Trivitas: Voters directly verifying votes

Sergiu Bursuc, Gurchetan S. Grewal, and Mark D. Ryan

School of Computer Science, University of Birmingham, UK
s.bursuc@cs.bham.ac.uk,research@gurchetan.com,m.d.ryan@cs.bham.ac.uk

Abstract. Individual verifiability is the ability of an electronic voting system to convince a voter that his vote has been correctly counted in the tally. Unfortunately, in most electronic voting systems the proofs for individual verifiability are non-intuitive and, moreover, need trusted devices to be checked. Based on the remote voting system JCJ/Civitas, we propose Trivitas, a protocol that achieves direct and end-to-end individual verifiability, while at the same time preserving coercion-resistance.

Our technical contributions rely on two main ideas, both related to the notion of credentials already present in JCJ/Civitas. Firstly, we propose the use of trial credentials, as a way to track and audit the handling of a ballot from one end of the election system to the other end, without increased complexity on the voter end. Secondly, due to indistinguishability of credentials from random values, we observe that the association between any credential and its corresponding vote can be made public at the end of the election process, without compromising coercion-resistance. The voter has more intuitive and direct evidence that her intended vote has not been changed and will be counted in the final tally.

Keywords. Electronic voting, Individual verifiability, Trial votes, Intuitive verifiability

1 Introduction

The concept of *election outcome verifiability* has emerged as a vital ingredient in electronic voting systems. This has arisen partly because some implementations have been shown vulnerable to outcome manipulation, e.g. [8, 5]. Another reason is that, in contrast with electronic banking and social networking, it is not easy to put mistakes right if they are uncovered after the results of the election have been declared. A third reason is that it is notoriously difficult to prove that the *software* which might be running behind the scenes has the expected properties; it is more practical to verify the results produced by the software, than the software itself. For this reason, election outcome verifiability is sometimes referred to as *software independence*.

Election verifiability in presence of a bulletin board may be conveniently split in three notions [19]:

Individual verifiability [9, 21, 10, 2–4, 1]. To any voter, individual verifiability should offer the possibility to verify that her cast ballot has been correctly

recorded and tallied by the system. Ideally, the voter should also have an assurance that her cast ballot correctly encodes her cast vote. In some systems this additional assurance is offered by an option to audit a ballot [3, 4, 1] or a ballot form [10] before casting a vote.

Universal verifiability [17, 20, 14, 15]. To any external observer, universal verifiability should offer the possibility to verify that all recorded votes have been correctly tallied.

Eligibility verifiability [18, 11, 19]. To any external observer, eligibility verifiability should offer the possibility to verify that the set of tallied votes corresponds to votes cast by eligible voters. Eligibility may be enforced at any stage in the system: either during the casting of a vote [10], or during its recording [1], or during its tallying [18, 11]. Accordingly, eligibility verifiability will pertain to the corresponding stage.

One problem with the way *individual verifiability* is usually achieved is that the voter does not see her vote at the time she visits the bulletin board. This is the case in Helios [1], Pret-a-Voter [10], JCJ/Civitas [18, 11], and others. The reason is in order to achieve the property of coercion-resistance, which asserts that the voter shouldn't be able to prove to a potential coercer how she voted. Therefore, individual verifiability is achieved by indirect means; the voter can check that the encrypted ballot is present, and has some other evidence (perhaps based on auditing) that the encrypted ballot really represents her vote. Moreover, after the ballots are anonymized, the voter looses track even of her encrypted ballot.

Voters are likely to find this indirect achievement of individual verifiability unsatisfactory. This feeling has arisen in the focus groups that were held as part of the EPSRC project *Trustworthy Voting Systems* [23]. Four hour-long managed discussions among groups of about 10 citizens were arranged by a professional facilitator, with the aim of soliciting people's opinions about Pret-a-Voter. In at least two of the discussions, participants questioned the worthwhileness of checking the presence of their ballot on the bulletin board, given that they did not have any direct evidence that the ballot really contained their vote. The issue has also been mentioned by Adida and Neff [2], where the requirement that verifiability should be *direct* and *end to end* has been highlighted.

Our contribution. We introduce Trivitas, an adaptation of JCJ/Civitas. We show how the credentials of JCJ/Civitas can be adapted to improve individual verifiability. In particular, we show how voters can see their own vote in plaintext, making the verification experience more direct, and more intuitive.

Our first idea is the notion of trial votes. A trial vote is a vote that is cast along with real votes, but will not be counted. It will be decrypted and exposed along the way, in a way that is traceable by the voter that cast it. Since most of the system components are not able to distinguish trial and real votes, it gives confidence in the correct handling of all votes. This is an extension of the ideas of auditing in [10, 1], with the crucial difference that the auditing process is performed not only in the phase of casting a ballot, but is spread throughout the whole election process. In other words, a trial credential will function as a

marker whose sign is that the handling of this ballot should be made transparent, by e.g. decrypting and publishing its contents at every stage. There are a few technical difficulties with that, because trial ballots can fulfill their role only as long as they are not identified as such by possibly corrupted agents in the voting system. To avoid this problem, we make use of the fact that the decryption key is distributed among a set of trustees and propose to decrypt trial ballots by running a decryption mix [9, 10] among the trustees.

Our second idea is based on the following observation: real credentials are indistinguishable for anyone (exept for the voter) from trial credentials and fake credentials (an element that JCJ/Civitas introduces to enable coercion-resistance). Therefore, without compromising coercion-resistance, for each ballot (be it real, trial or fake) we can publish after anonymization (done by a re-encryption mixnet [17]) its corresponding credential and the decrypted vote. This allows a voter to verify that all its votes have made it into the final tally: its real vote, its trial vote and its possible fake votes. There is again a technical difficulty, related to eligibility verifiability and coercion-resistance: trial votes and fake votes have to be eliminated from the final tally in a publicly verifiable way, hence a coercer could observe that the credential obtained from the voter is fake. To avoid this problem we make use again of a decrytion mix run by the trustees: there is no way to link the decrypted contents of a ballot to ballots that will be eliminated to enforce eligibility.

Outline of the paper. In section 2 we describe the cryptographic primitives used in JCJ/Civitas and in section 3 we review its design and its solutions for election verifiability. Then, we make specific our critique of individual verifiability, that can be extended to systems like Pret a Voter [10] and Helios [1]. In section 4, we describe our proposal. In section 5, we show initial ideas about how trial credentials could be used to improve universal verifiability and recoverability. Finally, in section 6 we argue that changing JCJ/Civitas in the way that we propose does not compromise the coercion-resistance guarantees of the original system, and we give a hint of how the proof of [18] could be adapted to prove coercion-resistance for the new system.

