True Trustworthy Elections: Remote Electronic Voting Using Trusted Computing

Matt Smart and Eike Ritter

School of Computer Science
University of Birmingham
{m.j.smart,e.ritter}@cs.bham.ac.uk

Abstract. We present a new remote, coercion-resistant electronic voting protocol which satisfies a number of properties previously considered contradictory. We introduce trusted computing as a method of ensuring the trustworthiness of remote voters, and provide an extension to our protocol allowing revocable anonymity, on the grounds of it being a legal requirement in the United Kingdom.

1 Introduction

One of the driving factors for electronic elections is *remote voting*—the notion that a voter can vote from any location. Achieving this whilst also achieving coercion resistance (i.e., allowing the voter to vote in the presence of a coercer, without being able to prove how they are voting, or whether their vote is valid) is very difficult, especially when also considering voter anonymity: not only do the authorities need to be convinced that each voter is running the correct voting protocol, but the voters must also be convinced that the authorities are behaving correctly.

It is very important that *revocable anonymity* in electronic voting—the ability to link a ballot back to its voter—be given adequate consideration. In the UK, it is a legal requirement that it should be possible for the election authorities to link a ballot to its voter [6, p. 106]. Only we currently consider this notion [30].

In this paper, we present a protocol which uses *trusted computing* to achieve assurances as to the state of the voter's (remote) machine, whilst also permitting revocable anonymity, and satisfying the other standard requirements of e-voting protocols. We use the Direct Anonymous Attestation (DAA) protocol [8] to provide a mechanism for cryptographically assuring the authorities of the state of a remote platform (run by a voter), whilst also assuring the voter that her vote is counted anonymously. No *remote* voting protocol has considered the state of a voting machine before, though the notion has been suggested at a high level, without implementation detail [9], and some 'polling station'-type protocols (not suitable for remote voting) exist [13]. Arguably, if a voter's machine can be made to display false statements to a voter, then there is simply no point in making the rest of the protocol secure: the user's machine is the 'weak link' in the chain. Indeed, if the voter's machine is compromised by a trojan (or such), then irrespective of the protocol being implemented, any remote voting protocol is inherently insecure. Use of the Trusted Platform Module (TPM) mitigates this risk.

1.1 Related Work

No work besides our own [30] provides revocable anonymity in electronic voting. Only a small amount of work provides coercion-resistant, remote electronic voting. Civitas [10] is a good example, based on the JCJ voting protocol [20]. However, it does not seem scalable—every encrypted vote requires several expensive plaintext equivalence tests, and credential generation requires the voter to contact every tallier. This complexity suggests that further modification to include revocable anonymity would be unwise.

Further, despite a discussion of using trusted computing and Direct Anonymous Attestation for peer-to-peer networks [4], and preliminary discussions of trusted computing in electronic voting [2, 9, 3, 19], we found very few actual protocols: [13] give a protocol which uses the TPM to provide trustworthy 'polling station' DRE (Direct Recording Electronic voting) machines. Whilst their solution is interesting, we are interested in *remote* voting only, and find that the amount of trust placed in a number of entities by the authors is too high: the election authority, tallying authority and precinct judge all need to be fully trusted, and nothing is done to mitigate the possibility of a voter visiting the DRE machine multiple times in an election. The use of a single Platform Vote Ballot (PVB) key to sign the vote storage area also introduces a weak point for the security of all votes on a single DRE. [32] presents a discussion and high level, basic protocol for remote voting using the TPM. [25, 26] also present high-level discussions of in-person polling station voting protocols using trusted hardware. It should be noted that the work of [25] is not receipt-free, and that of [26] places complete trust in the authorities.

Many electronic voting protocols [29, 14, 20] rely on anonymous channels, or anonymous and untappable channels [23], to satisfy some security properties. In our work, we require an anonymous channel in the voting phase. Many protocols use mix networks for this [10, 27, 29, 7, 17], which provide effective anonymity as long as at least one participant in the mix is honest. We deliberately do not specify the method of implementation for our anonymous channel, but suggest that a Tor/Onion Routing network [12], or any other protocol allowing *bidirectional* anonymous communication would be suitable.

We, like many previous protocols, use probabilistic homomorphic encryption and re-encryption to ensure universal verifiability and unlinkability of ballots (through decryption of a product of encrypted votes) [5, 11, 20, 10], which naturally lends itself to threshold cryptography, affording us a greater level of assurance against corrupted talliers. These protocols require, for remote voting, that the voter is not observed at the "very moment of voting" [22].

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 evidences the fact that a voter *has voted successfully.* [20] and implementations thereof [10] 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. Not only does the protocol itself use several inefficient, expensive Plaintext Equivalence Tests throughout, but revocation of anonymity would require a further PET between the credential supplied with a vote and every credential on the voter list, followed by a collusion with the registrar. [22] 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.

Revocable anonymity is a concept which has been considered at great length in other fields, such as digital cash [18, 21]. In digital cash, it is particularly important that it should be possible to both link an electronic coin to the person who spent it once the transaction has occurred , and link a person's identity to all coins available to him. One manner in which this can be done is to encode an encrypted copy of the coin owner's identity into every coin. Requiring two or more parties to perform encryption, including a judge [18], ensures that a user's anonymity will not be revoked unless there is sufficient legal cause. In our work, we protect the voter's identity using a similar mechanism.

