When is a test not a proof?

Eleanor McMurtry[1], Olivier Pereira[2], and Vanessa Teague[3]

[1] University of Melbourne, emcmurtry@student.unimelb.edu.au
[2] Université catholique de Louvain, ICTEAM, olivier.pereira@uclouvain.be
[3] Thinking Cybersecurity Pty. Ltd. and the Australian National University, vanessa@thinkingcybersecurity.com

Abstract. A common primitive in election and auction protocols is a *plaintext equivalence test* (PET) in which two ciphertexts are tested for equality of their plaintexts, and a verifiable proof of the test's outcome is provided. The most commonly-cited PETs require at least one honest party, but many applications claim *universal verifiability*, at odds with this requirement. If a *test* that relies on at least one honest participant is mistakenly used in a place where a *universally verifiable proof* is needed, then a collusion by all participants can insert a forged proof of equality into the tallying transcript. We show this breaks universal verifiability for the JCJ/Civitas scheme among others, because the only PETs they reference are not universally verifiable. We then demonstrate how to fix the problem.

1 Introduction

We consider the distinction between *universal verifiability* and *distributed trust*, in particular for Plaintext Equivalence Tests (PETs). Commonly-cited PETs such as [17] and [19] require at least one honest participant. Although this is clear enough in the original papers, numerous uses of PETs assume universal verifiability, which requires that a convincing proof of an untrue fact cannot be fabricated even if all the authorities collude. Many e-voting and verifiable auction protocols use a Plaintext Equivalence *Test* (PET, with distributed trust) as if it were a Plaintext Equivalence *Proof* (PEP), when it is not. They require their PET to have universal verifiability, but reference only PETs that do not have this property. Specifically, in Jakobsson and Juels [17] a collusion of all authorities can forge a "proof" that two encrypted values are equal when they are not. If in a real implementation this PET was used instead of a PEP, the consequence would completely undermine universal verifiability for:

- the JCJ e-voting scheme [18] and its implementation, Civitas [8];
- a linear-time modification to JCJ [23];
- the Pretty Good Democracy scheme [21];
- the Caveat Coercitor scheme [12];
- the Selections scheme [7];
- the Cobra scheme [11]; and
- several verifiable auction schemes [2], [6], [20]

In each case, the forged PEP enables a collusion among all tallying authorities to exclude valid votes or include invalid votes while passing verification, thus breaking universal verifiability. We stress that this is not a fault in the PET of [17], but in the misalignment between the properties it provides and the properties it is assumed to have when used in the above schemes.

We show further that this problem combines with an existing issue in the Civitas implementation of the Fiat-Shamir transform to disastrous effect, allowing colluding authorities to also forge a proof that two encrypted values are *not* equal when in fact they are.

1.1 Addressing the Problem

PETs with distributed trust, including [17], are generally easy to transform into a universally-verifiable Plaintext Equality Proof (PEP). We demonstrate how a faked positive test can be detected in [17] with an additional check in the verification procedure, and that there is a negligible soundness error, *i.e.* probability of a false positive.

1.2 Our Contribution

We examine each protocol in turn, explaining the implications of using a PET that can be forged when all tallying authorities collude. We then explain how to patch the verification algorithm in the Jakobsson-Juels PET so that its transcript does allow universal verifiability. We also explore the implications of weak Fiat-Shamir transforms in one protocol, and explain how this can be fixed.

Mathematical details are provided in the following section. An examination of each affected scheme is in Section 4. The correction of the PET verification algorithm and a proof of universal verifiability are in Section 5.

2 The Jakobsson-Juels PET

We demonstrate that if all the trustees of a secret key collude, they can produce a valid plaintext equivalence proof for two ciphertexts that are not encryptions of the same value. We begin by reviewing the details of a PET from the widely-referenced Jakobsson-Juels paper [17], and explain why it does not form a PEP.

2.1 Plaintext Equivalence Test

Let $(\mathbb{G}, g, p, q, y)$ be the parameters for ElGamal encryption, with g a generator for the group $\mathbb{G}$, and $q = \langle g \rangle$, *i.e.* $a^q = 1 \bmod p$ for all $a \neq 0$ where p and q are large prime numbers. The public key is $y = g^x$ for secret key x. Consider two ElGamal ciphertexts $(a_1, b_1) = (g^{r_1}, m_1 y^{r_1})$ and $(a_2, b_2) = (g^{r_2}, m_2 y^{r_2})$ where $m_1 \neq m_2$.

For ease of exposition, suppose there are two trustees T_1 and T_2 who collude (but it works just as well for more trustees). Each trustee has its own share

x_i $(i = 1, 2)$ of the secret key, corresponding to its own public key $y_i = g^{x_i}$, arranged so that $x_1 + x_2 = x \bmod q$ and hence $y_1 y_2 = y \bmod p$.

They will produce a false proof that $m_1 = m_2 \bmod q$. We (publicly) set

$$C = (a_1/a_2, b_1/b_2) = \left(g^{r_1 - r_2}, (m_1/m_2) y^{r_1 - r_2}\right)$$

The main idea of the PET is that the two trustees will each raise (each element of) C to a random power, prove in zero knowledge that they indeed used the same power on all the elements of C, multiply all their exponentiated values together, then decrypt the result, using another Zero Knowledge Proof to prove they decrypted correctly. Call the trustees' random exponents ρ_1 (for T_1) and ρ_2 (for T_2). If ρ_1 and ρ_2 are chosen randomly, and if the decrypted value $(m_1/m_2)^{\rho_1 + \rho_2} = 1$, then with high probability[4] $m_1/m_2 = 1$. So as long as at least one trustee is honest, the proof is both sound and privacy-preserving. A detailed example is shown in Figure 1.

Exponential ElGamal A common variant of ElGamal, called *exponential ElGamal*, encrypts a message m by first computing g^m and then encrypting this value as usual in order to create additively-homomorphic ciphertexts. We will work with this variant when discussing protocols that use it.

2.2 Why the PET Is Not a Proof

The PET described above does not form a plaintext equivalence *proof*. If the trustees collude and set $\rho_1 = -\rho_2 \bmod q$ (easy to do in a setting like ElGamal where q is public), then they turn C into an encryption of 1 because $(m_1/m_2)^q = 1$ even when $m_1 \neq m_2$. In fact, they turn C into the trivial ciphertext $(1, 1)$, which looks unusual to a human eye but is a perfectly valid ElGamal encryption of 1. Hence the colluding trustees, even if they decrypt honestly, can produce a false proof that $m_1 = m_2$.

