Traceable Electronic Voting

Paulo Ferreira

Instituto Superior Técnico, Portugal
pmpf@ist.utl.pt

Abstract

Electronic voting technology enables institutions to deploy anonymous elections over wide networks in ways that secure voter privacy and election robustness while offering a wide range of new benefits that come with modern information systems. We propose the introduction of a new feature to voting systems – a feature we have come to name traceability - that allows votes from the same voter to be related across elections, without compromising voter anonymity. A brief survey is presented to functionally compare modern approaches to electronic voting, and we then focus our attention on one particular approach - blind signatures - for which we establish a traceability mechanism. We describe the resulting protocol and explain its design. Finally, we attempt to discuss both applicability and shortcomings of the system we have designed.

Keywords: E-voting, Electronic surveys, Blind Signatures, Demographics

Resumo

A tecnologia de voto electrónico torna hoje possível às organizações realizar eleições anónimas sobre redes de computadores de pequena e larga escala, mantendo a privacidade do votante e a da segurança da eleição, com a panóplia de mais-valias que advêm dos sistemas de informção modernos. Neste trabalho, propõe-se a introdução de uma nova característica ao voto electronico – a que chamamos rastreabilidade – que permite correlacionar votos do mesmo votante em eleições distintas, sem comprometer o anonimato do votante. Procura-se apresentar um panorama das abordagens de voto electronico que permita uma comparação funcional entre elas e um enquadramento geral. A partir de uma destas abordagens (a que recorre a assinaturas cegas), desenvolve-se então um mecanismo de rastreabilidade Descreve-se o protocolo resultante, procurando explicar os passos envolvidos no desenho desse protocolo. Finalmente, discute-se a aplicabilidade e as limitações do sistema desenvolvido.

Keywords: E-voting, Voto Electrónico, Questionário Electronico, Inquérito Electrónico, Assinaturas cegas

Table of Contents

Table of Figures

1. Introduction

The use of Voting and Polling[1] has been a popular issue for debate over the last decades, due both to the promise of clearer elections, easier voting and lower abstention rates and to the increasingly distributed and digital nature of modern organizations. Many countries have already adopted some national or regional form of electronic voting, and companies now frequently use computerized ways to poll employees and costumers.

Electronic voting initiatives attempt to offer democratic communities the properties of conventional voting (soundness and anonymity) with the added benefits of modern information systems. These added benefits include speed, ease-of-use, the possibility of conducting elections over distributed communication networks and the promise of more transparent, verifiable election processes. Electronic voting systems thus provide the simultaneous delivery of the involved security requirements and of a practical recollection/analysis of results, a combination that is difficult to achieve in conventional elections. All this and the low cost of deployments keeps the adoption count growing and opens new possibilities for democratic decision-making and opinion polling.

There is a broad scope of electronic voting models. They vary on the level of technology they use and the nature of that technology. On one end, there are paper-based solutions that rely only on paper up to the counting step, and only apply computer/electronic technology to scan the bulk of submitted papers and count them. This hybrid model allows nations and other institutions with established voting traditions to perform a smooth transition to electronic voting, obtaining part of the benefits with a minimum of technology shock. On the other end there is full-fledged computer voting, for which everything happens in the digital domain; the voter's interface with the election is through software running on the voter's own home hardware and over a wide area network such as the internet. The latter stretch of the spectrum provides the lowest cost of deployment and the greatest level of flexibility, as no special equipment is required and components may be setup with non-dedicated systems machinery.

The present work focuses on a software-based, networked scenario. We attempt to briefly describe several existing approaches with this profile. Afterwards, we choose one over which to implement a new feature we are about to introduce.
We now proceed to describe this new feature.

---

[1] The terms 'voting' and 'polling' will henceforth be used interchangeably throughout the document with the same semantics, though 'polling' may include systems with less question/answer restrictions, whereas 'voting' might more often than not refer to more strict, multiple-choice based polling. We will also aim to look at processes from a more general, 'polling' perspective, expecting a poll to include free text fields or binary data.

2. Traceability

2.1. Motivation, Example Scenarios

Let us conceive a universe where college students evaluate the courses they take every semester by taking an anonymous authorized poll using a secure e-voting system. Supposing everything works as promised, at the end of every course students submit their evaluations confident that their anonymity is being preserved, and college administration and faculty can read the anonymous results without identifying students. They can elaborate on those results, but only to a limited extent - they can not correlate between different courses.

