents a proof of mutual knowledge of the plaintexts contained in the ciphertexts c and c' ($c = \{m\}_k^r$ and $c' = \{m'\}_k^{r'}$).
- $\mathsf{ProofMix}_l(\vec{r};\ \vec{c}_1, \vec{c}_2, k)$ where $\vec{r}, \vec{c}_1, \vec{c}_2$ are tuples of length l, represents a proof that $\vec{c}_2$ is obtained from ciphertexts $\vec{c}_1$ by mixing (i.e. applying some permutation) and reencryption ($\vec{r}$ is the collection of random values used for reencryption), i.e. $\vec{c}_2[\pi(i)] = \mathsf{reenc}(\vec{c}_1[i], k, \vec{r}[i])$, for some permutation π of $\{1,\ldots,l\}$.

C. Protocol Description

The protocol participants are as described in Section V. In what follows, we assume that i ranges over the set $\{0,\ldots,m\}$ and j ranges over $\{0,\ldots,k\}$. For a participant a, we will write

$\mathsf{sig}_a\{m\}$ instead of $\mathsf{sig}_{k_a}\{m\}$. We will also write $\mathrm{pub}(a)$ instead of $\mathrm{pub}(k_a)$.

Setup phase. We do not model here the first part of the setup phase, where the supervisor posts the ballot design (the set of valid votes), identifies the tellers by posting their public keys, and posts the electoral roll (the set of authorized voters). Instead, we assume that the public keys of the voting authorities, the ballot design, and the electoral roll are fixed. Below, we describe the remaining steps of this phase.

The tabulation tellers collectively generate a public key for a distributed encryption scheme and post it on the bulletin board:

$$\text{(KGen1)}\quad \mathsf{T}_j \to \mathsf{B}:\ \ \mathsf{sig}_{\mathsf{T}_j}\{h(y_j)\}$$
$$\text{(KGen2)}\quad \mathsf{T}_j \to \mathsf{B}:\ \ \mathsf{sig}_{\mathsf{T}_j}\{y_j,\ \mathsf{KnowPriv}(x_j;y_j)\}$$

where $y_i = \mathrm{pub}(x_i)$ and x_i is a random value, the private key share of T_j. After the first step, all the tellers wait until all commitments are available. After the second step, they verify all the proofs. Now, $(y_1 \times \cdots \times y_k)$ is the distributed public key of $\mathsf{T}_1, \ldots, \mathsf{T}_k$. We will refer to this key by K_T.

Next, each registration teller R_j randomly generates *credential* shares s_{ij}, for each voter v_i, and posts these shares:

$$\text{(Cred)}\quad \mathsf{R}_j \to \mathsf{B}:\ \ \mathsf{sig}_{\mathsf{R}_j}\{i, S_{ij}\} \quad \text{(for each } i,j)$$

where $S_{ij} = \{s_{ij}\}^{r_{ij}}_{K_\mathsf{T}}$ with randomness r_{ij}. The *public credential of* v_i is now publicly computable as $S_i = (S_{i1} \times \cdots \times S_{ik})$.

Registration phase. Voters register to acquire their private credentials:

$$\text{(Reg1)}\quad \mathsf{v}_i \to \mathsf{R}_j:\ \ \mathsf{request}$$
$$\text{(Reg2)}\quad \mathsf{R}_j \to \mathsf{v}_i:\ \ s_{ij},\ \bar{r}_{ij},\ D_{ij}$$

where $\bar{r}_{ij} = (r_{ij} + w_{ij})$, for randomness w_{ij}, and $D_{ij} = \mathsf{DVRP}(\delta_{ij}, w_{ij};\ S_{ij}, S'_{ij}, K_\mathsf{T}, \mathrm{pub}(\mathsf{v}_i))$, for $S'_{ij} = \{s_{ij}\}^{\bar{r}_{ij}}_{K_\mathsf{T}}$ ($\equiv_E \mathsf{reenc}(S_{ij}, K_\mathsf{T}, w_{ij})$) and randomness δ_{ij}. The voter v_i verifies this proof, which shows that S'_{ij} is a reencryption of S_{ij}. Given all the private shares, v_i computes his/her private credential as $s_i = (s_{i0} \times \cdots \times s_{ik})$.