Adding a simple check for the trivial ciphertext $(1, 1)$ solves the problem and produces a universally verifiable Proof of Plaintext Equality (PEP). We prove this in Section 5.

However, the PEP is strongly dependent on the soundness of its ZKPs. If these are not properly implemented, the PEP is not sound.

3 Flaws in a Practical Implementation of the PEP

One particularly influential implementation of the PEP and zero knowledge proofs discussed above[5] is Civitas [8]. In the following section we discuss how weaknesses in Civitas' implementations further undermine its PEPs, even if we ensure the PEP does not produce the trivial ciphertext.

[4] The test may fail if by pure bad luck, $\rho_1 = -\rho_2 \bmod q$. This happens only with the negligible probability $1/q$, where q is large.

[5] Indeed, one of the *only* implementations we were able to find.

3.1 Use of Zero Knowledge Proofs (ZKPs)

A practical implementation of the Jakobsson-Juels PET outlined above depends on two zero knowledge proofs. **First**, when trustee T_i takes $C = (a_1/a_2, b_1/b_2)$ and produces a randomised output $((a_1/a_2)^{\rho_i}, (b_1/b_2)^{\rho_i})$, an observer has no way to guarantee that T_i did indeed raise both elements to the same power. To this end, a zero knowledge proof must be used to prove the equality of two discrete logarithms. Set $d = a_1/a_2, d_i = d^{\rho_i}$ and $e = b_1/b_2, e_i = e^{\rho_i}$; then we must prove

$$\mathsf{dlog}_d d_i = \mathsf{dlog}_e e_i$$

Second, the trustees (collectively) need to prove that they faithfully decrypted the resulting ciphertext $(a, b) = (\prod_i d_i, \prod_i e_i)$. They do this by publishing $a_i = a^{x_i}$ and a proof that

$$\mathsf{dlog}_g y_i = \mathsf{dlog}_a a_i$$

guaranteeing that a really was raised to the power of the secret key to form a_i.

We will use $\mathsf{EqDlogs}(a, b, x, y)$ to mean a proof that $\mathsf{dlog}_a x = \mathsf{dlog}_b y$.

3.2 Making Equivalent Ciphertexts Look Different

The plaintext equivalence test (used as a proof) from the Civitas technical report [8] is reproduced as Figure 1 below.

B.3 Main protocols

PROTOCOL: Plaintext Equivalence Test (PET)

Due to:	Jakobsson and Juels [51]
Principals:	Tabulation tellers TT_i
Public input:	$c_j = \mathsf{Enc}(m_j; K_{\mathrm{TT}}) = (a_j, b_j)$ for $j \in \{1, 2\}$
Private input (TT_i):	Private key share x_i
	Let $R = (d, e) = (a_1/a_2, b_1/b_2)$
Output:	If $m_1 = m_2$ then 1 else 0

1. TT_i: $z_i \leftarrow \mathbb{Z}_q^*$; $(d_i, e_i) = (d^{z_i}, e^{z_i})$
2. TT_i: Publish $\mathsf{Commit}(d_i, e_i)$
3. TT_i: Barrier: wait until all commitments are available
4. TT_i: Publish (d_i, e_i) and proof $\mathsf{EqDlogs}(d, e, d_i, e_i)$
5. TT_i: Verify all commitments and proofs
6. Let $c' = (\prod_i d_i, \prod_i e_i)$
7. All TT: Compute $m' = \mathsf{DistDec}(c')$ using private key shares
8. If $m' = 1$ then output 1 else output 0

Fig. 1. PET protocol specification from the Civitas technical report

We demonstrate that trustees can produce a valid proof that two plaintexts encrypt different values, even though they are encryptions of the same value.

Here, we proceed in a single trustee setting, but this can be trivially extended to the multiple trustee setting.

We take advantage of a weaknesses in the way the Fiat-Shamir transform is implemented as part of the decryption proof in the Civitas documentation (Step 7 of Figure 1). The proof is specified as follows, for an ElGamal public key (g, y) where $y = g^x$, a ciphertext (a, b), and a decryption factor d that is supposed to be equal to a^x:

- Select a random $r \in \mathbb{Z}_q$ and compute the commitment $(r_0, r_1) = (g^r, a^r)$
- Compute the challenge $e = H(y, d, r_0, r_1)$
- Compute the response $f = r + ex$

The proof is considered to be valid if the following two equations are satisfied:

$$g^f = r_0 h^e \qquad a^f = r_1 d^e$$

and if e can be recomputed correctly. This is supposed to form a proof

$$\mathsf{EqDlogs}(g, a, y, d)$$

Note that the ciphertext's elements a and b have not been included in the hash, so a malicious prover can choose them after computing the challenge. Here is a strategy that a malicious trustee could use to produce a ciphertext that is an encryption of 1 and prove that it is an encryption of something else.

- Select random $r, s, t \in \mathbb{Z}_q$ and set $(r_0, r_1) = (g^r, g^s)$ and $d = g^t$
- Compute the challenge $e = H(y, d, r_0, r_1)$
- Compute the response $f = r + ex$
- Compute $a = g^{(s+et)/f}$

This is enough to make a proof that passes verification, and will prove that d is a valid decryption factor for a with respect to the public key (g, y), even though it is not the case (with overwhelming probability): d is just picked as a random group element. At this stage, we still have the freedom to choose any b we like in order to complete our ciphertext. In particular, we can set $b = y^{(s+et)/f}$, which guarantees that (a, b) is an encryption of 1. Thus we can produce a valid proof that (a, b) decrypts to b/d for arbitrary d, even though it is actually an encryption of 1.

In order to use this to devise a fake PEP, the cheating prover needs to know the randomness used to generate the ciphertexts. Take two ciphertexts (a_1, b_1) and (a_2, b_2) that we know are encryptions of the same plaintext and for which we know the randomness; say, $a_1 = g^{r_1}$ and $a_2 = g^{r_2}$ for some r_1, r_2. We will take their quotient and raise it to a "random" power $\rho = \rho_1 + \rho_2$ as per the PET:

$$(a, b) = \left(\left(\frac{a_1}{a_2} \right)^{\rho}, \left(\frac{b_1}{b_2} \right)^{\rho} \right)$$

However, we will cheat to ensure that

$$\rho = \frac{s + et}{f} \frac{1}{r_1 - r_2}$$

We are left with $a = g^{(s+et)/f}$ as per our cheated proof. Because we may freely choose d, we can now produce a valid proof that this decrypts to some value other than 1, and thus a valid proof that the ciphertexts encrypt different plaintexts.