2 Cryptographic primitives

JCJ/Civitas relies on the following cryptographic primitives. We do not detail the structure of zero-knowledge proofs, because it is not relevant for our purposes.

Distributed El-Gamal [7]. The encryption scheme being used is El-Gamal over a multiplicative group of integers modulo a prime $p = 2kq+1$, where q is also prime. The plaintext space and the ciphertext space $\mathcal{M}$ is the order q subgroup of $\mathbb{Z}_p^*$ (actually, some encoding has to be done when one wants the plaintext space to be $\mathbb{Z}_q$, but we can ignore such details here). Let g be the generator of $\mathcal{M}$. Then, a private key is a number $x \in \mathbb{Z}_q^*$ and the corresponding public key is $k = g^x \bmod p$. The encryption of a message m with a public key k is a pair

$(g^r \bmod p, m \cdot k^r \bmod p)$ where r is a fresh random number in $\mathbb{Z}_q^*$. To decrypt a ciphertext (a, b) the holder of the private key x computes $\frac{b}{a^x} \bmod p$.

JCJ/Civitas distributes the secret key (relying on e.g. [22]) 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. To decrypt a ciphertext (a, b), each of T_i publishes a partial decryption share $a_i = a^{x_i} \bmod p$. The final decryption can then be publicly computed as $\frac{b}{a_1 \cdot \ldots \cdot a_n}$.

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 $\mathsf{dec}(m, x)$, or simply by $\mathsf{dec}(m)$ when the key is clear from the context.

Decryption proof. Along with partial decryption shares a_i the holders of private key shares can send a zero-knowledge proof (equality of discrete logarithms) of the fact that they have used the correct key share to construct a_i. This allows any observer to check that the final result of the decryption is correctly computed, i.e. that decryption of $\{m\}_k$ is performed with the key $\mathsf{priv}(k)$ and gives the result m.

Re-encryption and mix nets. Given a ciphertext (a, b) constructed using the public key k any party can compute another ciphertext (a', b') such that (a', b') encrypts the same plaintext as (a, b) using the same key k: choose a random $r \in \mathbb{Z}_q^*$ and let $(a', b') = (a \cdot g^r \bmod p, b \cdot k^r \bmod p)$. We will denote the re-encryption of a ciphertext m with a random r by $\mathsf{renc}(m, r)$.

A re-encryption mix net $\mathcal{M}$ 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 $\mathcal{S}$. That is, $\mathcal{S}'$ is a formed by re-encryption of elements in a permutation of $\mathcal{S}$. Formally, 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 member of $\mathcal{M}$ is trusted not to be controlled by an intruder, he is unable to discover the permutation σ.

Mix proof, e.g. [17]. Given two sequences of ciphertexts $\mathcal{S} = m_1, \ldots, m_k$ and $\mathcal{S}' = m'_1, \ldots, m'_k$, a zero-knowledge mix proof shows that $\mathcal{S}'$ is a correct re-encryption mix of $\mathcal{S}$, but without revealing (a non-negligible part of) σ.

Plaintext equivalence test [16]. Consider two ciphertexts (a_1, b_1) and respectively (a_2, b_2), encrypted with the same key k, whose plaintexts are t_1 and respectively t_2. Assume that the private part $\mathsf{priv}(k)$ is distributed among $T_1, \ldots, T_n$. Then $T_1, \ldots, T_n$ can run a protocol to determine if $t_1 = t_2$ without them being able to learn t_1 or t_2: roughly, they compute $(a, b) = (\frac{a_1}{a_2}, \frac{b_1}{b_2})$ and perform a distributed decryption of (a, b); if the result of the decryption is 1, then $t_1 = t_2$; otherwise, $t_1 \neq t_2$.

For two ciphertexts m_1 and m_2, we will denote by $\mathsf{PET}(m_1, m_2) = \mathsf{true}$ if the plaintext equivalence test holds for m_1 and m_2.

PET proof [16]. Decryption proofs for the distributed decryption performed in a plaintext equivalence test can be used to attest that the test has been cor-

rectly performed, i.e. that $\mathsf{PET}(m_1, m_2) = \mathsf{true}$ if and only if $\mathsf{dec}(m_1, \mathsf{priv}(k)) = \mathsf{dec}(m_2, \mathsf{priv}(k))$.

3 Individual verifiability in JCJ/Civitas

3.1 Overview of JCJ/Civitas

The main idea in JCJ/Civitas is 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. It is assumed that at least one of the registrars will not be corrupted by the coercer and that the voter can communicate with this registrar using an untappable channel. To evade coercion, the voter has the ability to generate a fake credential, that is indistinguishable from a real one for the coercer. The participants of the protocol are

$\mathcal{R}$ - the set of registrars, whose role is to authenticate eligible voters and help generate their credentials.
$\mathcal{T}$ - the set of trustees, 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 ditributed decryption and plaintext equivalence tests.
$\mathcal{V}_1, \ldots, \mathcal{V}_n$ - the set of eligible voters.
$\mathcal{M}$ - a re-encryption mix net, whose role is to anonymize the set of cast ballots before verification of their eligibility and their decryption.
$\mathcal{B}$ - the bulletin board, 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.

A coercer may control some of $\mathcal{V}_1, \ldots, \mathcal{V}_n$, some of $\mathcal{R}$, some of $\mathcal{T}$ and some of $\mathcal{M}$ but not all. To achieve coercion-resistance, at least one of $\mathcal{R}$, one of $\mathcal{T}$ and one of $\mathcal{M}$ has to be outside the control of the coercer. A summary of the protocol is as follows, complemented by solid lines in figures 1 and 2. There are three phases: registration, voting and tabulation.

Registration. Election starts with trustees generating the public key KT of the election in a distributed manner, such that no minority of trustees can recover the private key $\mathsf{priv}(\mathsf{KT})$ [22] and the decryption is distributed [7]. The public part of the key is published on the bulletin board. By running a separate protocol with each of the registrars, the voter V_i obtains the private part c_i of her credential, together with a non-transferable proof P_i of the fact that the public part $\{c_i\}_{\mathsf{KT}}$, that is published in the electoral roll ER, correctly encodes the private part.

Voting. The ballot contains the encryption of the private credential c_i (with the key KT and with a different random than in ER) and the encryption of the intended vote v_i (with the key KT). To prevent the re-use of the same credential

by a party that does not hold the private part, the ballot contains additionally a proof $\mathsf{P_{cv}}$ of the fact that its creator knows both c_i and v_i. Additionally, P_{corr} proves that v_i is a valid vote, according to the specification decided by election authorities.

