Remote Electronic Voting with Revocable Anonymity

Technical Report *CSR-09-06*
University of Birmingham

Matt Smart and Eike Ritter

July 15, 2009

Abstract

We present a new remote, coercion-free electronic voting protocol which satisfies a number of properties previously considered contradictory. We introduce (and justify) the idea of revocable anonymity in electronic voting, on the grounds of it being a legal requirement in the United Kingdom, and show a method of proving the validity of a ballot to a verifier in zero knowledge, by extension of known two-candidate proofs.

1 Introduction

It is undoubtedly a challenge to design electronic voting protocols that satisfy what is an ever-growing list of requirements that are difficult to achieve simultaneously. Many governments have begun to adopt electronic voting with a view to improving voter turnout, with hardly any success. One of the driving factors for electronic voting is *remote voting*—the requirement that a citizen can vote from any location. This is unfortunately very difficult to achieve whilst minimising the potential for voter coercion. Further, how is it possible to satisfy voter privacy (anonymity) whilst also allowing the voter to verify that her vote has been counted? Can we assure ballot correctness for any number of candidates?

An especially important property in electronic voting is *anonymity* (privacy)—the notion that no voter should be linkable to their ballot. In this work, we introduce *revocable anonymity* to electronic voting — the notion that it should be possible to link one's identity to one's ballot, but *only* with the agreement of a Judge and a quorum of mutually distrusting parties.

In the UK, it is a legal requirement that it should be possible for the election authorities to link a ballot to its voter [2, p. 106]. To our knowledge, no work has previously considered this notion (equivalent to revocable anonymity in electronic voting), but it seems important to do so. One can envisage a situation in which, since voters are entirely anonymous, an attacker can vote on behalf of people who should be unable to vote, but, for whatever reason, are still on the electoral roll. In a 2005 postal vote scandal in Birmingham, UK, "possibly well over 2,000 of the votes cast" were fraudulent and illegitimate for one ward alone [30]. We feel that permitting anonymity revocation in extreme circumstances is fundamental to reducing election fraud such as this.

1.1 Our Contribution

In this work, we introduce a remote electronic voting protocol which satisfies several properties considered important in electronic voting.

- First, one of the larger problems we address is *revocable anonymity*: achieving this whilst satisfying several other properties is non-trivial. For example, some protocols give a voter's identity in plaintext with an encrypted ballot on a tallying board. Clearly, this means that an attacker can determine *whether* Alice has voted. If we want complete coercion resistance, this fact cannot be made clear. Further, keeping some record of a voter's identity in any way risks her privacy (and hence coercion-resistance) being compromised. In this work, we introduce revocable anonymity in the context of electronic voting, whilst re-

taining coercion-resistance. Our protocol permits legitimate voters to maintain their privacy entirely, whilst also allowing *individual* and *universal verifiability*.

- Second, through a novel use of designated verifier signatures and designated verifer proofs of re-encryption, we achieve *remote* voting with coercion-resistance and receipt-freeness, whilst maintaining the legitimate voter's privacy, and avoiding the need for anonymous or untappable channels, or tamper-resistant hardware. We refute the claim of Benaloh and Tuinstra [1] that a voter must be physically isolated at the point of voting (indeed, it is shown by Canetti and Gennaro [4] that coercion-freeness can be achieved via deniable encryption), but merely state that the voter must be able to vote *only once* without observation. What is necessary, it seems, is that any observer cannot be *convinced* that Alice has voted correctly, irrespective of whether or not he is watching Alice when she votes.

  Whilst a Tamper-Resistant Randomiser, or other trusted smart-card device, would partially solve the problem of coercion-resistance, it is unrealistic to expect every voter to use a smart card from any location. The "lack of deployed base of smart card readers", and potential need for voter financial investment [25] makes this approach undesirable. Similarly, whilst protocols using scanned paper ballots are very impressive, sociologically they do little to improve the current situation—one motive for electronic voting is to increase turnout via remote voting. If another motive is to save money, we should consider: "optical ballots need the same number of polling stations and they will be more expensive" [11].

- Third, we provide an extension of previous schemes to prove ballot validity for two-candidate elections [9], to multiple candidate elections, and give a proof of our extension.

The protocol we present achieves the above properties, as well as the standard electronic voting properties (completeness, uniqueness, coercion-resistance, fairness, and legitimate-voter privacy).

1.2 Related Work

In our experience, there is no work which provides revocable anonymity in electronic voting, and little work which provides large-scale, coercion-resistant, remote electronic voting ([7] is a good example, but does not seem scalable). We here discuss how previous authors have satisfied some of the important properties of e-voting without revocable anonymity, and our strategy to achieve them.

Many electronic voting protocols [29, 14, 20, 7, 12, 33] rely on anonymous channels, or anonymous and untappable channels [23], to satisfy some security properties. When considering voting over the Internet (an inherently insecure medium), one needs to think about how an anonymous channel could be implemented in the first place.

Attempts have been made to achieve anonymous channels with mix networks [5, 27, 21, 29, 3, 18, 7, 20], which provide effective anonymity, but can often be slow, inefficient, complex and subject to single points of failure (in the case of decryption mixes). Indeed, it has been argued [32] that for an Internet-based voting protocol, there is simply no way to reliably implement an anonymous communication channel over the Internet. Volkamer and Krimmer [32] suggest that IP address tracking or trojan horse viruses alone mean that any attempt at an anonymous channel would always suffer from some weakness.

Thankfully, in our work, we do not need to use anonymous *or* untappable channels (which are, when from voter to talliers, a very strong assumption), relying instead on various designated verifier proofs to satisfy voter verifiability whilst maintaining coercion-resistance and privacy.

In our work, we follow the scheme of many previous protocols using homomorphic encryption to ensure universal verifiability and unlinkability of ballots [1, 8, 9, 16, 33, 20, 22, 7, 20], which naturally lends itself to threshold cryptography, affording us a greater level of assurance against corrupted talliers. These protocols, along with some of those already mentioned, require, for remote voting, that the voter is not observed at the "very moment of voting" [22]. Indeed, Benaloh and Tuinstra state that "physical separation of the voter from possible coercive agents is fundamental to any uncoercible election protocol" [1, p. 550].

Lee *et al.* [22, 21], amongst others, suggest

the use of a *tamper-resistant randomiser*—smart card—and non-voter-observation at the point of voting, to guarantee coercion-resistance. An alternative is to have every voter use a public voting booth which either uses a smart card, as above, or a paper ballot which is optically processed by machine [5, 27, 26, 6, 24].

We note that any protocol providing a list of voters' identities with encrypted ballots could provide revocable anonymity, given the collusion of all parties needed to perform decryption. However, such a list clearly violates full coercion-resistance, as the fact that a voter has voted successfully can be determined by anyone. Juels *et al.* [20] and implementations thereof [7] involve talliers only keeping a list of votes at the end of the election (discarding the previous stage's encrypted credentials), thus severing the direct link between voter and vote. Revocation of anonymity would require a highly inefficient Plaintext Equivalence Test between the credential supplied with a vote and every credential on the voter list, followed by a collusion with the registrar. Lee *et al.* [21] would allow for revocation, *but* subject to collusion of the administrator, the entire mix and n talliers. The nature of usage of the bulletin board in the protocol also suggests that full coercion-resistance is not possible, as the fact that Alice has voted is plainly visible . Prêt à Voter [27] and similar schemes do not offer revocation at all, since Alice's choice of ballot paper is random, and as any identifying information is destroyed (by Alice), she cannot be linked to her ballot. In any case, no other protocol discusses revocable anonymity at all, to our knowledge.

In our work, we provide a protocol assuring coercion-resistance and full remote voting (subject to computer access), through use of designated verifier signatures [28, 19] and designated verifier proofs of re-encryption [17], using a new version of anonymous coercion credentials [20].

For a protocol to be fully coercion-resistant, a coercer should not be able to determine that a voter has *even voted successfully*. For a protocol to afford remote voting, it should allow a voter anywhere with a connection to the Internet to vote, without the need for a sealed voting booth (but merely with the requirement that the voter can vote *once* unobserved). To our knowledge, there is no scalable protocol available which satisfies these criteria without further assumptions: either anonymous channels/voting booths, or several trust assumptions.

1.2.1 Protocol Schema

We present a two-phase protocol, where voters do not need to synchronise between phases they are actively involved in. Our reasoning for splitting into two phases is to preserve the anonymity of the legitimate voter, henceforth referred to as Alice. In the first phase, voters register in person to receive eligibility tokens with designated verifier signatures, and form ElGamal encryptions of ballots, submitting them to a bulletin board. A member of a semi-trusted tallier group re-encrypts Alice's vote.

In the second phase, Alice receives a designated verifier proof of re-encryption (along with some other fake proofs), and her re-encrypted vote is posted to another bulletin board with an encrypted version of her identity. Alice can then check her vote has been included, or contact a Judge otherwise.

Once all votes are posted to the second bulletin board, a tally is calculated and announced. A simple schematic diagram of the protocol is given in Figure 1.