This will not only look perfectly fine, but it will also be infeasible to distinguish these ciphertexts and proofs from those expected in the system, unless we know x of course. This prevents the easy resolution discussed in Section 1.1.

In summary, a cheating prover, given two arbitrary equivalent ciphertexts, can forge a proof that they are different. This requires knowing the randomness used to generate the ciphertexts.

3.3 Making Encryptions of Different Messages Look Equivalent

The same issue can be exploited to produce proofs that two ciphertexts encrypting different messages actually encrypt the same message in a way that is undetectable to a public observer. This time, we will exploit the use of the weak Fiat-Shamir transform in the EqDlogs proof in the plaintext equivalence test (Step 4 of Figure 1). Again, for simplicity, we focus on the setting where we have a single trustee, which is malicious. This trustee can proceed as follows:

- Select a random encryption of 1 as $c' = (d_1, e_1) = (g^t, y^t)$
- Produce an EqDlogs proof as follows:
  1. Select random $r, s \in \mathbb{Z}_q$ and set $(r_0, r_1) = (g^r, y^s)$
  2. Compute the challenge $e_p = H(d_1, e_1, r_0, r_1)$
  3. Pick a random response $f \in \mathbb{Z}_q$
  4. Set $R = (d, e) = (g^{(r+te_p)/f}, y^{(s+te_p)/f})$

  and observe that (e_p, f) makes an $\mathsf{EqDlogs}(d, e, d_1, e_1)$ proof that passes verification despite the fact that $\log_d(d_1) \neq \log_e(e_1)$ with overwhelming probability.
- Complete the fake PEP by picking any pair of ciphertexts c_1, c_2 whose quotient is R, and producing a decryption proof for c' in a perfectly honest way.

Since c' is indeed an encryption of 1, the test will conclude that c_1 and c_2 are encryptions of the same plaintext, despite the fact that they encrypt different plaintexts with overwhelming probability. Below are two practical ways to exploit this.

If we are given one ciphertext with unknown randomness c_1 (say on a web bulletin board), we can construct a second ciphertext $c_2 = Rc_1$ so that $c_1/c_2 = R$, where we may freely choose R as above. Colluding tellers could exploit this by adding an extra ballot to the mix, and producing a PEP that falsely claims the targeted vote was cast with a duplicate credential. Note that this would require knowledge of some valid credential. This can be done by a colluding majority of authorities, where a PEP is meant to reveal the cheat, but of course we have broken the PEP [23].

If we know the randomness for both c_1 and c_2 (say z_1 and z_2) encrypting g^{m_1} and g^{m_2} (Civitas uses exponential ElGamal here) so that $c_1 = (g^{z_1}, g^{m_1} g^{xz_1})$ and $c_2 = (g^{z_2}, g^{m_2} g^{xz_2})$, we

1. choose t and set $\rho = \rho_1 + \rho_2 = \frac{r+te_p}{f(z_1-z_2)}$
2. choose f and set $w = \frac{f}{x}\frac{m_1}{m_2}$ (so that $f = wx\frac{m_2}{m_1}$)
3. choose r and set $s = r + w$

Now when we produce the re-randomised quotient $R = (c_1/c_2)^\rho$, we need to satisfy

$$R = \left(g^{\rho(z_1-z_2)}, g^{m_1/m_2} g^{x\rho(z_1-z_2)}\right)$$

We must satisfy $\rho(z_1-z_2) = (r+te_p)/f$ and $m_1/m_2 + x\rho(z_1-z_2) = x(s+te_p)/f$ as per Step 4 of the cheated proof above. The former is satisfied by definition of ρ. For the latter:

$$\begin{aligned}
\frac{m_1}{m_2} + x\rho(z_1 - z_2) &= \frac{m_1}{m_2} + x\frac{r+te_p}{wx\frac{m_2}{m_1}} && \text{(Eqn 1)} \\
&= \frac{m_1}{m_2}\left(1 + \frac{r+te_p}{w}\right) \\
&= \frac{m_1}{m_2}\frac{r+w+te_p}{w} \\
&= \frac{m_1}{m_2}(s+te_p)\frac{x}{f}\frac{m_2}{m_1} && \text{(Eqns 2, 3)} \\
&= x(s+te_p)/f
\end{aligned}$$

Colluding tellers could exploit this in the same way as the previous attack, but without needing to add an extra ballot to the mix.

3.4 Summary and Implications of these Vulnerabilities

If every party performing the PET colludes, they can

– produce a false proof that any two ciphertexts are equivalent, without needing to know the randomness used to generate them.

This problem can be prevented by adding a check for the trivial ciphertext to the verification algorithm.

Using the Fiat-Shamir weakness, they can additionally

– produce a false proof that two equivalent ciphertexts are different, if they know the randomness used to generate them (Section 3.2)
– produce a false proof that two ciphertexts are equivalent (Section 3.3), if
  - they know the randomness used to generate them, or
  - they can generate one of them (in which case there is no need to know the randomness of the other).

These further attacks will produce plausible non-trivial ciphertexts.

4 Why this Undermines Universal Verifiability in JCJ/Civitas and Other Protocols

Universal verifiability (UV) means that any observer can confirm an election's tally matches its cast votes; in particular, any voter could perform this verification [9]. See [3,1,22] for early work on this concept.

If the verification specification does not consider the case where all trustees of the PEP collude to produce a false outcome, then only the trustees can verify the plaintext equivalence proof — since they know whether they colluded. Therefore, missing this step results in a scheme whose tallying phase is not universally verifiable.

4.1 JCJ/Civitas

We investigate whether Civitas [8], an implementation of a well-known voting protocol we refer to as JCJ [18], correctly performs the check for a trivial ciphertext in its PEPs. The key idea behind the JCJ protocol is to assign each voter an encrypted credential. To provide coercion resistance, a voter can provide the coercer with an invalid credential, and the coercer cannot tell that the credential is not valid. Voters submit their encrypted votes and encrypted credentials, together with proofs of plaintext knowledge, to a bulletin board. To tally votes, the tabulation tellers:

1. check the proofs of plaintext knowledge for submitted votes,
2. identify (using a PEP) and exclude any votes with duplicate credentials,
3. perform a cryptographic mix on the encrypted votes,
4. identify (using a PEP) and exclude any votes cast with a credential that does not match any on the list of valid credentials, and
5. decrypt the resulting list of valid votes (but not the corresponding credentials).