In [30], we present a coercion-free remote electronic voting protocol which permits the voter to vote anonymously, whilst maintaining coercion-resistance and voter verifiability, and the ability to revoke her anonymity should the need arise. One of the shortcomings of the protocol is that it requires a certain level of trust in the first set of talliers (*viz.*, that $\mathbb{T}_1$ does not reveal the link between a ballot and its reencryption, only encrypts Alice's identity correctly, and only posts valid ballots to the bulletin board), in order to assure that collaboration between both tallier sets could result in Alice being linked to her ballot only with the cooperation of a judge. The authors also assumed that the platform the Alice voted from was always trustworthy (as is common in remote voting).

It is possible to reduce the amount of trust required in the talliers if we reduce the amount of information that needs to be kept private. [30] also makes no use of Trusted Computing or the TPM: this limits the amount of trust that can be placed in any remote client. In our work, we present a new approach, which uses the TPM to enhance the trustworthiness of clients, further reducing the need for trust in the talliers.

1.2 Our Contribution

In this work, we introduce the first practical work on a *remote* electronic voting protocol which uses *trusted computing* (specifically, the TPM and Direct Anonymous Attestation protocol). As we have already mentioned, a number of existing works discuss the applicability of trusted computing and the TPM to electronic voting. We are the first to extend this to remote electronic voting whilst also providing a detailed protocol to do so, leading to several contributions:

– A remote voting protocol allowing authorities to be convinced of the state of the voter's machine, and allowing anonymity revocation via the TPM

– A protocol allowing Alice to remain anonymous, whilst satisfying her eligibility to vote via a novel use of the DAA protocol
– An extension to the protocol allowing a voter to be traced to her vote, should the legal need arise, but only with the co-operation of a judge.

Protocol Schema We present a three-phase protocol, where voters do not need to synchronise between phases. In the first phase, our legitimate voter, Alice, registers *in person* to vote, In the next phase, she and her trusted platform module (TPM) execute the DAA *Join* protocol [8] and receive a certificate proving her eligibility to vote (the certificate is split into three parts, divided between Alice and her TPM).

In the final phase, Alice and her TPM execute the DAA *Sign* protocol in order to complete her vote, which is sent as an ElGamal encryption with a proof of its validity. Voting authorities execute the DAA *Verify* protocol, after which Alice's vote is re-encrypted, and she receives back a designated verifier proof of that re-encryption. Should Alice need to, she can request assistance from the Judge.

1.3 Structure

In §2, we define a number of preliminaries, and a number of primitives we make use of. In §3, we give the participants, trust model and threat model for our work. In §4 we present our protocol, and we give a brief list of the requirements that we have satisfied in §5. Finally we conclude.

2 Preliminaries

In this paper, we assume the availability of the following cryptographic primitives and protocols. Note that, like many papers in the field which adopt a standard Dolev-Yao model, we make the assumption that the cryptography in the primitives below is perfect.

2.1 Trusted Computing

A *trusted computer* is one that, through the use of a *trusted platform module* (TPM), and other technologies such as memory curtaining, sealed storage and *remote attestation*, removes reliance on the end user to prove that his computer is secure. The benefits of its use for remote applications requiring secure information flow and data handling are clear.

In the field of remote electronic voting (that is, voting from any internet-connected terminal), for example, we might require that a user can only vote from a machine that is running the correct voting software, for obvious reasons. We could do this by providing each voter with a bootable operating system 'live CD'-type disc[1].

[1] We note that, as suggested by [13], security of any protocol that obtains software and private keys from removable media is vulnerable to compromise. This issue can be mitigated by having the TPM compare a publicly known signed hash of the intended executable code with a hash the TPM itself generates. In fact, the user could make this comparison.

However, we naturally still require that the voter using the trusted machine remains *anonymous*, whilst still being able to demonstrate that the machine she is voting from is trustworthy.

For brevity we do not elaborate on the structure of, or commands of the TPM here. The reader is directed to [31, **?**] for further information. For our purposes, it is sufficient to state that actions performed by the TPM are trustworthy.

Direct Anonymous Attestation *Attestation* in our context is the notion that some verifier wishes to be convinced that Alice is using a machine which contains a valid, permitted TPM. Later, this TPM can prove that Alice's machine is running the correct software, while allowing Alice to remain anonymous.

Direct Anonymous Attestation (DAA) is the solution currently built into the TPM specification. The protocol is complex, and we advise that the uninitiated reader consult [8] for a full explanation. On a high level, DAA is split into three sub-protocols: *join*, *sign* and *verify*. In the *join* protocol, a host and a TPM gain *attestation* (a Camenisch-Lysyanskaya signature) on a secret value, chosen by the TPM, demonstrating that the host's machine contains a valid TPM.

In the *sign* protocol, the host and TPM use a proof of knowledge of this attestation to anonymously prove that they gained this verification, possibly attesting to the state of their machine in the process, and producing a DAA Signature on some message (generally a key). This signature is verified in the final stage of the protocol.

2.2 Threshold ElGamal Cryptosystem

For encryption of actual votes, we use an exponential ElGamal encryption scheme under a q-order multiplicative subgroup $G_q = \langle g \rangle$ 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. We detail in §4 how votes are encrypted, but also note that we use $\{m\}_k$ to denote encryption of m under key k, where the encryption scheme is unimportant. In parts of our protocol, we use a (t, n)-threshold decryption scheme analogous to that of [11], such that a majority t out of n key-share holders would have to collude to decrypt. For brevity we do not discuss this here.