1.3 Structure

In §2, we define a number of preliminaries, including the terminology used, and a number of primitives which we make use of. In §3, we give the participants, trust model and threat model for our work. We present some requirements in §4, and our protocol in §5. Finally, we sketch proofs of security in §6, and then conclude.

2 Preliminaries

In this paper, we assume the availability of the following cryptographic primitives. Note that we are working in the formal model, not in provable security. Therefore we make the assumption that the cryptography in the primitives below is perfect (excepting Lemma 1).

2.1 Threshold ElGamal Encryption Scheme

We use a standard ElGamal encryption scheme [10] under a q-order multiplicative subgroup $G_q = \langle g \rangle$

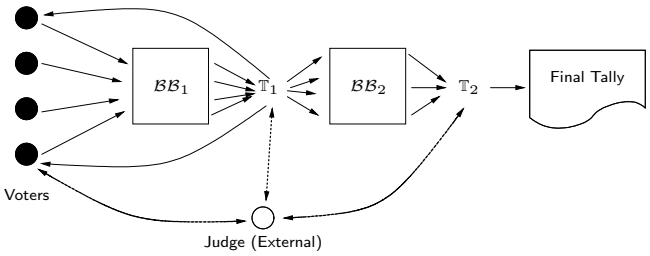

Figure 1: A schematic for our protocol.

of $\mathbb{Z}_p^*$, generated by an element $g \in \mathbb{Z}_p^*$, where p and q are suitably large primes, and $q|(p-1)$. All agents a in the protocol have a private key s_a of which only they have knowledge. Each agent has a corresponding public key $h_a = g^{s_a}$ where g is a known generator of the subgroup. Public keys are common knowledge to all users.

Given a message $m \in \mathbb{Z}_p$, its encryption is denoted as $(x, y) = (g^\alpha, h_a^\alpha m)$, where α is a random value in $\mathbb{Z}_q$. To decrypt (x, y), the intended receiver calculates $m = y/x^{s_a}$ using his private key. For our work, we consider only a 1-out-of-L (plurality) voting scheme (where L is the number of candidates that can be chosen from). We use ElGamal re-encryption and Designated Verifier Proofs of Re-encryption, which are discussed later.

In our protocol, we use a (t, n)-threshold decryption scheme (as shown in [9]) to increase voter security. In a standard ElGamal threshold decryption scheme, a secret key s is shared between n talliers $T_i : 1 \leq i \leq n$. Each tallier possesses a share s_i of s, and publishes $h_i = g^{s_i}$ as a commitment to his share. Each s_i is chosen such that the secret key can be reconstructed from a subset Λ of t shares of s, using Lagrange coefficients $\lambda_{i,\Lambda}$ [9]:

$$s = \sum_{i \in \Lambda} s_i \lambda_{i,\Lambda}, \qquad \lambda_{i,\Lambda} = \prod_{l \in \Lambda \setminus \{i\}} \frac{l}{l-i}$$

Hence a quorum of size t has to agree and cooperate in order to decrypt any value. The public key $h = g^s$ is broadcast to all agents.

To decrypt a message (x, y) without giving a single agent the whole private key s, each tallier in the quorum does the following:

1. Broadcast $w_i = x^{s_i}$, and proves (in zero-knowledge) the equality of the discrete logs
$$\log_g h_i = \log_x w_i$$
2. Where Λ is a subset of talliers who passed the proof (i.e., the quorum needed), the plaintext is decrypted as
$$m = \frac{y}{\prod_{i \in \Lambda} w_i^{\lambda_{i,\Lambda}}}$$

For more information on threshold ElGamal decryption, the reader is directed to Cramer *et al.* [9].

2.2 Strong Designated Verifier Signature Scheme

We adopt the designated verifier signature scheme of Saeednia *et al.* [28] due to its efficient nature, but others would be acceptable. We use designated verifier signatures to enable a prover (Bob, or any one of the first-round tellers in our case) to prove a statement to a verifier (Alice) by proving the validity of a signature: However, Alice is unable to prove the signature's validity to *anyone* else, on the grounds that she could have produced it herself [28, p. 43].

The parameters for the scheme are the same as those for ElGamal encryption: a large prime p; q such that $q|(p-1)$; a generator $g \in \mathbb{Z}_p^*$ of order q, and a one-way hash function hash that outputs values in $\mathbb{Z}_q$. Every user a has a secret key s_a and a corresponding public key $h_a = g^{s_a} \mod p$.

In order to generate a designated verifier signature on a message m for Alice, Bob selects $k \in_R \mathbb{Z}_q$, $e \in_R \mathbb{Z}_q^*$ and calculates

$$\begin{aligned} c &= h_{\mathsf{Alice}}^k \\ r &= \mathsf{hash}(m, c) \\ v &= ke^{-1} - rs_{\mathsf{Bob}} \mod q \end{aligned}$$

The triple (r, v, e) is now the signature of m [28]. We denote this designated verifier signature as $\mathsf{DVSign}_i(m)$, where the signature is generated by i.

Alice is able to verify the signature's correctness by checking the equation

$$\mathsf{hash}(m, (g^v h_{\mathsf{Bob}}^r)^{es_{\mathsf{Alice}}} \mod p) \stackrel{?}{=} r$$

Clearly, no-one other than Alice can verify the signature (it uses her private key). However, Alice is able to select a random $v' \in \mathbb{Z}_q$, $r' \in \mathbb{Z}_q^*$ and simulate the entire transcript herself (the reader is directed to the authors' paper for proof of this). As a result, Alice could have generated the signature herself, and no third party will be convinced of the validity of any signature from Bob that Alice claims to be valid [28, p. 45].

Even if Alice reveals her secret key to a third party, she cannot convince that party of the validity of the signature, since she herself could have simulated the signature transcription. Further, no party is able to reveal the contents of the signature without Alice's secret key.

2.3 Proof of Equality of Discrete Logarithms

In order to prevent an attack in our voting scheme, we require that the voter demonstrates to a verifier that her vote is of the correct form (without revealing what the vote is).

As we discuss later, a voter's vote is of the form $(x, y) = (g^\alpha, h_{\mathbb{T}_2}^\alpha g^{M^{i-1}})$ where $\alpha \in_R \mathbb{Z}_q$, M is the maximum number of voters and i represents the position in the list of candidates of the voter's chosen candidate. Alice needs to prove, in zero knowledge, that she is sending to the bulletin board some value for y where the exponent of g is in $\{M^0, \ldots, M^{L-1}\}$ where L is the number of candidates. If we did not have such a proof, any voter could spoil the election by adding spurious coefficients to the exponent, thereby voting several times.

We hence show that the ballot (x, y) is of valid form, as specified in the parameters of the election:

$$(x, y) = (g^\alpha, h_{\mathbb{T}_2}^\alpha m) : \ m \in \{g^{M^0}, \ldots, g^{M^{L-1}}\}$$

Cramer *et al.* [9] demonstrate this via a witness indistinguishable proof of knowledge of the relation:

$$\log_g x = \log_{h_{\mathbb{T}_2}}(y/m_0) \vee \log_g x = \log_{h_{\mathbb{T}_2}}(y/m_1)$$

for an election with only two candidates [9].

This *proof of validity* is described for an interactive, two-candidate scenario in [8, 9, 15]. Using the Fiat-Shamir Heuristic [13], the authors convert the interactive protocol into a non-interactive one [9]. However, the scheme provided in these papers is for votes with only two possible outcomes.

The proof of validity for a two-candidate scenario, where a vote $(x, y) = (g^\alpha, h^\alpha m) : m \in \{m_0, m_1\}$ and the prover knows the value of m, holds, proving that (x, y) is of the proscribed form, providing Alice submits a vote $v \in \{m_0, m_1\}$, as it provides a witness-indistinguishable proof for the relation given above. The prover knows a witness for either the left or right part, according to the choice of m.

We can extend the two-candidate scenario to L candidates, providing a proof for the relation given by

$$\log_g x = \log_{h_{\mathbb{T}_2}}(y/g^{M^0}) \vee \ldots$$
$$\ldots \vee \log_g x = \log_{h_{\mathbb{T}_2}}(y/g^{M^{L-1}})$$

We adapt the non-interactive proof of ballot validity to a scheme for a *multi-candidate* election. In Figure 2, we give a generalised adaptation (**G-PEQDL**) of the above proof of equality of discrete logarithms scheme where Alice votes for candidate k $(1 \leq k \leq L)$ with $(x, y) = (g^\alpha, h^\alpha g^{M^{k-1}})$. This is the only place where we extend one of the primitives we use, and as such we provide a proof for our extension:

Lemma 1. (Security of Generalised PEQDLs) The proof of validity for an L-vote scenario, holds by extension of the above.

Proof. Referring to the protocol (Figure 2), Alice generates the values α, ω and r_i, d_i (for $i = 1, \ldots, k-1, k+1, \ldots, L$) at random, where she has voted for the k^{th} candidate out of L.