We can immediately see that a flawed PEP could be abused by the tabulation tellers to cheat in the election.

In its technical report [8], Civitas explicitly claims universal verifiability regardless of whether the tellers collude on page 10: "Trust Assumption 6. There exists at least one honest tabulation teller. If all the tellers were corrupted, then the adversary could trivially violate coercion resistance by decrypting credentials and votes. This assumption is not needed for verifiability, even if all the tellers collude or are corrupted—the proofs posted by tellers during tabulation will reveal any attempt to cheat." Its verification specification, page 43 of the technical report reproduced as Figure 1 here, does not mention checking for a $(1, 1)$ ciphertext which is necessary for this property.

How did this happen? The JCJ paper has a proof of universal verifiability of the overall voting protocol that is not exactly wrong, but leaves a trap for the unwary implementer. It says of their PETs that "we model the ideal properties

of the primitive as an oracle [...] with the property of public verifiability",[6] citing the Jakobsson-Juels paper [17]. However, the protocol in that paper does not have the UV property, and does not claim it, proving instead that it is sound under the assumption that at least one participant is honest. (It also does not explicitly mention that even with this assumption, the protocol has a negligible but non-zero soundness error.)

JCJ also cites [19], which in its Section 4.4 describes another PET, also with the explicit assumption that fewer than a threshold of trustees are compromised; however, this paper correctly points out that the protocol has a negligible but non-zero soundness error. So if JCJ were implemented using a PEP with universal verifiability, then (we believe) the overall protocol would also be UV, but since its only cited example is a PET that is not UV, a natural implementation risks failing UV — as indeed the Civitas implementation does.

The incorrect PEP together with the Civitas scheme's weak Fiat-Shamir transform allows various attacks that undermine universal verifiability for JCJ, such as:

- falsely proving that two different credentials are duplicates, hence excluding them at Step 2,
- falsely proving that a valid credential is different from everything on the list of valid credentials, hence excluding it at Step 4,
- falsely proving that an invalid credential is equivalent to something on the list of valid credentials, hence including it at Step 4.

4.2 A Linear-Time Enhancement to JCJ

In [23], an enhancement to the JCJ protocol is proposed whereby tallying can be done in linear time instead of a worst-case quadratic time. They "assume the application of publicly verifiable group threshold mechanisms whenever registering or tallying authorities perform joint computations", which could be taken to mean that our attack is outside their threat model; however this seems at odds with the notion of universal verifiability, and the majority of their discussion of security properties relies on that of JCJ. So the protocol is not broken as such, but does lay a similar trap for the unsuspecting reader: they are referred to JCJ, which claims UV but refers the reader to Mix & Match.

4.3 Introduction to Cryptography Textbook

The textbook *Introduction to Cryptography: Principles and Applications* (3rd edition) [10] discusses distributed plaintext equivalence tests in Section 4.7, basing their discussion on JCJ (and referencing Mix & Match). They unfortunately make the same oversight, commenting that "the test may give a wrong result only if $z = \sum_{i=1}^{n} z_i = [0]$" and that "this happens only with the negligible probability $1/q$", not accounting for the case where every party colludes. When discussing

[6] Public verifiability is a synonym for UV.

privacy of the distributed PEP, they explain "the privacy of $\mathcal{P}k$-encrypted ciphertexts and the threshold decryption are guaranteed if at least t of the n parties are honest and no coalition of t parties behaves dishonestly", but do not discuss the implications of dishonest parties on the *integrity* of the proof.

4.4 Pretty Good Democracy

Pretty Good Democracy [21,15] also uses Plaintext Equivalence Tests in a critical part of the tallying process, claiming that Counted-as-Cast and Tallied Correctly "can be verified from the Bulletin Board using standard techniques, which do not require trusting any of the authorities." But it refers only to the PETs of [17], which do require trusting at least one authority to be honest. A collusion of all tallying authorities could fake a match with a correct vote code, thus tricking a voter into thinking that the vote had been properly received when in fact an invalid vote had been substituted.

4.5 Selections

The Selections protocol [7] falls victim to a similar trap. It focuses on over-the-shoulder coercion-resistance, meaning that an adversary who is present during vote casting cannot coerce the voter. To achieve this, the paper mentions that the voter must trust at least one "registrar". However, it

1. explicitly claims end-to-end verifiability;
2. mentions the protocol can work with distributed registrars; and
3. clarifies its trust assumption by saying the trusted registrar must not collude with a coercer, but *may still misbehave in any other regard.*

In particular, the trustees that perform the tallying are not required to be honest, and the security assumptions for them are left unclear — they may or may not be the same entities as the registrars.

The tallying protocol (Algorithm 3 in the paper) uses a plaintext equality test among tallying trustees, and explicitly refers the reader to Mix & Match despite the lack of universal verifiability.

4.6 Cobra

The Cobra protocol [11] has the same problem as Selections because it is based on it, claiming to have universally verifiable proofs yet referring the reader to Juels and Jakobsson for a plaintext equality test.

4.7 Caveat Coercitor

Caveat Coercitor [12] aims to provide a new property called *coercion evidence* by giving voters an opportunity to signal that they have been coerced. A voter can cancel their vote (assuming it has been cast in a way that did not represent their

true wishes) by simply re-voting with the same credential but a different vote. This nullifies the vote and is interpreted as evidence of coercion. The talliers produce a tally showing which votes were cast with repeated credentials, for which they use PEPs. Those that are duplicates are not included in the final count. The exact protocol is similar to JCJ, except that when a credential is used twice to cast the same vote it is included (once). The talliers:

1. check the proofs of plaintext knowledge for submitted votes;
2. identify (using a PET) any votes with duplicate credentials:
   (a) identify (using a PET) whether all the corresponding votes are equal, in which case one is included;
   (b) or, if there are at least two different votes cast with the same credential, exclude them all.
3. perform a cryptographic mix on the encrypted votes;
4. identify (using a PET) and exclude any votes cast with a credential that does not match any from the list of valid credentials; and
5. decrypt the resulting list of valid votes (but not the corresponding credentials).