Tabulation. Before tabulation starts, the proofs of cast ballots are verified and ballots with invalid proofs are discarded. The valid ballots and the electoral roll are then sent to a re-encryption mix net for anonymization. Credentials from anonymized ballots are compared 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. Duplicate elimination is done by plaintext equivalence tests and can be performed either before, or after the mix. Finally, the decided set of countable votes is decrypted, and the corresponding decryption proofs are posted on the bulletin board.

3.2 Election verifiability

Universal verifiability and **eligibility verifiability** are achieved by proofs posted on the bulletin board:

- Mix proofs allow auditors $\mathcal{A}$ to verify that all ballot coming out of the mix net $\mathcal{M}$ (i.e. belonging to the set $\mathsf{MixedBallots}$) corresponds to a recorded ballot (i.e. belonging to the set $\mathsf{CastBallots}$), and also that no recorded ballot has been discarded.
- PET proofs and proofs $\mathsf{P_{cv}}$ allow $\mathcal{A}$ to verify that only votes coming from eligible voters are kept on the bulletin board for final decryption (i.e. in the set $\mathsf{CountableVotes}$).
- Decryption proofs allow $\mathcal{A}$ to verify that all countable votes have been correctly decrypted.

Individual verifiability is achieved as follows:

1. $\mathcal{V}$ must trust her voting machine that her cast ballot correctly encodes her vote.
2. $\mathcal{V}$ can check the bulletin board to see that the cast ballot has been correctly recorded.
3. Universal verifiability of mix nets ensures that all the recorded votes are correctly mixed, and therefore the vote cast by $\mathcal{V}$ is included in the set of mixed ballots.
4. Universal verifiability of PETs ensures that at least one copy of $\mathcal{V}$'s ballot is kept after the elimination of duplicates. Moreover, the proof P_i that the voter obtains during registration ensures that her private credential corresponds to a public credential in the electoral roll ER. Universal verifiability of mix nets ensures that a re-encryption of her public credential is also present in the anonymized electoral roll MER. Universal verifiability of PETs ensures that valid ballots are not eliminated when credentials are checked against the electoral roll MER. Altogether, these give to $\mathcal{V}$ an assurance that her

Fig. 1. JCJ/Civitas (solid lines) and additions of Trivitas (dotted lines): Registration and Voting phases

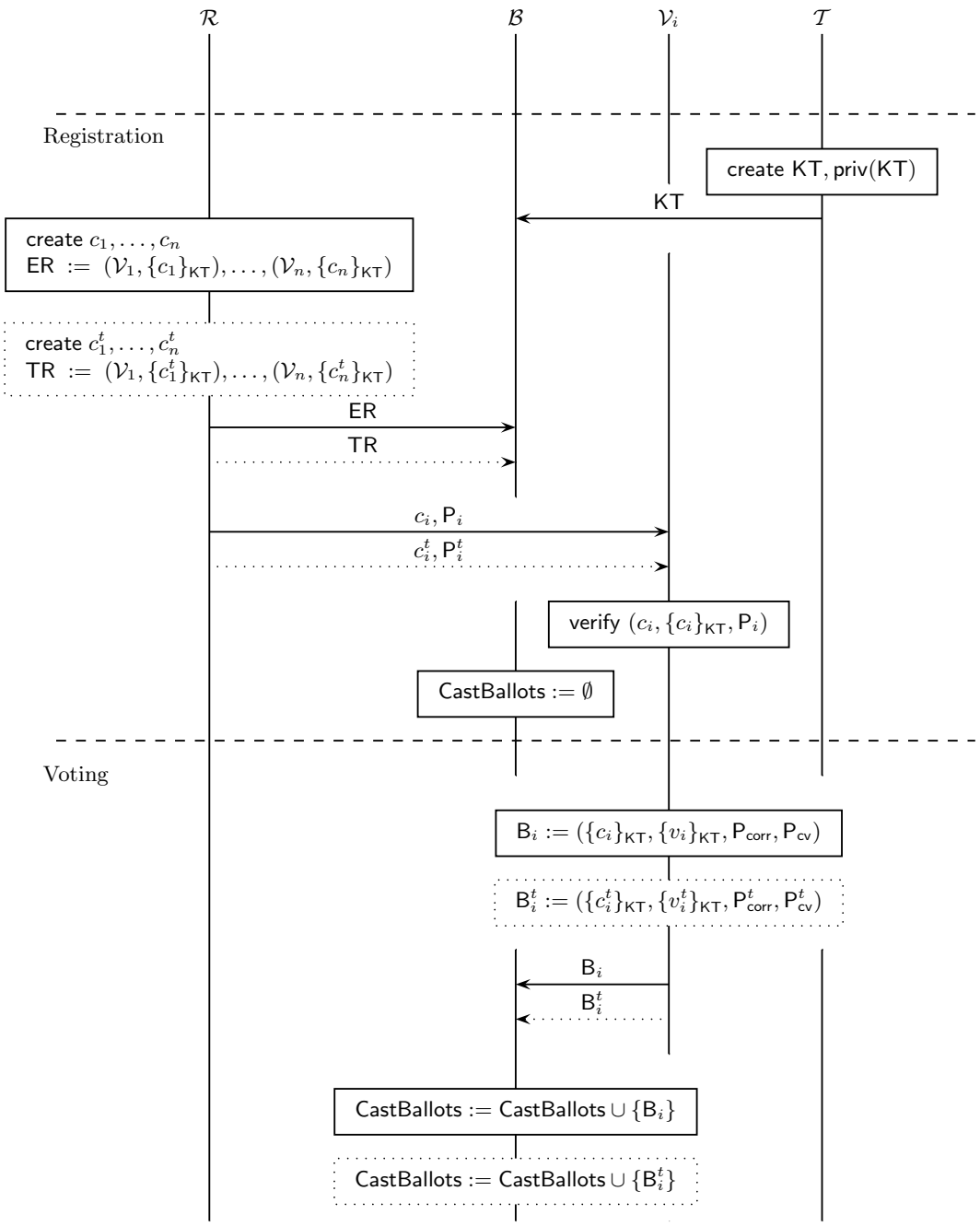
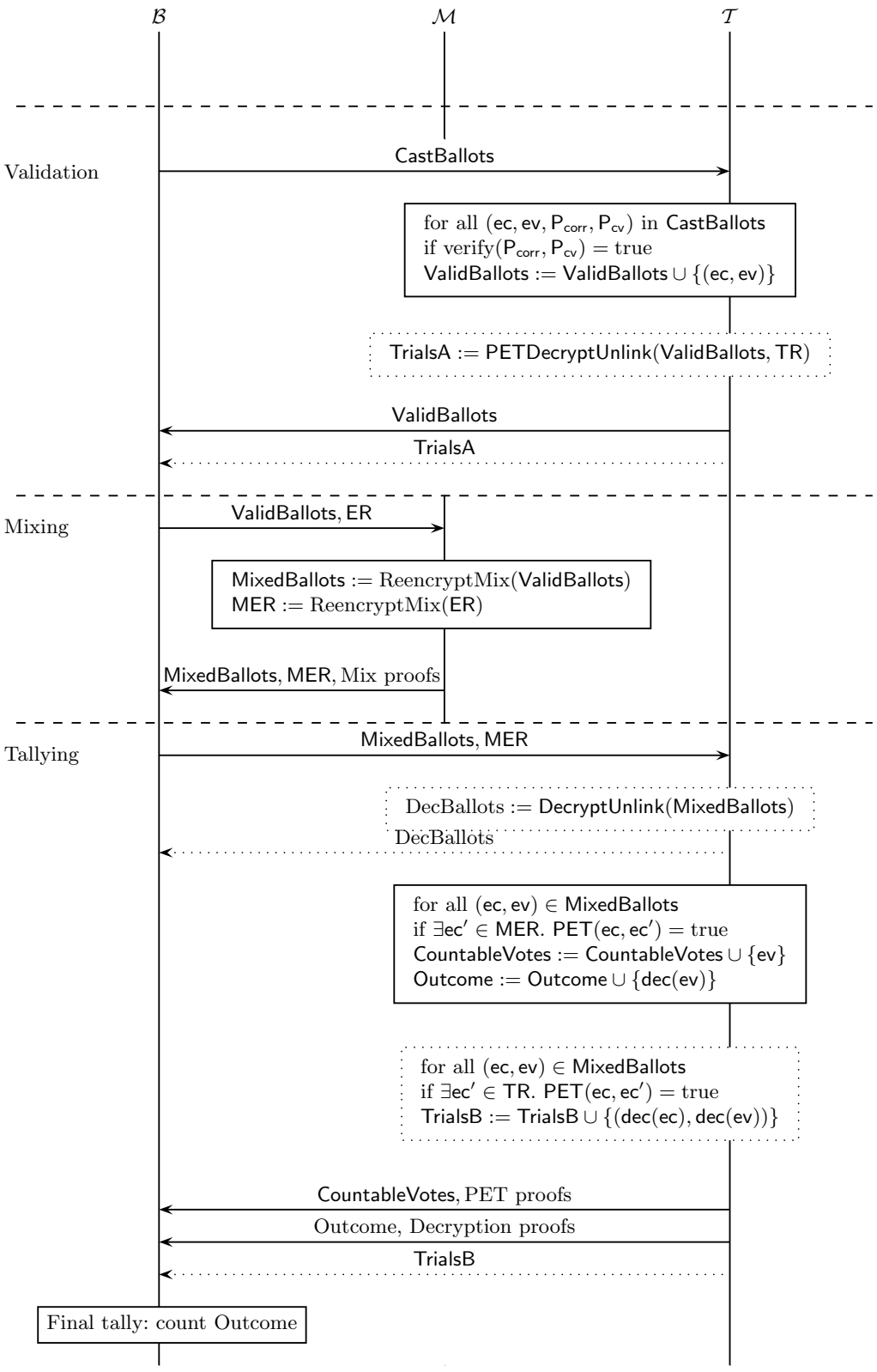