She uses $x = g^\alpha, y = h_{\mathbb{T}_2}^\alpha g^{M^{k-1}}$ as with her actual vote (note that the value of α does not change), and proceeds to generate $a_k = g^\omega, b_k = h_{\mathbb{T}_2}^\omega$. All other a_i for $1 \leq i \leq L$ are calculated as $g^{r_i} x^{d_i}$, and all other $b_i \leftarrow h_{\mathbb{T}_2}^{r_i} (\frac{y}{g^{M^{i-1}}})^{d_i}$.

Finally Alice generates

$$c \leftarrow \mathsf{hash}(h_{\mathsf{Alice}}, x, y, a_1, b_1, \ldots, a_L, b_L)$$

Using this value she can generate d_k by subtracting all other d_i values from c, and finally $r_k \leftarrow \omega - \alpha d_k$. Alice sends all a, b, d and r values to the verifier.

The verifier now generates c in the same way (note that this value is not sent to him), and trivially this c should equal the sum of all d_i, as Alice manipulated d_k to make this the case. The verifier now checks each value of a_i, b_i:

1. For all $a_{i\neq k}, b_{i\neq k}$: a_i trivially equals $g^{r_i}x^{d_k}$ and b_i trivially equals $h_{\mathbb{T}_2}^{r_i}(\frac{y}{g^{M^{i-1}}})^{d_i}$, as these values were calculated in the same manner by Alice
2. For a_k:
   (a) The value Alice calculates is $a_k = g^{\omega}$
   (b) The verifier makes the comparison:

$$\begin{aligned} a_k &\stackrel{?}{=} g^{r_k}x^{d_k} \\ &\stackrel{?}{=} g^{\omega-\alpha d_k}g^{\alpha d_k} \\ &\stackrel{?}{=} g^{\omega-\alpha d_k+\alpha d_k} \\ &\stackrel{?}{=} g^{\omega} \end{aligned}$$

   which succeeds if Alice is honest.
3. For b_k:
   (a) The value Alice calculates is $b_k = h_{\mathbb{T}_2}^{\omega}$
   (b) The verifier makes the comparison:

$$\begin{aligned} b_k &\stackrel{?}{=} h_{\mathbb{T}_2}^{r_k}(\frac{y}{g^{M^{k-1}}})^{d_k} \\ &\stackrel{?}{=} h_{\mathbb{T}_2}^{\omega-\alpha d_k}(\frac{h_{\mathsf{Alice}}^{\alpha}g^{M^{k-1}}}{g^{M^{k-1}}})^{d_k} \\ &\stackrel{?}{=} h_{\mathbb{T}_2}^{\omega-\alpha d_k}h_{\mathbb{T}_2}^{\alpha d_k} \\ &\stackrel{?}{=} h_{\mathbb{T}_2}^{\omega-\alpha d_k+\alpha d_k} \\ &\stackrel{?}{=} h_{\mathbb{T}_2}^{\omega} \end{aligned}$$

   which succeeds if Alice is honest.

If Alice is honest and submits a vote $(x, y) = (g^{\alpha}, h_{\mathbb{T}_2}^{\alpha}m)$ where $m \in \{g^{M^0}, \ldots, g^{M^{L-1}}\}$, she will provide a witness-indistinguishable proof of the relation given by

$$\log_g x = \log_{h_{\mathbb{T}_2}}(y/g^{M^0}) \vee \log_g x = \log_{h_{\mathbb{T}_2}}(y/g^{M^1}) \vee \ldots \vee \log_g x = \log_{h_{\mathbb{T}_2}}(y/g^{M^{L-1}})$$

The only way that Alice could be dishonest to her advantage is to make y equal to (for example) $h^{\alpha}g^{30M^{k-1}}$, thereby voting 30 times. Similarly she could attempt to vote for more than one candidate ($y = h^{\alpha}g^{M^{k-1}+M^{k-2}}$). This attack cannot work. In the verification phase, b_k could not be equal to $h_{\mathbb{T}_2}^{r_k}(\frac{y}{g^{M^{k-1}}})^{d_k}$ if the value of the exponent of g in y is not in $\{M^0, M^1, \ldots, M^{L-1}\}$. Thus, one or more calculations for b_i would fail, and the proof would be rejected.

The G-PEQDL proof is honest-verifier zero knowledge as with the two-party scenario presented in [9, p. 487], to whose proof ours is analagous. □

2.4 Designated Verifier Re-encryption Proofs

The properties of the ElGamal encryption scheme allow re-encryption (randomisation) of ciphertexts. Given a ciphertext (x, y), another agent is able to generate $(x_f, y_f) = (xg^{\beta}, yh^{\beta}) : \beta \in_R \mathbb{Z}_q^*$. It is known that given two ElGamal ciphertexts, without knowledge of the private key or the re-encryption factor β, determining any re-encryption relationship between the ciphertexts is hard under the DDH assumption.

In our protocol, we use an ElGamal re-encryption to preserve the voter's anonymity. However, the voter needs to have some conviction that her vote has been counted (individual verifiability). We achieve this via a *Designated Verifier Re-encryption Proof* (DVRP): such a proof convinces Alice that a given re-encrypted ciphertext is equivalent to that she generated, whilst not convincing any third party. We adopt the scheme used by Lee *et al.*[21, 22, 17]: if $(x, y) = (g^{\alpha}h^{\alpha}m)$ is a ciphertext of a message m as described above, $(x_f, y_f) = (xg^{\beta}, yh^{\beta})$ is a re-encryption of (x, y). The prover, P (the agent that does the re-encryption) needs to demonstrate to Alice that (x_f, y_f) is equivalent to (x, y) in such a manner that m is not revealed, and this proof is not transferable. P therefore does the following:

1. Selects $d, j, u \in_R \mathbb{Z}_q$
2. Calculates $(a, b) = (g^d, h_P^d)$ and $\sigma = g^j h_{\mathsf{Alice}}^u$, where h_{Alice} is Alice's public key as before.
3. Calculates $c = \mathsf{hash}(a, b, \sigma, x_f, y_f)$ and $z = d - \beta(c + j)$
4. Sends (c, j, u, z) to Alice

Alice then merely needs to verify that

$$c = \mathsf{hash}(g^z(\frac{x_f}{x})^{c+j}, h^z(\frac{y_f}{y})^{c+j}, g^j h_{\mathsf{Alice}}^u, x_f, y_f)$$

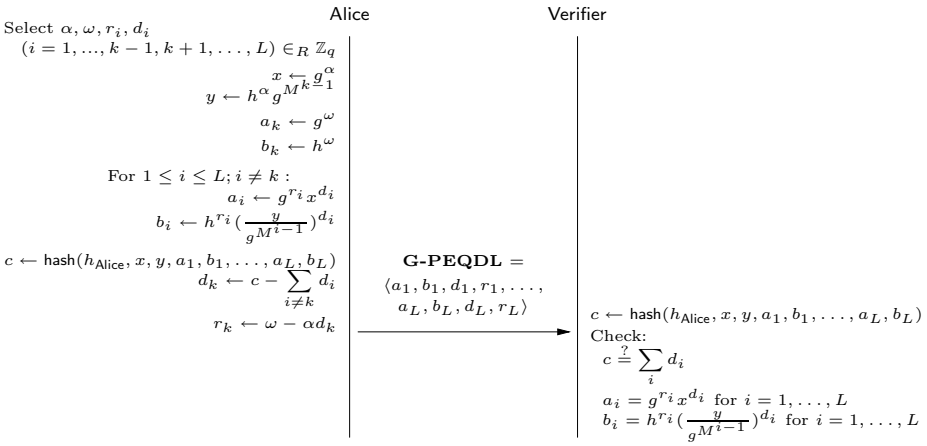

Figure 2: Our generalised non-interactive proof of ballot validity for a vote for candidate k

As detailed in [22, 17], Alice is able to generate this proof for herself as she knows her own private key (σ is a trapdoor commitment for j and u), meaning that no-one (other than Alice) can be convinced by it. Indeed, Alice can insert 'fake' proofs into any communication meant for her, to fool observers.

We note that in order to assure full *remote* coercion-resistance for Alice, she needs to protect her private key in some way, so that the DVRP cannot be verified by a coercer. Possible solutions to this are detailed in Appendix A.

3 Protocol Model

We now detail the model for our protocol. A simple schematic diagram was given in Figure 1.

3.1 Participants

Our protocol is modelled with 5 kinds of participant. A participant (agent) is an interactive polynomial-time random computation. All agents are able to communicate via a network, which is not secure or anonymous in any way.

The participants are as follows:

- **Voters**. The protocol allows M voters $v_i \in \{v_0, v_1, ..., v_{M-1}\}$ to vote. Alice is an honest voter who wishes to vote anonymously. She is able to vote many times, but *once* unobserved. Eligible voters' public keys are publicly known.

- **First Round Bulletin Board/First Round Talliers**. Our protocol uses two separate *bulletin boards*. A standard bulletin board is a public broadcast channel with memory. The first bulletin board we use is writable only by *voters*. All voters send an encrypted vote and signed proof of validity to this board, which we denote as $\mathcal{BB}_1$.