The idea is that coerced voters give up their vote, but that "unforgeable evidence about the degree of coercion that took place is included in the election output." If the talliers can fake apparently-equal PEPs for credentials or votes that are actually different, there are a number of ways they can cheat.

- In Step 2 they can take two unique credentials and fake a proof that they are equal, thus nullifying one of them (if the votes are equal) or both (if the votes are different).
- In Step 2a they can take two different votes cast with the same credential and fake a proof that the votes are equal, thus getting one of them included when both should be excluded. This is particularly important because this is exactly the case that Caveat Coercitor is designed to flag as coercion and exclude — instead, a faked PEP allows it to be treated as a repeat-casting of the same vote, so a vote is accepted and no coercion is flagged.
- Like JCJ/Civitas, in Step 4 Caveat Coercitor uses PEPs to test whether a vote's credential is equal to one of those on the list of valid credentials, so just as in JCJ/Civitas, a faked PEP for equality again allows corrupt authorities to stuff the ballot by pretending that an invalid credential is valid.

It is not entirely clear whether the possibility for all talliers to collude and fake PEPs is included in the trust model of Caveat Coercitor. The paper refers to the PETs as proofs, (petproof), and does not give an explicit construction, but cites the distributed-trust-based test of [17].

On p.371 (Sec B) it says "We assume that one mix server in M, one tallier in T and the voting machines are honest." The talliers correspond to the authorities for the PET, so if they are assuming that at least one tallier is honest then our attack is not within the security model. However, the paper in several places claims universal/public verifiability, such as "The talliers will execute an

algorithm whose output can be used by any external observer to determine the amount of coercion," (p.371) and "Observers can perform universal and eligibility verification, because the computations of algorithm 3 are publicly verifiable," (p.375). The crucial Lemma is Lemma 4, which considers whether the set of accepted votes corresponds correctly to the set of those that were cast with a unique credential or a repeated credential for the same vote. The proof (in the Appendix) simply treats the PETs as perfect and does not mention the honest tallier. The paper concludes by saying that "A major feature of the system is that the degree of coercion that actually took place is publicly verifiable." The assumptions and claims are thus somewhat ambiguous and inconsistent.

In summary, the degree of coercion is not universally verifiable because without at least one trustworthy tallier, evidence of coercion can be hidden.

4.8 Universally Verifiable Auctions

Several cryptographic auction protocols also use the PET from [17] while claiming universal verifiability, and therefore are are vulnerable to the same attack. In a verifiable auction there are n players who place a secret bid, and engage in an interaction to determine which was the highest bid (and who placed it) with convincing evidence that this was done correctly. There are two distinct settings: one where this evidence is enough for *each player* to be sure of the result, and one where it is enough for *any public observer* to verify the result.

For example, the scheme of Abe et al. [2] is claimed to be UV but uses the Mix & Match PET, so it cannot have this property. The protocol of Bradford et al. [6] also claims UV, citing Mix & Match for its PET as well as [16], an almost-identical PET (which is also not UV). Most notably, Quaglia et al. [20] present a method for converting a verifiable election protocol into a verifiable auction protocol; they use Helios as a case study, in which a plaintext equivalence proof is used to construct a universally verifiable auction from mixnet tallying. However, they once again cite Mix & Match as their PET, so fail to achieve UV.

5 Correcting the Problems to Achieve UV

5.1 The Fiat-Shamir Transform

When using the Fiat-Shamir transform to convert an interactive zero knowledge proof to a non-interactive equivalent, it is crucial that one includes the full statement to be proved in the hash [4]. Focusing on Civitas, for the decryption proof it uses $H(y, d, g^r, a^r)$ when it should also include a, b, b^r. For the equality of discrete logs it uses $H(v, w, a, b)$ (where $a = f^z$ and $b = h^z$ for a random factor z) when it should also include f, h. Note that these inclusions break the attacks outlined above.

5.2 The Correct Plaintext Equivalence Proof

It is easy to fix the weakness in the distributed PET by simply checking that the ciphertext is not $(1, 1)$. However, this check is not present in either the original

Mix & Match paper by Jakobsson & Juels (which does not claim UV), nor JCJ (which assumes UV and points to Mix & Match), nor the Civitas verification code (for which the TR claims UV). We outline the setting of the Civitas PET (a direct implementation of the Mix & Match PET) below.

Let $x_i \leftarrow \mathbb{Z}_q^*$ be the private key shares and $y_i = g^{x_i} \bmod p$ be the public key shares such that

$$Y = \prod_i y_i \bmod p = \prod_i g^{x_i} \bmod p = g^{\sum_i x_i \bmod q} \bmod p$$

is the public key. We assume $\mathsf{EqDlogs}(d, e, x, y)$ is an existentially sound[7] noninteractive zero knowledge proof that $\mathsf{dlog}_d x = \mathsf{dlog}_e y$. We stress that EqDlogs as presented in [8] is **not** existentially sound due to the issue in its implementation of the Fiat-Shamir transform. However, [4,5] give an argument that a correctly implemented transform including all inputs to the protocol in the hash function results in an existentially sound version.

We are given as public input ciphertexts

$$c_j = (a_j, b_j) = (g^{r_j} \bmod p, g^{m_j} Y^{r_j} \bmod p) \text{ for } j \in \{1, 2\}$$

The goal is to prove in a universally-verifiable manner whether $m_1 = m_2 \bmod q$.

Let $(d, e) = (a_1/a_2, b_1/b_2)$. Each teller TT_i is supposed to randomly choose $z_i \in \mathbb{Z}_q^*$, calculate their share $(d_i, e_i) = (d^{z_i}, e^{z_i})$, and publish a commitment $\mathsf{Commit}(d_i, e_i)$. [8] A distributed decryption is then performed on c', where each TT_i publishes their share $a'_i = a'^{x_i} \bmod p$ and a proof $\mathsf{EqDlogs}(g, a', y_i, a'_i)$, guaranteeing they faithfully calculated a'_i from a private key share x_i.

Under the assumption that the Fiat-Shamir weakness is addressed (see [4] for details on this), we present a proof that the PET in Civitas is universally verifiable when a check for trivial ciphertexts are added, following the approach of [5] (sections 19-20). The corrected protocol is shown in Protocol 1, in which $\mathcal{P}$ is the prover (playing the role of all tellers simultaneously since we allow the possibility of collusion) and $\mathcal{V}$ is the public verifier. We will construct the protocol to allow both proofs of both equality and inequality; this will make the argument a little more complex, but much more general.