We have selected our cryptosystem for the voting section of our protocol because of the ease of tallying that a multiplicative homomorphic cryptosystem provides. The use of an exponential encryption scheme permits simple tallying, but requires the solving of a discrete logarithm. Of course, this becomes more difficult with increasing exponent size. With modern computing power and methods such as the Index Calculus and Baby-step Giant-step algorithms for computation of discrete logarithms, we do not see this to be an issue, but it would also be possible to hold several smaller 'regional' elections, rather than one large election, with our protocol to alleviate this problem. We also note that alternative threshold cryptosystems, such as that of Paillier [24], could be used for very large, country-wide elections.

2.3 Threshold Signature Scheme

In order to ensure that eligibility and uniqueness are always satisfied in our protocol, we employ a $(t, n)-$threshold signature scheme during the voting phase of the protocol. A threshold signature scheme works in a similar way to a threshold decryption scheme: of n possible talliers, t must collude to generate a signature on a message. The scheme that we adopt is not of great consequence, but the one used by [15] has good verification properties and fits in well with the exponential ElGamal cryptosystem that we use.

2.4 Anonymous Channel

Due to the nature of our protocol, we require an anonymous, bidirectional channel, so that Alice can both send her vote anonymously, and receive proofs of her vote having been counted. We note that standard mix networks are not designed to receive replies, but onion routing-based networks are [12].

2.5 Strong Designated Verifier Signature Scheme

We adopt the designated verifier signature scheme of [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 talliers 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]. For brevity we do not discuss the scheme here, but direct the reader to [28] instead. We denote by $\mathsf{DVSign}_{\mathsf{a}\to\mathsf{b}}(m)$ a designated verifier signature on a message m produced by party a and intended for reading by party b.

2.6 Proof of Equality of Discrete Logarithms

In order to prevent an attack in our voting scheme (voting for several candidates or for one candidate multiple times with the same ballot), we require that the voter demonstrates to a verifier that her vote is of the correct form (without revealing what the vote is).

A voter's vote is of the form $(x, y) = (g^{\alpha}, h_{\mathbb{T}_v}^{\alpha} g^{M^{i-1}})$ 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 adopt our Generalised Proof of Equality of Discrete Logarithms (G-PEQDL) scheme [30, pp. 43-44] in order for Alice to provide such a proof (with the small change that the challenge value c is formed using Alice's TPM's public AIK, rather than h_{Alice}), and refer the reader to this paper.

2.7 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 a re-encryption $(x_f, y_f) = (xg^{\beta}, yh^{\beta})$, where $\beta \in_R \mathbb{Z}_q^*$.

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) based on a fresh keypair that Alice selects: 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 [22, 16], such that the prover, P (the agent that does the re-encryption) demonstrates to Alice that (x_f, y_f) is equivalent to (x, y) in such a manner that the original message is not revealed, and this proof cannot convince any other entity. The reader is directed to the above papers for more details.

3 Protocol Model

3.1 Participants

Our protocol is modelled with four kinds of participants. All participants are able to communicate via a network, which is not untappable.

- **Voters.** The protocol allows M voters $v_i \in \{v_0, v_1, \ldots, v_{M-1}\}$ to vote. Alice is an honest voter who wishes to vote anonymously. She can vote an unlimited number of times, but must be able to vote *once* unobserved.
- **Administrator.** The (in-person) administrator $\mathbb{A}$ is a single entity, responsible for ensuring that Alice receives a random number of paper *validity cards* containing validity tokens δ_j. We expand upon this in the next section. $\mathbb{A}$ is responsible for identifying Alice, but not for determining her eligibility to vote.
- **Registrar.** The registrar $\mathbb{R}$ is a single agent, possessing a secret key $s_{\mathbb{R}}$. Note that we assume a bottleneck will *not* occur here, but we could equally use a group of identical registrar agents to mitigate such a problem.
  The registrar is responsible for ensuring, via the DAA *Join* protocol, that Alice is eligible to vote, and has not attempted to register already. The registrar will send Alice a voter group membership certificate, with which she can prove to the talliers that her vote is permitted.
- **Talliers.** The talliers, $\mathbb{T} = \{\mathbb{T}_1, \ldots, \mathbb{T}_n\}$, are a group of agents (disjoint from $\mathbb{R}$) who authorise the addition of each submitted ballot to the *bulletin board*, via the DAA *sign* and *verify* protocols. Each tallier has a copy of a secret key $s_{\mathbb{T}}$, with which he determines the validity of votes, and a *share* of a secret key $s_{\mathbb{T}_v}$, with which he collaborates with a quorum of $\mathbb{T}$ in order to decrypt the end tally, once the election is finished. These keys are unrelated—we use them to ensure that no single tallier has access to an individual vote. $\mathbb{T}$ are also responsible for re-encrypting votes, and sending proof of this to Alice.

3.2 Trust Model

We make the following assumptions in our protocol:

1. The TPM and the manufacturers of the TPM (the root of trust), are trusted to behave as intended by the protocol
2. All parties trust that $\mathbb{T}$ will not reveal the link between a ballot (x, y) and its re-encryption (x_f, y_f)

3. All voters trust that the validity of any given δ value will not be revealed by $\mathbb{A}$, *except* to members of $\mathbb{T}$ via a designated verifier signature
4. All parties trust that each voter will only be permitted to submit *one* validity card to the secured box for each election
5. All parties trust that $\mathbb{R}$ will not issue group membership certificates to ineligible voters, and will only do so once for eligible voters
6. All participants trust that the Judge will only authorise revocation of anonymity in appropriate circumstances
7. Alice trusts the Judge to honestly state whether votes have been counted

3.3 Threat Model

We now consider the potential threats that could affect our protocol, based on the attacker's capabilities. We address how these threats are managed in §4. Note that, as mentioned earlier, we assume perfect cryptography.