  The *first-round talliers* $\mathbb{T}_1$ are a *semi-trusted* group of agents[1], each possessing an ElGamal secret key $s_{\mathbb{T}_1}$ in its entirety, which any one of them can use to remove the first layer of encryption on Alice's vote[2]. We assume that each instance would be busy enough, and that votes would be batched before sending to $\mathcal{BB}_2$, so that timing attacks would be ineffective. Our justification for having multiple members of $\mathbb{T}_1$ is to prevent a bottleneck of computational power, but if this problem were ignored, we could equally substitute the group for a single entity.

  The first round talliers are responsible for ensuring that Alice's vote is valid according to the set of valid possible votes, not coerced, and not a double-vote. They are unable to see Alice's actual vote token. $\mathbb{T}_1$ encrypt Alice's identity,

[1] We discuss our need for trusting $\mathbb{T}_1$ in §3.1.1.

[2] The size of $\mathbb{T}_1$ would need to be determined empirically depending on the size of the electorate. Since each member of the group has a copy of the same key, the size only affects how much of a bottleneck (in terms of computational power) $\mathbb{T}_1$ is.

should revocation be required.

- **Second Round Bulletin Board/Second Round Talliers**. The second bulletin board $\mathcal{BB}_2$ is viewable by all users of the protocol, and writable only by $\mathbb{T}_1$. It lists only the re-encrypted (valid) votes in a random permutation. The votes themselves, (x, y), are encrypted with the public key of the second round talliers.

  The *second-round talliers* are a group of agents (disjoint from $\mathbb{T}_1$) who decrypt the ballots listed on the second round bulletin board using threshold ElGamal with a shared key $s_{\mathbb{T}_2}$. The second round talliers will also publish the final tally.

- **Anonymity Teller Group**. As well as each being separate groups $\mathbb{T}_1, \mathbb{T}_2$, the tallier groups form part of a larger group which deals only with the voter's anonymity. This group contains an equal number of members of $\mathbb{T}_1$ and $\mathbb{T}_2$ and is simply denoted $\mathbb{T}$. As such, it has a public key $g^{s_{\mathbb{T}}}$ and associated private key $s_{\mathbb{T}}$, where the private key is distributed amongst all members as before. In this case, to decrypt, a quorum of a size t_{id}, greater than the size of either $\mathbb{T}_1$ or $\mathbb{T}_2$, will need to collude to decrypt. Note that this decryption is only ever needed when a voter's identity needs to be traced, as our protocol is optimistic. Further, a voter's anonymity cannot be revoked without the agreement of the quorum *and* the Judge.

- **Judge**. The Judge is an entity of the protocol that is rarely used. She has two purposes:

  1. If Alice cannot find her re-encrypted vote on the bulletin board, she asks the Judge for verification.
  2. The Judge also authorises anonymity revocation (having been presented with appropriate evidence of the need for revocation) in order to deliberately link a ballot to a voter, by applying her private key for a decryption.

  Note that the Judge may be external to the protocol, and is trusted. Further, since the protocol is optimistic, the Judge is only used in a minority of cases, where a voter's identity needs to be revealed, or Alice cannot find her vote on the bulletin board.

3.1.1 (Partially) Trusting $\mathbb{T}_1$

Our original intention with this work was to minimise trust completely: i.e., not to trust any group of talliers. In one of the earlier iterations of the protocol, $\mathbb{T}_1$ was a distributed group of talliers sharing a private key, and performing threshold decryptions and signatures for each vote.

Our initial work concerned allowing $\mathbb{T}_1$ to verify that Alice was a legitimate voter, and re-encrypt her vote, without any one tallier having enough knowledge of the final re-encryption to be able to link Alice's original vote to her final, re-encrypted one. One approach might have been to have each vote re-encrypted t times, once by each of t members of a quorum of talliers, forwarding each intermediate re-encryption (and proof thereof) to the next tallier in a chain. This has a number of problems, most important being that the final tallier in the chain gains knowledge of the full re-encryption, defeating the object of threshold decryption for $\mathbb{T}_1$.

We found that the best solution was to *partially* trust $\mathbb{T}_1$. This means that we trust that $\mathbb{T}_1$:

- will not reveal the link between Alice's ballot (x, y) and her re-encrypted ballot (x_f, y_f), except by request of the Judge;
- will make valid encryptions of voter identities when forming $\overline{\mathsf{id}}$ tags;
- will only sign and post to $\mathcal{BB}_2$ ballots which are valid;
- will not reveal the validity of any vote based on its δ value (validity token), or permit unauthorised access to the pool of δ values.
- will act honestly in communications with the Judge (no other honest communications are required than those stated here);

Note therefore that $\mathbb{T}_1$ *at no point* has access to Alice's unencrypted vote. We further do not trust $\mathbb{T}_1$ to reliably send communications—if messages do not arrive as expected, the voter can detect this.

A number of protocols make other assumptions which, as a result of partially trusting $\mathbb{T}_1$, we do

not need to make. In [20], votes are received without interference, implying that voters cast ballots on an anonymous channel. Many other protocols trust an anonymous, often untappable channel [7, 29, 12, 14, 16, 8] to not leak any information to observers, whether via a mix or not. Others assume the availability of a trusted Smart-Card or 'randomiser' to perform re-encryptions and proofs thereof [16, 12, 22], or some other trusted 'honest verifier' that is not allowed to co-operate with a mix network or other parties [34]. Civitas [7] requires that a voter trust at least one registration teller, *and* their voting client. A far larger majority trust that the voter cannot be observed at all during voting (perhaps via a voting booth), which is an assumption we are not prepared to make in order to provide remote voting — we merely assume that Alice can vote unobserved once (but can otherwise vote many times observed).

One of the harder challenges we faced was to integrate revocable anonymity into electronic voting — without trusting $\mathbb{T}_1$, we feel it would not have been possible to integrate this property, as Alice's ID would always be traceable by her, or a corrupt member of $\mathbb{T}_1$. As already discussed, being able to link a ballot to a voter is a legal requirement in Britain [2, 31]. We wished to do this in such a fashion that legitimate voters' votes could remain completely coercion-resistant (i.e., an attacker could not even tell *whether* a voter had voted), meaning that Alice's identity had to be encrypted in some way. Only the first-round talliers could do this encryption (Alice could lie about her identity, or could link her encrypted ID to her vote later on), but, if we were to use an untrusted first-round tallier quorum, every tallier would have access to Alice's ID, allowing her ballot to be linked by any coercer having access to the tallier group.

By partially trusting $\mathbb{T}_1$, we are able to avoid these problems, whilst making no more assumptions than the protocols above do, and assuming no anonymous channels. In future work, we will attempt to remove even this trust.

3.2 Trust Model

We make the following assumptions in our protocol:

1. All parties trust that $\mathbb{T}_1$ will not reveal the link between a ballot (x, y) and its re-encryption (x_f, y_f), or which δ values are valid
2. All parties trust that $\mathbb{T}_1$ will perform valid encryptions of each voter's identity, to afford anonymity revocation
3. The Judge and $\mathbb{T}_2$ trust that $\mathbb{T}_1$ will only sign and post to $\mathcal{BB}_2$ ballots which are valid
4. The Judge trusts that $\mathbb{T}_1$ will accurately and honestly send any data requested by it, to the Judge
5. All participants trust that the Judge will only authorise revocation of anonymity in appropriate circumstances
6. Alice trusts that she will receive one (and only one) valid voting token, along with several invalid ones, from the first-round talliers during registration.
7. Alice trusts the Judge to honestly state whether votes have been counted
8. All parties trust that voter identities will be stored correctly (and securely) on the second-round bulletin board

Note that we have already assumed that: $\mathbb{T}_1$ will batch votes before sending to $\mathcal{BB}_2$, to prevent timing attacks; Alice can vote once unobserved; and a t-sized quorum of $\mathbb{T}_2$ will not collude to break fairness or decrypt ballots until voting is over.

3.3 Threat Model

In this section, we consider the potential threats that could affect our protocol, based on the attacker's capabilities. We address how these threats are managed in §5. As to the assumptions we make about the attacker's strength based on the strength of the cryptography we use, we assume perfect cryptography.

Note that in our protocol, the attacker can assume the role of any entity (except the Judge). He is able to corrupt up to $t-1$ talliers where collusion is required to decrypt messages (and t is the threshold size for that quorum). All channels are public, so the attacker can:

1. Read messages

2. Decrypt and read any message m, subject to having the correct decryption key s for an encrypted message $(g^{\alpha}, g^{\alpha s}m)$
3. Intercept messages
4. Inject bad ballots in the first phase, and spurious messages generally
5. Temporarily block messages (although we assume resilient channels for liveness)

4 Requirements

We next give a list of requirements that our protocol should satisfy, followed by a discussion of how these requirements will prevent certain attacks on the protocol.

4.1 Protocol Requirements

We now present the requirements that we wish to satisfy in our work.

