2013 IEEE Symposium on Security and Privacy

Caveat Coercitor: coercion-evidence in electronic voting

Gurchetan S. Grewal and Mark D. Ryan
School of Computer Science,
University of Birmingham, UK
research@gurchetan.com,
m.d.ryan@cs.bham.ac.uk

Sergiu Bursuc
Centre for Secure
Information Technologies,
Queen's University of Belfast, UK
s.bursuc@qub.ac.uk

Peter Y. A. Ryan
Faculté des Sciences, de la Téchnologie
et de la Communication,
Université du Luxembourg
peter.ryan@uni.lu

***Abstract*—The balance between coercion-resistance, election verifiability and usability remains unresolved in remote electronic voting despite significant research over the last few years. We propose a change of perspective, replacing the requirement of coercion-resistance with a new requirement of coercion-evidence: there should be public evidence of the amount of coercion that has taken place during a particular execution of the voting system.**

We provide a formal definition of coercion-evidence that has two parts. Firstly, there should be a *coercion-evidence test* that can be performed against the bulletin board to accurately determine the degree of coercion that has taken place in any given run. Secondly, we require *coercer independence*, that is the ability of the voter to follow the protocol without being detected by the coercer.

To show how coercion-evidence can be achieved, we propose a new remote voting scheme, Caveat Coercitor, and we prove that it satisfies coercion-evidence. Moreover, Caveat Coercitor makes weaker trust assumptions than other remote voting systems, such as JCJ/Civitas and Helios, and has better usability properties.

***Keywords*-Coercion resistance; coercion evidence; electronic voting; verifiable elections; security protocols; security models; usability**

I. Introduction

A. Background and motivation

Current research in electronic voting focuses on trying to design usable voting systems that satisfy the properties of *coercion-resistance* and *verifiability*. Coercion-resistance is a fundamental, and strong, property of electronic voting systems. It states that a voter should be able to cast his vote as intended, even in presence of a coercer that may try to force him to cast a different vote. On the other hand, verifiability allows to verify that the outcome of the election reflects correctly the voters choice. A voter can verify that the ballot has been correctly recorded on the bulletin board, and anyone can verify that the recorded ballots from eligible voters are tallied correctly.

Especially in remote voting, coercion-resistance and verifiability are extremely hard to achieve together and this leads to schemes of questionable usability or to trust assumptions that are difficult to meet in practice. Perhaps the system that comes closest to satisfying both those properties in their strongest form is JCJ/Civitas [16], [12]. However, it does so at the cost of usability:

- the voters are required to run a multi-party protocol with registrars in order to construct their credentials.
- to avoid coercion, the voters must be capable of creating and passing off fake credentials and fake proofs.
- a trusted registrar and an untappable channel for that registrar are necessary.

Another prominent system is Helios [2], [3], which is very usable but is for low-coercion elections, and has only very weak methods to resist coercion. For example, in some variants of Helios (and also in the Estonian system), only the last vote counts. This is supposed to help a voter resist coercion by re-voting. However, if the voting credentials have been leaked, an attacker can always override a legitimate vote.

Moreover, in any E-voting system, there may be cases where the voter is subject to *silent coercion*, which means being coerced without noticing. This can happen when the voting credentials are leaked for various reasons: dishonest registrars, insecure communication channels, malware on the voting machine. In that case, the voter can be subject to an impersonation attack and the intended vote may be replaced by a vote chosen by the attacker. This is a more general form of coercion which is not addressed even in a strong system like JCJ/Civitas, because there is no way for the voter to find out that the voting credentials have been compromised. Variations of JCJ/Civitas [11], [4] aim to improve the usability of voter strategies for achieving coercion-resistance, but they do not consider the problem of silent coercion explicitly. In Helios as well, if the credentials of a voter have been leaked, they can be used to cast a vote on voter's behalf.

B. Coercion-evidence

This paper takes as its starting point the observation that it has been impossible to achieve all the requirements simultaneously. Since result verifiability and usability may be considered "non-negotiable", we consider a setting in which coercion-resistance may be relaxed. Nevertheless, to defend against coercion (both explicit coercion and silent coercion), we propose the property of *coercion-evidence*. This means that unforgeable evidence about the degree of

1081-6011/13 $26.00 © 2013 IEEE
DOI 10.1109/SP.2013.32

coercion that took place is included in the election output. Authorities, observers and voters can examine this evidence and use it to determine whether the election result carries a mandate for the winning candidate. Thus, election authorities can decide to consider the election as valid or not, leading to disincentivisation of coercion.

In some situations, dishonest voters could try to disrupt the election by faking being coerced. Since an individual voter can only fake his or her own coercion, this strategy would be effective only if the winning margin is low. Although this doesn't prevent coercion evidence, it could lower the attractiveness of our system. We will discuss practical mitigations as well as research perspectives to address this challenge.

C. Caveat Coercitor

We propose Caveat Coercitor, an electronic voting scheme intended to support remote (e.g. internet) voting and to be practically deployable. By shifting away from the conventional wisdom of coercion-resistance in favour of coercion-evidence, it tries to find a "sweet spot" between security and usability. Coercion-evidence also means that the system disincentivises coercion. In Caveat Coercitor, the most the coercer can achieve is to cancel a voter's vote. This is rather weaker than the usual notion of forced abstention, because the number of such cancelled votes is revealed as part of the election output. Since the election result will be considered valid only if those cancelled votes would not have affected it, this means that the coercer cannot significantly affect the outcome of the election.

Caveat Coercitor borrows some ideas from JCJ/Civitas, in particular the notion of private and public credentials, but at the same time it is designed such that, unlike JCJ/Civitas:

- The generation of credentials does not have to be distributed, making it simpler and more usable.
- The registration phase needs to be secured only with "best effort" confidentiality of credentials. If this security is broken, coercion may be possible, but it will be evident. Credentials can be sent by post, SMS or email. Effort should be made to keep the credentials confidential, but even if this does not succeed the core property of coercion evidence is not broken.
- This property does not rely on trustworthiness of any registrar, and does not require untappable channels during registration. We require the channel to be *resilient*: voters will receive their credentials, perhaps after multiple attempts.
- Voters can directly verify that their private credential matches the public one on the bulletin board (voters can be given the randoms used in the encryption). There is no need for zero-knowledge proofs.
- The system does not require the use of fake credentials. If a voter is coerced, he can give away his real credential and vote normally. Coercion will be evident in that case.
- The system addresses the problem of silent coercion. If a voting credential has been leaked and misused by an intruder, this will be evident to any external observer.

Our contributions. 1) We introduce the property of *coercion-evidence* and we propose a general formal definition (section III). 2) We propose a new remote voting scheme, *Caveat Coercitor*, that achieves coercion-evidence (section IV). 3) We weaken the trust assumptions for remote electronic voting and we discuss how coercion-evidence is useful in practice. We also hint how Caveat Coercitor could be adapted to address the long-standing problem of an untrusted voting device (section V). 4) We perform a rigorous analysis of the fact that Caveat Coercitor satisfies coercion-evidence (section VI).

II. Cryptographic primitives

We will rely on the following cryptographic primitives in this paper:

Distributed El-Gamal [8]: We will consider the El-Gamal encryption scheme, where the secret key can be distributed (relying on e.g. [20]) among a set of trustees $T_1, \ldots, T_n$. In that case, the private key is split as $x = x_1 + \ldots + x_n$ and each of T_i holds a secret share x_i.

The El-Gamal encryption of a plaintext m with a public key k and random r will be denoted in the following by $\{m\}_k^r$, or simply by $\{m\}_k$ when r is not important or is clear from the context. The private part of a public key k will be denoted by $\mathsf{priv}(k)$. The decryption of a ciphertext m with a private key x will be denoted by $\mathsf{dec}(m, x)$.

We consider an exponential version of El-Gamal where ciphertexts can be homomorphically combined to compute the addition of plaintexts: we have $\{m_1\}_k^{r_1} * \{m_2\}_k^{r_2} = \{m_1 + m_2\}_k^{r_1+r_2}$.

Re-encryption and mix nets [10], [15]: Given a ciphertext $\{m\}_k^r$ constructed using the public key k any party can compute another ciphertext $\{m\}_k^{r+r'}$ that encrypts the same plaintext using the same key k, by using a new random r'. We will denote the re-encryption of a ciphertext m with a given random r' by $\mathsf{renc}(m, r')$.

A re-encryption mix net is a set of agents $\mathcal{M}$ that takes as input a sequence of ciphertexts $\mathcal{S} = m_1, \ldots, m_k$ and outputs a sequence of ciphertexts $\mathcal{S}' = m'_1, \ldots, m'_k$ that is a re-encryption mix of $\mathcal{S}$. Specifically, $\mathcal{S}'$ is a formed by re-encryption of elements in a permutation of $\mathcal{S}$: there is a permutation σ of $\{1, \ldots, k\}$ and a sequence of randoms $r_1, \ldots, r_k$ such that $m'_1 = \mathsf{renc}(m_{\sigma(1)}, r_1), \ldots, m'_k = \mathsf{renc}(m_{\sigma(k)}, r_k)$. Moreover, if at least one element of $\mathcal{M}$ is not to be controlled by an adversary $\mathcal{C}$, the permutation σ remains secret to $\mathcal{C}$.

Plaintext equivalence test [14]: Given two ciphertexts $\{m_1\}_k^{r_1}$ and respectively $\{m_2\}_k^{r_2}$, encrypted with the same key k, whose plaintexts are m_1 and respectively m_2, a

plaintext equivalence test (pet) allows the holders of the decryption key to demonstrate that $m_1 = m_2$, without revealing the decryption key or any information about m_1 or m_2. For two ciphertexts c_1 and c_2, we will denote by $\mathsf{pet}(c_1, c_2) = \mathsf{ok}$ iff the plaintext equivalence test holds for c_1 and c_2. Key holders only provide pets as indicated by the protocol.

Zero-knowledge proofs for verifiability: Decryption, re-encryption mixnets and plaintext equivalence tests can be accompanied by non-interactive zero-knowledge proofs that attest of the fact that these operations have been correctly performed, without revealing sensitive information like the decryption key. Zero-knowledge proofs are crucial in order to ensure universal verifiability of the election, while preserving user privacy. We will see that they are essential for coercion-evidence as well. We will denote by $\mathsf{petproof}(c_1, c_2, res)$ a zero-knowledge proof that the result of the plaintext equivalence test applied to c_1 and c_2 is res.

III. Coercion-evidence

In this section we do not fix the model that is used to specify security protocols. The definition can be instantiated in any computational or symbolic model. We assume that the model defines the notion of a run and of a bulletin board for an e-voting system. As usual, we assume that there is an attacker (or coercer, intruder) that controls the communication network. When a message is sent to the environment, it is assumed to be in the control of the attacker.

We assume that each eligible voter $\mathcal{A}$ has a unique voting credential $s_{\mathcal{A}}$. There is a registration phase where the voters can obtain their correct voting credentials. In other words, we assume a registration protocol that ensures availability and integrity of voting credentials. We do not assume that this protocol ensures the secrecy of voting credentials. In particular, the voting credentials of several voters may have been leaked during registration.

The voting phase allows a voter with voting credentials s and the intended vote v to execute a program $\mathcal{V}(s, v)$, whose definition depends on the protocol. $\mathcal{V}(s, v)$ may allow a voter to cast one or multiple ballots, abstain or report coercion. We say that a voter $\mathcal{A}$, with credential $s_{\mathcal{A}}$ and intended vote $v_{\mathcal{A}}$, follows the specification of the protocol if its interaction with the protocol consists in executing $\mathcal{V}(s_{\mathcal{A}}, v_{\mathcal{A}})$.

Definition 1 (coerced, dishonest, free voters): Let τ be a run of an electronic voting system and $\mathcal{A}$ be a voter with credential $s_{\mathcal{A}}$ who intends to vote $v_{\mathcal{A}}$. We say that the voter $\mathcal{A}$ is:

- *coerced* (to vote for $v_{\mathcal{C}}$), if
  - a ballot with credential $s_{\mathcal{A}}$ and vote $v_{\mathcal{C}}$, with $v_{\mathcal{C}} \neq v_{\mathcal{A}}$, is present on the bulletin board in the voting phase
  - $\mathcal{A}$ follows the specification of the protocol in the run τ.
- *dishonest*, if $\mathcal{A}$ does not follow the specification of the protocol in the run τ.
- *free*, if $\mathcal{A}$ is not coerced in the run τ

From our definitions, it follows that honest voters always cast a vote for their intended choice, and they do not cast a vote for a different candidate. If a voter $\mathcal{A}$ is coerced (resp. dishonest), we let $s_{\mathcal{A}} \in \delta_c(\tau)$ (resp. $s_{\mathcal{A}} \in \delta_d(\tau)$) and we will sometimes say that $s_{\mathcal{A}}$ is a coerced (resp. dishonest) credential. Thus,

- $\delta_c(\tau)$ is the set of credentials for coerced voters.
- $\delta_d(\tau)$ is the set of credentials for dishonest voters.