In our protocol, the attacker can assume the role of any entity, except the Judge or $\mathbb{A}$. 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 (the mix network is tappable, but anonymous), so the attacker can:

1. Read and intercept 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. Inject bad ballots in the voting phase, and spurious messages generally
4. Temporarily block messages (although we assume resilient channels for liveness)

4 Protocol

Our protocol has three stages. Diagrams, where necessary, are given in Figures 1, 2 and 3. We use a number of TPM commands in our protocol: these are denoted `as such`. We do not modify the TPM API in any way.

In-Person Registration (Fig 1) In order to begin voting, Alice first has to apply *in person* to vote, with the administrator $\mathbb{A}$. This can be at any point before the election. Once her identity is confirmed by $\mathbb{A}$, Alice is observed selecting a number, r, of *validity cards* from a box. r is generated randomly by $\mathbb{A}$ when Alice's identity is confirmed. These cards are pieces of paper with a perforation down the middle, and the same value $\delta_j : j \in \{0, \ldots, r-1\}$ printed on each side. Alice selects (mentally) one of the cards, whose δ value, δ_A, will denote her intended vote. She separates the card along the perforation, and places half of it into a secure box, retaining the other half. The bin must be designed to accept only one card per voter. Note that Alice does not need to bring any device (viz., the machine containing her TPM) with her: she merely needs to select cards.

With the remaining cards, Alice separates each card and places one half of each card into a shredder. Again, we must ensure that Alice destroys half of each validity card that she has not chosen to denote her intended vote.

Alice leaves in-person registration with several halves of validity cards. She has a mental note of which is valid (and could, in fact, discard or hide that one), but cannot prove which is valid to any observer. As she took a random number of cards, an observer cannot force her to vote once with every card she selected. Note that only $\mathbb{A}$ has access to the secure box, and that the voter has no way to prove how many cards she selected.

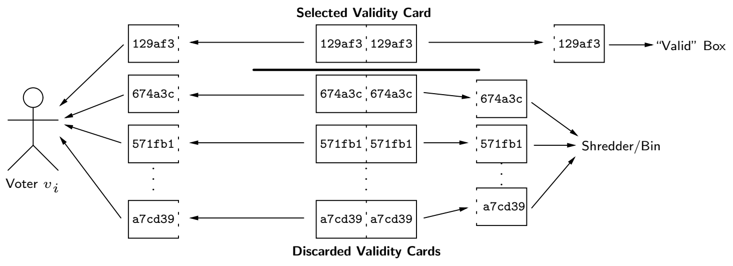


Fig. 1. In Person Registration

We note that our approach to voter registration is unconventional for a remote voting scheme. However, it removes the unrealistic requirement for an electronic untappable channel to the administrator (like that suggested by [10]), instead using a physical registration in which the voter can take part at any time before the election. This approach is clearly also more "user-friendly". Our design considerably reduces the trust we need to place in $\mathbb{A}$ (he now only knows which δ values are valid, not for whom, so we need only ensure that he does not release this information).

Join Alice and her TPM, $\text{TPM}_{\text{Alice}}$, execute the DAA *Join* protocol: this is as with the DAA *Join* protocol [8], and so we do not provide a diagram to represent it. The only change that we make is that, before issuing a certificate to a voter, the registrar $\mathbb{R}$ must ensure that said voter is eligible to vote. The communication channel with $\mathbb{R}$ does not need to be anonymous. We however adopt the requirement of [8] that the channel must be 'authentic' between $\text{TPM}_{\text{Alice}}$ and $\mathbb{R}$: i.e., the registrar must be sure that it is communicating with the correct TPM. Such authenticity can be achieved using the TPM's *endorsement key* (EK) for initial communications [8].

The product of the Join protocol is a membership certificate generated by $\mathbb{R}$. With this certificate, Alice can prove that she is a member of the group of legitimate voters, and is therefore allowed to vote. She will have registered using a unique pseudonym, and will use a different pseudonym to vote, making her registration and voting unlinkable. We refer to [8] for more detail.

Voting (Fig 2) The protocol by which Alice and her TPM vote is shown in Figure 2. If we assume that Alice can be tracked by an attacker with a global view of the network (and thus, the ability to see the IP address Alice votes from), then we must use an anonymous channel to preserve Alice's coercion-resistance and privacy.

First, we begin with an execution of the DAA *Sign* protocol (denoted as such in Figure 2—again, we omit the detail of the DAA Sign protocol for brevity, and refer the reader to [8] instead).

The outcome of the *Sign* protocol is a signature σ which convinces the talliers $\mathbb{T}$ that Alice's machine contains a TPM, and that she is a certified, eligible member of the voters group. Alice's TPM generates an *attestation identity key* $\text{AIK}_{\mathsf{Alice}}$ which is sent to $\mathbb{T}$ as part of the DAA signature, and will be used to prove authenticity of later messages. Note that this AIK is not linkable to Alice in any way, and the communication with $\mathbb{T}$ is similarly unlinkable [8].

With the *Sign* protocol complete, $\mathbb{T}$ can then query Alice's TPM as to the state of her machine. To do this, any member $\mathbb{T}_i$ of $\mathbb{T}$ begins an *encrypted transport session* between itself and Alice's TPM directly (note that Alice does not see the result of any transactions that occur here). $\mathbb{T}_i$ selects a challenge nonce c_v, and requests a hash of the current state of the TPM's registers, using the command `TPM_QUOTE`, and including the challenge. The TPM responds with the appropriate data. If $\mathbb{T}_i$ is satisfied that the machine is in the correct state, it requests that the TPM create a new keypair, bound to the correct TPM register (PCR) states. This means that, when a decryption is needed using this key, it can only occur if the TPM's PCRs are in the correct state. We denote the *handle* of this key as k_A, and note that the key is asymmetric, the private part being accessible only to the TPM.