1. **Eligibility** Only eligible voters should be able to vote
2. **Uniqueness** Only one vote per voter should be counted
3. **Receipt-Freeness** The voter should be given no information which can be used to demonstrate to a coercer how *or if* they have voted, *after* voting has occurred
4. **Coercion-Resistance** It should not be possible for a voter to prove how they voted or even if they are voting, even if they are able to interact with the coercer during voting
5. **Verifiability**
   (a) **Individual Verifiability** A voter should be able to verify that their vote has been counted correctly
   (b) **Universal Verifiability** A voter should be able to verify that all votes have been counted correctly
6. **Fairness** No-one can gain any information about the result of the tally until the end of the voting process and publication of votes
7. **Vote Privacy** Neither the authorities nor any other participant should be able to link any ballot to the voter having cast it, *unless* the protocol to revoke anonymity has been invoked
   (a) **Revocable Anonymity** It should be possible for an *authorised entity* (or collaboration of entities, for us) to reveal the identity of a voter by linking his vote to him.
8. **Remote Voting** Voters should not be restricted by physical location

5 Protocol

Our voting protocol has four stages. We begin here with a brief discussion of the purpose of each stage, and then follow with a more detailed explanation.

In the first stage (which may be at any point before the voting session begins), the first round talliers generate and produce a designated-verifier signature on each of a collection of values δ_i, which are sent to the voter. Each has a designated verifier signature (as described in §2.2) $\mathsf{DVSign}_{\mathbb{T}_1}(\delta_i)$ paired with it; however only one of these signatures is valid.

In the second stage, voters provide a tuple containing an encrypted ballot, with a signed G-PEQDL to show ballot correctness and validity, to bulletin board $\mathcal{BB}_1$.Any member of $\mathbb{T}_1$, at random, decrypts a tuple, checks the signature provided with the vote (without being able to view the vote itself), and ensures that the voter is eligible. They then perform a random re-encryption of the ballot.

A number of designated verifier re-encryption proofs are sent back to the voter (so that an observer cannot determine if Alice's vote has been posted or not), and the re-encrypted vote, with an encrypted copy of the voter's public key, and a hash of these signed by one of the first-round talliers, is posted to a public bulletin board $\mathcal{BB}_2$ once a large enough batch is ready to send.

In the third stage, once voting has finished, a quorum of the second round talliers decrypts the votes and announces the result.

The protocol (summarised in Figures 3 and 1) proceeds in three stages. Note that unless otherwise specified, transactions between Alice and $\mathbb{T}_1$ are *not* encrypted, and channels are not private. Encryptions of a message m with the public key of entity a, usually denoted $(x, y) = (g^{\alpha}, h_a^{\alpha}m)$ are

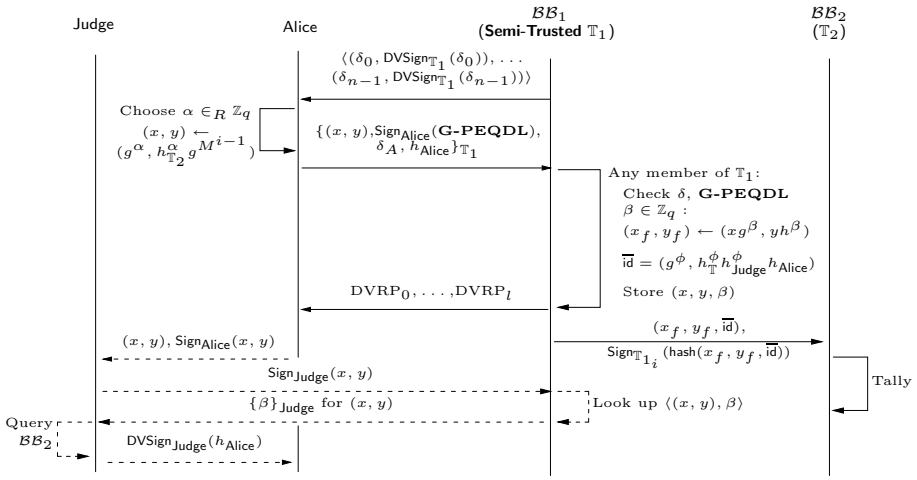

Figure 3: Our protocol. Note that dashed lines indicate a non-compulsory part of the protocol (voter complaints), and that the first communication ($\mathbb{T}_1 \rightarrow$ Alice is in-person).

denoted $\{m\}_a$ (excepting the encrypted vote value itself), for convenience.

Stage 1: Ballot Validity Tokens

The protocol begins with Alice registering in person to vote, with any of the first round talliers $\mathbb{T}_1$. They will give to her a collection of a *random number of* values δ_i. Each has a designated verifier signature $\mathsf{DVSign}_{\mathbb{T}_1}(\delta_i)$ paired with it, which has been generated by a member of $\mathbb{T}_1$. However, only one of these signatures is valid (clearly, only the voter with the correct private key can verify this fact). Alice hence receives a string

$$\langle(\delta_0, \mathsf{DVSign}_{\mathbb{T}_1}(\delta_0)), (\delta_1, \mathsf{DVSign}_{\mathbb{T}_1}(\delta_1)), \ldots, (\delta_{n-1}, \mathsf{DVSign}_{\mathbb{T}_1}(\delta_{n-1}))\rangle$$

As will be discussed, the coercion-resistance Alice enjoys increases with the size of n (i.e., the probability that the attacker can guess the correct δ value decreases with $|n|$). Note that Alice would be able to generate designated verifier signatures at her liberty. Alice is able to calculate which of the signatures is valid for the value paired with it; the tallier will also submit, to a private 'pool' of valid δ values, the correct value for each voter. If Alice votes under coercion, since she received a random number of δ values, an observer cannot force her to use all values (she could conceal one or more, or arbitrarily insert values). Hence she simply votes using invalid δ values.

If she later votes without coercion[3], she sends a signature of this value with her vote as a 'proof' of validity. Upon checking for eligibility, the talliers simply check the pool to see if a value is present[4] If she were to send a value for which the DV-Signature was incorrect when sent to her, this would alert the first-round talliers that her vote was made under coercion, which would alter their response to her. However, a coercer would not be able to distinguish a valid δ value from an invalid one, as he has no way of determining whether Alice herself made the designated verifier signature, or indeed whether the signature is valid.

Stage 2: Encrypted Vote Posting

As with other voting protocols using homomorphic encryption, we choose the form of the ballot in such a way that decryption of all ballots multiplied together leads to a simple tally of votes. A vote for the i^{th} candidate is given as $g^{M^{i-1}}$, where M is the maximum number of voters.

Voter Alice selects a value $\alpha \in_R \mathbb{Z}_q$, and encrypts her vote for candidate i using the public

[3]Note that we must assume that Alice is able to vote unobserved, but she only needs to do this once.

[4]We assume that δ values issued are unique, and that the space for selection of these values is large enough that Alice could not select another valid δ at random.

key of the *second round talliers*, to give $(x, y) = (g^{\alpha}, h_{\mathbb{T}_2}^{\alpha} g^{M^{i-1}})$. She groups this with the correct δ value δ_A, and her public key h_{Alice}. Finally, she calculates the Generalised Proof of Equality of Discrete Logarithms (see §2.3) for her ballot (x, y) to prove that the vote is of correct form, and produces a standard ElGamal signature on this. This tuple $\langle (x, y), \mathsf{Sign}_{\mathsf{Alice}}(\textbf{G-PEQDL}), \delta_A, h_{\mathsf{Alice}} \rangle$ is encrypted with the public key of the first-round talliers, and posted to the first round bulletin board, $\mathcal{BB}_1$.

Stage 3: Validity Checking

Once Stage 2 is complete, any member $\mathbb{T}_{1_i}$ of $\mathbb{T}_1$ removes the first layer of encryption on each vote on the first-round bulletin board. That tallier then:

1. verifies that the vote is legitimate, by ensuring that the δ value given is present in the pool of valid δ values[5]. Note that because the votes themselves are encrypted for $\mathbb{T}_2$, the first-round talliers cannot see *how* a voter votes — merely *that* a voter has attempted to vote.

2. verifies the G-PEQDL supplied with the ballot (x, y) to determine that Alice's vote is a single vote for a single valid candidate in the election

Once the validity of a ballot is assured, and any invalid ballots are disposed of, $\mathbb{T}_{1_i}$ re-encrypts (x, y) with a random factor β to give (x_f, y_f). That member also encrypts Alice's public key[6] by doing the following:

- Select a random $\phi \in_R \mathbb{Z}_q$
- Using the joint public key for both sets of talliers $h_{\mathbb{T}}$, and the Judge's public key, form

$$\overline{\mathsf{id}} = (g^{\phi}, h_{\mathbb{T}}^{\phi} h_{\mathsf{Judge}}^{\phi} h_{\mathsf{Alice}})$$

The tallier then continues. He:

3. generates a signature on $\mathsf{hash}(x_f, y_f, \overline{\mathsf{id}})$, and concatenates this with $(x_f, y_f, \overline{\mathsf{id}})$ to form the final message string.

[5] We presume that the 'valid δ' pool is inaccessible to coercers.

[6] Alice's public key is not taken 'as read' by the tallier concerned—it is compared against a list of public keys, and is of course also used to verify the signature Alice provides.