The number $|\delta_c(\tau)|$ is called the degree of coercion in the run τ.

A coerced voter is one whose credentials have been leaked and misused by an intruder. This may have happened by explicit coercion or by silent coercion. We assume that an honest voter would nevertheless obtain the voting credentials and cast a vote normally for the intended choice. We consider voters to be coerced only if their credentials have been used to cast a vote for a candidate that is different from their intended choice.

A dishonest voter can misbehave in arbitrary ways, and appear to be coerced even when he is not. It is also possible that a voter did not follow the protocol due to a genuine mistake, but for simplicity we will consider such voters dishonest. Note that a free voter can also be dishonest.

We consider an equivalence relation on runs, called *indistinguishability*, that models the inability of an observer to tell the difference between two runs that differ on some private data. For two runs τ and τ', we denote by $\tau \sim \tau'$ if they are in the indistinguishability relation (defined in section VI). A run τ is by definition *complete* if its corresponding bulletin board contains the outcome of the election.

Definition 2 (coercion-evidence): An electronic voting system $\mathcal{EVS}$ satisfies *coercion-evidence* if:

1) **Coercion-evidence test:** there exists a test $\mathsf{ce}(\_)$ that takes as input data on the bulletin board and outputs a number such that, for every complete run τ of $\mathcal{EVS}$, we have
$$|\delta_c(\tau)| \leq \mathsf{ce}(\tau) \leq |\delta_c(\tau)| + |\delta_d(\tau)|$$
2) **Coercer independence:** Let $\mathcal{A}$ be a voter with credential $s_{\mathcal{A}}$ who intends to vote for $v_{\mathcal{A}}$. Then, for every run τ of $\mathcal{EVS}$ where
   - the voter $\mathcal{A}$ follows the protocol by executing $\mathcal{V}(s_{\mathcal{A}}, v_{\mathcal{A}})$
   - for every candidate v_i, there is a free honest voter $\mathcal{A}_i$ that follows the protocol by executing $\mathcal{V}(s_{\mathcal{A}_i}, v_i)$

there exists a run τ', where $\mathcal{A}$ does not execute $\mathcal{V}(s_{\mathcal{A}}, v_{\mathcal{A}})$, such that $\tau \sim \tau'$.

The first part of coercion-evidence requires the existence of a test that can be applied to data that is available on the bulletin board in order to determine the degree of coercion in a given run. Note however that we do not require the test to return the exact degree of coercion, but allow for an over-aproximation. Indeed, for an external party, there is no way of telling the difference between a coerced voter and a dishonest voter who simulates being coerced. That is why we have to allow an upper bound of $|\delta_c(\tau)| + |\delta_d(\tau)|$.

The second part of coercion-evidence requires that the coercer can not make a distinction between a run τ where the voter has followed the protocol normally and has cast a vote for the desired candidate, in spite of being coerced, and a run τ' where the voter has followed the coercer's instructions. We require that for each coerced voter there exists a free honest voter, so that in each case there is at least one vote for the coerced choice, otherwise the coercer would trivially detect that the coerced voter did not obey the instructions.

Note that our definition also counters vote buying and achieves effective receipt-freeness. A receipt for a ballot with valid credentials and a certain candidate, even if that ballot is cast, is not useful to a coercer because it does not guarantee that the vote will be counted.

One might imagine a system that allows voters to declare that they are being coerced, perhaps with a tick-box on the ballot form, or a separate channel by which to report coercion later. This may work if a voter knows they are being coerced. But such a system does not satisfy coercion-evidence according to our definition, since our definition requires evidence even in the case of silent coercion.

IV. Caveat Coercitor

Caveat Coercitor is a variation of JCJ/Civitas[16], [12]. We first recall JCJ/Civitas and then describe how Caveat Coercitor deviates from it.

A. Overview of JCJ/Civitas

JCJ/Civitas is designed around the notion of credentials (with a private and a public part), that allow eligible voters to authenticate their ballots. To allow coercion-resistance, JCJ/Civitas distributes credential generation among a set of parties called registrars. To resist coercion, the voter has the ability to generate a fake credential and a fake proof, that look as real to the coercer. To ensure this, it is assumed that at least one of the registrars is not corrupted by the coercer and that the voter can communicate with this registrar using an untappable channel.

The participants of the protocol are

- the set of registrars $\mathcal{R}$, whose role is to authenticate eligible voters and help generate their credentials.
- the set of talliers $\mathcal{T}$, whose role is to generate and publish the public key of the election. Each of them holds a secret share of the corresponding private key, that will be used for distributed decryption and plaintext equivalence tests.
- the set of eligible voters $\mathcal{A}_1, \ldots, \mathcal{A}_n$.
- a re-encryption mix net $\mathcal{M}$, whose role is to anonymize the set of cast ballots before verification of their eligibility and their decryption.
- the bulletin board $\mathcal{B}$, whose role is to record the manipulation of ballots at all stages of the election, from their recording to their tallying. It also records proofs of correct ballot handling submitted by $\mathcal{R}, \mathcal{T}$ and $\mathcal{M}$, that can be checked by external auditors.

Trust assumptions. A coercer may control some of $\mathcal{A}_1, \ldots, \mathcal{A}_n$, some of $\mathcal{R}$, some of $\mathcal{T}$ and some of $\mathcal{M}$. For a voter $\mathcal{A}_i$ to achieve coercion-resistance, at least one $\mathcal{A}_j$, $j \neq i$, one of $\mathcal{R}$, one of $\mathcal{T}$, one of $\mathcal{M}$ and the voting machines must be outside the control of the coercer. Moreover, there must be an untappable channel from $\mathcal{A}_i$ to the trusted registrar, so that the coercer can not get hold of the real credential, and an anonymous channel for the voter to submit the ballot. The voter must also trust the voting machine to correctly construct the voting ballot and to verify the zero-knowledge proofs provided by registrars.

A summary of the protocol is as follows:

Initialisation. The election starts with talliers $\mathcal{T}$ generating the public key pk of the election in a distributed manner, such that no minority of talliers can recover the private key $\mathsf{priv}(pk)$ [20] and the decryption is distributed [8]. The public part of the key is published on the bulletin board.

Voting credentials. In order to cast their vote, every eligible voter has a private credential. Private credentials will be denoted by the letter s (decorated with various indices). For a given private credential s, there exists a corresponding public credential $\{s\}^r_{pk}$, which will be published on the electoral roll. We will denote public credentials (and their re-encryptions) by the letter S (decorated with indices).

Registration. By running a separate protocol with each of the registrars, the voter $\mathcal{A}$ obtains a private share $s^i_{\mathcal{A}}$ of her voting credential. Let $m = |\mathcal{R}|$. The private voting credential of the voter is the sum of all private credential shares, i.e. $s_{\mathcal{A}} = s^1_{\mathcal{A}} + \ldots + s^m_{\mathcal{A}}$, and the public credential is the homomorphic combination of all public credential shares registered on the electoral roll, i.e. $\{s_{\mathcal{A}}\}^r_{pk} = \{s^1_{\mathcal{A}}\}^{r_1}_{pk} * \ldots * \{s^m_{\mathcal{A}}\}^{r_m}_{pk} = \{s^1_{\mathcal{A}} + \ldots + s^m_{\mathcal{A}}\}^{r_1+\ldots+r_m}_{pk}$.

Validity of voter credentials. Each registrar also provides the voter with a non-transferable proof P_{corr} of the fact that the public share $\{s^i_{\mathcal{A}}\}^{r_i}_{pk}$, that is published on the electoral roll ER, correctly encodes the private part $s^i_{\mathcal{A}}$. The proof can be verified on the voting machine of the voter.

Resisting coercion. The voter has the ability to construct a fake credential by replacing the credential share $s^i_{\mathcal{A}}$ of a

trusted registrar with a fake credential share ${s'}^i_{\mathcal{A}}$. The voter also has the ability (with the help of the voting machine) to construct a fake proof P'_{corr} showing that $\{s^i_{\mathcal{A}}\}^{r_i}_{pk}$ is an encryption of ${s'}^i_{\mathcal{A}}$.

Voting. The ballot $(\{s_{\mathcal{A}}\}^{r_s}_{pk}, \{v\}^{r_v}_{pk}, P_{\mathsf{sv}}, P_{\mathsf{corr}})$, from the voter $\mathcal{A}$, contains the encryption of the private credential $s_{\mathcal{A}}$ (with the key pk and with a different random than in the electoral roll) and the encryption of the intended vote v (with the key pk). To prevent the re-use of the same credential by a party that does not hold the private part, the ballot contains additionally a zero-knowledge proof P_{sv} of the fact that its creator knows both s and v. Additionally, a zero-knowledge proof P_{corr} proves that v is a valid vote, according to the specification decided by election authorities. The ballot $(\{s_{\mathcal{A}}\}^{r_s}_{pk}, \{v\}^{r_v}_{pk}, P_{\mathsf{sv}}, P_{\mathsf{corr}})$ is constructed by the voting machine and submitted to the bulletin board.

Verification of proofs and mixing. Before tabulation starts, the zero-knowledge proofs P_{sv} and P_{corr} of cast ballots are verified (e.g. by the talliers) and ballots with invalid proofs are discarded. The valid ballots (without the proofs) and the electoral roll are then sent to the re-encryption mix net $\mathcal{M}$ for anonymization.

Tallying. Credentials from anonymized ballots are compared, using plaintext equivalence tests, to credentials from the anonymized electoral roll, to ensure that votes to be counted are cast by eligible voters only. If multiple ballots are submitted with the same credentials, only one copy is kept according to a predefined policy, e.g. only the last vote counts. Finally, the decided set of countable votes is decrypted.

The re-encryption mix, the plaintext equivalence tests and the decrypted votes are accompanied by zero-knowledge proofs that ensure the operations have been correctly performed.

B. Caveat Coercitor: registration, voting and mixing

Caveat Coercitor follows the same scheme as JCJ/Civitas with the following differences.

Trust assumptions. We assume that one mix server in $\mathcal{M}$, one tallier in $\mathcal{T}$ and the voting machines are honest. However, to achieve coercion-evidence, Caveat Coercitor does not have to assume that any of the registrars are trusted, and neither does it require any untappable channel between voters and registrars. The registration channel is assumed only to allow the voters to obtain their credentials, and it may allow the coercer to read honest messages and inject fake messages. We discuss our trust assumptions in more detail in section V.

The **initialisation** phase and the structure of the voting ballots are exactly the same as in JCJ/Civitas.

Registration. The voters can simply receive their private credentials by email or by post. Their creation does not have to be distributed.

Validity of voter credentials. Because the voter is not required to keep the private credential secret at all cost, the proofs that the public share $\{s\}^r_{pk}$ of the credential is a correct encryption of the private share s can be significantly simplified. In fact, the voter can simply obtain from the registrar the random number r that has been used in the encryption algorithm. Then, the voter can use any device to verify that the encryption has been correctly performed.

Resisting coercion. In Caveat Coercitor voters do not have to "resist" coercion. To make coercion "evident" a voter does not need to perform any additional task other than to cast a vote for the desired candidate. He can give away the voting credential to the coercer, if asked.

Voting. The construction of voting ballots is similar to the one used in JCJ/Civitas. The fundamental difference is in how Caveat Coercitor handles multiple ballots from the same voter: a voter may submit multiple ballots, constructed on possibly different voting devices, provided they are all for the same candidate. At most one vote for one candidate will be counted for each credential. Precisely one vote will be counted if all the ballots for a given credential contain a vote for the same candidate. No votes will be counted for a credential if there are ballots for that credential corresponding to at least two distinct candidates. Instead, that credential will be detected by the coercion-evidence test.

Verification of proofs and mixing. The zero-knowledge proofs of cast ballots are verified. Before mixing, the ballots with invalid proofs are eliminated.

C. Caveat Coercitor: coercion-evidence and tallying

We have at this stage a set of anonymized ballots to be tallied and an anonymized electoral roll. In addition to eliminating ballots with fake credentials, now we need to determine the coerced credentials, remove the corresponding ballots from the ballots to be tallied and output evidence of coercion.