Next, Alice generates a fresh ElGamal keypair, $(s_v, h_v = g^{s_v})$. She then sends a message `votetoken` to $\mathbb{T}$. `votetoken` contains Alice's vote, in the form of an exponential ElGamal encryption $(x, y) = (g^{\alpha}, h^{\alpha}_{\mathbb{T}_v} g^{M^{i-1}})$, where she is selecting the i^{th} candidate, her chosen δ_A value (should she be voting according to her own wishes) or any other δ value (if she is being coerced), the public part of the aforementioned key h_v, and the **G-PEQDL** proof that her vote is for one valid candidate only. The tallier $\mathbb{T}_k$ that receives Alice's vote now checks whether it was sent under coercion. To do this, he sends $\delta, \mathsf{sign}_{\mathbb{T}}(\delta)$ to $\mathbb{A}$. $\mathbb{A}$ checks whether the δ value received is in the secure box, and if so, sends a correct designated verifier signature of the value, $\mathsf{DVSign}_{\mathbb{A}\to\mathbb{T}_k}(\delta)$. If the δ value is not found in the box (meaning Alice sent a vote under coercion), an incorrect designated verifier signature is returned to $\mathbb{T}$. Again, only $\mathbb{T}_k$ can determine this, and cannot prove this fact to an observer.

Once Alice's vote is determined to be non-coerced, her G-PEQDL proof is checked by $\mathbb{T}_k$. If this is invalid, her vote is discarded. If the G-PEQDL is correct, Alice's vote is re-encrypted using a re-encryption factor $\beta \in_R \mathbb{Z}_q$. If her vote was not coerced, Alice is sent a tuple of designated verifier proofs of re-encryption (DVRPs), produced using the public key h_v Alice generated earlier. One of these is valid for Alice's re-encrypted vote; the others are valid for other votes already on the bulletin board[2]. Each DVRP is separately encrypted using the public part of the key k_A which Alice's TPM generated. This means that Alice is free to have her machine generate re-encryption proofs herself (the nature of the proof is such that the entity for whom the proof is designated can use her private key—s_v in this case—to generate further DVRPs), to fool coercers.

[2] Vote submissions are batched so that there are always enough votes on the bulletin board to do this: talliers can agree a policy beforehand as to how the first few votes are posted.

The re-encrypted (x_f, y_f) is sent to a threshold of talliers in $\mathbb{T}$, along with the re-encryption factor and the G-PEQDL proof. If that threshold agree, they jointly generate a signature on (x_f, y_f), and the vote and its signature are placed on the bulletin board.

If Alice's vote *was* coerced, she is sent several DVRPs as before. However, this time, $\mathbb{T}_k$ produces DVRPs based on re-encryptions of votes on the bulletin board that were not Alice's. Note that the DVRPs Alice receives use a key which she freshly generated (to prevent her being identified). Each DVRP is encrypted with a key for which only Alice's TPM has the private part. As a consequence, Alice needs to load the correct key into the TPM (using TPM_LoadKey2), and then requests the TPM to decrypt each DVRP ciphertext, using TPM_UnSeal.

At this point, it should be noted that the keypair k_A generated by the TPM was bound to a certain set of PCR states. If this set of states is not in place at the time of DVRP decryption with TPM_UnSeal, decryption cannot occur. This ensures not only that Alice still uses the same TPM, but also that no rogue software is executed after Alice casts her vote.

Alice can then check to see if any one of the DVRPs represent valid re-encryptions, checking the bulletin board. Note that *every* re-encryption will be on the bulletin board, but only Alice can be convinced that any vote is hers. If she does not find her vote, Alice may contact the Judge, who will contact $\mathbb{T}$. The Judge may further allow Alice to vote again, under his supervision.

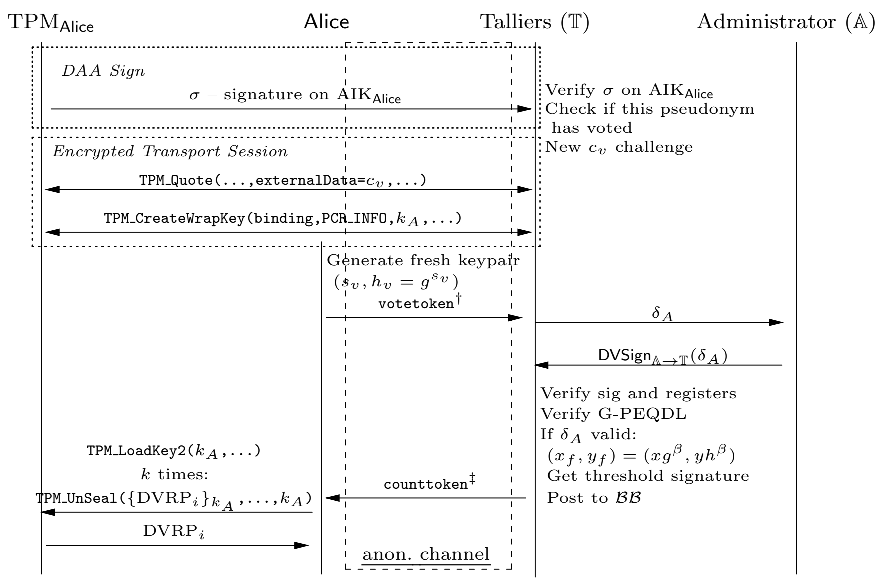
When voting is complete, the product of all encrypted votes is calculated by $\mathbb{T}$ as $(X, Y) = (\prod_{j=1}^{l} x_{f_j}, \prod_{j=1}^{l} y_{f_j})$. This product is calculable by any observer. The final tally is calculated by a quorum (size t) of $\mathbb{T}$ colluding to decrypt this product, 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 every vote is threshold-signed on the bulletin board, observers are convinced that every vote is genuine.