The tallier responsible for the re-encryption sends Alice a designated-verifier re-encryption proof (DVRP) that her vote has been included on the public bulletin board as (x_f, y_f), along with a number of other correct DVRPs, which are not valid for Alice (only she will be able to determine this). Note that if Alice's sent δ value were invalid, the tallier would send Alice only invalid DVRPs, meaning that an attacker could not determine whether her vote was invalid simply by messages received by Alice.

The tallier will then personally store the values $\langle (x, y), \beta \rangle$, and mark on the private electoral roll that Alice has voted (for example, by adding a signature of her public key). This information will never be released, except to the Judge as proof that Alice's vote was counted. The tuple $\langle x_f, y_f, \overline{\mathsf{id}}, \mathsf{sign}_{\mathbb{T}_1}(\mathsf{hash}(x_f, y_f, \overline{\mathsf{id}})) \rangle$ is posted to the second-round talliers' bulletin board. Alice is able to check the second bulletin board to ensure her vote appears and the signature on it is valid, but cannot convince anyone else of this fact (nor can she decrypt the re-encrypted vote). Any entity can check that a vote on the bulletin board is valid by verifying the signature for the hash of that vote.

Stage 4: Tallying

Once all DVRPs have been sent to their respective voters, it is simple for the second-round talliers $\mathbb{T}_2$ to decrypt votes. First, each $\langle (x_f, y_f), \overline{\mathsf{id}} \rangle$ is checked against its signed hash. Those not matching are ignored in tallying. A quorum of t talliers jointly decrypt a product

$$(X, Y) = (\prod_{j=1}^{l} x_{f_j}, \prod_{j=1}^{l} y_{f_j})$$

(without any single member having access to the private key, as discussed in §2.1), and then post the product to a publicly viewable place. The quorum threshold-decrypt the resulting tally, giving $g^{r_1 M^0 + r_2 M^1 + \ldots + r_L M^{L-1}}$, and $r_1, \ldots, r_L$ as the final tally. Note that since the second-round bulletin board is publicly viewable, any party can verify that any vote must have been correct, by comparing the published hash to the values given with it.

Anonymity Revocation

We have built into our protocol the ability to recover a voter's identity after the voting process is complete, but only with the co-operation of the Judge and a quorum of $\mathbb{T}$, the anonymity group. When Alice's vote is submitted to $\mathcal{BB}_2$, part of it is a token

$$\overline{\mathsf{id}} = (g^{\phi}, h_{\mathbb{T}}^{\phi} h_{\mathsf{Judge}}^{\phi} h_{\mathsf{Alice}})$$

If, in the tallying phase of the protocol, any ballot is found to be illegal (or if, for any other reason, anonymity has to be revoked), a quorum of members of the *anonymity tallier group* $\mathbb{T}$ need to collude (note that the t_{id} value for this threshold decryption should be higher than the size of either $\mathbb{T}_1$ or $\mathbb{T}_2$).

$$\frac{h_{\mathbb{T}}^{\phi} h_{\mathsf{Judge}}^{\phi} h_{\mathsf{Alice}}}{g^{\phi s_{\mathbb{T}}}} = h_{\mathsf{Judge}}^{\phi} h_{\mathsf{Alice}}$$

The Judge must now be sent the token, with appropriate evidence justifying anonymity revocation. The Judge can then divide by $g^{\phi s_{\mathsf{Judge}}}$ to give the voter's identity.

Voter Complaints

A disadvantage of using designated-verifier re-encryption proofs is that Alice cannot prove the validity of the proof she receives from the first-round talliers that her vote has been re-encrypted as (x_f, y_f), which she may need to do if she cannot find her re-encrypted vote on $\mathcal{BB}_2$.

A solution we might adopt would be for Alice to receive a 1-out-of-L re-encryption proof [17], which is requested by Alice after all votes are posted to the board. However, such a proof is quite laborious and would allow an attacker to see that Alice's vote was counted. Instead, Alice sends her original (x, y) to the Judge. The Judge requests the stored β, δ_A from the first-round talliers, and can then use these to check that Alice's vote was counted. If Alice's vote is counted, the Judge sends her δ_A with a valid designated verifier signature for δ_A. Otherwise, he makes the designated verifier signature invalid. Only Alice can determine this fact, and can again insert valid signatures arbitrarily. If Alice's vote is shown to have not been counted, we could also allow her to collude with the Judge to submit a vote a second time—in this manner, if her vote is again not counted, the Judge can take further action.

6 Informal Security Analysis

We now give an informal sketch of security proofs for our protocol. Our proofs of security assume the correctness of various cryptographic primitives and assumptions (discussed in §2)—note that although these primitives assume the provable security model, we work in the formal model, thus assuming that cryptography is perfect. With the exception of Lemma 1, we make no further assumptions about, or changes to, the primitives that we use.

6.1 Assumptions

We first give a number of assumptions based on the security of the primitives we have used.

6.2 Requirements Verification

We now discuss how we have satisfied the security requirements given in §4.1. We assume that in all traces, and any number of runs of, the protocol, an eligible voter voting involves at most one occurrence of that voter's public key being found on the electoral register; i.e., a voter receives exactly one valid signed δ value. We further assume, as mentioned above, that cryptography is perfect, as we adopt the formal model.

Property 1 (Coercion Resistance)**.** The protocol achieves *coercion-resistance*; i.e., the voter is unable to co-operate in *any way* with a coercer (excepting the Judge).

We have from the protocol that the only token Alice receives back from the talliers is a *designated verifier* proof of re-encryption, which proves *only to Alice* that her vote has been re-encrypted as (x_f, y_f). We have from §2.4 above that Alice is unable to use this proof to convince anyone else of the re-encryption.

As a result, although she receives proof that her vote has been recorded, and can verify this fact individually, she cannot prove to anyone else how she voted, or even if she voted—Alice is able to generate the DVRP herself.

At the beginning of the protocol, during in-person registration, one of the talliers[7] ($\mathbb{T}_1$) produces several δ values for each voter, with a threshold designated verifier signature for each, only one of which is valid. We have from §2.2 above that only Alice, the intended recipient of the threshold designated verifier signature can distinguish it as

- valid, and
- not produced by Alice herself.

Thus when Alice selects a δ value and sends it with her vote, she cannot co-operate with the coercer to show that it is valid; i.e., that she has voted legitimately at all. Note that Alice can continue to generate δ values and seemingly valid designated verifier signatures for them *ad infinitum*, but only one value, that with a correct tallier signature, will ever be valid.

If Alice is forced to vote multiple times, only the vote with the valid δ value will be counted. There is no way for the coercer to know which vote was genuine. He is not even able to determine if Alice's vote has been posted through timing attacks, as $\mathbb{T}_1$ batches re-encryptions before sending them to $\mathcal{BB}_2$.

Finally, Alice receives no useful proof of how (or if) she voted: she merely receives a designated verifier re-encryption proof, with which she alone can determine if her vote is to be counted.

Since coercion-resistance is strictly a stronger property than receipt-freeness, we can show that our protocol is receipt-free:

Property 2 (Receipt-Freeness)**.** The protocol achieves *receipt-freeness*; i.e., the voter is given no information with which she can prove how (or if) she voted.

For coercion-resistance, we require that Alice cannot cooperate in *any way* with a coercer to demonstrate how, or if, she votes legitimately. For receipt freeness, we merely require that Alice does not receive a receipt showing the result of her voting attempt. Since we have shown that this is not possible in order to prove coercion-resistance, it is trivial to see that receipt-freeness is satisfied.

Property 3 (Correctness and Eligibility)**.** The protocol is *correct* and satisfies *eligibility*; i.e., there is no trace of the protocol resulting in a successfully counted vote, from Alice, for candidate i, that did not begin with Alice voting for i. Further, there exists one (and only one) trace of the protocol resulting in a vote for candidate a that began with Alice voting for candidate a.

To prove this requirement, we need to demonstrate that there is no way that any two or more parties can collude illegitimately to defraud Alice or the authorities. Nor is it possible for Alice to actively collude with another voter or observer. We consider collusions between the parties shown in Table 4, and discuss them below. Bob represents any other voter, or coercer.

	Bob	$\mathbb{T}_{1_i}$	$\mathbb{T}_{2_i}$
Alice	i	ii	iii
$\mathbb{T}_{\mathbf{1_{j\neq i}}}$	iv	v	vi
$\mathbb{T}_{\mathbf{2_{j\neq i}}}$	vii	viii	ix

Figure 4: Table detailing possible collaborations between protocol entities. Note that as we partially trust $\mathbb{T}_1$, some collaborations are ineffective.

i. *Alice and any other voter or attacker (Bob).* We consider an attack where Alice colludes with any other voter or attacker to attempt to vote several times, or claim that her vote has not been counted.

   The **G-PEQDL** proof that Alice signs and sends to the first-round talliers demonstrates that for Alice's vote (x, y), y is in $\{h_{\mathbb{T}_2}^{\alpha} g^{M^0}, \ldots, h_{\mathbb{T}_2}^{\alpha} g^{M^{L-1}}\}$ where L is the number of candidates. For any vote to be forwarded to the second-round bulletin board, this proof must hold. Thus it is not possible for Alice to vote for more than one candidate at a time.