Coercion-evidence. Let τ be a run of Caveat Coercitor up to this phase. We let

- $\mathsf{BB}_{\mathsf{aer}}(\tau)$ be the anonymized electoral roll with each element of the form $\{s\}^r_{pk}$, for some private credential s. $\{s\}^r_{pk}$ are re-encryptions of public credentials and we denote them by (decorations of) the letter S.
- $\mathsf{BB}_{\mathsf{cast}}(\tau)$ be the set of cast ballots with valid zero-knowledge proofs of correctness. Each element of $\mathsf{BB}_{\mathsf{cast}}(\tau)$ is of the form $(\{s\}^{r1}_{pk}, \{v\}^{r1'}_{pk}, P_{\mathsf{sv}}, P_{\mathsf{corr}})$, for some private credential s and vote v. Encrypted votes will be denoted by (decorations of) the letter V.
- $\mathsf{BB}_{\mathsf{tally}}(\tau)$ be the set of anonymized ballots to be authorized relying on $\mathsf{BB}_{\mathsf{aer}}(\tau)$ and then tallied. Each element of $\mathsf{BB}_{\mathsf{tally}}(\tau)$ is of the form $(\{s\}^{r2}_{pk}, \{v\}^{r2'}_{pk})$, for some private credential s and vote v.

The talliers will execute an algorithm whose output can be used by any external observer to determine the amount

of coercion, while at the same time allowing coercer-independence. Specifically, the talliers will compute the set of credentials in $\mathsf{BB}_{\mathsf{aer}}(\tau)$ for which there are at least two ballots with two different votes in $\mathsf{BB}_{\mathsf{tally}}(\tau)$, i.e. the set

$$\mathsf{BB}_{\mathsf{ce}}(\tau) = \{S \in \mathsf{BB}_{\mathsf{aer}}(\tau) | \exists (S_1, V_1), (S_2, V_2) \in \mathsf{BB}_{\mathsf{tally}}(\tau). \\ \mathsf{pet}(S, S_1) = \mathsf{pet}(S, S_2) = \mathsf{ok}, \mathsf{pet}(V_1, V_2) \neq \mathsf{ok}\}$$

There is a simple way to compute $\mathsf{BB}_{\mathsf{ce}}(\tau)$, shown in algorithm 1. The idea is to first group together all the ballots that correspond to a given credential, relying on plaintext equivalence tests, and then detect if among these there are two ballots representing distinct votes. At the end of algorithm 1, the set $\mathsf{ce}_{\mathsf{cc}}^{\mathsf{cred}}(\tau)$ should be equal to $\mathsf{BB}_{\mathsf{ce}}(\tau)$ (we will prove this in section VI) and the zero-knowledge proofs in $\mathsf{ce}_{\mathsf{cc}}^{\mathsf{zkp}}(\tau)$ should allow anyone to verify that the set $\mathsf{ce}_{\mathsf{cc}}^{\mathsf{cred}}(\tau)$ has been correctly computed.

Algorithm 1 Coercion-evidence: algorithm that reveals too much

Input: $\mathsf{BB}_{\mathsf{aer}}(\tau), \mathsf{BB}_{\mathsf{tally}}(\tau)$ (taken from the bulletin board)
Output: $\mathsf{ce}_{\mathsf{cc}}^{\mathsf{cred}}(\tau), \mathsf{ce}_{\mathsf{cc}}^{\mathsf{zkp}}(\tau)$
$\mathsf{ce}_{\mathsf{cc}}^{\mathsf{cred}}(\tau) := \emptyset; \mathsf{ce}_{\mathsf{cc}}^{\mathsf{zkp}}(\tau) := \emptyset$
for $S \in \mathsf{BB}_{\mathsf{aer}}(\tau)$ **do**
  $\mathcal{B}_S := \emptyset$ // $\mathcal{B}_S$ is the set of ballots corresponding to S
  for $(S', V) \in \mathsf{BB}_{\mathsf{tally}}(\tau)$ **do**
    if $\mathsf{pet}(S, S') = \mathsf{ok}$ **then**
      $\mathcal{B}_S := \mathcal{B}_S \cup \{(S', V)\}$
      $\mathsf{ce}_{\mathsf{cc}}^{\mathsf{zkp}}(\tau) := \mathsf{ce}_{\mathsf{cc}}^{\mathsf{zkp}}(\tau) \cup \mathsf{petproof}(S, S', \mathsf{yes})$
  for $(S_1, V_1) \in \mathcal{B}_S, (S_2, V_2) \in \mathcal{B}_S$ **do**
    if $\mathsf{pet}(V_1, V_2) \neq \mathsf{ok}$ **then**
      $\mathsf{ce}_{\mathsf{cc}}^{\mathsf{cred}}(\tau) := \mathsf{ce}_{\mathsf{cc}}^{\mathsf{cred}}(\tau) \cup \{S\}$
      $\mathsf{ce}_{\mathsf{cc}}^{\mathsf{zkp}}(\tau) := \mathsf{ce}_{\mathsf{cc}}^{\mathsf{zkp}}(\tau) \cup \mathsf{petproof}(V_1, V_2, \mathsf{no})$
return $\mathsf{ce}_{\mathsf{cc}}^{\mathsf{cred}}(\tau), \mathsf{ce}_{\mathsf{cc}}^{\mathsf{zkp}}(\tau)$ (to the bulletin board)

Now, the *coercion-evidence test* $\mathsf{ce}_{\mathsf{cc}}(\tau)$ in Caveat Coercitor, described in Algorithm 2, consists in verifying that $\mathsf{cc}_{\mathsf{ce}}^{\mathsf{cred}}(\tau)$ has been correctly computed (basically, this means verifying the zero-knowledge proofs for the plaintext equivalence tests) and outputting $|\mathsf{cc}_{\mathsf{ce}}^{\mathsf{cred}}(\tau)|$ as the observed degree of coercion.

Algorithm 2 Coercion-evidence test

Input: $\mathsf{ce}_{\mathsf{cc}}^{\mathsf{cred}}(\tau), \mathsf{ce}_{\mathsf{cc}}^{\mathsf{zkp}}(\tau)$ (taken from the bulletin board)
Output: $\mathsf{ce}_{\mathsf{cc}}(\tau)$
for $\mathsf{zkp} \in \mathsf{ce}_{\mathsf{cc}}^{\mathsf{zkp}}(\tau)$ **do**
  if verify $(\mathsf{zkp}) \neq ok$ **then**
    fail
$\mathsf{ce}_{\mathsf{cc}}(\tau) := \mathsf{count}(\mathsf{ce}_{\mathsf{cc}}^{\mathsf{cred}}(\tau))$
return $\mathsf{ce}_{\mathsf{cc}}(\tau)$ (to the bulletin board)

The algorithm 1 provides data for the coercion-evidence test and allows coercer independence, but let us see at what cost coercer independence is achieved. Indeed, note that the number of ballots that correspond to every credential in $\mathsf{BB}_{\mathsf{aer}}(\tau)$ is revealed. Now, assume the following coercion strategy: the coercer instructs the voter $\mathcal{A}$ to abstain and then casts a very large number $n_{\mathcal{A}}$ of ballots using $s_{\mathcal{A}}$. Therefore, if the voter disobeyed and has cast a vote using $s_{\mathcal{A}}$, the output of the algorithm 1 will allow the coercer to observe that for some credential $n_{\mathcal{A}}+1$ ballots have been cast. Now, to obtain coercer-independence, there should be some free voter $\mathcal{A}'$ that has cast $n_{\mathcal{A}}$ ballots using $s'_{\mathcal{A}}$: the coercer will see $n_{\mathcal{A}}$ ballots for some credential, and will not be able to tell whether that credential is $s_{\mathcal{A}}$ or not.

Although in theory coercer independence can be achieved in this way, the practical aspect of this approach is questionable. Therefore, we propose another algorithm to compute $\mathsf{BB}_{\mathsf{ce}}(\tau)$, which outputs less information about the set of tallied ballots (Algorithm 3). Instead of computing the set of all ballots that have been cast with every credential $s_{\mathcal{A}}$, we carefully guide the computation to determine if there are two distinct votes for $s_{\mathcal{A}}$. In particular, if there are multiple ballots with credential $s_{\mathcal{A}}$ for the same candidate, only one of them will be determined. We do the minimal amount of computations required to determine if a credential is coerced or not, and then go to the next credential.

We assume there are l candidates and that a list $\mathsf{can}_1, \ldots, \mathsf{can}_l$ of candidate names encrypted with pk has been computed and passed through a mixnet. These encrypted candidate names are part of the input for Algorithm 3. The algorithm first groups the ballots according to the candidate that they represent and then for every credential it checks whether it belongs to at least two groups.

To achieve coercer independence for a voter $\mathcal{A}$ in the context of Algorithm 3, all we need is the presence of a free voter $\mathcal{A}'$ that can either cast a normal vote or cancel his ballots by casting two different votes: the coercer can not tell the difference between the ballots of $\mathcal{A}$ and those of $\mathcal{A}'$. If the coercer has cast a large number of ballots $n_{\mathcal{A}}$ with $s_{\mathcal{A}}$ and the voter $\mathcal{A}$ has cast a different vote with $s_{\mathcal{A}}$, the free voter $\mathcal{A}'$ just needs to cast a vote for the candidate desired by the coercer: there is no way for the coercer to detect that there are $n_{\mathcal{A}}+1$ ballots with credential $s_{\mathcal{A}}$ and that they have been discarded as coercion-evidence.

Tallying. For a ciphertext c and a set of ciphertexts M, we will denote by $c \in_{\mathsf{pet}} M$ the fact $\exists c' \in M.\mathsf{pet}(c, c') = \mathsf{ok}$. Let $\mathsf{BB}_{\mathsf{valid}}(\tau) = \mathsf{BB}_{\mathsf{aer}}(\tau) \setminus \mathsf{BB}_{\mathsf{ce}}(\tau)$ be the set of anonymized credentials that are not coerced. Now, for every $(S, V) \in \mathsf{BB}_{\mathsf{tally}}(\tau)$,

- either $S \in_{\mathsf{pet}} \mathsf{BB}_{\mathsf{valid}}(\tau)$. In this case, since $S \notin_{\mathsf{pet}} \mathsf{BB}_{\mathsf{ce}}(\tau)$, we are certain that all ballots that correspond to S contain a vote for the same candidate. Only one of them should be counted.
- or else $S \in_{\mathsf{pet}} \mathsf{BB}_{\mathsf{ce}}(\tau)$. In this case, S corresponds to a coerced credential and no ballot should be counted for S.

Algorithm 3 Coercion-evidence: algorithm that reveals enough

Input: $\mathsf{BB}_{\mathsf{aer}}(\tau), \mathsf{BB}_{\mathsf{tally}}(\tau), \mathsf{can}_1 \ldots \mathsf{can}_l$
(taken from the bulletin board)
Output: $\mathsf{ce}^{\mathsf{cred}}_{\mathsf{cc}}(\tau), \mathsf{ce}^{\mathsf{zkp}}_{\mathsf{cc}}(\tau)$
$\mathsf{ce}^{\mathsf{cred}}_{\mathsf{cc}}(\tau) := \emptyset; \mathsf{ce}^{\mathsf{zkp}}_{\mathsf{cc}}(\tau) := \emptyset$
// Step 1: group ballots according to the vote that they encode
for $i := 1$ to l **do**
  $\mathcal{B}_i := \emptyset$
  for $(S, V) \in \mathsf{BB}_{\mathsf{tally}}(\tau)$ **do**
    if $\mathsf{pet}(\mathsf{can}_i, V) = \mathsf{ok}$ **then**
      $\mathcal{B}_i := \mathcal{B}_i \cup \{(S, V)\}$
      $\mathsf{ce}^{\mathsf{zkp}}_{\mathsf{cc}}(\tau) := \mathsf{ce}^{\mathsf{zkp}}_{\mathsf{cc}}(\tau) \cup \mathsf{petproof}(\mathsf{can}_i, V, \mathsf{yes})$
// Step 2: for each S, check if it occurs in $\mathcal{B}_i, \mathcal{B}_j$, $i \neq j$
for $S \in \mathsf{BB}_{\mathsf{aer}}(\tau)$ (*) **do**
  for $i := 1$ to l **do**
    for $(S_1, V_1) \in \mathcal{B}_i$ **do**
      if $\mathsf{pet}(S, S_1) = \mathsf{ok}$ **then**
        $\mathsf{ce}^{\mathsf{zkp}}_{\mathsf{cc}}(\tau) := \mathsf{ce}^{\mathsf{zkp}}_{\mathsf{cc}}(\tau) \cup \mathsf{petproof}(S, S_1, \mathsf{yes})$
        for $j := i + 1$ to l **do**
          for $(S_2, V_2) \in \mathcal{B}_j$ **do**
            if $\mathsf{pet}(S, S_2) = \mathsf{ok}$ **then**
              $\mathsf{ce}^{\mathsf{cred}}_{\mathsf{cc}}(\tau) := \mathsf{ce}^{\mathsf{cred}}_{\mathsf{cc}}(\tau) \cup \{S\}$
              $\mathsf{ce}^{\mathsf{zkp}}_{\mathsf{cc}}(\tau) := \mathsf{ce}^{\mathsf{zkp}}_{\mathsf{cc}}(\tau) \cup \mathsf{petproof}(S, S_2, \mathsf{yes})$
              go to next credential in $\mathsf{BB}_{\mathsf{aer}}(\tau)$ at (*)
return $\mathsf{ce}^{\mathsf{cred}}_{\mathsf{cc}}(\tau), \mathsf{ce}^{\mathsf{zkp}}_{\mathsf{cc}}(\tau)$ (to the bulletin board)

- or else $S \notin_{\mathsf{pet}} \mathsf{BB}_{\mathsf{aer}}(\tau)$. In this case, S corresponds to a non-eligible credential and no ballot should be counted for S

Therefore, we have the following simple algorithm for tallying: for all credential $S \in \mathsf{BB}_{\mathsf{valid}}(\tau)$, find the first ballot $(S', V) \in \mathsf{BB}_{\mathsf{tally}}(\tau)$ such that $\mathsf{pet}(S, S') = \mathsf{ok}$; then, decrypt V.