Fig. 2. JCJ/Civitas (solid lines) and additions of Trivitas (dotted lines): Tabulation phase



ballot is identified as coming from an eligible voter and is not eliminated during the enforcement of eligibility.
5. Finally, universal verifiability of distributed decryption ensures that $\mathcal{V}$'s ballot is correctly decrypted and tallied.

Some systems, e.g Pret a Voter [10] and Helios [1], improve the first point with a *cut-and-choose* mechanism, that allows the voter to audit a ballot before casting a vote.

Our critique of individual verifiability in JCJ/Civitas, Pret a Voter and Helios refers more generally to systems that rely on universal verifiability to achieve end-to-end individual verifiability. The points 3-5 above require complex mathematical operations and the corresponding verification algorithms must be run on trusted devices. Moreover, even if the corresponding zero-knowledge proofs are rigorously tailored to ensure the desired properties, the ordinary voter may be left wondering if her vote has actually been counted in the final tally. While auditors may be expected to have access to trusted devices and to understand the concepts behind zero-knowledge proofs, we do not consider these assumptions satisfactory when individual verifiability is considered.

Let us also note the following limiting aspect of the cut-and-choose mechanism in [3, 10] and [1]. In all these systems, the audited ballot is handled as a real ballot, but only up to the point when the voter decides to audit it, after which it is discarded. Only one ballot gets to be cast and submitted to the bulletin board. In our proposal, the audited ballot, that we call a *trial* ballot, will be handled in the same way as a real ballot at each phase of the protocol, while additionally playing its role as an audit ballot. In particular, it will be posted on the bulletin board before mixing and handled subsequently in the mix and in the decryption. The traditional cut-and-choose guarantees are recovered in this setting by tracking the trial ballot and the corresponding decrypted vote on the bulletin board before the mix.

4 Trivitas: trial credentials and universal decryption

The first proposal of Trivitas is the notion of trial credentials. A trial credential is a credential that allows a voter to cast a vote that will not be counted in the final tally, but will appear on the bulletin board at several stages of the tabulation phase. Its purpose is to allow the voter to gain confidence in the correct operation of the system. Trial ballots are identifiable as such only by a threshold set of trustees, thus any component of the system has to treat the set of all ballots in the same way. We show how trial credentials can be implemented in the context of JCJ/Civitas and show their immediate benefit for individual verifiability.

Moreover, we propose another addition to JCJ/Civitas, independent of trial credentials, that brings a further improvement to individual verifiability: we decrypt and publish the content of all ballots after the mix. Therefore, for every ballot that a voter has cast (real, trial or fake), she can verify that the corresponding credential and vote occur on the bulletin board after the mix. This

gives a direct evidence to the voter that her ballots, and most importantly her real ballot, have been correctly constructed and processed by the mix network.

4.1 Overview of proposed additions

The additions that we propose in each phase are the following. They are also sketched in dotted lines in figures 1 and 2.