Let us suppose they find out the student population is divided between those who highly evaluate the importance of mathematics and those who believe mathematics is of little relevance for their curriculum. Let us further imagine that the same happens with physics. It is easy to imagine someone suggesting 'These results mean that some students like theoretical courses, and others don't.' - implying a correlation between answers for the two courses. To prove this, we would have to find a way to trace votes between elections.

As another example, imagine a country where a referendum is taken to legalize light drugs. Shortly after, another one is conduced on the decriminalizing of abortion, and the results are numerically close. Some observers might claim 'Most of the population is being conservative and generally averse to change'. If we could prove, for instance, that the majority of people who voted to reject the second referendum voted approval on the first one, we would have proven this assumption wrong.

2.2. The Concept of Traceability and Traceability Tokens

To establish these correlations between votes in different elections, we propose the introduction of **traceable tokens**, or **pseudonyms**, as identifiers of voter origin that should be separated and impossible to associate with the voter's true identity, but remain constant for each voter throughout elections.
It is a very simple notion, but we have found no literature on the subject. It is likely that work on traceability exists under different naming.

2.3. Traceability (token) Requirements

Consistency: A voter should not be able to submit votes with different pseudonyms in different elections.
Collision-free: Pseudonyms from different voters should not collide.

Secrecy: Pseudonyms must not be stored or accessible alongside data that may reveal voter identity. This means it must never be available to administrators.
Coercion: To prevent proof of vote, pseudonyms must be unknown to voters. If a voter is able to prove his vote, he may be coerced to vote in a predefined way. For this reason, the pseudonym should be handled in a secure and transient way by voter software.

2.4. Levels of traceability

We can think of three different levels of traceability:

1. **Weak**: users can include pseudonym in their votes in different elections, but there is no way to verify they keep using the same token, so voters are given the liberty to choose whether their votes are to be associated. There is also no way to verify that no two users share the same token, so votes from different voters may be incorrectly associated. Under the requirement definitions above, weak traceability systems provide no **consistency**. This level of traceability can be trivially added whenever there is flexibility to provide a free-form field in the ballot: the user simply uses a predefined form to fill in, if he opts to, a pseudonym of his choice, without any verification.

2. **Strong**: users are forced to use their token consistently throughout elections, and that token is guaranteed to be unique.

3. Conventional voting requirements

To successfully design or use an electronic voting system, the requirements expected of the system should be established early on.

We may recognize, among the expected requirements, different requirement scopes:

- **Core Requirements**, those consensually established as intrinsic to the problem of voting
- **Usability Requirements**, those that measure ease-of-use for a system (usually centered on the effort or knowledge demanded of the voter).
- **Operational/Resilience requirements**, evaluating how a system depends on the technological/operational platform, how much demanding it is on resources, and how far it is able to work properly under partial failure of those resources or deliberate attacks on the protocol.

3.1. Core Requirements

Accuracy

A system is said to be accurate if it can assure: (1) No legitimate vote can be ignored in the final tally; (2) No illegitimate vote can be counted as valid; (3) Votes can not be tampered with (changed) after submission.

Democracy

Democracy is understood to be the proper (equal) weighting of votes: (1) All legitimate voters can vote. (2) Legitimate votes are counted only once (all votes have equal weight).

Privacy

The ability of a system to assure: (1) No party can associate a vote with the originating voter (voting is anonymous); (2) Voters may not prove how they have voted.

Special attention should be drawn on this second privacy property. When it is required, the inability of a voter to prove the contents of their vote is intended to avoid the selling/extortion of votes. The problem with this requirement is that it is incompatible with free-text fields in ballots (any unconstrained answering can be used to mark votes and later prove their authorship), and it also often limits verifiability: if users know how they voted but cannot prove it, even if they verify their vote was miscounted they can not prove that either. Systems designed in a way that does not allow for vote proof (as traditional elections have so far been) are also called receipt-free.

Verifiability

Verifiability may be either (1) Universal, so that voters, administrative entities and third-parties can verify each vote was properly authorized and counted for; or (2) Partial, enabling only some of the involved parties (a constituted observer or an administrator) to verify results, or possibly permitting only the verification of part of the results (as when a voter is only able to verify the correctness of his own vote).