V. Coercion-evidence in Caveat Coercitor

A. Trust assumptions.

We will show in section VI that coercion-evidence holds in caveat coercitor under the following trust assumptions:

- The voting machines of coerced voters and of some free voters are not compromised.
- The registration channel allows voters to obtain their voting credentials, and voters verify them to ensure that they are correct. In particular, this implies that voters can cast a valid ballot for their intended choice.
- There is an anonymous channel that allow voters to cast their vote as intended.
- There exists at least one honest mix server $\mathcal{M}$
- There exists at least one honest tallier $\mathcal{T}$

The trust assumptions about one honest mix server and one honest tallier are standard and probably necessary in any system based on mixnets and on public-key cryptography. We explain later in this section how Caveat Coercitor may be adapted to relax the first assumption about the trusted voting machine.

A big novelty of our trust assumptions, in contrast with Helios and JCJ/Civitas, is that voting credentials need to be secured only with "best effort". Thus, even if the voting channel where the voter obtains his credential is compromised, or if the voter is coerced to reveal the private credential, coercion-evidence still holds. (However, if the security is insufficient, a lot of coercion will be detected.) This also means we do not have to trust the registrars to keep the credential secret. More generally, this addresses the problem of silent coercion discussed in the introduction, which is common to Helios, JCJ/Civitas and the Estonian internet voting system.

Informally, coercion-evidence holds given these trust assumptions because:

- The coerced voters follow the instruction to cast a vote for their intended candidate. Since they have access to a trusted voting device, this ensures that, for all coerced voters, coercion-evidence is input in the system during the voting phase. It will be detected later.
- Additionally, a trusted voting device means that the coercer can not observe that a voter has cast a ballot, and this is essential for coercer-independence.
- The voter verifies that his ballot has reached the bulletin board and zero-knowledge proofs for mixnets and plaintext equivalence tests are publicly checked. This ensures that coercion-evidence that is input in the system during the voting phase is not lost on the way from the voters to the talliers. The role of zero-knowledge proofs is also to ensure that coercion-evidence in ballots to be tallied corresponds indeed to ballots cast in the voting phase, and not introduced in the system during mixing.
- Voters receive and verify that their credentials are valid. This ensures that their ballots are not discarded before the final tally, and will result either in a counted vote, or in coercion-evidence.
- The honest mixer and the honest tallier will help in achieving coercer-independence: the honest mixer will ensure that the ballots cast by the coercer or the coerced voter can not be tracked to determine how they have been handled in the tallying phase; the honest tallier ensures that the ballots are decrypted and plaintext equivalence tests are performed only as specified by the protocol.

B. Example.

Suppose that events unfold as depicted in Table I. Among 48M voters, about 44M voters submit ballots with vote for only one candidate, while about 4.2M voters (or their coercers) submit ballots containing votes for different candidates. Thus, most voters have not been coerced, and have cast possibly multiple ballots for a single candidate. They

	No. of credentials	%age of credentials
Total number of eligible credentials	48,783,530	100%
Number of credentials that have voted for a single candidate	44,539,363	91.3%
Number of credentials that have voted for at least two candidates	4,244,167	8.7%

Table I
AN EXAMPLE DISTRIBUTION OF BALLOTS. A TABLE IN THIS FORMAT IS OUTPUT BY THE SYSTEM AT THE END OF THE ELECTION.

account for 91.3% of voters. The rest 8.7% of voters have cast ballots for different candidates. These may be voters that have been forced to release their credentials, forced to cast vote to a candidate but they also managed to cast vote to their intended candidate or they may simply have decided to submit two ballots for different candidates. So either they were coerced or they did not follow the specifications of the protocol and thus were dishonest. The ballots from these 8.7% of voters will all be discarded and 91.3% of the ballots are taken to determine the outcome. The system counts one vote for the (single) candidate represented by each of the set of ballots that corresponds to each credential. The margins by which the winner beats the other candidates can be considered in order to determine whether the election result should carry.

To understand the reason for this way of counting, let us distinguish the following two cases:

- If a credential has been used once or more times, but each time to vote for the same candidate, a vote for that candidate is counted. Indeed, there are three subcases:
  - The voter has cast multiple ballots for the same candidate;
  - The attacker obtained voter's credentials and has cast a ballot for the same candidate as the voter; or
  - The voter knowingly abstained and the attacker obtained her credentials and cast ballots for one candidate.

  The output of the system does not allow any voter, coercer or observer to distinguish between these subcases, but if the voter behaved correctly and cast a ballot(the third case is eliminated), then his vote is counted.
- If a credential has been used to cast votes for multiple candidates, none of its votes is counted. The following sub-cases are indistinguishable:
  - The voter has cast multiple ballots for several different candidates;
  - The voter has cast multiple ballots for one candidate, the attacker obtained her credentials and has cast a vote for a different candidate;
  - The voter and the attacker each have cast votes for several different candidates; or
  - The voter knowingly abstained and the attacker cast votes on her behalf for multiple different candidates.

In each of these cases, either the voter is dishonest or the voter is coerced. The corresponding votes are therefore not counted and the evidence is recorded for the authorities to make a decision.

C. Discussion

Disincentivisation of coercion: As demonstrated, a coercer that wishes to achieve a particular outcome is faced with a dilemma. Assuming the security of the cryptography, and assuming that the voter is not dissuaded from casting her own ballot by invalid threats, the best the coercer can do is try to force a large number of annulled votes. However, the number of annulled votes he forces will be detected and announced, and an analysis will be performed to check whether those votes will materially affect the outcome. If not, the election will be considered a success and the declared outcome will have been shown to be robust against the degree of coercion attempted.

Since annulled votes in Caveat Coercitor are evident, they do not have the power that forced abstentions have in other systems. If the coercer has a strategy of annulling votes that he believes would be votes for a particular candidate, it won't work, because the final results will be interpreted as meaning 'there was a lot of coercion' rather than 'there were a lot of abstentions'.

Disruption of the election: In the definition of coercion-evidence, we have taken into account the fact that dishonest voters may pretend to be coerced, and therefore the coercion-evidence test may overestimate the actual degree of coercion. In some situations, a set of dishonest voters could rely on this in order to challenge the validity of the election. Especially when the difference between the winner and the runner-up of the election is small, a minority of voters would suffice to cause a disruption.

To deter voters from disrupting the election in practice, one option could be making voluntarily voting for two different candidates an offence. Both technical and administrative measures should be carefully designed to find an acceptable balance between discouraging dishonest behavior and encouraging resistance to coercion.

Another option is to have a probabilistic interpretation of coercion-evidence, depending on various data that can be gathered about the election (voter intentions, audit logs, etc.). This approach would allow to separate coercion-evidence into different threads, one of which would be evidence of actual coercion. More research on this idea is needed.

Individual and universal verifiability: Individuals can verify that their ballot is present on the bulletin board,

and will be counted. If the voter was coerced, their ballot will be included in the coercion evidence test. Observers can perform universal and eligibility verification, because the computations of algorithm 3 are publicly verifiable. Additionally, observers can verify the figures given in the table.

Towards untrusted voting machines: Although in this paper we assume for simplicity that a voter has access to a trusted voting machine, Caveat Coercitor has the potential to be adapted to provide coercion evidence in the context of untrusted machines, and this is our main topic for future research. We give some hints about directions here. For instance, assume a voter has access to n voting devices and the voter is unsure which one can be trusted for integrity. That is not a problem for Caveat Coercitor: the instruction for the voter is to simply cast his vote on any number of available devices. As long as *one out of n* devices is not integrity-compromised (the voter does not need to know which one),

- either the vote will be counted for the preferred candidate;
- or coercion will be recorded.

This deals with integrity. For voting clients that are privacy-corrupted, the situation is more difficult. A voter needs to take whatever steps are necessary to prevent the client communicating with the coercer. This may involve using the client in a Faraday cage, and resetting it or even destroying it afterwards, depending on the coercer's power.

Systems like JCJ/Civitas and Helios are not as well adapted to the untrusted client, because they are not designed to be as tolerant of coercion as Caveat Coercer. Our system allows coercion (but makes it evident).

VI. Analysis

According to definition 2, we prove that the coercion-evidence test of Caveat Coercitor is adequate (section VI-A) and that Caveat Coercitor allows coercer independence (section VI-B). We give proof sketches in this section and complete proofs in the appendix.

A. Coercion-evidence test

If $\mathsf{ce}_{\mathsf{cc}}(\tau)$ is the result of applying the coercion-evidence test in Caveat Coercitor for a run τ, we have to show that $|\delta_c(\tau)| \leq \mathsf{ce}_{\mathsf{cc}}(\tau) \leq |\delta_c(\tau)| + |\delta_d(\tau)|$.

For a credential s and a vote v, we will denote by $\mathcal{B}(s,v)$ the set of possible ballots (including the zero-knowledge proofs) with credential s and vote v. Thus, $\mathcal{B}(s,v)$ is the set of tuples of the form $(\{s\}^r_{pk}, \{v\}^{r'}_{pk}, P_{\mathsf{sv}}, P_{\mathsf{corr}})$, where r, r' are random numbers and P_{sv}, P_{corr} are the corresponding zero-knowledge proofs for this ballot. We denote by $\mathcal{B}_0(s,v)$ the set of possible ballots with credential s and vote v, without the zero-knowledge proofs.

For a run τ of Caveat Coercitor, recall that $\mathsf{BB}_{\mathsf{cast}}(\tau)$ is the set of cast ballots with valid zero-knowledge proofs and $\mathsf{BB}_{\mathsf{tally}}(\tau)$ is the set of ballots to be tallied in the run τ. Let us denote by $\mathcal{E}(\tau)$ the set of private credentials for eligible voters in a run τ. We define

$$\begin{aligned}
\mathcal{I}_{\mathsf{cast}}(\tau) = \quad & \{s \mid \exists v, v'.v \neq v', \mathcal{B}(s,v) \cap \mathsf{BB}_{\mathsf{cast}}(\tau) \neq \emptyset \\
& \;\&\; \mathcal{B}(s,v') \cap \mathsf{BB}_{\mathsf{cast}}(\tau) \neq \emptyset\} \\
\mathcal{I}_{\mathsf{tally}}(\tau) = \quad & \{s \mid \exists v, v'.v \neq v', \mathcal{B}_0(s,v) \cap \mathsf{BB}_{\mathsf{tally}}(\tau) \neq \emptyset \\
& \;\&\; \mathcal{B}_0(s,v') \cap \mathsf{BB}_{\mathsf{tally}}(\tau) \neq \emptyset\} \\
\mathcal{F}_{\mathsf{cast}}(\tau) = \quad & \{s \mid \exists v.\mathcal{B}(s,v) \cap \mathsf{BB}_{\mathsf{cast}}(\tau) \neq \emptyset\} \setminus \mathcal{E}(\tau) \\
\mathcal{F}_{\mathsf{tally}}(\tau) = \quad & \{s \mid \exists v.\mathcal{B}_0(s,v) \cap \mathsf{BB}_{\mathsf{tally}}(\tau) \neq \emptyset\} \setminus \mathcal{E}(\tau)
\end{aligned}$$

In words, $\mathcal{I}_{\mathsf{cast}}(\tau)$ and respectively $\mathcal{I}_{\mathsf{tally}}(\tau)$ represent the set of (inconsistent) credentials that have at least two corresponding ballots on the bulletin board in the voting phase and in the tallying phase respectively. The set $\mathcal{F}_{\mathsf{cast}}(\tau)$ represents fake (i.e. invalid) credentials that have been used to cast a vote, while the set $\mathcal{F}_{\mathsf{tally}}(\tau)$ is formed of fake credentials that correspond to ballots that are to be tallied.