Registration. Mirroring the set of real credentials $c_1, \ldots, c_n$ we assume that registrars generate a set of *trial credentials* $c_1^t, \ldots, c_n^t$. The set $c_1^t, \ldots, c_n^t$ is constructed and distributed to voters following the same protocols as for $c_1, \ldots, c_n$, thus we can assume the same security properties: in particular, trial credentials are indistinguishable from real credentials and fake credentials, for anyone but for the voter that receives its shares. In addition to the electoral roll ER, now we have a *trial roll* TR, that contains the public parts of the trial credentials $\{c_1^t\}_{\mathsf{KT}}, \ldots, \{c_n^t\}_{\mathsf{KT}}$.

Voting. In addition to $\mathcal{B}_i$ as in JCJ/Civitas, $\mathcal{V}_i$ constructs a trial ballot $\mathcal{B}_i^t = (\{c_i^t\}_{\mathsf{KT}}, \{v_i^t\}_{\mathsf{KT}}, \mathsf{P}_{\mathsf{corr}}^t, \mathsf{P}_{\mathsf{cv}}^t)$ and uploads both $\mathcal{B}_i$ and $\mathcal{B}_i^t$ to the bulletin board (at implementation level, it has to be decided if a ballot construction form would be used twice or if the system would allow the construction of both ballots at the same time).

Tabulation. At the time of ballot validation (i.e. just after the voting phase ends), the trustees $\mathcal{T}$ additionally perfom a PET test of each recorded ballot against the trial roll. For all ballots for which this PET returns true, the trustees decrypt the corresponding credential and the corresponding vote and publish them on the bulletin board: this is the set $\mathsf{TrialsA}$ in figure 2. Formally, trustees publish on the bulletin board the result of $\mathsf{PETDecryptUnlink}(\mathsf{CastBallots}, \mathsf{TR})$, where the motivation, specification and the algorithm for $\mathsf{PETDecryptUnlink}$ are discussed in section 4.3.

The set of all the cast ballots (that includes the trial ballots) are sent to the mixnet $\mathcal{M}$ for anonymization. Just after the mix and before the PET tests for eligibility enforcement, we propose for all ballots to be decrypted and their corresponding decrypted credentials and decrypted votes to be posted on the bulletin board: this is the set $\mathsf{DecBallots}$ in figure 2. To preserve coercion-resistance of the system, this has to be done in a way that does not link the published credentials and votes to the corresponding ballot. We propose the use of a decryption mix $\mathsf{DecryptUnlink}(\mathsf{MixedBallots})$, whose idea is discussed in section 4.3.

At the time of eligibility enforcement, credentials in ballots are additionally tested against the trial roll TR. If a ballot is identified as trial, it is not discarded but is labeled as such on the bulletin board. Finally, all ballots that remain on the bulletin board after eligibility enforcement are decrypted and only votes that do not correspond to trial ballots are tallied. If a ballot is labeled as a trial ballot, the corresponding credential is also decrypted. The decrypted trial ballots after the tabulation form the set $\mathsf{TrialsB}$ in figure 2.

4.2 Individual verifiabiliy in Trivitas

A voter can trace his trial vote in each phase of the system: it should be present on the bulletin board in the set TrialsA, after the voting phase, and in the set TrialsB, after the tabulation phase. Moreover, relying on the decryption of all the ballots after the mix, the result of which is the set DecBallots on the bulletin board, the voter can check that the encryption and the mix has been correct for all of his cast ballots: the one with a real credential, the one with a trial credential and possibly the ones with fake credentials. Hence, fake credentials can also be used for the purpose of end-to-end individual verifiability. In summary, the voter $\mathcal{V}_i$ can check that:

	Verifiability test	**Assured property**
$\mathcal{IV}_1$	The pair (c_i^t, v_i^t) occurs in the set TrialsA on the bulletin board	The machine has correctly encoded $\mathcal{V}_i$'s votes and $\mathcal{V}_i$'s ballots have been correctly recorded on the bulletin board
$\mathcal{IV}_2$	The pairs $(c_i, v_i), (c_i^t, v_i^t)$ and all (c_i^f, v_i^f) occur in the set DecBallots on the bulletin board	All of $\mathcal{V}_i$'s submitted ballots have been input in the mixnet $\mathcal{M}$ and have been correctly processed and output by $\mathcal{M}$
$\mathcal{IV}_3$	The pair (c_i^t, v_i^t) occurs in the set TrialsB on the bulletin board	$\mathcal{V}_i$'s intended vote occurs in the final outcome

Let us argue why all these tests are valid, in the sense that, if they are satisfied for the voter $\mathcal{V}_i$, then the claimed properties hold with high probability for all of $\mathcal{V}_i$'s ballots: trial, real and fake. We leave rigorous proofs along the lines of [18, 19] as future work, and perform only an informal analysis in the following. As in JCJ/Civitas, we assume that either one member of $\mathcal{T}$ is honest or else that auditors check decryption proofs (*). However, this is transparent for the voter, who performs her own verification.

$\mathcal{IV}_1$. Assumption (*) ensures that the decryption of trials is correct: the published trial pair is indeed the content of $\mathcal{V}_i$'s trial ballot, that is present on the bulletin board. Then, the fact that a trial credential is indistinguishable from a real credential ensures that a cheating voting machine or a cheating bulletin board has to make a random guess, thus having at least a 50% probability of being detected.

$\mathcal{IV}_2$. Assumption (*) ensures that the set of published pairs (DecBallots) corresponds to the decryption of ballots output by $\mathcal{M}$ (MixedBallots). Therefore, $\mathcal{IV}_2$ assures that all of $\mathcal{V}_i$'s ballots are correctly output by the mix. Moreover, note that $\mathcal{IV}_2$ increases the assurance offered by $\mathcal{IV}_1$, because $\mathcal{V}_i$ can check the correct construction and transmission of all her ballots. Still, $\mathcal{IV}_1$ is useful to detect a potential problem as early as possible and also to identify more precisely the elements of the system that have caused the problem. For instance, we will see in the next section how $\mathcal{IV}_1$ allows for recoverability when a problem is detected before the mix.

$\mathcal{IV}_3$. The parallel decryption of trial votes gives some evidence for the voter that votes are not arbitrarily eliminated during eligibility enforcement. This is formally ensured by assumption (*).