Anonymity Revocation (Fig 3) In [30], we introduced the notion of revocable anonymity in electronic voting: i.e., that a voter could be linked to his ballot when this link was authorised by a Judge. Being able to link a voter to his/her ballot is a legal requirement in the United Kingdom.

The changes that we make to the protocol in Figure 2 in order to provide revocable anonymity are quite simple. We begin with a small change to the registration protocol. Once the DAA *Join* part of the protocol is complete, the registrar $\mathbb{R}$ sends Alice an encryption of her ID with the Judge's public key, $\overline{\mathsf{id}} = \{\mathsf{id}\}_{\mathsf{Judge}}$. $\mathbb{R}$ also sends a signature of this encryption, $\mathsf{Sign}_{\mathbb{R}}(\overline{\mathsf{id}})$ to Alice.

The voting protocol completes the DAA *Sign* protocol as before. Alice then sends the encryption and signature thereof to $\mathbb{T}$, who verify the signature and store the ciphertext. She then extends a TPM PCR with the value of $\overline{\mathsf{id}}$ using TPM_Extend (this is equivalent to hashing the current value of the chosen register, concatenated with $\overline{\mathsf{id}}$). $\mathbb{T}$ can ensure Alice has done this, by ensuring that the value received from TPM_Quote is that which would be expected for a correct machine state *concatenated with* the encrypted ID value.

Voting then proceeds as normal: Alice's identity is re-encrypted by $\mathbb{T}$ and printed on the bulletin board next to her vote. Should revocation be required, a


$\text{TPM}_{\mathsf{Alice}}$
Alice
Talliers ($\mathbb{T}$)
Administrator ($\mathbb{A}$)
DAA Sign
σ – signature on $\text{AIK}_{\mathsf{Alice}}$
Verify σ on $\text{AIK}_{\mathsf{Alice}}$
Check if this pseudonym has voted
New c_v challenge
Encrypted Transport Session
TPM_Quote(...,externalData=c_v,...)
TPM_CreateWrapKey(binding,PCR_INFO,k_A,...)
Generate fresh keypair
$(s_v, h_v = g^{s_v})$
votetoken[†]
δ_A
$\mathsf{DVSign}_{\mathbb{A}\to\mathbb{T}}(\delta_A)$
Verify sig and registers
Verify G-PEQDL
If δ_A valid:
$(x_f, y_f) = (xg^{\beta}, yh^{\beta})$
Get threshold signature
Post to $\mathcal{BB}$
TPM_LoadKey2(k_A,...)
k times:
TPM_UnSeal($\{\text{DVRP}_i\}_{k_A}$,...,k_A)
counttoken[‡]
DVRP_i
anon. channel

† : votetoken $= \langle (x, y) = (g^{\alpha}, h_{\mathbb{T}_v}^{\alpha} g^{M^{i-1}}), \mathbf{G} - \mathbf{PEQDL}, h_v, \delta_A \rangle$
‡ : counttoken $= \langle k_A, \{\text{DVRP}_0\}_{k_A}, \ldots, \{\text{DVRP}_k\}_{k_A} \rangle$, one valid for h_v if δ_A correct

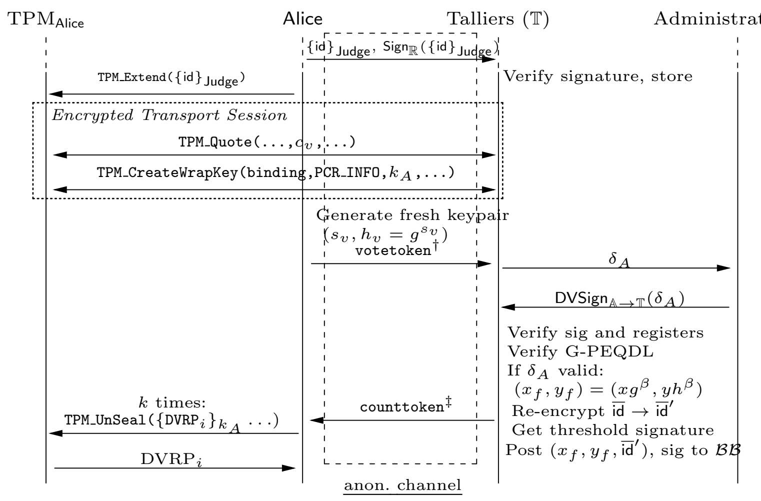
Fig. 2. The Voting Protocol `vote1` (without revocable anonymity). Ellipses suggest standard uses of unchanged TPM commands: only salient parameters are given.

member $\mathbb{T}_k$ of $\mathbb{T}$ sends the tuple $\overline{\mathsf{id}}$ to the Judge, along with appropriate evidence justifying revocation. The Judge is then free to revoke Alice's anonymity and take further action against her. Note that in order to preserve Alice's anonymity, we add a trust requirement that $\mathbb{R}$ does not collude with $\mathbb{T}$ to reveal Alice's identity, and always provides the correct identity for a voter (since $\mathbb{R}$ is trusted to perform the DAA Join protocol correctly, this is not a large increase in trust). Note that Alice could later contact the Judge to determine whether her anonymity had been revoked or not. This does not, to us, constitute full auditability, as Alice needs to contact a third party to audit her vote. We discuss approaches to achieving auditable revocable anonymity in the conclusion.