A first lemma shows that all the coerced credentials are contained in the set $\mathcal{I}(\tau)$ and that, additionally, the set $\mathcal{I}(\tau)$ may only contain fake or dishonest credentials:

Lemma 1: For all run τ of Caveat Coercitor, we have

$$\delta_c(\tau) \subseteq \mathcal{I}_{\mathsf{cast}}(\tau) \subseteq \delta_c(\tau) \cup \delta_d(\tau) \cup \mathcal{F}_{\mathsf{cast}}(\tau)$$

The proof of the first inclusion relies on the definition of coerced voters and on the trust assumption that the coerced voters have access to an honest machine to cast their vote. Thus, at least two different votes are present for each coerced credential on the bulletin board: one coming from the voter and another one coming from the coercer. The second inclusion can be shown by a simple case analysis.

Secondly, we show that the set of fake credentials and the set of inconsistent credentials can not be changed in the mixnet.

Lemma 2: For all run τ of Caveat Coercitor in which the mixnet proofs are valid, we have

$$\mathcal{I}_{\mathsf{tally}}(\tau) = \mathcal{I}_{\mathsf{cast}}(\tau) \text{ and } \mathcal{F}_{\mathsf{tally}}(\tau) = \mathcal{F}_{\mathsf{cast}}(\tau)$$

The proof is an easy consequence of the fact that verifiable mixnets can not modify the content of ballots. Another easy observation is that the set $\mathsf{BB}_{\mathsf{ce}}(\tau)$, defined in section IV-C, has the same cardinality as the set of inconsistent credentials that are not fake:

Lemma 3: For all run τ of Caveat Coercitor, we have

$$|\mathsf{BB}_{\mathsf{ce}}(\tau)| = |\mathcal{I}_{\mathsf{tally}}(\tau) \setminus \mathcal{F}_{\mathsf{tally}}(\tau)|$$

Finally, we show that the coercion-evidence test provided by Caveat Coercitor captures exactly $|\mathsf{BB}_{\mathsf{ce}}(\tau)|$, i.e. we show that the Algorithm 3 from section IV-C is correct. Let $\mathsf{ce}^{\mathsf{cred}}_{\mathsf{cc}}(\tau)$ be the set of credentials output by the Algorithm 3 for a run τ.

Lemma 4: For all run τ of Caveat Coercitor, we have

$$\mathsf{ce}^{\mathsf{cred}}_{\mathsf{cc}}(\tau) = \mathsf{BB}_{\mathsf{ce}}(\tau)$$

To prove this lemma, we show that every credential in $\mathsf{BB}_{\mathsf{ce}}(\tau)$ will be detected in the step 2 of the Algorithm 3 and added to the set $\mathsf{ce}^{\tau}_{\mathsf{cc}}$. Moreover, we observe that every credential that has been used to cast a vote for a single candidate will not be part of $\mathsf{ce}_{\mathsf{cc}}(\tau)$, and therefore $\mathsf{ce}_{\mathsf{cc}}(\tau) \subseteq \mathsf{BB}_{\mathsf{ce}}(\tau)$.

Recall that, by definition, we have $\mathsf{ce}_{\mathsf{cc}}(\tau) = |\mathsf{ce}^{\mathsf{cred}}_{\mathsf{cc}}(\tau)|$. Putting all the lemmas together, we obtain the first part of coercion-evidence according to definition 2. Corollary 1 shows that the output of the coercion-evidence test in Caveat Coercitor is a correct estimate of the degree of coercion in any run, up to the number of dishonest voters:

Corollary 1: For all run τ of Caveat Coercitor, we have

$$|\delta_c(\tau)| \leq \mathsf{ce}_{\mathsf{cc}}(\tau) \leq |\delta_c(\tau)| + |\delta_d(\tau)|$$

Proof: From lemma 1 and lemma 2, we have $\delta_c(\tau) \subseteq \mathcal{I}_{\mathsf{cast}}(\tau)$ and $\mathcal{I}_{\mathsf{cast}}(\tau) = \mathcal{I}_{\mathsf{tally}}(\tau)$. Therefore, we deduce $\delta_c(\tau) \subseteq \mathcal{I}_{\mathsf{tally}}(\tau)$. By definition, all credentials in $\delta_c(\tau)$ are valid, i.e. $\delta_c(\tau) \subseteq \mathcal{E}(\tau)$, and thus $\delta_c(\tau) \cap \mathcal{F}_{\mathsf{tally}}(\tau) = \emptyset$. Hence, we obtain $\delta_c(\tau) \subseteq \mathcal{I}_{\mathsf{tally}}(\tau) \setminus \mathcal{F}_{\mathsf{tally}}(\tau)$. Considering the cardinality of these sets, we obtain $|\delta_c(\tau)| \leq |\mathcal{I}_{\mathsf{tally}}(\tau) \setminus \mathcal{F}_{\mathsf{tally}}(\tau)| = |\mathsf{BB}_{\mathsf{ce}}(\tau)| = |\mathsf{ce}^{\mathsf{cred}}_{\mathsf{cc}}(\tau)| = \mathsf{ce}_{\mathsf{cc}}(\tau)$, where we have used lemma 3 for the first equality and lemma 4 for the second equality. This way we get $|\delta_c(\tau)| \leq \mathsf{ce}_{\mathsf{cc}}(\tau)$.

From lemma 1, we have $\mathcal{I}_{\mathsf{cast}}(\tau) \subseteq \delta_c(\tau) \cup \delta_d(\tau) \cup \mathcal{F}_{\mathsf{cast}}(\tau)$. Using lemma 2, we get $\mathcal{I}_{\mathsf{tally}}(\tau) \subseteq \delta_c(\tau) \cup \delta_d(\tau) \cup \mathcal{F}_{\mathsf{tally}}(\tau)$. Subtracting $\mathcal{F}_{\mathsf{tally}}(\tau)$ from both sides, we have $\mathcal{I}_{\mathsf{tally}}(\tau) \setminus \mathcal{F}_{\mathsf{tally}}(\tau) \subseteq \delta_c(\tau) \cup \delta_d(\tau) \setminus \mathcal{F}_{\mathsf{tally}}(\tau)$. Furthermore, by definition we have $\delta_c(\tau) \cup \delta_d(\tau) \subseteq \mathcal{E}(\tau)$ and therefore $\delta_c(\tau) \cup \delta_d(\tau) \setminus \mathcal{F}_{\mathsf{tally}}(\tau) = \delta_c(\tau) \cup \delta_d(\tau)$. Thus, we have $\mathcal{I}_{\mathsf{tally}}(\tau) \setminus \mathcal{F}_{\mathsf{tally}}(\tau) \subseteq \delta_c(\tau) \cup \delta_d(\tau)$. Considering the cardinality of these sets and applying lemma 3 and lemma 4, we deduce $\mathsf{ce}_{\mathsf{cc}}(\tau) \leq |\delta_c(\tau)| + |\delta_d(\tau)|$ and we can conclude the proof. ∎

B. Coercer independence

Let $\mathcal{A}$ be a voter with credential $s_{\mathcal{A}}$ who intends to vote for $v_{\mathcal{A}}$. We have to show that, for every run τ of Caveat Coercitor where

- the voter $\mathcal{A}$ follows the protocol by executing $\mathcal{V}(s_{\mathcal{A}}, v_{\mathcal{A}})$
- for every candidate v_i, there is a free honest voter $\mathcal{A}_i$ that follows the protocol by executing $V(s_{\mathcal{A}_i}, v_i)$

there exists a run τ', where $\mathcal{A}$ does not execute $\mathcal{V}(s_{\mathcal{A}}, v_{\mathcal{A}})$, such that $\tau \sim \tau'$.

First we show how τ' can be constructed given τ and then we show that $\tau \sim \tau'$.

A run τ can be characterized by the sequence of messages that have been sent over the network in τ. For every message in a run τ, either the message is sent by the attacker (or by a party under the control of the attacker), or else it is sent by an honest agent, and all the attacker can do is to learn that message. A run τ can be seen as a sequence of (partial) runs $\tau = \tau_1 \dots \tau_n$. In Caveat Coercitor, we split a run in a sequence corresponding to the phases of the system: $\tau = \tau_{\mathsf{vote}}.\tau_{\mathsf{er}}.\tau_{\mathsf{mix}}.\tau_{\mathsf{tally}}$, where τ_{vote} is the list of ballots with valid zero-knowledge proofs that are cast on the bulletin board in the voting phase, τ_{er} is the electoral roll, τ_{mix} is the output of the re-encryption mix net and τ_{tally} is the output that talliers compute in the tallying phase, which includes data for the coercion-evidence test and the final outcome

Construction of the run τ'. There are four possible cases for the run τ:

- the coercer cast a vote with credential $s_{\mathcal{A}}$ for a candidate $v_{\mathcal{C}}$, with $v_{\mathcal{C}} \neq v_{\mathcal{A}}$
- the coercer did not cast a vote with credential $s_{\mathcal{A}}$
- the coercer cast a vote with credential $s_{\mathcal{A}}$ for the candidate $v_{\mathcal{A}}$
- the coercer cast at least two different votes with credential $s_{\mathcal{A}}$

For simplicity, we only consider the first case, which is also the most likely. The other three cases can be handled in a similar way. We sketch the main ideas of the construction in the following and we give a complete description in the appendix.

Assume the coercer has cast a ballot $b_{s_{\mathcal{A}}, v_{\mathcal{C}}} \in \mathcal{B}(s_{\mathcal{A}}, v_{\mathcal{C}})$. By definition of the run τ, we know that the voter $\mathcal{A}$ has cast a ballot $b_{s_{\mathcal{A}}, v_{\mathcal{A}}} \in \mathcal{B}(s_{\mathcal{A}}, v_{\mathcal{A}})$ and that there is a free honest voter $\mathcal{A}'$ that has cast a ballot $b_{s_{\mathcal{A}'}, v_{\mathcal{C}}} \in \mathcal{B}(s_{\mathcal{A}'}, v_{\mathcal{C}})$. Thus, the ballots corresponding to $s_{\mathcal{A}}$ are canceled because they are counted as coercion-evidence, while there is a vote counted for $v_{\mathcal{C}}$ that corresponds to $s_{\mathcal{A}'}$.

On the other hand, a requirement of the definition is that the voter $\mathcal{A}$ abstains from voting in the run τ'. Therefore, a vote for $v_{\mathcal{C}}$ corresponding to $s_{\mathcal{A}}$ is counted in the final tally. In order to obtain $\tau \sim \tau'$, we need at least to have the same outcome in the runs τ and τ'. Hence, we need the free voter $\mathcal{A}'$ to cancel his vote in the run τ': we assume $\mathcal{A}'$ casts a ballot $b_{s_{\mathcal{A}'}, v_{\mathcal{C}}} \in \mathcal{B}(s_{\mathcal{A}'}, v_{\mathcal{C}})$, as in the run τ, and in addition a ballot $b_{s_{\mathcal{A}'}, v_{\mathcal{A}}} \in \mathcal{B}(s_{\mathcal{A}'}, v_{\mathcal{A}})$. The rest of the cast ballots in τ' are assumed to be exactly the same as in τ.

Thus, the amount of coercion-evidence and the final outcome of τ' are exactly the same as in τ. What we need in addition in order to achieve coercer independence is that:

- the coercer is not able to trace any of the cast ballots and detect that they where handled as coercion evidence
- the coercer is not able to decrypt the ballots before they are mixed

The first point is achieved by our trust assumptions that the voter $\mathcal{A}$ can cast his ballot on an anonymous channel and that at least one mix server is honest. The anonymous channel ensures that the coercer can not distinguish the two runs during the voting phase. Later, in the run τ', the honest mix server will permute the ballots so that coercion-evidence and the votes for $v_{\mathcal{C}}$ show up in the same places as in τ. The second point is achieved by out trust assumption that at least one tallier is honest. Therefore, the ballots will be

decrypted only as specified by the protocol, after the mixing occurs.

In conclusion, if $\tau = \tau_{\mathsf{vote}}.\tau_{\mathsf{er}}.\tau_{\mathsf{mix}}.\tau_{\mathsf{tally}}$, then we let $\tau' = \tau'_{\mathsf{vote}}.\tau'_{\mathsf{er}}.\tau'_{\mathsf{mix}}.\tau'_{\mathsf{tally}}$, where

- τ'_{vote} is as τ_{vote}, with the difference that the voter $\mathcal{A}$ abstains from voting and the free voter $\mathcal{A}'$ cancels his vote.
- $\tau'_{\mathsf{er}} = \tau_{\mathsf{er}}$
- τ'_{mix} is as τ_{mix}, with the difference that the honest mixer swaps some ballots corresponding to $s_{\mathcal{A}}$ with some ballots corresponding to $s_{\mathcal{A}'}$, and also swaps the credentials of $\mathcal{A}$ and of $\mathcal{A}'$ while anonymizing the electoral roll.
- $\tau'_{\mathsf{tally}} = \tau_{\mathsf{tally}}$. This is possible because, by construction of the runs τ_{vote} and τ'_{vote}, it is the case that they determine the same outcome. Since there is one tallier that is not under the control of the coercer, all the coercer can do is to observe the final outcome computed according to the specification of the protocol.