   Further, when Alice votes, she is recorded as having voted, providing her δ value is valid (if not, her vote is discarded). The first-round talliers store a record that Alice has voted, along with the re-encryption factor used for her. This can be used for later proof that she has voted, meaning Alice can neither vote more than once, nor falsely claim to have not had her vote counted.

ii. *Alice and any first-round tallier.* In this attack, Alice could attempt to collude with $\mathbb{T}_{1_i}$ to have

[7] This choice can again be made at random.

an invalid ballot accepted, or to vote twice. Our model dictates that we trust that $\mathbb{T}_1$ will only sign or accept ballots that are valid, hence this attack cannot occur. An attack in which Alice is given more than one valid δ token at registration is also prevented by the assumption that Alice will receive only one of these.

iii. *Alice and any second-round tallier.* We envisage an attack in which Alice colludes with a second-round tallier to bypass the security given by the first-round talliers, in order to request that her vote is altered. Second-round talliers are unable to write to the second-round bulletin board—in fact, only first-round talliers can. A threshold-size quorum of $\mathbb{T}_2$ is needed to change votes, should Alice wish to collude with a second-round tallier to do this.

Further, Alice's re-encrypted vote is listed with a signed hash on the bulletin board—this could not be altered without invalidating the vote.

iv. *An attacker and any first-round tallier.* In this attack, an attacker might collude with a first-round tallier to force it to obtain information about voters, or to ignore votes. Our trust model dictates that $\mathbb{T}_1$ will reveal no relation between ballots and their re-encryptions. Further, since Alice is able to vote once without observation, an attacker will not necessarily know which votes belong to her. If her vote is somehow ignored, Alice will notice that it has not been posted to $\mathcal{BB}_2$, and can complain.

v. *Two first-round talliers.* Two first round talliers have no more power than one, hence there is no attack in this scenario.

vi. *A first- and second-round tallier.* The only possible attack here is for the first- and second-round tellers to collude to reveal the link between voter and vote, or to alter individual votes. This would require the collusion of a quorum of $\mathbb{T}_2$ to decrypt an individual ballot, which we can safely assume will not occur. Since votes are batched by $\mathbb{T}_1$, a timing attack could not be mounted by either party.

vii. *Any other voter/attacker and any second-round tallier.* Here we envisage an attack in which an attacker colludes with any member of the second-round talliers $\mathbb{T}_{2_i}$, in order to alter Alice's vote. Due to the nature of the DVRPs Alice receives from $\mathbb{T}_1$, an attacker cannot determine what her re-encrypted vote looks like. Thus he cannot tell $\mathbb{T}_{2_i}$ which vote to alter. Further, votes cannot be amended without a quorum of members of $\mathbb{T}_2$, the bulletin board cannot be deleted from, and the signed hash of each vote ensures that such modification is not possible.

viii. *A second- and first-round tallier.* See (vi).

ix. *Two second-round talliers.* This attack would involve two (or any number less than t) talliers colluding to alter votes. A threshold-sized quorum of talliers is needed to decrypt or alter votes, and second-round talliers cannot write to or delete from $\mathcal{BB}_2$. Thus there is no possible attack.

Property 4 (Fairness)**.** The protocol is *fair*; i.e., no party can gain any information about the voting tally until all voting is complete.

We have from the protocol that all votes are encrypted with the key of the *second-round* talliers, and from §2.1 that votes encrypted with the group public key $h_{\mathbb{T}_2}$ can only be decrypted by a quorum of size t of that group collaborating.

Further, we have from Properties 1 and 2 that Alice cannot prove how she voted to anyone, nor is she able to co-operate with any coercer. §2.4 shows that despite proving the validity of a vote (i.e., that it is one vote for a valid candidate), the first round talliers are unable to ascertain whom Alice has voted for—Alice cannot use the DVRP she receives from $\mathbb{T}_{1_i}$ to prove how she voted, or indeed if her vote was accepted.

Thus, Alice cannot prove to any party how she has voted. As a tally is not released by the second-round talliers until voting is complete, no observer is able to gain any information (other than that only valid votes were accepted) until voting is complete.

Property 5 (Individual Verifiability)**.** The protocol allows a voter to verify that their vote has been counted in the tally.

When Alice's vote is submitted to the talliers, she receives back a Designated Verifier Re-Encryption

Proof, showing that her vote (x, y) has been re-encrypted as (x_f, y_f). She also receives several other designated verifier proofs for other voters' valid votes, posted to the bulletin board. We have from §2.4 that only Alice is convinced by the proof intended for her, and she cannot use it to prove how she voted to anyone else (as she could have herself generated the proof). Only Alice can be convinced that the other proofs are invalid for her.

If Alice's vote is not found on the second-round bulletin board, or is incorrectly signed, she contacts the Judge. The Judge receives the re-encryption factor used for Alice (β) and the δ_A value which she sent. He can then check the second-round bulletin board, and send δ_A and a designated verifier signature of it to Alice. We have from §2.2 that only Alice can ascertain that the designated verifier signature is valid. If it is, and if the δ_A value that Alice sent matches the one she receives, she can be assured that her vote has been counted, without assuring any coercer.

Property 6 (Universal Verifiability)**.** A voter is able to verify that *all votes* have been counted correctly.

We achieve universal verifiability quite trivially due to the properties of the ElGamal homomorphic cryptosystem we use [10, 9]. Any party is able to check that any vote is valid due to the signed hash given with it (as $\mathbb{T}_1$ is trusted to produce valid signatures only on valid ballots, verifiers can be assured of ballots' validities). Further, any party can generate the product of votes and compare this to the product announced by the talliers — as we use threshold cryptography, we can reasonably assume that the collusion of the required quorum of members of $\mathbb{T}_2$ will not occur, except to legally announce the tally.

Property 7 (Voter Privacy)**.** The protocol achieves *voter privacy.*

We have from §2.1 that threshold ElGamal encryption is secure for any number of coerced quorum members less than t. We also have from the protocol that $\mathbb{T}_1$ is trusted not to reveal any link between a ballot and its re-encryption. Thus, Alice's actual ballot cannot be decrypted until it reaches the second-round bulletin board. By this time, it has been randomly re-encrypted so that no link can be made to Alice, and she cannot prove how she voted to anyone. The only record of her identity, $\overline{\mathsf{id}}$, is doubly encrypted such that only a quorum of the Anonymity Teller Group, in collusion with the Judge, can link Alice to her ballot.

In the first stage of voting, Alice receives back only a designated verifier re-encryption proof, which preserves her privacy as her vote cannot be linked to the re-encrypted vote by anyone but her. Further, should she complain to the Judge, she receives a proof from the Judge that only she can verify.

Property 8 (Revocable Anonymity)**.** The protocol achieves *revocable anonymity.*

Should Alice's identity need to be revoked, it is simple for the Judge, with appropriate evidence, to select a ballot from the second-round bulletin board and decrypt it as shown in §5, without affecting any other ballot. Since the first-round talliers keep a record of all re-encrypted votes, it is possible for the Judge to trace a ballot back to its voter, or trace a ballot forwards from its voter's original (x, y) ballot.

Property 9 (Eligibility)**.** The protocol achieves *eligibility*; i.e., only eligible voters may vote.

Every voter (and their public key) are listed on a private electoral roll. When Alice votes, she signs a **G-PEQDL** and sends her δ value. Knowledge of a signing key *and* which δ value is correct ensures that Alice is eligible—the talliers will discard votes that are illegitimate.

Property 10 (Uniqueness)**.** The protocol achieves *uniqueness*; i.e., eligible voters may only successfully vote once.

There are a number of factors which ensure uniqueness is achieved. Each voter is listed on the electoral roll, and receives only one valid δ_A token. As soon as a voter votes, they are marked on the electoral roll as having voted, providing their **G-PEQDL** and δ_A values are as expected. If a voter is coerced and sends an invalid δ_A value, their vote is not counted and they will be able to vote again. The value that the voter receives back will simply be a string of invalid designated verifer proofs.

By the assumption at the start of this section, every voter receives only one valid δ value.

7 Conclusion

We have presented an election protocol providing what we believe to be the first scheme for revocable anonymity, whilst also allowing the voter coercion-free remote voting and verifiability, as well as legitimate-voter privacy. We require no untappable channels, and achieve an efficient 1-out-of-L scheme, integrating an extension of two-candidate discrete proofs of logarithm equality to L parties.

Our protocol also satisfies *remote voting*: as long as the voter is connected to the Internet, they are able to vote from any location, under the assumption that they can vote unobserved once. This, we argue, is a very natural assumption to make.

We envisage that, given the remote nature of our protocol, it could be implemented across the Internet. Future work will concentrate on removing the need to trust any single party (except the Judge) at all, and on enhancing the remote-voting nature of the protocol—we might consider how to ensure that the machine the voter votes from can be trusted by the voter. One option might be to allow the user to vote only through a signed applet.