Flexibility

The flexibility of a system is understood as the freedom it allows for in the structure/contents of ballots. Flexible systems are those that do not impose any restrictions on the formats of neither questions nor answers.

3.2. Usability Requirements

Ease-of-use

This evaluates how easy it is to vote, including all steps necessary (registering, preparing, understanding the rules and tasks involved, actually voting and verifying results, where possible).

Mobility

Mobility is the ability of a system to take votes from different physical locations.

3.3. Operational Requirements and Robustness

Availability

In a practical implementation, availability refers to: (1) How a system maintains functionality and stability from the beginning to the end of an election; (2) The ability of a system to receive votes from all voters equally and at any time when the election is open.

Collusion Resistance

A system is as resistant to collusion as it is hard to gather enough parties to effectively compromise the core requirements, mostly considering Privacy and Availability. Measurements of collusion resistance are often given in terms of how many entities are necessary to collusion.

Scalability

Scalability measures how well a system applies to larger scales. Scaling is said to be efficient if computational/communication costs are linear to the number of voters.

4. Available (modern) approaches to e-voting

We will now attempt to provide a brief overview of the e-voting landscape and how it may interest us for the development of a traceable e-voting system.
First we give concise descriptions of the available tools used by the various solutions we have come to know of, and afterwards we attempt to describe each in a succinct way. For each one, we will try to summarize its shortcomings and how we see it meet the applicability for our tracing goal.
Afterwards, we try to describe the workings of Blind Signature protocols, as we have chosen that approach to develop traceability on, presenting the interactions involved and the basic operation of each component.
We end by naming the succession of blind signature protocol implementations we studied up to the present moment, providing a brief description and references for each.

4.1. Techniques used in E-Voting

This section presents a brief list of the main building blocks of e-voting protocols. Most of them involve at least intermediate mathematics to be completely understood, but all have a relatively simple use. Some reference pointers are included.

4.1.1. Public key Cryptography

Public key cryptography [RSA78] is the most essential building block in all modern e-voting protocols for both signing and encrypting. Essentially, a user holds a mathematically related public/private key pair, such that a message encrypted with one can only be decrypted by the other. To send a message secretly to a user, it is enough to encrypt it with the user's public key. To sign a message, a user encrypts it with his/her private key (decrypting it with the public key allows anyone to verify authorship).

4.1.2. Anonymous Channels

Anonymous channels allow information to be sent in a way that makes it impossible to trace who sent it. Examples include Onion Routing [Reed96] or remailing systems. Most E-voting protocols assume, unless stated otherwise, that communication is done through anonymous channels.

4.1.3. Bulletin Boards

A bulletin board is any public repository on which anyone (of a given public) can write and read, but can not remove or change information published. This can be used, for instance, to publish public keys (or private keys made public at the end of an election) or election results.

4.1.4. Digital Hashes

A digital hash function provides a way to fingerprint a large block of data in a small number of bytes. Secure hash algorithms are designed so that it is very hard to tamper input in such a way as to keep the same hash output. They are widely used for signing or zero-knowledge proofs (see below).

4.1.5. Zero Knowledge Proofs

Zero Knowledge proofs [GMR85] are protocols/mechanisms to prove knowledge of information without revealing it. Zero Knowledge proving is deeply related to bit commitment. For a soft, entertaining introduction, see [Naor99].

4.1.6. Bit Commitment

Bit commitment schemes are ways to assure information will not be changed by a user without forcing the user to reveal it. An obvious example is a secure hash function: if we have a hash function of some data, we can be sure that data stays untampered under penalty of hash mismatching, even if we do not it.

4.1.7. Homomorphism

Homomorphism, in e-voting, relates to the use of the preservation of an aggregating function in a special data space. The typical case is the resort to a special kind of cipher e for which e(x1) + e(x2) = e(x1+x2). For the first paper on an approach to e-voting based on Homomorphism by Cohen and Fischer, see [CF85].

4.1.8. Mix-Nets

A Mix-Net is a group of servers each of which has the purpose of hiding the correspondence between input and output. In E-Voting, it is used to destroy information on the origin of a vote. For the original introduction to Mix-Nets by David Chaum, see [Cha81].