Indistinguishability of τ and τ'. Although the actions performed and observed by the coercer are the same in τ and in τ', it may be the case that the coercer could distinguish τ from τ' by performing various computations on the messages that where sent in τ and τ': encryption, re-encryption, checking zero-knowledge proofs, combining messages, etc. In order to reason about the knowledge of the attacker, we therefore need a formal model of messages and of all possible computations that an attacker may perform. We adopt a variant of the applied pi-calculus [1], used by the protocol verifier ProVerif [6], [7].

We specify for ProVerif the messages that can be built in Caveat Coercitor, the cryptographic primitives and the possible actions of the attacker. Then, ProVerif allows us to verify that all the pairs of runs of a *bi-process* are in *observational equivalence*. Observational equivalence models the indistinguishability relation between runs that we are interested in: $\tau \sim \tau'$ if and only if any computations applied to τ and to τ' lead to the same observations. A bi-process is a pair of applied pi-calculus processes (P, P'), see e.g. [1], that share the same structure and differ only on the messages that they handle. A bi-process (P, P') generates pairs of runs (τ, τ'), where τ is a run of P, τ' is a run of P' and τ and τ' have been generated by executing the same actions in P and in P'.

We observe that the runs τ and τ' constructed in section VI-A are of the form $\tau = \tau_0.\tau_{\mathsf{tally}}$ and $\tau' = \tau'_0.\tau_{\mathsf{tally}}$, i.e. the tally in the two runs is exactly the same. Therefore, in order to show that $\tau \sim \tau'$, it is sufficient to show that $\tau_0 \sim \tau'_0$, where $\tau_0 = \tau_{\mathsf{vote}}.\tau_{\mathsf{er}}.\tau_{\mathsf{mix}}$ and $\tau'_0 = \tau'_{\mathsf{vote}}.\tau'_{\mathsf{er}}.\tau'_{\mathsf{mix}}$. Accordingly, we specify two process P_{cc} and P'_{cc} such that the bi-process $(P_{\mathsf{cc}}, P'_{\mathsf{cc}})$ generates all the pairs of runs of the form $(\tau_{\mathsf{vote}}.\tau_{\mathsf{er}}.\tau_{\mathsf{mix}})$ and $(\tau'_{\mathsf{vote}}.\tau'_{\mathsf{er}}.\tau'_{\mathsf{mix}})$ as constructed in section VI-A. We show with ProVerif that observational equivalence holds for the given bi-process $(P_{\mathsf{cc}}, P'_{\mathsf{cc}})$ (the ProVerif code is available online [1]), and therefore we can conclude the indistinguishability of any pair of runs (τ, τ'):

Corollary 2: For any pair of runs (τ, τ') of Caveat Coercitor constructed as described in section VI-A, we have $\tau \sim \tau'$.

From corollary 1 and corollary 2, we can conclude:

Theorem 1: Caveat Coercitor satisfies coercion-evidence under the trust assumptions from section V-A.

VII. Related work

Formal definitions: A formal definition of coercion-resitance based on observational equivalence is proposed in [13]. They prove that the Lee protocol [19] satisfies it. Another definition based on observational equivalence is proposed in [5], where an automated proof with ProVerif has been carried out for JCJ/Civitas. A more detailed, but not automated, analysis of JCJ/Civitas based on an epistemic approach is performed in [18]. In a computational model, coercion-resistance has been defined and proved for JCJ/Civitas in [16], [12].

Caveat Coercitor can readily be compared with other systems that allow Internet voting, such as JCJ/Civitas and Helios.

JCJ/Civitas [16], [12]: The system makes a strong assumption that there is an untappable channel for registration. It also assumes that the voter can run a multiparty protocols and keep real credentials secret all the time. Under these assumptions, it is strongly resistant to coercion, and is fully verifiable by voters and observers. Several variants of JCJ/Civitas improve the usability of the aspects related to verifiability [9], [21] and coercion-resistance [11].

On the other hand, coercion-evidence in Caveat Coercitor is not dependent on the secrecy of voting credentials, leading to more realistic assumption about the distribution of credentials. The registration phase, the voting phase and the resistance to coercion become simpler and more usable.

Helios 2.0 [2], [3]: This system is designed for low-coercion elections. It makes a few efforts to resist potential coercion, for example by keeping secret from voters the randoms in their ballots, but these efforts are easily defeated. On the positive side, Helios 2.0 enjoys individual and universal verifiability (but not eligibility verifiability). The most interesting feature of Helios is its high usability, which has been demonstrated by running large elections without failure. Caveat Coercitor is designed to be as usable as Helios (indeed, it can have the same front end and voter experience). Moreover, Caveat Coercitor does not have the restriction to low-coercion elections of Helios.

VIII. Conclusion

We propose coercion-evidence - a new property for e-voting systems that can replace coercion-resistance and

[1] markryan.eu/research/caveat-coercitor/

improve usability of electronic voting systems. Our formal definition is general enough to permit the verification of coercion-evidence in various systems and different security models. We also propose Caveat Coercitor - a voting system in which coercion is only weakly resisted, but made evident. The ability of a coercer is limited, because the best the coercer can achieve is the annulment of a voter's vote. A major feature of the system is that the degree of coercion that actually took place is publicly verifiable, provided the coerced voters follow the instructions of the protocol and cast votes for their desired candidate. This means that the coercer has only little incentive to coerce. Any observer can check if the coercion (that is, the annulled votes) could have made a material difference to the outcome.

To the best of our knowledge, our work is the first to propose a remote voting system with incoercibility properties that does not rely on an untappable channel. It is also the first paper to realise the importance of considering "silent" coercion (coercion unknown to the voter being coerced, for example by leaked credentials) together with coercion that is known to the voter.

Future work: As already indicated, the idea of coercion tolerance (with evidence) seems well-suited to addressing the problem of how to vote on untrusted machines. We speculated a bit about this in our discussion section, and intend to work on it further. We would also like to develop implementations, perhaps based on those of Helios or JCJ/Civitas, and see how they work in practice. It would also be interesting to explore how the concept of coercion evidence could be applied in other contexts than E-voting.

IX. Acknowledgements

We thank Jeremy Clark, Miriam Paiola and the anonymous reviewers for very useful comments. This work has been partially funded by the UK Engineering and Physical Sciences Research Council (EPSRC), under the project "TVS: Trustworthy Voting Systems" (EP/G02684X/1), and the Luxembourg National Research Fund (FNR), under the project SeRTVS-C09/IS/06.

References

[1] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *Proceedings of the 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, January 2001.

[2] Ben Adida. Helios: Web-based open-audit voting. In Paul C. van Oorschot, editor, *USENIX Security Symposium*, pages 335–348. USENIX Association, 2008.

[3] Ben Adida, Olivier Pereira, Olivier De Marneffe, and Jean-Jacques Quisquater. Electing a university president using open-audit voting: Analysis of real-world use of helios. In *In Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE)*, 2009.

[4] Roberto Araujo, Sébastien Foulle, and Jacques Traoré. A practical and secure coercion-resistant scheme for internet voting. In David Chaum, Markus Jakobsson, Ronald L. Rivest, Peter Y. A. Ryan, Josh Benaloh, Miroslaw Kutylowski, and Ben Adida, editors, *Towards Trustworthy Elections*, volume 6000 of *Lecture Notes in Computer Science*, pages 330–342. Springer, 2010.

[5] Michael Backes, Catalin Hritcu, and Matteo Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *CSF*, pages 195–209, 2008.

[6] Bruno Blanchet. Automatic proof of strong secrecy for security protocols. In *IEEE Symposium on Security and Privacy*, pages 86–100, 2004.

[7] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, 2008.

[8] Felix Brandt. Efficient cryptographic protocol design based on distributed el gamal encryption. In Dongho Won and Seungjoo Kim, editors, *ICISC*, volume 3935 of *Lecture Notes in Computer Science*, pages 32–47. Springer, 2005.

[9] Sergiu Bursuc, Gurchetan S. Grewal, and Mark D. Ryan. Trivitas: Voters directly verifying votes. In Kiayias and Lipmaa [17], pages 190–207.

[10] David Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. In Dimitris Gritzalis, editor, *Secure Electronic Voting*, volume 7 of *Advances in Information Security*, pages 211–219. Springer, 2003.

[11] Jeremy Clark and Urs Hengartner. Selections: Internet voting with over-the-shoulder coercion-resistance. In George Danezis, editor, *Financial Cryptography*, volume 7035 of *Lecture Notes in Computer Science*, pages 47–61. Springer, 2011.

[12] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a secure voting system. In *IEEE Symposium on Security and Privacy*, pages 354–368. IEEE Computer Society, 2008.

[13] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.

[14] Markus Jakobsson and Ari Juels. Mix and match: Secure function evaluation via ciphertexts. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 2000.

[15] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of the 11th USENIX Security Symposium*, pages 339–353, Berkeley, CA, USA, 2002. USENIX Association.

[16] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In Vijay Atluri, Sabrina De Capitani di Vimercati, and Roger Dingledine, editors, *WPES*, pages 61–70. ACM, 2005.

[17] Aggelos Kiayias and Helger Lipmaa, editors. *E-Voting and Identity - Third International Conference, VoteID 2011, Tallinn, Estonia, September 28-30, 2011, Revised Selected Papers*, volume 7187 of *Lecture Notes in Computer Science*. Springer, 2012.

[18] Ralf Küsters and Thomasz Truderung. An Epistemic Approach to Coercion-Resistance for Electronic Voting Protocols. In *2009 IEEE Symposium on Security and Privacy (S&P 2009)*, pages 251–266. IEEE Computer Society, 2009.

[19] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. In Jong In Lim and Dong Hoon Lee, editors, *ICISC*, volume 2971 of *Lecture Notes in Computer Science*, pages 245–258. Springer, 2003.

[20] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

[21] Michael Schläpfer, Rolf Haenni, Reto E. Koenig, and Oliver Spycher. Efficient vote authorization in coercion-resistant internet voting. In Kiayias and Lipmaa [17], pages 71–88.

APPENDIX A.
PROOFS FOR SECTION VI-A

Lemma 1: For all run τ of Caveat Coercitor, we have

$$\delta_c(\tau) \subseteq \mathcal{I}_{\mathsf{cast}}(\tau) \subseteq \delta_c(\tau) \cup \delta_d(\tau) \cup \mathcal{F}_{\mathsf{cast}}(\tau)$$

Proof: First we prove that $\delta_c(\tau) \subseteq \mathcal{I}_{\mathsf{cast}}(\tau)$. Let $s \in \delta_c(\tau)$ be the credential of coerced voter $\mathcal{A}$ in a run τ. By definition of coerced voters and since by our assumptions $\mathcal{A}$ has access to a trusted voting device, there exist two ballots in $\mathsf{BB}_{\mathsf{cast}}(\tau)$ corresponding to s and two different votes v, v': $\mathcal{B}(s,v) \cap \mathsf{BB}_{\mathsf{cast}}(\tau)$, $\mathcal{B}(s,v') \cap \mathsf{BB}_{\mathsf{cast}}(\tau)$ and $v \neq v'$. Therefore, we have $s \in \mathcal{I}_{\mathsf{cast}}(\tau)$.

Let us now prove that $\mathcal{I}_{\mathsf{cast}}(\tau) \subseteq \delta_c(\tau) \cup \delta_d(\tau) \cup \mathcal{F}_{\mathsf{cast}}(\tau)$. Let $s \in \mathcal{I}_{\mathsf{cast}}(\tau)$ and assume that $s \notin \mathcal{F}_{\mathsf{cast}}(\tau)$. We show that $s \in \delta_c(\tau) \cup \delta_d(\tau)$. By definition of $\mathcal{I}_{\mathsf{cast}}(\tau)$, there exist $v_1, \ldots, v_n$, such that $n > 1$, $v_i \neq v_j$ and $\mathcal{B}(s,v_i) \cap \mathsf{BB}_{\mathsf{cast}}(\tau) \neq \emptyset$, for all $1 \leq i, j \leq n$. By definition of coerced and dishonest voters, we have

- if all ballots that are cast by $\mathcal{A}$ belong to $\mathcal{B}(s,v_i)$, for a unique i, $1 \leq i \leq n$, then $\mathcal{A}$ is coerced and $s \in \delta_c(\tau)$.
- otherwise, either $\mathcal{A}$ has cast two different votes or $\mathcal{A}$ did not cast a vote, and we have $s \in \delta_d(\tau)$.