5 Protocol Properties

For brevity, we do not go into detail about the properties achieved by this protocol (we aim to include a detailed, formal verification in a future version of this work). The protocol satisfies the following requirements:

- **Coercion-Resistance**. Voters cannot prove how they are voting, even if interacting with the voter during voting. The protocol prevents a coercer from determining even if a voter has voted, even when the coercer is physically present
- **Verifiability**. The protocol allows voters to determine that their vote was counted (individual verifiability), and allows all observers to determine that the tally accurately represents all cast votes (universal verifiability). The


TPM$_{\mathsf{Alice}}$
Alice
Talliers ($\mathbb{T}$)
Administrator ($\mathbb{A}$)
$\{\mathsf{id}\}_{\mathsf{Judge}}, \mathsf{Sign}_{\mathbb{R}}(\{\mathsf{id}\}_{\mathsf{Judge}})$
Verify signature, store
TPM_Extend($\{\mathsf{id}\}_{\mathsf{Judge}}$)
Encrypted Transport Session
TPM_Quote(...,c_v,...)
TPM_CreateWrapKey(binding,PCR_INFO,k_A,...)
Generate fresh keypair
$(s_v, h_v = g^{s_v})$
votetoken†
δ_A
$\mathsf{DVSign}_{\mathbb{A}\to\mathbb{T}}(\delta_A)$
Verify sig and registers
Verify G-PEQDL
If δ_A valid:
$(x_f, y_f) = (xg^\beta, yh^\beta)$
Re-encrypt $\overline{\mathsf{id}} \to \overline{\mathsf{id}}'$
Get threshold signature
Post $(x_f, y_f, \overline{\mathsf{id}}')$, sig to $\mathcal{BB}$
k times:
TPM_UnSeal($\{\mathrm{DVRP}_i\}_{k_A}$...)
counttoken‡
DVRP$_i$
anon. channel

† : votetoken $= \langle (x, y) = (g^\alpha, h^\alpha_{\mathbb{T}_v} g^{M^{i-1}}), \mathbf{G-PEQDL}, \delta_A, h_v \rangle$

Fig. 3. Changes to the Voting Protocol `vote2` (with revocable anonymity)

ability to verify that one's vote is counted as cast is generally considered contradictory to one's vote being private (voter privacy) and being unable to prove how one is voting (coercion-resistance).

– **Fairness**. No-one gains any information about the tally until the end of the voting process.
– **Voter Privacy**. No participant can link a voter to their ballot, unless the revocation protocol has been invoked:
  • **Revocable Anonymity**. An authorised entity (or collection thereof) can link a voter to her ballot.
– **Remote Voting**. Voters are not restricted by physical location, providing they have a computer with a TPM and Internet access.

6 Conclusions and Future Work

We have presented an electronic voting protocol providing what the first scheme that uses trusted computing to guarantee the security of the remote voter's machine, whilst also allowing the voter coercion-free remote voting and verifiability, as well as legitimate-voter privacy, and enabling authorities to revoke a voter's anonymity under certain circumstances. Further, the remote nature of our protocol allows voters to cast their votes from any computer with a TPM even with a physically present coercer, providing they can vote unobserved once (a minimal requirement, as otherwise a coercer could always simulate the voter, or trivially suppress her ability to vote).

In the form described in protocol `vote2`, our protocol does not permit genuine *auditable* revocable anonymity. In an ideal scenario, we would like to supply $\mathbb{T}$ with a 'sealed envelope' containing Alice's identity. If the envelope is opened, Alice can see this later. A simple solution involves Alice remaining online throughout the election, her own TPM encrypting and decrypting her identity. In this manner, the TPM would know when Alice's identity was traced, and could inform Alice. If Alice wished for her vote to be counted, she would leave her machine on, rather than risk her TPM being unreachable. Of course, the problem is that a rogue Alice would rather risk turning her machine off and having her vote uncounted than being imprisoned for voting fraud. For us, this is not an acceptable implementation of auditability, and hence we leave this to future work. We note that preliminary work on the 'digital sealed envelope' problem has been recently considered by [1]. Our next step will be to build a prototype implementation of this work in Java, beginning with a TPM simulator.