Protocol 1 PlaintextEquivalenceProof

Setup: public ElGamal parameters $(\mathbb{G}, p, q, g)$; *public key shares* y_i *for* $i \in [n]$ *(where* n *is the number of tellers). Corresponding private key shares* x_i *for* $i \in [n]$.

1. $\mathcal{P}$ *sends ciphertexts* $c_j = (a_j, b_j)$ *for* $j \in \{1, 2\}$ *to* $\mathcal{V}$.
2. $\mathcal{P}, \mathcal{V}$ *calculate* $d = a_1/a_2, e = b_1/b_2$.
3. $\mathcal{P}$ *sends* $(d_i, e_i), \mathsf{EqDlogs}(d, e, d_i, e_i)$ *to* $\mathcal{V}$.
4. $\mathcal{P}, \mathcal{V}$ *calculate* $c' = (a', b') = (\prod_i d_i, \prod_i e_i)$.

[7] Also called *adaptively sound* in other literature.

[8] The commitment is elided from Protocol 1, as it is not relevant in the case that every teller colludes.

5. $\mathcal{P}$ *sends* $a'_i, \mathsf{EqDlogs}(g, a', y_i, a'_i)$ *to* $\mathcal{V}$ *with a bit* IsEq *corresponding to whether the proof is of equality (*$\mathsf{IsEq} = 1$*) or inequality (*$\mathsf{IsEq} = 0$*).*
6. $\mathcal{V}$ *calculates*
$$M = \frac{b'}{\prod_i a'_i} \bmod p$$
7. $\mathcal{V}$ *verifies the* $\mathsf{EqDlogs}$ *ZKPs (as will be made precise below with* Check_{EqD}*)*
8. *If* $\mathsf{IsEq} = 1$, $\mathcal{V}$ *outputs* $\begin{cases} \mathsf{accept} & \text{if } M = 1\text{, the ZKPs pass verification, and } c' \neq (1,1) \\ \mathsf{reject} & \text{otherwise} \end{cases}$
If $\mathsf{IsEq} = 0$, $\mathcal{V}$ *outputs* $\begin{cases} \mathsf{accept} & \text{if } M \neq 1 \text{ and the ZKPs pass verification} \\ \mathsf{reject} & \text{otherwise} \end{cases}$

We show that the combination of ZKP facts and equations checked directly by $\mathcal{V}$ implies that the two plaintexts are equal (mod q).

Lemma 1. *In the setting of Protocol 1,* $m_1 = m_2 \bmod q$ *if and only if* $M = 1$, $c' \neq (1,1)$, $\mathsf{dlog}_d d_i = \mathsf{dlog}_e e_i$, *and* $\mathsf{dlog}_g y_i = \mathsf{dlog}_{a'} a'_i$ *for all* $i \in [n]$.

Proof. With the setup done, an observer may compute (explicitly labelling modulo operations):

$$\begin{aligned} c' = (a', b') = \left(\prod_i d^{z_i}, \prod_i e^{z_i}\right) &= (d^{z \bmod q}, e^{z \bmod q}) \text{ where } z = \sum_i z_i \\ &= \left(\left(\frac{a_1}{a_2}\right)^{z \bmod q}, \left(\frac{b_1}{b_2}\right)^{z \bmod q}\right) \\ &= \left(\left(\frac{g^{r_1}}{g^{r_2}}\right)^{z \bmod q} \bmod p, \left(\frac{g^{m_1}}{g^{m_2}}\frac{y^{r_1}}{y^{r_2}}\right)^{z \bmod q} \bmod p\right) \end{aligned}$$

as well as

$$\begin{aligned} A' = \prod_i a'_i = \prod_i \left(\frac{g^{r_1}}{g^{r_2}}\right)^{z x_i \bmod q} &\pmod p \\ = \left(\frac{g^{r_1}}{g^{r_2}}\right)^{z \sum_i x_i \bmod q} = \left(\frac{Y^{r_1}}{Y^{r_2}}\right)^{z \bmod q} &\pmod p \end{aligned}$$

and thereby recover

$$\begin{aligned} M = \frac{b'}{A'} = \frac{\left(\frac{g^{m_1}}{g^{m_2}}\frac{Y^{r_1}}{Y^{r_2}}\right)^{z \bmod q}}{\left(\frac{Y^{r_1}}{Y^{r_2}}\right)^{z \bmod q}} = \left(\frac{g^{m_1}}{g^{m_2}}\right)^{z \bmod q} &\pmod p \\ = 1 \text{ if and only if } \frac{g^{m_1}}{g^{m_2}} = 1 \bmod p \textbf{ or } z = 0 \bmod q & \end{aligned}$$

In particular, note that if $z = 0 \bmod q$, $c' = (1,1)$ which trivially decrypts to 1 regardless of whether $m_1 = m_2$.

5.3 Security Proof for the Corrected PEP

We follow [5] for formalising the security requirements on the ZKPs, as well as the plaintext equivalence proof system itself. The general idea is to decide what a "proof" should mean mathematically: a statement whose truth can be checked by a public observer. We will argue that any efficient adversary should have only a negligible probability of forging a proof for a false statement.

Definition 1. ***Non-interactive proof system***
Let $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ be an effective relation, where if $(x, y) \in \mathcal{R}$, y is a statement and x is a witness for the statement. A ***non-interactive proof system*** *for $\mathcal{R}$ is a pair of algorithms* $(\mathsf{Gen}, \mathsf{Check})$ *where*

1. Gen *is an efficient probabilistic algorithm that is invoked as $\pi \leftarrow_{\mathcal{R}} \mathsf{Gen}(x, y)$, where $(x, y) \in \mathcal{R}$, and π belongs to some proof space $\mathcal{PS}$;*
2. Check *is an efficient deterministic algorithm that is invoked as $\mathsf{Check}(y, \pi)$, where $y \in \mathcal{Y}$ and $\pi \in \mathcal{PS}$; the output of* Check *is either* accept *or* reject*. If $\mathsf{Check}(y, \pi) = \mathsf{accept}$, we say π is a valid proof for y.*