Hence we can conclude that $s \in \delta_c(\tau) \cup \delta_d(\tau) \cup \mathcal{F}_{\mathsf{cast}}(\tau)$. ■

Lemma 2: For all run τ of Caveat Coercitor in which the mixnet proofs are valid, we have

$$\mathcal{I}_{\mathsf{tally}}(\tau) = \mathcal{I}_{\mathsf{cast}}(\tau) \text{ and } \mathcal{F}_{\mathsf{tally}}(\tau) = \mathcal{F}_{\mathsf{cast}}(\tau)$$

Proof:

From definitions, we observe that to conclude the lemma it is sufficient to show that for all credential s and all vote v, we have $|\mathcal{B}(s,v) \cap \mathsf{BB}_{\mathsf{cast}}(\tau)| = |\mathcal{B}_0(s,v) \cap \mathsf{BB}_{\mathsf{tally}}(\tau)|$. In words, this means that no ballots have been lost or added to the tally during the mixing phase. This follows immediately from the validity of mixnet proofs. ■

Lemma 3: For all run τ of Caveat Coercitor, we have

$$|\mathsf{BB}_{\mathsf{ce}}(\tau)| = |\mathcal{I}_{\mathsf{tally}}(\tau) \setminus \mathcal{F}_{\mathsf{tally}}(\tau)|$$

Proof: Let $\mathsf{BB}^{priv}_{\mathsf{ce}}(\tau)$ be the set of private credentials that corresponds to the public credentials in $\mathsf{BB}_{\mathsf{ce}}(\tau)$. We note that $|\mathsf{BB}^{priv}_{\mathsf{ce}}(\tau)| = |\mathsf{BB}_{\mathsf{ce}}(\tau)|$ and therefore it is sufficient to show that $\mathsf{BB}^{priv}_{\mathsf{ce}}(\tau) = \mathcal{I}_{\mathsf{tally}}(\tau) \setminus \mathcal{F}_{\mathsf{tally}}(\tau)$.

We have $s \in \mathsf{BB}^{priv}_{\mathsf{ce}}(\tau) \Leftrightarrow \exists \{s\}^r_{pk} \in \mathsf{BB}_{\mathsf{aer}}(\tau)$ and $(\{s\}^{r_1}_{pk}, V_1), (\{s\}^{r_2}_{pk}, V_2) \in \mathsf{BB}_{\mathsf{tally}}(\tau)$ such that $\mathsf{pet}(V_1, V_2) \neq \mathsf{ok} \Leftrightarrow s \in \mathcal{E}(\tau)$ and $\exists v_1 \neq v_2$ such that $\mathcal{B}_0(s,v_1) \cap \mathsf{BB}_{\mathsf{tally}}(\tau) \neq \emptyset$ and $\mathcal{B}_0(s,v_2) \cap \mathsf{BB}_{\mathsf{tally}}(\tau) \neq \emptyset$ $\Leftrightarrow s \in \mathcal{I}_{\mathsf{tally}}(\tau) \setminus \mathcal{F}(\tau)$. ■

Lemma 4: For all run τ of Caveat Coercitor, we have

$$\mathsf{ce}^{\mathsf{cred}}_{\mathsf{cc}}(\tau) = \mathsf{BB}_{\mathsf{ce}}(\tau)$$

Proof: We prove first that $\mathsf{BB}_{\mathsf{ce}}(\tau) \subseteq \mathsf{ce}^{\mathsf{cred}}_{\mathsf{cc}}(\tau)$. If $S^0 \in \mathsf{BB}_{\mathsf{ce}}(\tau)$, then there exist $(S^0_1, V^0_1), (S^0_2, V^0_2) \in \mathsf{BB}_{\mathsf{tally}}(\tau)$ such that $\mathsf{pet}(S^0, S^0_1) = \mathsf{pet}(S^0, S^0_2) = \mathsf{ok}, \mathsf{pet}(V^0_1, V^0_2) \neq \mathsf{ok}$. As $\mathsf{pet}(V^0_1, V^0_2)) \neq \mathsf{ok}$ there exist $\mathsf{can}_i, \mathsf{can}_j$ with $1 \leq i, j \leq$ such that $\mathsf{pet}(\mathsf{can}_i, V^0_1) = \mathsf{ok}$ and $\mathsf{pet}(\mathsf{can}_j, V^0_2) = \mathsf{ok}$. After the Step 1 of the Algorithm 3, we have $(S^0_1, V^0_1) \in \mathcal{B}_i$ and $(S^0_2, V^0_2) \in \mathcal{B}_j$. Without loss of generality, let us suppose that $i < j$ and that i, j are the smallest indices with these properties.

Therefore, during the step 2 of the Algorithm 3, when S^0 is considered for the role of S, $(S^0_1, \mathcal{V}^0_1) \in \mathcal{B}_i$ and $(S^0_2, \mathcal{V}^0_2) \in \mathcal{B}_j$ can be considered for the role of (S_1, V_1) and (S_2, V_2). Thus, we have $S \in \mathsf{ce}^{\mathsf{cred}}_{\mathsf{cc}}(\tau)$.

To prove the inverse inclusion, consider $S \in \mathsf{cc}^{\mathsf{cred}}_{\mathsf{ce}}(\tau)$. This means there exist $\mathcal{B}_i, \mathcal{B}_j$ such that there are $(S_1, V_1) \in \mathcal{B}_i$, $(S_2, V_2) \in \mathcal{B}_j$ and $\mathsf{pet}(\mathsf{can}_i, V_1) = \mathsf{ok}$, $\mathsf{pet}(\mathsf{can}_j, V_2) = \mathsf{ok}$ and $\mathsf{pet}(S_1, S) = \mathsf{ok} = \mathsf{pet}(S, S_2)$. Therefore, we can conclude $S \in \mathsf{BB}_{\mathsf{ce}}(\tau)$. ■

APPENDIX B.
PROOFS FOR SECTION VI-B

Construction of the run τ'. The main idea in the construction of τ' is to rely on the free voter $\mathcal{A}'$ in the voting phase so that the ballots of $\mathcal{A}'$ in the run τ' look like the ballots of $\mathcal{A}$ in the run τ, and the other way around. Furthermore, we rely on an honest re-encryption mix server to ensure that the coercer can not track the ballots back to $\mathcal{A}$ or to $\mathcal{A}'$. The honest mix server swaps the ballots of $\mathcal{A}$ and $\mathcal{A}'$ and also their credentials in the anonymized electoral roll. An honest tabulation tallier ensures that the tallying is performed according to the specification of the protocol, both in τ and in τ'.

For a run $\tau = \tau_{\mathsf{vote}}.\tau_{\mathsf{er}}.\tau_{\mathsf{mix}}.\tau_{\mathsf{tally}}$ as the one in section VI-B, we let $\tau' = \tau'_{\mathsf{vote}}.\tau_{\mathsf{er}}.\tau'_{\mathsf{mix}}.\tau'_{\mathsf{tally}}$, where τ'_{vote}, τ'_{mix} and τ'_{tally} are defined in the following.

Voting phase. We have by definition at least three particular ballots present in τ_{vote} (recall that we consider the case when the coercer has cast a vote for $v_{\mathcal{C}}$, $v_{\mathcal{C}} \neq v_{\mathcal{A}}$, using the credential $s_{\mathcal{A}}$): two with credential $s_{\mathcal{A}}$ and one with credential $s_{\mathcal{A}'}$. Without loss of generality, we fix an order for these three ballots on the bulletin board. Then, we have:

$$\tau_{\mathsf{vote}} = L_1.b_{s_{\mathcal{A}},v_{\mathcal{C}}}.L_2.b_{s_{\mathcal{A}},v_{\mathcal{A}}}.L_3.b_{s_{\mathcal{A}'},v_{\mathcal{C}}}.L_4$$

where $b_{s_{\mathcal{A}},v_{\mathcal{C}}} \in \mathcal{B}(s_{\mathcal{A}}, v_{\mathcal{C}})$, $b_{s_{\mathcal{A}},v_{\mathcal{A}}} \in \mathcal{B}(s_{\mathcal{A}}, v_{\mathcal{A}})$, $b_{s_{\mathcal{A}'},v_{\mathcal{C}}} \in \mathcal{B}(s_{\mathcal{A}'}, v_{\mathcal{C}})$ and L_1,L_2,L_3,L_4 are lists of ballots. Furthermore, for any $1 \leq \ell \leq 4$, if there is a ballot in L_i that corresponds to the credential $s_{\mathcal{A}'}$, then the corresponding vote is $v_{\mathcal{C}}$.

Let $c_{\mathcal{A}}$ and respectively $c_{\mathcal{A}'}$ be the public credentials that correspond to the private credentials $s_{\mathcal{A}}$ and $s_{\mathcal{A}'}$. Then, without loss of generality, we can assume that the electoral roll is

$$\tau_{\mathsf{er}} = S_1.c_{\mathcal{A}}.S_2.c_{\mathcal{A}'}.S_3$$

where S_1, S_2 and S_3 are the public credentials of other voters. Now, for some $b_{s_{\mathcal{A}'},v_{\mathcal{A}}} \in \mathcal{B}_{s_{\mathcal{A}'},v_{\mathcal{A}}}$, let us consider τ'_{vote} defined as follows

$$\tau'_{\mathsf{vote}} = L_1.b_{s_{\mathcal{A}},v_{\mathcal{C}}}.L_2.b_{s_{\mathcal{A}'},v_{\mathcal{A}}}.L_3.b_{s_{\mathcal{A}'},v_{\mathcal{C}}}.L_4$$

Hence, τ'_{vote} is the same as τ_{vote}, with the difference that $\mathcal{A}$ now follows the instructions of the coercer and does not cast a vote for the intended candidate, while the free voter $\mathcal{A}'$ becomes dishonest, and casts an additional ballot for the candidate intended by $\mathcal{A}$.

Mixing phase. According to our trust assumptions, there is at least one mix server that is not controlled by the coercer. Let $\tau_{\mathsf{mix}} = \tau^0_{\mathsf{mix}}.\tau^1_{\mathsf{mix}}.\tau^h_{\mathsf{mix}}.\tau^2_{\mathsf{mix}}$, where:

- τ^0_{mix} is the input to the mix network, i.e. $\tau^0_{\mathsf{mix}} = \tau^0_{\mathsf{vote}}.\tau_{\mathsf{er}}.\tau_{\mathsf{cand}}$, where τ^0_{vote} contains the ballots from τ_{vote}, but not their corresponding zero-knowledge proofs, and τ_{cand} is a list of encrypted candidate names (to be used in the coercion-evidence test, cf Algorithm 2).
- τ^1_{mix} is the output of (dishonest) mix servers that are present in the mixnet before the honest mix server.
- τ^h_{mix} is the output of the honest mix server.
- τ^2_{mix} is the output of (dishonest) mix servers that are present in the mixnet after the honest mix server.

Now, we consider $\tau'_{\mathsf{mix}} = \tau'^0_{\mathsf{mix}}.\tau'^1_{\mathsf{mix}}.\tau'^h_{\mathsf{mix}}.\tau'^2_{\mathsf{mix}}$, where:

- $\tau'^0_{\mathsf{mix}} = \tau'^0_{\mathsf{vote}}.\tau'_{\mathsf{er}}.\tau'_{\mathsf{cand}}$, where τ'^0_{vote} contains the ballots from τ'_{vote}, but not their corresponding zero-knowledge proofs,, and $\tau'_{\mathsf{er}} = \tau_{\mathsf{er}}, \tau'_{\mathsf{cand}} = \tau_{\mathsf{cand}}$.
- τ'^1_{mix} is constructed from τ'^0_{mix} by applying exactly the same operations as for constructing τ^1_{mix} from τ^0_{mix}. This possible because these operations consist only in performing re-encryptions and applying permutations to the set of ballots: they do not depend on the content of ballots.
- τ'^h_{mix} is constructed as follows. Recall that $\tau^0_{\mathsf{mix}} = \tau^0_{\mathsf{vote}}.\tau_{\mathsf{er}}.\tau_{\mathsf{cand}}$, where τ^0_{vote} is τ_{vote} without the zero-knowledge proofs of correctness, i.e. we have

$$\tau^0_{\mathsf{vote}} = L^0_1.b^0_{s_{\mathcal{A}},v_{\mathcal{C}}}.L^0_2.b^0_{s_{\mathcal{A}},v_{\mathcal{A}}}.L^0_3.b^0_{s'_{\mathcal{A}},v_{\mathcal{C}}}.L^0_4$$

where we denote by b_0 (resp. L^0) a ballot b (resp. a list of ballots L) stripped of its proofs. Let τ^{ih}_{mix} be the input for the honest mix server in τ_{mix}, i.e. the output of the last dishonest server in τ^1_{mix}. Relying on the zero-knowledge proofs that have to be provided by any mix server (including the dishonest ones), we deduce that $\tau^{ih}_{\mathsf{mix}} = \rho_1.\rho_2.\rho_3$ where ρ_1, ρ_2 and respectively ρ_3 are re-encryption mixes of τ^0_{vote}, τ_{er} and respectively τ_{cand}, with some permutations chosen by the coercer. Thus, we have

$$\begin{array}{rcl} \rho_1 & = & L_1.w_1.L_2.w_2.L_3.w_3.L_4 \\ \rho_2 & = & S_1.t_1.S_2.t_2.S_3 \end{array}$$

where $w_1.w_2.w_3$ is a re-encryption mix of $b^0_{s_{\mathcal{A}},v_{\mathcal{C}}}.b^0_{s_{\mathcal{A}},v_{\mathcal{A}}}.b^0_{s'_{\mathcal{A}},v_{\mathcal{C}}}$ and $t_1.t_2$ is a re-encryption mix of $c_{\mathcal{A}}.c_{\mathcal{A}'}$. Furthermore, $\tau^h_{\mathsf{mix}} = \rho^h_1.\rho^h_2.\rho^h_3$, where, for every $1 \leq i \leq 3$, ρ^h_i is a re-encryption mix of ρ_i, with some permutation chosen by the honest mix server. Let:

$$\begin{array}{rcl} \rho^h_1 & = & L^h_1.w^h_1.L^h_2.w^h_2.L^h_3.w^h_3.L^h_4 \\ \rho^h_2 & = & S^h_1.t^h_1.S^h_2.t^h_2.S^h_3 \end{array}$$

where $w^h_1.w^h_2.w^h_3$ is a re-encryption mix of $w_1.w_2.w_3$ and $t^h_1.t^h_2$ is a re-encryption mix of $t_1.t_2$.