References

[1] Josh Benaloh and Dwight Tuinstra. Receipt-Free Secret-Ballot Elections (Extended Abstract). In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 544–53, Montreal, 1994. ACM.

[2] Robert Blackburn. *The Electoral System in Britain*. Macmillan, London, 1995.

[3] Dan Boneh and Philippe Golle. Almost Entirely Correct Mixing with Applications to Voting. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 68–77, Washington DC, 2002. ACM.

[4] Ran Canetti and Rosario Gennaro. Incoercible Multiparty Computation. In *Proceedings, 37th Annual Symposium on Foundations of Computer Science*, pages 504–13. IEEE, 1996.

[5] David Chaum, Peter Y. A. Ryan, and Steve Schneider. A Practical, Voter-verifiable Election Scheme. In *Computer Security — ESORICS 2005 10th European Symposium on Research in Computer Security*, volume 3679 of *Lecture Notes in Computer Science*, pages 118–39, Milan, September 2005. Springer-Verlag.

[6] David Chaum, Jeroen van de Graaf, Peter Y. A. Ryan, and Poorvi L. Vora. High Integrity Elections. Cryptology ePrint Archive, Report 2007/270, 2007. `http://eprint.iacr.org/`.

[7] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a Secure Voting System. In *Proceedings, 2008 IEEE Symposium on Security and Privacy*, pages 354–68. IEEE, 2008.

[8] Ronald Cramer, Matthew Franklin, Berry Schoenmakers, and Moti Yung. Multi-Authority Secret-Ballot Elections with Linear Work. In *Advances in Cryptology - EUROCRYPT'96. Proceedings*, pages 72–83, Berlin, 1996. Springer-Verlag.

[9] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In *Advances in Cryptology - EUROCRYPT '97. Proceedings*, pages 103–18, Berlin, 1997. Springer-Verlag.

[10] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–72, 1985.

[11] Jordi Barrat Esteve. A preliminary question: Is e-voting actually useful for our democratic institutions? What do we need it for? In *Electronic Voting 2006, 2nd International Workshop: Proceedings*, volume P-86 of *Lecture Notes in Informatics*, pages 51–60, Bregenz, Austria, 2006. Gesellschaft für Informatik.

[12] Chun-I Fan and Wei-Zhe Sun. An efficient multi-receipt mechanism for uncoercible anonymous electronic voting. *Mathematical and Computer Modelling*, 48:1611–1627, 2008.

[13] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Proceedings, Advances in Cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, New York, 1987. Springer-Verlag.

[14] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A Practical Secret voting Scheme for Large Scale Elections. In *Advances in Cryptology - AUSCRYPT '92. Workshop on the Theory and Application of Cryptographic Techniques Proceedings*, pages 244–51, Berlin, 1993. Springer-Verlag.

[15] Alejandro Hevia and Marcos Kiwi. Electronic jury voting protocols. *Theoretical Computer Science*, 321(1):73–94, 2004.

[16] Martin Hirt. *Multi-Party Computation: Efficient Protocols, General Adversaries and Voting*. PhD thesis, ETH Zurich, 2001.

[17] Martin Hirt and Kazue Sako. Efficient Receipt-Free Voting Based on Homomorphic Encryption. In *Advances in Cryptology - EUROCRYPT 2000. Proceedings*, volume 1807 of *Lecture Notes in Computer Science*, pages 539–56, Bruges, Belgium, 2000. Springer-Verlag.

[18] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making Mix Nets Robust for Electronic Voting by Randomised Partial Checking. In *Proceedings of the 11th USENIX Security Symposium*, pages 339–53, Berkeley, 2002. USENIX Assoc.

[19] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated Verifier Proofs and Their Applications. In *Proceedings, EUROCRYPT '96*, pages 143–154. Springer-Verlag, 1996.

[20] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-Resistant Electronic Elections. In *WPES'05: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, pages 61–70, New York, 2005. ACM.

[21] Byoungcheon Lee, Colin Boyd, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *In Proc. of Information Security and Cryptology (ICISC'03)*, volume 2971 of *Lecture Notes in Computer Science*, pages 245–258. Springer-Verlag, 2004.

[22] Byoungcheon Lee and Kwangjo Kim. Receipt-Free Electronic Voting Scheme with a Tamper-Resistant Randomizer. In *Proceedings of ICISC2002*, pages 389–406. Springer-Verlag, 2002.

[23] Tatsuaki Okamoto. Receipt-Free Electronic Voting Schemes for Large Scale Elections. In B. Christianson *et al.*, editor, *Security Protocols Workshop*, volume 1361 of *Lecture Notes in Computer Science*, pages 25–35. Springer-Verlag, 1997.

[24] Ronald Rivest and Warren Smith. Three Voting Protocols: ThreeBallot, VAV, and Twin. In *Proceedings of Electronic Voting Technology Workshop, 2007*, pages 1–14, Boston, MA, 2007.

[25] Aviel D. Rubin. Security Considerations for Remote Electronic Voting. *Communications of the ACM*, 45(12):39–44, 2002.

[26] Peter Y.A. Ryan. Prêt à Voter With a Human-Readable, Paper Audit Trail. Technical Report CS-TR: 1038, Newcastle University, 2007.

[27] Peter Y.A. Ryan and S.A. Schneider. Prêt à Voter with re-encryption mixes. In *Computer Security—ESORICS 2006. Proceedings*, volume 4189 of *Lecture Notes in Computer Science*, pages 313–26, Hamburg, 2006. Springer-Verlag.

[28] Shahrokh Saeednia, Steve Kremer, and Olivier Markowitch. An Efficient Strong Designated Verifier Signature Scheme. *Information Security and Cryptology — ICISC*, 2971:40–54, 2003.

[29] Kazue Sako and Joe Kilian. Receipt-Free Mix-Type Voting Scheme: A practical solution to the implementation of a voting booth. In *Advances in Cryptology - EUROCRYPT'95. Proceedings*, pages 393–403, Berlin, 1995. Springer-Verlag.

[30] John Stewart. A Banana Republic? The Investigation into Electoral Fraud by the Birmingham Election Court. *Parliamentary Affairs*, 59(4):654–667, 2006.

[31] The Electoral Commission. Factsheet: Ballot Secrecy, December 2006.

[32] Melanie Volkamer and Robert Krimmer. Secrecy forever? Analysis of Anonymity in Internet-based Voting Protocols. In *Proceedings, First International Conference on Availability, Reliability and Security, ARES 2006*, pages 340–347, Vienna, 2006. IEEE.

[33] Stefan G. Weber, Roberto Araújo, and Johannes Buchmann. On Coercion-Resistant Electronic Elections with Linear Work. In *Proceedings, 2007 2nd International Conference on Availability, Reliability and Security*, pages 908–16, Vienna, 2007. IEEE.

[34] Aneta Zwierko and Zbigniew Kotulski. A Light-Weight e-Voting System with Distributed Trust. *Electronic Notes in Theoretical Computer Science*, 168:109–26, 2007.

A Assuring Mobile Coercion-Resistance

As noted in §2.4, to assure full coercion resistance if Alice does not vote from a voting booth, she needs some way to protect her private key, so that an attacker cannot simply take it and verify each designated verifier re-encryption proof she receives. We here discuss alternatives for achieving such assurance.

There are two possible scenarios which could occur when Alice votes remotely. In the first instance, she may *choose* to give her private key away for some reward. In this case we can simply solve the problem by introducing some 'economic disincentive' [7], as in the Estonian electronic voting system, in which the key a user uses to vote is also that which produces signatures.

In the second instance, Alice may be forced to give up her private key. This is a serious attack, since she cannot 'fake' her key: the public counterpart of her private key is common knowledge. It seems, in fact, that this attack is a limitation of *all* electronic voting protocols which attempt remote voting—in order to achieve coercion-resistance, there must be some token which only the voter has knowledge of, else she could simply be simulated by the coercer. A possible solution is to not use Alice's private key for the relevant communication. Alice would be forced to register *in person* to vote with a member of $\mathbb{T}_1$ (though it should be noted that this could be done at any time before the election). When she does this, her identity is verified, and she is given a new public/private keypair $(s_{\mathsf{Alice}-v}, h_{\mathsf{Alice}-v})$, which essentially acts as a session key for Alice's vote. The public part of this keypair is stored with Alice's name on the private electoral roll.

Now, when Alice's vote is re-encrypted, $\mathbb{T}_1$ simply forms the Designated Verifier Re-encryption Proof setting $\sigma = g^j h^u_{\mathsf{Alice}-v}$. If Alice is not being coerced, she can verify the DVRP in the same way as usual using her new keypair. If she *is* being coerced, she is free to generate a keypair of her

own, $(s'_{\mathsf{Alice}-v}, h_{\mathsf{Alice}-v})$, and insert seemingly valid DVRPs, using $s'_{\mathsf{Alice}-v}$ to do so, into the communication from $\mathbb{T}_1$.

Note that if we do need to make the extra assumption that Alice must pre-register in person, we must add to our trust model the assumption that $\mathbb{T}_1$ does not leak Alice's 'session' voting key $h_{\mathsf{Alice}-v}$ to coercers.