We will assume that $\mathsf{EqDlogs}$ is corrected to satisfy this requirement. Given public input f, h, v, w we would like to prove that we know x such that $v = f^x$ and $w = h^x$. After choosing a random z, the statement (for prime p, q) is $(p, q, f, h, v, w, a = f^z \bmod p, b = h^z \bmod p)$ with witness x. We will publish $c = H(f, h, v, w, a, b)$ and $r = z + cx \bmod q$. We then have algorithms

- Gen_{EqD} which maps $(x, p, q, f, h, v, w, a, b)$ to $\pi = (c, r)$
- $\mathsf{Check}_{EqC}(p, q, f, h, v, w, a, b, \pi)$ which outputs accept if and only if $f^r = av^c \bmod p$ and $h^r = bw^c \bmod p$ (otherwise, it outputs reject)

From here we will use $\mathsf{EqDlogs}(f, h, v, w)$ less formally to mean "the proof output by $\mathsf{Gen}_{EqD}(x, p, q, f, h, v, w, a, b)$" where context makes x, p, q clear.

$\mathsf{PlaintextEquivalenceProof}$ also satisfies this definition: given ciphertexts $c_1 = (a_1, b_1), c_2 = (a_2, b_2)$, ElGamal parameters $(\mathbb{G}, p, q, g)$, and public key shares y_i, the statement is

$$(\mathbb{G}, p, q, g, c_1, c_2, \{y_i\}_{i \in [n]}, \mathsf{IsEq})$$

with witnesses $\{x_i\}_{i \in [n]}$. The algorithms are:

- Gen_{PEP} which maps $(\{x_i\}_{i \in [n]}, \mathbb{G}, p, q, g, c_1, c_2, \{y_i\}_{i \in [n]}, \mathsf{IsEq})$ to

$$\pi = \{(d_i, e_i, a'_i, \mathsf{EqDlogs}(d, e, d_i, e_i), \mathsf{EqDlogs}(g, a', y_i, a'_i))\}_{i \in [n]}$$

- $\mathsf{Check}_{PEP}(\mathbb{G}, p, q, g, c_1, c_2, \{y_i\}_{i \in [n]}, \mathsf{IsEq}, \pi)$ which is as per Steps 2, 4, 6, 7, 8 of Protocol 1, with additional checks on the ElGamal parameters:
  1. Check that $(\prod_i d_i \bmod p, \prod_i e_i \bmod p) \neq (1, 1)$.
  2. Check that $d = a_1/a_2$ and $e = b_1/b_2$ in $\mathsf{EqDlogs}(d, e, d_i, e_i)$, and that it is a valid proof.
  3. Check that g has order q in $\mathbb{G}$, that $a' = \prod_i d_i$ in $\mathsf{EqDlogs}(g, a', y_i, a'_i)$, and that it is a valid proof.

4. If $\mathsf{IsEq} = 1$, output $\begin{cases} \mathsf{accept} & \text{if the above checks succeed and } \prod_i \frac{e_i}{a_i'} = 1 \bmod p \\ \mathsf{reject} & \text{otherwise} \end{cases}$

If $\mathsf{IsEq} = 0$, output $\begin{cases} \mathsf{accept} & \text{if the above checks succeed and } \prod_i \frac{e_i}{a_i'} \neq 1 \bmod p \\ \mathsf{reject} & \text{otherwise} \end{cases}$

The requirement we will use for universal verifiability is that of *existential soundness*:[9] the adversary who produces our proof should not be able to falsify a proof even if they freely choose all of the parameters, given a verifier who holds only public knowledge [5, Def. 20.2]. If we can achieve this, then we satisfy universal verifiability.

Definition 2. ***Existential soundness***
*Let $\Phi = (\mathsf{Gen}, \mathsf{Check})$ be a non-interactive proof system for $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ with proof space $\mathcal{PS}$. To attack Φ, an adversary $\mathcal{A}$ outputs a statement $y \in \mathcal{Y}$ and a proof $\pi \in \mathcal{PS}$. We say that the adversary wins the game if $\mathsf{Check}(y, \pi) = \mathsf{accept}$ but there is no witness x such that $(x, y) \in \mathcal{R}$. We define $\mathcal{A}$'s advantage with respect to Φ, denoted $\mathsf{niESadv}[\mathcal{A}, \Phi]$, as the probability that $\mathcal{A}$ wins the game. Φ is **existentially sound** if for all efficient adversaries $\mathcal{A}$, $\mathsf{niESadv}[\mathcal{A}, \Phi]$ is negligible.*

Theorem 1. *Assume* EqDlogs *is an existentially sound non-interactive proof system. For all efficient adversaries $\mathcal{A}$ with security parameter k, suppose that for some negligible $\varepsilon(k)$, we have: $\mathsf{niESadv}[\mathcal{A}, \mathsf{EqDlogs}] < \varepsilon(k)$.*
Then PlaintextEquivalenceProof *is existentially sound.*

Proof. Consider an efficient adversary $\mathcal{A}$ with security parameter k for whom

$$\mathsf{niESadv}[\mathcal{A}, \mathsf{PlaintextEquivalenceProof}] = P(k)$$

Suppose $\mathcal{A}$ outputs a proof π for the false statement $(\mathbb{G}, p, q, g, c_1, c_2, \mathsf{IsEq})$.

Case 1: $\mathsf{IsEq} = 1$. Then if $\mathsf{Check}_{PEP}(\mathbb{G}, p, q, g, c_1, c_2, \mathsf{IsEq}, \pi)$ outputs accept, by definition $m_1 \neq m_2 \bmod q$ and $\frac{g^{m_1}}{g^{m_2}} = 1 \bmod p$.

Case 2: $\mathsf{IsEq} = 0$. Then if $\mathsf{Check}_{PEP}(\mathbb{G}, p, q, g, c_1, c_2, \mathsf{IsEq}, \pi)$ outputs accept, by definition $m_1 = m_2 \bmod q$ and $\frac{g^{m_1}}{g^{m_2}} \neq 1 \bmod p$.