On the other hand, according to our construction of τ'^1_{mix}, the input to the honest mix server in τ'_{mix} is $\tau'^{ih}_{\mathsf{mix}} = \rho'_1.\rho_2.\rho_3$, where

$$\rho'_1 = L_1.w'_1.L_2.w'_2.L_3.w'_3.L_4$$

and $w'_1.w'_2.w'_3$ is a re-encryption mix of $b^0_{s_{\mathcal{A}},v_{\mathcal{C}}}.b^0_{s_{\mathcal{A}'},v_{\mathcal{A}}}.b^0_{s_{\mathcal{A}'},v_{\mathcal{C}}}$ such that, for every $1 \leq i \leq 3$, if $w'_i \in \mathcal{B}_0(s_{\mathcal{A}}, v_{\mathcal{C}})$ or $w'_i \in \mathcal{B}_0(s_{\mathcal{A}'}, v_{\mathcal{C}})$, then $w'_i = w_i$. Thus, as in the case of the pair $(\tau_{\mathsf{vote}}, \tau'_{\mathsf{vote}})$, the only difference between $\tau'^{ih}_{\mathsf{mix}}$ and τ^{ih}_{mix} is that, for some $1 \leq i \leq 3$, we have $w'_i \in \mathcal{B}_0(s_{\mathcal{A}'}, v_{\mathcal{A}})$ and $w_i \in \mathcal{B}_0(s_{\mathcal{A}}, v_{\mathcal{A}})$.

In the run τ' the output of the honest mix server must be $\tau'^h_{\mathsf{mix}} = \rho'^h_1.\rho'^h_2.\rho'^h_3$, corresponding to re-encryption mixes of ρ'_1, ρ_2 and ρ_3. Relying on the fact that the mix server producing τ'^h_{mix} is honest, we can choose particular permutations and particular randoms for the re-encryption mixes that produce τ'^h_{mix}. We choose these such that we have the following:

$$\begin{array}{rcl} \rho'^h_1 & = & L^h_1.w'^h_1.L^h_2.w'^h_2.L^h_3.w'^h_3.L \\ \rho'^h_2 & = & S^h_1.t'^h_1.S^h_2.t'^h_2.S^h_3 \end{array}$$

where $w'^h_1.w'^h_2.w'^h_3$ is a re-encryption mix of $w'_1.w'_2.w'_3$ and $t'^h_1.t'^h_2$ is a re-encryption mix of $t_1.t_2$ such that:

- ${w'}_i^h \in \mathcal{B}_0(s_{\mathcal{A}'}, v_{\mathcal{C}}) \Leftrightarrow w_i^h \in \mathcal{B}_0(s_{\mathcal{A}}, v_{\mathcal{C}})$.
- ${w'}_i^h \in \mathcal{B}_0(s_{\mathcal{A}'}, v_{\mathcal{A}}) \Leftrightarrow w_i^h \in \mathcal{B}_0(s_{\mathcal{A}}, v_{\mathcal{A}})$
- ${w'}_i^h \in \mathcal{B}_0(s_{\mathcal{A}}, v_{\mathcal{C}}) \Leftrightarrow w_i^h \in \mathcal{B}_0(s_{\mathcal{A}'}, v_{\mathcal{C}})$
- $\mathsf{pet}({t'}_1^h, c_{\mathcal{A}}) = \mathsf{ok} \Leftrightarrow \mathsf{pet}(t_1^h, c_{\mathcal{A}'}) = \mathsf{ok}$
- $\mathsf{pet}({t'}_2^h, c_{\mathcal{A}'}) = \mathsf{ok} \Leftrightarrow \mathsf{pet}(t_2^h, c_{\mathcal{A}}) = \mathsf{ok}$

Intuitively, the two ballots from $\mathcal{A}'$ and the ballot from $\mathcal{A}$ are re-arranged in the run τ'_{mix} by the honest mix server so that the plaintext equivalence tests performed for coercion-evidence will succeed for the same positions in τ' as in τ, and also the vote for $v_{\mathcal{C}}$ is revealed in the same position. Furthermore, the credentials of $\mathcal{A}$ and $\mathcal{A}'$ are swapped on the electoral roll, so that the credential that is marked as coerced is at the same position in both runs.

- ${\tau'}_{\mathsf{mix}}^2$ is constructed from ${\tau'}_{\mathsf{mix}}^h$ by performing exactly the same operations as for constructing τ_{mix}^2 from $\tau_m {ix}^h$. This is possible because the operations performed by a re-encryption mix server do not depend on the content of ballots.

Tallying phase. Let $\tau_{\mathsf{mix}}^{\mathsf{out}}$ and respectively ${\tau'}_{\mathsf{mix}}^{\mathsf{out}}$ be the outcome of the re-encryption mix net after the (partial) runs $\tau_{\mathsf{vote}}.\tau_{\mathsf{mix}}$ and $\tau'_{\mathsf{vote}}.\tau'_{\mathsf{mix}}$. Note that $\tau_{\mathsf{mix}}^{\mathsf{out}} = \tau_{\mathsf{vote}}^{\mathsf{out}}.\tau_{\mathsf{er}}^{\mathsf{out}}.\tau_{\mathsf{cand}}^{\mathsf{out}}$ and ${\tau'}_{\mathsf{mix}}^{\mathsf{out}} = {\tau'}_{\mathsf{vote}}^{\mathsf{out}}.{\tau'}_{\mathsf{er}}^{\mathsf{out}}.{\tau'}_{\mathsf{cand}}^{\mathsf{out}}$, where for all $a \in \{\mathsf{vote}, \mathsf{er}, \mathsf{out}\}$, τ_a^{out} and respectively ${\tau'}_a^{\mathsf{out}}$ is a re-enncryption mix of τ_a and respectively τ'_a. Furthermore, by construction we have:

- $\tau_{\mathsf{cand}}^{\mathsf{out}} = {\tau'}_{\mathsf{cand}}^{\mathsf{out}}$
- $\tau_{\mathsf{er}}^{\mathsf{out}} = t_1 \dots t_n$ and ${\tau'}_{\mathsf{er}}^{\mathsf{out}} = t'_1 \dots t'_n$ such that for every $1 \leq i \leq n$, we have
  - either $t'_i = t_i$
  - or else $\mathsf{pet}(t'_i, c_{\mathcal{A}}) = \mathsf{ok}$ and $\mathsf{pet}(t_i, c_{\mathcal{A}'}) = \mathsf{ok}$
  - or else $\mathsf{pet}(t'_i, c_{\mathcal{A}}) = \mathsf{ok}$ and $\mathsf{pet}(t_i, c_{\mathcal{A}'}) = \mathsf{ok}$
- $\tau_{\mathsf{vote}}^{\mathsf{out}} = w_1 \dots w_m$ and ${\tau'}_{\mathsf{vote}}^{\mathsf{out}} = w'_1 \dots w'_m$ such that for every $1 \leq i \leq m$, we have
  - either $w'_i = w_i$
  - or else $w'_i \in \mathcal{B}_0(s_{\mathcal{A}'}, v_{\mathcal{C}})$ and $w_i \in \mathcal{B}_0(s_{\mathcal{A}}, v_{\mathcal{C}})$
  - or else ${w'}_i \in \mathcal{B}_0(s_{\mathcal{A}'}, v_{\mathcal{A}})$ and $w_i \in \mathcal{B}_0(s_{\mathcal{A}}, v_{\mathcal{A}})$
  - or else ${w'}_i \in \mathcal{B}_0(s_{\mathcal{A}}, v_{\mathcal{C}})$ and $w_i \in \mathcal{B}_0(s_{\mathcal{A}'}, v_{\mathcal{C}})$

The trust assumptions for Caveat Coercitor ensure that there is at least one honest tallier. The honest tallier will follow the protocol and will not reveal his share of the decryption key to the coercer. Therefore, the only way to complete any run in the tallying phase for the coercer is to obey the specification of the protocol. Thus, we have $\tau_{\mathsf{tally}} = \tau_{\mathsf{valid}}.\tau_{\mathsf{buckets}}.\tau_c^1 \dots \tau_c^n.\tau_{\mathsf{dec}}$, where

- τ_{valid} are the ballots from $\tau_{\mathsf{vote}}^{\mathsf{out}}$ with eligible credentials
- τ_{buckets} is the outcome of plaintext equivalence tests that are performed to group the ballots according to the vote that they contain (step 1 of algorithm 2)
- for every $1 \leq i \leq n$, τ_c^i represents the outcome of plaintext equivalence tests performed to determine if the voter with credential t_i is coerced (step 2 of algorithm 2)
- τ_{dec} represents the result of decrypting ballots with valid credentials whose corresponding voters have not been coerced or dishonest.

Then, we let $\tau'_{\mathsf{tally}} = \tau'_{\mathsf{valid}}.\tau'_{\mathsf{buckets}}.{\tau'}_c^1 \dots {\tau'}_c^n.\tau'_{\mathsf{dec}}$, where:

- because $s_{\mathcal{A}}$ and $s_{\mathcal{A}'}$ are valid credentials by assumption, we note that the set of ballots with ineligible credentials is exactly the same in $\tau_{\mathsf{vote}}^{\mathsf{out}}$ and ${\tau'}_{\mathsf{vote}}^{\mathsf{out}}$. Therefore, if $\tau_{\mathsf{valid}} = \tau_{\mathsf{vote}}^{\mathsf{out}} \setminus S$, then we can choose $\tau'_{\mathsf{valid}} = {\tau'}_{\mathsf{vote}}^{\mathsf{out}} \setminus S$.
- from our observation about $(\tau_{\mathsf{vote}}^{\mathsf{out}}, {\tau'}_{\mathsf{vote}}^{\mathsf{out}})$ and $(\tau_{\mathsf{er}}^{\mathsf{out}}, {\tau'}_{\mathsf{er}}^{\mathsf{out}})$, it follows that the plaintext equivalence tests from algorithm 2 and the decryption of the final outcome give exactly the same results for $(\tau_{\mathsf{valid}}, \tau_{\mathsf{er}}^{\mathsf{out}})$ and $({\tau'}_{\mathsf{valid}}, {\tau'}_{\mathsf{er}}^{\mathsf{out}})$. Therefore, we can choose $\tau'_{\mathsf{buckets}} = \tau_{\mathsf{buckets}}, {\tau'}_c^i = \tau_c^i$, for all $1 \leq i \leq n$, and $\tau'_{\mathsf{dec}} = \tau_{\mathsf{dec}}$.

This way we finish the construction of τ'.

Indistinguishability of τ and τ'.

The ProVerif code available online at *markryan.eu/research/caveat-coercitor* verifies the observational equivalence of a bi-process $(P_{\mathsf{cc}}, P'_{\mathsf{cc}})$, which generates pairs of runs of the form (τ, τ'), as described in section VI-B. P_{cc} and P'_{cc} share the same structure and differ only on some (internal) messages. This difference can be expressed in ProVerif with the construct $\mathsf{choice}[u, v]$: the process P_{cc} (also called the left process) uses the term u, whereas the process P'_{cc} (the right process) uses the term v. To model the difference between the runs τ and τ' from Caveat Coercitor all we need is to make sure that the ballots that are cast are as expected. Then, ProVerif explores all the possible runs starting from that.