4.3 Anonymous PETs and distributed decryption with ciphertext-plaintext unlinkability

We now come back to two cryptographic components of the proposed system that have been left out in section 4.1: PETDecryptUnlink and DecryptUnlink. PETDecryptUnlink is used to decrypt trial ballots while keeping them indistinguishable from other ballots. This is necessary for being able to rely on trial ballots to audit the system even after they are decrypted. DecryptUnlink is used to decrypt all ballots, without revealing the link between individual ballots and their content. This is necessary to preserve coercion-resistance: otherwise, a coercer could detect that a ballot cast with a fake credential has been eliminated before the final tally.

Recall that votes and credentials are encrypted with a public key KT, whose corresponding private part $\mathsf{priv}(\mathsf{KT})$ is distributed among $\mathcal{T}$. In the following, we assume $\mathcal{T} = \{T_1, \ldots, T_n\}$. The specification for PETDecryptUnlink and DecryptUnlink is as follows:

PETDecryptUnlink

Input: $\mathcal{S} = (\mathsf{ec}_1, \mathsf{ev}_1), \ldots, (\mathsf{ec}_m, \mathsf{ev}_m)$ and $\mathsf{TR} = \mathsf{ec}'_1, \ldots, \mathsf{ec}'_k$

Output: $\mathcal{O} = \{(c, v) \mid \exists i_{\mathcal{S}} \in \{1, \ldots, m\},\ \exists i_{\mathsf{TR}} \in \{1, \ldots, k\},\ \mathsf{dec}(\mathsf{ec}_{i_{\mathcal{S}}}) = \mathsf{dec}(\mathsf{ec}'_{i_{\mathsf{TR}}}) = c\ \&\ \mathsf{dec}(\mathsf{ev}_{i_S}) = v\}$

Unlinkability: for all $(c, v) \in \mathcal{O}$, the index $i_{\mathcal{S}}$ of $(\{c\}_{\mathsf{KT}}, \{v\}_{\mathsf{KT}})$ in $\mathcal{S}$ is indistinguishable from a random number in $\{1, \ldots, m\}$.

DecryptUnlink

Input: $\mathcal{S} = (\mathsf{ec}_1, \mathsf{ev}_1), \ldots, (\mathsf{ec}_m, \mathsf{ev}_m)$

Output: $\mathcal{O} = \{(c, v) \mid \exists i_{\mathcal{S}} \in \{1, \ldots, m\}.\ \mathsf{dec}(\mathsf{ec}_{i_{\mathcal{S}}}) = c\ \&\ \mathsf{dec}(\mathsf{ev}_{i_S}) = v\}$

Unlinkability: for all $(c, v) \in \mathcal{O}$, the index $i_{\mathcal{S}}$ of $(\{c\}_{\mathsf{KT}}, \{v\}_{\mathsf{KT}})$ in $\mathcal{S}$ is indistinguishable from a random number in $\{1, \ldots, m\}$.

Our proposed implementation for PETDecryptUnlink and DecryptUnlink is an adaptation of the decryption mix idea present in [9, 10] to the case of distributed El-Gamal. This setting has already been studied in e.g. [13, 12], that show moreover how the shuffle can be made verifiable. However, since we do not require a verifiable shuffle for our application (a misbehavior during decryption would be detected by the voter by simply observing the trial credentials) our algorithms are more straightforward and do not provide zero-knowledge proofs. We only describe the algorithm for PETDecryptUnlink, the second algorithm being similar and more simple.

PETDecryptUnlink. For all et in TR, the parties $T_1, \ldots, T_n$ (holding private key shares $x_1, \ldots, x_n$) run the following protocol:

Initial phase (can be run publicly by any party).
Assume $\mathsf{et} = (a, b)$ and, for all $1 \leq i \leq m$, assume $\mathsf{ec}_i = (a_i, b_i)$. Compute and publish $p_1 = (\frac{a_1}{a}, \frac{b_1}{b}), \ldots, p_m = (\frac{a_m}{a}, \frac{b_m}{b})$. The input for T_1 in the next phase is $(p_1, \mathsf{ec}_1, \mathsf{ev}_1), \ldots, (p_m, \mathsf{ec}_m, \mathsf{ev}_m)$.

PET phase (being run privately and consequently by each of $T_1, \ldots, T_n$). Let $(p_1, \mathsf{ec}_1, \mathsf{ev}_1), \ldots, (p_m^p, \mathsf{ec}_m, \mathsf{ev}_m)$ be the input for T_i. Create new random numbers $r_1^p, \ldots, r_m^p \in \mathbb{Z}_q^*, r_1^c, \ldots, r_m^c \in \mathbb{Z}_q^*, r_1^v, \ldots, r_m^v \in \mathbb{Z}_q^*$ and compute

- $(c_1, d_1) = \mathsf{renc}(p_1, r_1^p), \ldots, (c_m, d_m) = \mathsf{renc}(p_m, r_m^p)$
- $\mathsf{ec}'_1 = \mathsf{renc}(\mathsf{ec}_1, r_1^c), \ldots, \mathsf{ec}'_m = \mathsf{renc}(\mathsf{ec}_m, r_m^c)$
- $\mathsf{ev}'_1 = \mathsf{renc}(\mathsf{ev}_1, r_1^v), \ldots, \mathsf{ev}'_m = \mathsf{renc}(\mathsf{ev}_m, r_m^v)$

Partially decrypt $(c_1, d_1), \ldots, (c_m, d_m)$, i.e. compute $d'_1 = \frac{d_1}{c_1^{x_i}}, \ldots, d'_m = \frac{d_m}{c_m^{x_i}}$.

Choose a permutation σ of $\{1, \ldots, m\}$ and publish

$$((c_{\sigma(1)}, d'_{\sigma(1)}), \mathsf{ec}'_{\sigma(1)}, \mathsf{ev}'_{\sigma(1)}), \ldots, ((c_{\sigma(m)}, d'_{\sigma(m)}), \mathsf{ec}'_{\sigma(m)}, \mathsf{ev}'_{\sigma(m)})$$

This is the input for T_{i+1}.

Decryption phase (run jointly by $T_1, \ldots, T_n$). For all $(p, \mathsf{ec}, \mathsf{ev})$ in the output of T_n: if $p = 1$, perform a distributed decryption of ec and of ev and make the result part of the output set: $\mathcal{O} := \mathcal{O} \cup \{(\mathsf{dec}(\mathsf{ec}), \mathsf{dec}(\mathsf{ev}))\}$.

If at least one of $T_1, \ldots, T_n$ behaves honestly, PETDecryptUnlink satisfies also the unlinkability requirement, as formalized and proved in [12].

5 Other properties

In this section we discuss other possible applications of trial credentials.