4.1.9. Blind Signatures

Blind signatures, first purposed by David Chaum in [Cha82], provide a way for a user to sign data it cannot read. To obtain a blind signature, the message author first blinds the message to be signed with a private blinding factor. The signer receives it blinded, and signs it as it would another message. After devolution, the author unblinds the message using the same factor used for blinding, and as a result obtains a valid signature on the original unblinded message. The first proposed use of blind signatures for e-voting in [FOO92] led to a series of variations with well-known implementations, in at chain at the end of each is REVS, on which our prototypical traceable voting system was developed.

The steps to requesting and providing blind signatures go as follows:

A message `m` is prepared for signing by calculating `m'`, which is submitted to the signer: $m' = Blind(X, m)$, where x is a secret blinding factor chosen randomly.

The signer signs the blinded message, producing a blind signature:

$s' = Sign(m')$

Upon receiving this blind signature, the final signature may be calculated by unblinding:

$s = Unblind(x, Sign(m'))$

For RSA, blind signing is available as:

$m' \equiv m \cdot (x^e) \left[\mathrm{mod}\, N\right]$, for any random `x` such that $\gcd(x, N) = 1$

$s' \equiv (m \cdot (x^e))^d \equiv m^d \cdot x \left[\mathrm{mod}\, N\right]$

and finally

$s \equiv x^{-1} \cdot s' \equiv m^d \left[\mathrm{mod}\, N\right]$

Where `x` is the secret blinding factor chosen randomly and (e,d) is a public/private key pair for RSA.

For more information on blind signatures, please see [Cha82].

4.2. Simplistic Voting Protocols

The following protocols do not meet the requirements we listed before, when we considered what an e-voting system should have. They are described merely to illustrate the challenges involved in protocol design without any particular tools.

4.2.1. A One Agency Protocol

In the simplest protocol conceivable, users submit their anonymous votes to a counter. At the end of the election, the counter publishes the results.
Problems: The counter cannot distinguish votes from different users, so anyone can vote. Furthermore, there is no way to assure each voter only votes once. Finally, there is no control that the tallier does not forge votes.
Traceability: Weak traceability is easy to implement by adding an extra free-form question to the ballot.

4.2.2. Another One Agency Protocol

It is easy to modify the previous example so that authorization and accuracy are supported, but with the cost of anonymity. The same protocol is executed with the users signing their votes. Now the tallier can verify votes come from different and legitimate users, but it also knows who voted for whom.
Problems: No anonymity; the tallier can see who votes what. It can also still forge votes.
Traceability: Not applicable.

We can infer from the two previous examples that anonymity and authorization can not be achieved using a single entity – Authentication involves identification, and identification destroys anonymity.

4.2.3. Two Agencies

Separating validation and counting, both anonymity and authorization can be achieved. Users vote by first identifying themselves with a Validator and requesting a unique random id. The Validator only gives one id to each voter. Users then submit their votes, with the unique ids attached, to the tallier. At the end of the election the Validator publishes a list of the given ids, dissociated from the voters identities. The tallier discards votes that are not in that list of votes with used ids, and publishes all vote/id pairs. At the end, users can verify their vote is published.
Problems: The Validator can still forge votes (if votes forged are less than the voters who abstain, this goes undetected), and if Validator and Administrator collude, they can identify votes. Additionally, attacks can be made using random ids to attempt to vote for other people (though this may be arbitrarily hardened by using large data and good random value generation).
Traceability: Traceability could be achieved by the Validator giving the tallier a second token apart from the random value, but then the Validator could use the new token to identify votes.

4.2.4. Two Agencies, late Tally

A variation of the two agency protocol described above includes encryption by the voter on the votes submitted, so that the results are first published and only then decrypted. The voters submit their votes and ids encrypted, the tallier publishes all encrypted votes, and only then do voters publish the decryption keys for their votes, for everyone to decrypt.

Problems: Apart from the problems of the previous protocol, this one includes the inconvenience of introducing an extra step and thus forcing the users to interact twice during the election.

Traceability: Same as for Two Agencies.

4.3. Full Protocols

The following protocols are full-fledged, i.e., they obey all the core requirements presented earlier, and thus represent viable e-voting solutions.