In either case $c' \neq (1, 1)$, and $\mathsf{EqDlogs}(d, e, d_i, e_i)$, $\mathsf{EqDlogs}(g, a', y_i, a_i')$ are valid proofs. We checked that $\langle g \rangle = q$, so Lemma 1 implies (at least) one of the following two statements is false for some $i \in [n]$:

(i) $\mathsf{dlog}_d d_i = \mathsf{dlog}_e e_i$

(ii) $\mathsf{dlog}_g y_i = \mathsf{dlog}_{a'} a_i'$

So $\mathcal{A}$ must have cheated in at least one instance of EqDlogs. We can now construct an adversary $\mathcal{B}$ (with security parameter k) to defeat EqDlogs. $\mathcal{B}$ runs $\mathcal{A}$ to obtain two proofs $\mathsf{EqDlogs}(d, e, d_i, e_i)$, $\mathsf{EqDlogs}(g, a', y_i, a_i')$. At least one is valid; we do not know which one, so $\mathcal{B}$ tosses a coin. On heads, $\mathcal{B}$ chooses $\mathsf{EqDlogs}(d, e, d_i, e_i)$, and on tails it chooses $\mathsf{EqDlogs}(g, a', y_i, a_i')$. $\mathcal{B}$ then outputs

[9] Also called *adaptive soundness*.

the chosen statement and proof. In this way, $\mathcal{B}$ wins with probability at least $P(k)/2$.

Therefore given that an efficient adversary attacking EqDlogs has advantage at most ε, an efficient adversary attacking PlaintextEquivalenceProof has advantage at most 2ε. □

6 Discussion and Conclusion

We have found a subtle misalignment of assumptions in a very influential paper (624 citations): they require a PET to have universal verifiability, but reference only PETs that do not have this property. As with many cryptographic protocols, this small unmet assumption completely undermines one of the protocol's primary security goals, in this case universal verifiability. This oversight affects a large number of follow-on protocols in exactly the same way, hence undermining their universal verifiability too. Furthermore, the issue is not unique to the electronic voting domain, but also appears in verifiable auction schemes. To our knowledge, other work has not noticed this important subtlety.

Although our most detailed analysis describes errors in the Civitas implementation, this is primarily because Civitas is the only one of the affected papers to provide a detailed technical report and code, rather than because the other projects understood the subtle assumption better. Comments in other papers suggest that, if the authors had implemented their designs, they would likely have made the same confusion between PETs with PEPs. This is further demonstrated by their reference to the PET of [17], which lacks the crucial property. Fortunately, the mistake is easily rectifiable — but if the Fiat-Shamir transform is used improperly by a protocol, a related attack opens up to catastrophically undermine any plaintext equivalence proofs used.

The discovery has significant parallels with Bernhard `et al.`'s discovery of errors in the implementation of the Fiat-Shamir heuristic that undermined soundness in the context of Helios [4], and again in the SwissPost-Scytl sVote scheme [14,13]. This suggests that errors of this nature are not unique to the schemes discussed in this paper, but may be indicative of a more systemic problem in the misalignment of assumptions between separate work.

Acknowledgements The research carried out by O. Pereira was partially supported by the F.N.R.S. PDR SeVoTe.

References

1. Abe, M.: Universally verifiable mix-net with verification work independent of the number of mix-servers. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 437–447. Springer (1998)
2. Abe, M., Suzuki, K.: M+ 1-st price auction using homomorphic encryption. In: International Workshop on Public Key Cryptography. pp. 115–124. Springer (2002)
3. Benaloh, J.: Verifiable secret-ballot elections (1988)

4. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 626–643. Springer (2012)
5. Boneh, D., Shoup, V.: A graduate course in applied cryptography. Draft 0.5 (2020)
6. Bradford, P.G., Park, S., Rothkopf, M.H., Park, H.: Protocol completion incentive problems in cryptographic vickrey auctions. Electronic Commerce Research **8**(1-2), 57–77 (2008)
7. Clark, J., Hengartner, U.: Selections: Internet voting with over-the-shoulder coercion-resistance. In: International Conference on Financial Cryptography and Data Security. pp. 47–61. Springer (2011)
8. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. In: 2008 IEEE Symposium on Security and Privacy (sp 2008). pp. 354–368. IEEE (2008)
9. Cortier, V., Galindo, D., Küsters, R., Mueller, J., Truderung, T.: Sok: Verifiability notions for e-voting protocols. In: 2016 IEEE Symposium on Security and Privacy (SP). pp. 779–798. IEEE (2016)
10. Delfs, H., Knebl, H.: Introduction to cryptography, vol. 3. Springer (2015)
11. Essex, A., Clark, J., Hengartner, U.: Cobra: Toward concurrent ballot authorization for internet voting. EVT/WOTE **12** (2012)
12. Grewal, G.S., Ryan, M.D., Bursuc, S., Ryan, P.Y.: Caveat coercitor: Coercion-evidence in electronic voting. In: 2013 IEEE Symposium on Security and Privacy. pp. 367–381. IEEE (2013)
13. Haenni, R.: Swiss Post Public Intrusion Test: Undetectable attack against vote integrity and secrecy. `https://e-voting.bfh.ch/app/download/7833162361/PIT2.pdf?t=1552395691` (Mar 2019)
14. Haines, T., Lewis, S.J., Pereira, O., Teague, V.: How not to prove your election outcome. In: 2020 IEEE Symposium on Security and Privacy (SP). pp. 784–800 (2019)
15. Heather, J., Ryan, P.Y., Teague, V.: Pretty good democracy for more expressive voting schemes. In: European Symposium on Research in Computer Security. pp. 405–423. Springer (2010)
16. Hevia, A., Kiwi, M.: Electronic jury voting protocols. Theoretical Computer Science **321**(1), 73–94 (2004)
17. Jakobsson, M., Juels, A.: Mix and match: Secure function evaluation via ciphertexts. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 162–177. Springer (2000)
18. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Towards Trustworthy Elections, pp. 37–63. Springer (2010)
19. MacKenzie, P., Shrimpton, T., Jakobsson, M.: Threshold password-authenticated key exchange. In: Annual International Cryptology Conference. pp. 385–400. Springer (2002)
20. Quaglia, E.A., Smyth, B.: Secret, verifiable auctions from elections. Theoretical Computer Science **730**, 44–92 (2018)
21. Ryan, P.Y., Teague, V.: Pretty good democracy. In: International Workshop on Security Protocols. pp. 111–130. Springer (2009)
22. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 393–403. Springer (1995)

23. Spycher, O., Koenig, R., Haenni, R., Schläpfer, M.: A new approach towards coercion-resistant remote e-voting in linear time. In: International Conference on Financial Cryptography and Data Security. pp. 182–189. Springer (2011)