5.1 Universal verifiability.

We propose the following universal verifiability test for Trivitas:

	Verifiability test	**Assured property**
$\mathcal{UV}$	All the trials published before the mix are in the set of decrypted ballots after the mix, i.e. TrialsA $\subseteq$ DecBallots, and they have the same number of occurences	The mixnet $\mathcal{M}$ is correctly processing all the ballots

We propose this test as an addition to the current universal verifiability proofs, not as a replacement: it is more efficient, but probably offers less assurance than traditional zero-knowledge proofs. On the other hand, this test could also be combined with other tests that offer as well lesser guarantees of correctness but better performance [6], in order to improve their assurance while preserving their efficiency.

Let us argue about the validity of $\mathcal{UV}$. Because PETDecryptUnlink(CastBallots, TR) does not give away what ballots among CastBallots are trials, $\mathcal{M}$ has to treat all the ballots in CastBallots uniformly. In particular, if it chooses to cheat on a subset of ballots in CastBallots, this subset is random. Therefore, if there are

enough trial ballots (this could be ensured for instance by letting observers insert any number of trials), a dishonest behaviour of $\mathcal{M}$ would be detected with high probability by the test $\mathcal{UV}$.

These arguments hold only when the voting machines are not corrupted. Otherwise, a possibly corrupted $\mathcal{M}$ could differentiate trial ballots from other ballots when they are decrypted. We address this problem by a variation of Trivitas that does not let the machine learn which ballots are trials, even when they are decrypted (section 5.3).

5.2 Recoverability from failed verification

What happens when individual verifiability fails, e.g. an incorrect trial vote is published along his trial credential? In general, this issue is quite complex, because it requires procedures to determine who is telling the truth: the voter or the voting system. For Trivitas, our proposed recoverability technique is straightforward and requires only a slight modification to the system: trials are decrypted and published in short time after the ballots are cast and the voter does not have to wait for the end of the voting phase to verify a trial. Then, if a voter observes a problem with her trial vote on the bulletin board, she should simply re-vote, using a potentially safer machine. The policy for handling duplicate votes would then be to consider only the last vote as being valid, because it is the vote in which the voter has the highest confidence.

However, like in the case of universal verifiability, this solution is not ideal, because a compromised machine could make a distinction between trial credentials and valid credentials, after trial ballots are decrypted. The variant of Trivitas in the next section addresses this isue.

5.3 The case of a compromised voting machine

We propose a variant of Trivitas whose aim is to allow universal verifiability and recoverability as discussed above, even in presence of compromised voting machines. The main property of this variant is that it preserves the secrecy of the trial credential, while still allowing the voter to verify a trial vote relying on that credential. The cost is a slightly more complicated voting and vote verification experience:

- along with c and c_t, the voter additionally receives (or constructs) two numbers: one corresponding to a random r and one to $\{r\}_{\mathsf{KT}}$. We may assume that the same protocol is run for obtaining c, c_t and r and hence that the value of r is secret and known only to the voter.
- when constructing a ballot, the voter inputs not only the credential and the vote, but also $\{r\}_{\mathsf{KT}}$.
- when decrypting trial ballots, the trustees $\mathcal{T}$ do not decrypt directly the credential $\{c_t\}_{\mathsf{KT}}$ but instead multiply it with $\{r\}_{\mathsf{KT}}$, to obtain $\{c_t \cdot r\}_{\mathsf{KT}}$ (relying on the homomorphic properties of El-Gamal) and decrypt it to $c_t \cdot r$. Hence, instead of looking for a pair (c_t, v_t) on the bulletin board (like in the

basic version of Trivitas), the voter would look for $(c_t \cdot r, v_t)$ (for usability, one can see that an additive homomorphism, also possible with El-Gamal, would be better here).

In this variation of Trivitas, even if the voting machine is compromised, it can not be used to identify which ballots are trials. Hence, trial ballots can also be used for universal verifiability. For recoverability, a trial credential could be used multiple times and the machine would still be forced to take a 50% chance of getting caught each time when it is cheating.

6 Coercion-resistance

In this section we discuss why Trivitas offers the same coercion-resistance guarantees as JCJ/Civitas. Coercion-resistance in JCJ/Civitas relies on the ability of the voter to create a fake credential c_f and a fake proof P_f that satisfy the following properties:

- given a pair (c', P'), a coercer can not determine if the pair represents a voter's real credential and proof (c, P_f) or if it represents a fake pair (c_f, P_f). This is due to the fact that at least one registrar is assumed to be honest and the communication channel used with that registrar is assumed to be untappable.
- if a ballot with a fake credential is submitted, it will be eliminated from the final tally in the tabulation phase, during eligibility enforcement. Crucially, all ballots have been mixed and re-encrypted and at least one member of the mix network is assumed to be honest. This ensures that a coercer can not observe to what credentials correspond the ballots that have not been included in the final tally.

As usual, we also have to assume that there are enough votes for each candidate, so that the coercer can not observe that the voter did not follow his instructions from the mere outcome of the election.

The first addition of Trivitas, trial credentials, does not affect the way in which real ballots and fake ballots are handled by the election system. The only observable difference for the coercer is the presence of decrypted trial ballots at every phase and this does not give any information about real ballots or fake ballots. In particular, the two properties mentioned above remain true in presence of trial ballots.

The second addition of Trivitas, universal decryption, is potentially more problematic for coercion-resistance, since it concerns all the recorded ballots. However, coercion-resistance is preserved by two crucial points:

- all ballots are decrypted, without making a difference between real credentials, fake credentials and trial credentials. This ensures that, in Trivitas as in JCJ/Civitas, the coercer can not determine if a credential is valid or not.

– the algorithm applied to decrypt all ballots is a decryption mix, i.e. we apply DecryptUnlink(MixedBallots). It may be surprising that a set of anonymized ballots is decrypted with a decryption mix. However, this is needed because trustees must eliminate fake ballots in a publicly verifiable way. In that case, if the coercer could additionally see the contents of all ballots, he could determine what credentials were invalid.

Towards a formal proof. Let us sketch how coercion-resistance proof for JCJ/Civitas [18] could be extended to cover Trivitas. To define coercion-resistance for an election system E in a computational model, [18] considers an ideal system $\mathsf{E}_{\mathcal{I}}$ where the outcome of the election is "magically" computed: the adversary can observe only the final outcome and is not able to influence anything more than vote choices for the compromised voters. Then, a system is said to satisfy coercion-resistance if the probability of a polynomial time adversary being able to determine if it has been cheated is roughly the same when the election is run by E as in the case when the election is run by $\mathsf{E}_{\mathcal{I}}$.