Voting phase. Each voter sends his/her *ballot* b_i containing his/her vote and public credential to all ballot boxes:

$$\text{(Vote)}\quad \mathsf{v}_i \to \mathsf{X}_j:\ \ b_i = \langle \{s_i\}^{r_i}_{K_\mathsf{T}}, \{v_i\}^{r'_i}_{K_\mathsf{T}}, P^i_V, P^i_K \rangle$$

where v_i is the vote chosen by v_i and the terms P^i_V and P^i_K are zero-knowledge proofs. The term $P^i_V = \mathsf{OneOf}_l(r'_i; \{v_i\}^{r'_i}_{K_\mathsf{T}}, K_\mathsf{T}, b_1, \ldots, b_l)$ shows that the vote v_i is well-formed with respect to the ballot design (v_i is one of the valid votes $b_1, \ldots, b_l$). The term $P^i_K = \mathsf{MutKnow}(s_i, v_i, r_i, r'_i; \{s_i\}^{r_i}_{K_\mathsf{T}}, \{v_i\}^{r'_i}_{K_\mathsf{T}}, K_\mathsf{T})$ shows that the submitter simultaneously knows s_i and v_i. We call the value v_i *the vote of* b_i and s_i the *ballot credential* of b_i; $\{v_i\}^{r'_i}_{K_\mathsf{T}}$ will be called *the encrypted vote of* b_i, and $\{s_i\}^{r_i}_{K_\mathsf{T}}$ will be called *the encrypted credential of* b_i.

Tabulation phase. Before the tabulation phase, each ballot box posts a commitment to its content on the bulletin board:

$$\text{(Comm1)}\quad \mathsf{X}_j \to \mathsf{B}:\ \ \mathsf{sig}_{\mathsf{X}_j}\{j, C_j\}$$

where $C_j = h(\mathrm{content}(\mathsf{X}_j))$. The supervisor then posts his own signatures on all these commitments, defining the set of votes to be tabulated:

$$\text{(Comm2)}\quad \mathsf{S} \to \mathsf{B}:\ \ \mathsf{sig}_{\mathsf{S}}\{j, C_j\}$$

Then, the tabulation tellers collectively tally the election:

(1) All tabulation tellers *retrieve* the ballots from all ballot boxes and the public credentials from the bulletin board. They also verify that the content of ballot boxes corresponds to the commitments posted in (Comm2).

(2) All tabulation tellers *check the proofs* in retrieved ballots and eliminate any ballot with an invalid proof.

Note that (1) and (2) are performed by each teller independently and that these steps are publicly computable. We denote by B the resulting set of ballots.

(3) *Duplicate elimination* is performed, by running $\mathsf{PET}(c, c')$, for all encrypted ballot credentials c, c' from distinct ballots in B:

$$\text{(PET1)}\quad \mathsf{T}_j \to \mathsf{B}:\ \ \mathsf{sig}_{\mathsf{T}_j}\{\alpha_j(c,c'),\ P_{\alpha_j}(c,c')\}$$

where $\alpha_j(c,c') = \mathsf{petshare}(c, c', x_j)$ and $P_{\alpha_j}(c,c') = \mathsf{ProofPETShare}(x_j; \alpha_j(c,c'), y_j, c, c')$. Now, each teller waits until all the tellers post their shares and verifies the proofs. The result of PET for c, c' is $\mathsf{distpet}(\alpha_0, \ldots, \alpha_k)$, which is publicly computable. For each two ballots for which PET holds true, only one is kept (according to some fixed policy).

(4) *Mixing ballots* is performed on the list of remaining ballots $\vec{u}_0$. Each tabulation teller in turn applies its own random permutation π^b_j with reencryption. We assume that $\vec{u}_j$ is the input to the j-th teller:

$$\text{(Mix1)}\quad \mathsf{T}_j \to \mathsf{B}:\ \ \mathsf{sig}_{\mathsf{T}_j}\{\vec{u}_{j+1},\ P_{u_j}\}$$

where $\vec{u}_{j+1}$ with $\vec{u}_{j+1}[\pi^b_j(i)] = \mathsf{reenc}(\vec{u}_j[i], K_\mathsf{T}, \vec{r}_j[i])$ is the resulting mix and $P_{u_j} = \mathsf{ProofMix}(\vec{r}_j;\ \vec{u}_j, \vec{u}_{j+1}, K_\mathsf{T})$, with $\vec{r}_j$ being a vector of random values. In a similar manner, *mixing of credentials* is performed on the list of public credentials.

(5) In this step, *invalid ballot elimination*, ballots without valid credentials are eliminated. For each ballot with the encrypted credential c, $\mathsf{PET}(c, c')$ is performed against every public credential c'. This is done similarly to (PET1). If the test fail for all c', the ballot is removed.

(6) Finally, *decryption* is performed for each of the remaining ballots, i.e., decryption is applied to the encrypted vote c of each of the remaining ballots (but not to the encrypted credentials):

$$\text{(Decr)}\quad \mathsf{T}_j \to \mathsf{B}:\ \ \mathsf{sig}_{\mathsf{T}_j}\{\gamma_j(c),\ P_{\gamma_j}(c)\}$$

where $\gamma_j(c) = \mathsf{dshare}(c, x_j)$ and $P_{\gamma_j}(c) = \mathsf{ProofDShare}(x_j;\ \gamma_j(c), y_j, c)$. Each teller waits until the remaining tellers submit their shares and verifies the proofs. Now, the decrypted vote is publicly computable as $v = \mathsf{distdec}(\gamma_0, \ldots, \gamma_k)$.