References

1. Ables, K., Ryan, M.D.: Escrowed Data and the Digital Envelope. In: Trust and Trustworthy Computing 2010. pp. 246–56. Springer Verlag (2010)
2. Alkassar, A., Sadeghi, A.R., Schultz, S., Volkamer, M.: Towards Trustworthy Online Voting. In: Proceedings, WISSec 2006, (2006)
3. Arbaugh, W.A.: The Real Risk of Digital Voting? Computer 37(12), 124–25 (2004)
4. Balfe, S., Lakhani, A.D., Paterson, K.G.: Trusted Computing: Providing Security for Peer-to-Peer Networks. In: Proceedings - Fifth IEEE Conference on Peer-to-Peer Computing. pp. 117–24. IEEE (2005)
5. Benaloh, J., Tuinstra, D.: Receipt-Free Secret-Ballot Elections (Extended Abstract). In: Proceedings, 26th ACM Symposium on the Theory of Computing. pp. 544–53. ACM, Montreal (1994)
6. Blackburn, R.: The Electoral System in Britain. Macmillan, London (1995)
7. Boneh, D., Golle, P.: Almost Entirely Correct Mixing with Applications to Voting. In: Proceedings, CCS 2002. pp. 68–77. ACM, Washington DC (2002)
8. Brickell, E., Camenisch, J., Chen, L.: Direct Anonymous Attestation. In: Proceedings, CCS 2004. pp. 132–45. ACM (2004)
9. Challener, D., Yoder, K., Catherman, R., Safford, D., Doorn, L.V.: A Practical Guide to Trusted Computing. IBM Press, Boston, MA (2008)
10. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a Secure Voting System. In: Proceedings, 2008 IEEE Symposium on Security and Privacy. pp. 354–68. IEEE (2008)
11. Cramer, R., Gennaro, R., Schoenmakers, B.: A Secure and Optimally Efficient Multi-Authority Election Scheme. In: Advances in Cryptology - EUROCRYPT '97. Proceedings. pp. 103–18. Springer-Verlag, Berlin (1997)
12. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: SSYM'04: Proceedings, 13th USENIX Security Symposium. pp. 21–38. USENIX Association (2004)
13. Fink, R.A., Sherman, A.T., Carback, R.: TPM meets DRE: reducing the trust base for electronic voting using trusted platform modules. IEEE Transactions on Information Forensics and Security 4(4), 628–37 (2009)
14. Fujioka, A., Okamoto, T., Ohta, K.: A Practical Secret voting Scheme for Large Scale Elections. In: Proceedings, AUSCRYPT '92. pp. 244–51. Springer-Verlag, Berlin (1993)
15. Harn, L.: Group-Oriented (t, n) Threshold Digital Signature Scheme and Digital Multisignature. In: IEE Proceedings—Computers and Digital Techniques. vol. 141, pp. 307–13 (1994)
16. Hirt, M., Sako, K.: Efficient Receipt-Free Voting Based on Homomorphic Encryption. In: Advances in Cryptology - EUROCRYPT 2000.. Lecture Notes in Computer Science, vol. 1807, pp. 539–56. Springer-Verlag, Bruges, Belgium (2000)

17. Jakobsson, M., Juels, A., Rivest, R.L.: Making Mix Nets Robust for Electronic Voting by Randomised Partial Checking. In: Proceedings, 11th USENIX Security Symposium. pp. 339–53. USENIX Assoc, Berkeley (2002)
18. Jakobsson, M., Yung, M.: Revokable and Versatile Electronic Money (Extended Abstract). In: Proceedings, CCS '96. pp. 76–87. ACM Press, New York (1996)
19. Jorba, A.R., Ruiz, J.A.O., Brown, P.: Advanced Security to Enable Trustworthy Electronic Voting. In: Proceedings, Third European Conference on E-Government. EJEG, Dublin, Ireland (2003)
20. Juels, A., Catalano, D., Jakobsson, M.: Coercion-Resistant Electronic Elections. In: Proceedings, WPES'05. pp. 61–70. ACM, New York (2005)
21. Kügler, D., Vogt, H.: Off-line Payments with Auditable Tracing. In: FC 2002. Lecture Notes in Computer Science, vol. 2357, pp. 269–81. Springer-Verlag, Berlin (2002)
22. Lee, B., Boyd, C., Kim, K., Yang, J., Yoo, S.: Providing receipt-freeness in mixnet-based voting protocols. In: Proceedings, ICISC 2003. Lecture Notes in Computer Science, vol. 2971, pp. 245–58. Springer-Verlag (2004)
23. Okamoto, T.: Receipt-Free Electronic Voting Schemes for Large Scale Elections. In: B. Christianson *et al.* (ed.) Security Protocols Workshop. Lecture Notes in Computer Science, vol. 1361, pp. 25–35. Springer-Verlag (1997)
24. Paillier, P.: Public-Key Cryptosystems Based on Discrete Logarithms Residues. In: Proceedings, Eurocrypt '99. Lecture Notes in Computer Science, vol. 1592. Springer (1999)
25. Paul, N., Tanenbaum, A.S.: Trustworthy Voting: From Machine to System. Computer 42(5), 35–41 (2009)
26. Rössler, T., Leitold, H., Posch, R.: E-Voting: A Scalable Approach using XML and Hardware Security Modules. In: Proceedings, 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service. pp. 480–85. IEEE (2005)
27. Ryan, P.Y., Schneider, S.: Prêt à Voter with re-encryption mixes. In: Computer Security—ESORICS 2006. Proceedings. Lecture Notes in Computer Science, vol. 4189, pp. 313–26. Springer-Verlag, Hamburg (2006)
28. Saeednia, S., Kremer, S., Markowitch, O.: An Efficient Strong Designated Verifier Signature Scheme. Information Security and Cryptology — ICISC 2971, 40–54 (2003)
29. Sako, K., Kilian, J.: Receipt-Free Mix-Type Voting Scheme: A practical solution to the implementation of a voting booth. In: Advances in Cryptology - EUROCRYPT'95. Proceedings. pp. 393–403. Springer-Verlag, Berlin (1995)
30. Smart, M., Ritter, E.: Remote Electronic Voting with Revocable Anonymity. In: Proceedings, ICISS 2009. Lecture Notes in Computer Science, vol. 5905, pp. 39–54. Springer (2009)
31. TCG: Trusted Computing Group: TPM Main: Parts 2 and 3, Version 1.2, Revision 116 (March 2011), http://bit.ly/camUwE
32. Volkamer, M., Alkassar, A., Sadeghi, A.R., Schulz, S.: Enabling the Application of Open Systems like PCs for Online Voting. In: Proceedings of the 2006 Workshop on Frontiers in Electronic Elections—FEE'06 (2006)