The proof of coercion-resistance is a reduction to (a variant of) the Decision-Diffie Hellman (DDH) assumption: no polynomial time algorithm can distinguish between a Diffie-Hellman tuple (g_1, g_1^x, g_2, g_2^x) and a random tuple (a, b, c, d). Then, the proof relies on a simulator $\mathcal{S}$ that behaves either as E or as $\mathsf{E}_{\mathcal{I}}$, depending on whether its input is a Diffie-Hellman tuple or not. If there would be a polynomial time adversary that breaks coercion-resistance, i.e. it has better chances of coercion when E is used instead of $\mathsf{E}_{\mathcal{I}}$, then that adversary could be used by the simulator $\mathcal{S}$ to determine if its input is a Diffie-Hellman tuple in polynomial time. This would break the DDH assumption.

To extend this proof to Trivitas all we have to do is to show that the simulator $\mathcal{S}$ of [18] can also execute the additional operations, i.e. the management of trial credentials and the universal decryption of all ballots after the mix. This is possible because the simulator of [18] holds the private key $\mathsf{priv}(\mathcal{T})$ and can therefore decrypt ballots at any time. This makes it possible to simulate both the audit of trial ballots and the universal decryption. Moreover, the same simulator can easily create trial credentials and trial ballots, this process being similar to the creation of real credentials and ballots.

7 Conclusion and future work

We have proposed several additions to JCJ/Civitas that improve its individual verifiability. We introduce trial credentials that offer to voters the ability to audit the election process at any stage: creation of ballots, their transmission to the bulletin board, their processing by the mixnet and their final decryption. Moreover, we observe that we can rely on the presence of fake ballots and trial ballots on the bulletin board after the mix to decrypt all ballots without compromising coercion-resistance. This certainly improves individual verifiability: to our knowledge, this is the first mixnet based system where a voter can directly verify that her actual vote is correctly recorded in the system after the mix. Not

only that, but she can also cast as many votes as she likes with fake credentials and check that they are all correctly output after the mix.

The idea of trial ballots is not necessarily specific to JCJ/Civitas. We believe it could be implemented in other electronic voting systems as well, although this requires further research. The universal decryption of ballots after the mix relies on the notion of credentials, to allow voters to identify their votes, and of fake credentials, to allow coercion-resistance. Credentials are also interesting for eligibility verifiability, possible in JCJ/Civitas but generally not possible in other systems. Hence, it would be interesting to investigate the possibility of adding a credential infrastructure on top of other E-voting protocols.

We also plan to develop ideas and variations sketched in section 5. In particular, putting the mechanism for recoverability in the hands of the voter, instead of third party organizations, looks more appealing both from the perspective of the voter and from the perspective of election authorities. Finally, the sketch of coercion-resistance proof from section 6 has to be completed, and this may open other research directions, relating iterative protocol development and the corresponding security proofs.

Acknowledgments. Thanks to Jeremy Clark, Aleksander Essex and anonymous referees for useful comments and interaction on the ideas of the paper. We gratefully acknowledge financial support from EPSRC, through the projects EP/G02684X/1 "Trustworthy Voting Systems" and EP/H005501/1 "Analysing Security and Privacy Properties".

References

1. Ben Adida. Helios: Web-based open-audit voting. In Paul C. van Oorschot, editor, *USENIX Security Symposium*, pages 335–348. USENIX Association, 2008.
2. Ben Adida and Andrew C. Neff. Ballot casting assurance. In *Usenix/ACCURATE Electronic Voting Technology Workshop, Vancouver, BC,Canada*, 2006.
3. Josh Benaloh. Simple verifiable elections. In *Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop*, pages 5–5, Berkeley, CA, USA, 2006. USENIX Association.
4. Josh Benaloh. Ballot casting assurance via voter-initiated poll station auditing. In *Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology*, pages 14–14, Berkeley, CA, USA, 2007. USENIX Association.
5. Matt Blaze, Arel Cordero, Sophie Engle, Chris Karlof, Naveen Sastry, Micah Sherr, Till Stegers, and Ka-Ping Yee. Source code review of the Sequoia voting system. In *Report commissioned as part of the California Secretary of State's Top-To-Bottom Review of California voting systems*, July 20, 2007.
6. Dan Boneh and Philippe Golle. Almost entirely correct mixing with applications to voting. In Vijayalakshmi Atluri, editor, *ACM Conference on Computer and Communications Security*, pages 68–77. ACM, 2002.
7. 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.
8. Joseph A. Calandrino, Ariel J. Feldman, J. Alex Halderman, David Wagner, Harlan Yu, and William P. Zeller. Source code review of the Diebold voting system. In

Report commissioned as part of the California Secretary of State's Top-To-Bottom Review of California voting systems, July 20, 2007.
9. David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy*, 2(1):38–47, 2004.
10. David Chaum, Peter Y. A. Ryan, and Steve A. Schneider. A practical voter-verifiable election scheme. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *ESORICS*, volume 3679 of *Lecture Notes in Computer Science*, pages 118–139. Springer, 2005.
11. 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.
12. Jun Furukawa. Efficient, verifiable shuffle decryption and its requirement of unlinkability. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 319–332. Springer, 2004.
13. Jun Furukawa, Hiroshi Miyauchi, Kengo Mori, Satoshi Obana, and Kazue Sako. An implementation of a universally verifiable electronic voting scheme based on shuffling. In Matt Blaze, editor, *Financial Cryptography*, volume 2357 of *Lecture Notes in Computer Science*, pages 16–30. Springer Berlin / Heidelberg, 2003. 10.1007/3-540-36504-4_2.
14. Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 368–387. Springer, 2001.
15. Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 539–556. Springer, 2000.
16. 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.
17. Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In Dan Boneh, editor, *USENIX Security Symposium*, pages 339–353. USENIX, 2002.
18. 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.
19. Steve Kremer, Mark Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, editors, *ESORICS*, volume 6345 of *Lecture Notes in Computer Science*, pages 389–404. Springer, 2010.
20. C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *ACM Conference on Computer and Communications Security*, pages 116–125, 2001.
21. C. Andrew Neff. Practical high certainty intent verification for encrypted votes, 2004.
22. 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.
23. Steve Schneider, Morgan Llewellyn, Chris Culnane, James Heather, Sriramkrishnan Srinivasan, and Zhe Xia. Focus group views on Pret a Voter 1.0. In *REVOTE, International Workshop on Requirements Engineering for Electronic Voting Systems*, 2011.