4.3.1. One Agency, ANDOS Distribution

Another variation purposes that a single agency distributes and counts votes, but the distribution is done with a special protocol that provides All-Or-Nothing disclosure (ANDOS) [BCR87], that assures all authorized users receive a valid voting id, and get that id in anonymity.

Problems: ANDOS requires all voters must participate in the distribution of validation tokens (this could be avoided at the cost of an extra election step where abstaining voters identify themselves – but thus breaking part of the privacy property). The validation id also serves as a receipt, so users can prove how they have voted. Another drawback is that ANDOS is said to be computationally demanding.

Traceability: Validation tokens could be repeatedly used throughout elections, but that would only work as long as the electorate didn't change, which is in practice unfeasible but for a rare number of elections.

4.3.2. The Homomorphism Approach

The approaches resorting to Homomorphism rely on the preservation of some aggregating function (typically addition), so that tallying can be performed without decrypting individual votes, and decryption is done on the tallied result. As votes are not individually decrypted, they can safely be signed for authorization and openly published. Some systems require that the voter also submit a zero-knowledge proof that the vote is valid, so as not to disrupt tallying. For references to implementations of Homomorphic approaches to e-voting, please follow the work of Hirt and Sako [HS00] and its developments.

Problems: The foremost problem with this approach is the restrictions it imposes on ballots to allow for a function that preserves tallying on the encrypted votes. Yes/No votes are straightforward, but multiple choice votes are more complex, and it is still unpractical to implement homomorphism on ballots with free text fields.
Traceability: As results cannot be given per vote, but only on the aggregated result, no tracing is applicable.

4.3.3. Mix-net based protocols

Mix-Nets essentially solve anonymity by having voters encrypt their votes successively with the public keys of the various Mix-Servers used and submitting votes to the server with the outermost key used. The last Mix-Server to decrypt a vote publishes it in a Bulletin Board.
For pointers to Mix-Net developments please refer to [JJR02], [PIK93], [SK95] or [OKST97]. [Joa05] includes a richer set of references on Mix-Nets.
Problems: The largest drawbacks with applying Mix-Nets to voting is the complexity of the zero-knowledge proofs required to validate correct behavior on the part of the mix-servers, and the low level of fault tolerance on Mix servers.
As for robustness, a variation has been purposed to tolerate less than a half of the servers to present faults, and work has been recently developed to make Mix-Net protocols more robust and practical.
Traceability: The original Mix-Net protocol proposed by Chaum suggests the inclusion of a pseudonym used to sign ballot but as with the ANDOS approach, traceability is restricted to fixed electorates - or at least growing electorates, as the removal of a voter prevents that pseudonyms can be reused.

4.3.4. Blind Signatures

Blind signature protocols rely on the fact that, with blind signatures, the validating authority does not know what it is actually signing, so the signed information can be submitted for tally and then openly published without compromising the voter's authority. The main problem is then the trust placed in validation - so as not to allow validators to forge votes - but this can be arbitrarily reduced if we distribute this trust among several validators by requiring the signature of a number of them.
Problems: The biggest problem with the blind signature scheme seems so far to be that the blinding factor and the blinded signature returned from the validator can be used to prove the vote. We should note this is only true if the voter can use a self-modified voting module to disclose the blinding factors used (which is commonplace when voting is done remotely). Research is being done at INESC-ID to establish a TCB on the voter module of a voting system to assure the voter must use an official module, and this official version can be designed not to reveal the blinding factors.

Traceability: The same way a validator blindly signs a ballot for the voter, it can sign a blinded version of the voter's traceability token (pseudonym), and refuse to sign two different blinded tokens for the same user, so that the user is always forced to use the first one chosen.

4.4. Choosing Blind Signatures and REVS for Traceability

From the protocols we have covered, it seems the most simple and effective way to alter an existing voting scheme to include traceability is to follow a blind signature protocol in a recent open implementation such as REVS. A simple design change can have Administrators perform pseudonym regulation, and most effort will be put on altering the voter module to encompass the extra interactions and validations.

We therefore used REVS to implement a working prototype of a traceable voting system.

It must be stated that REVS is the platform we had most information on, so our evaluation is based on limited scrutiny[2]. We should also mention that work on Mix-Nets has been developing rapidly, and if a solution is reached that is capable of braking the constraint of the election pseudonym - native to the protocol - to a first election's electorate, Mix-Nets may prove to be a competitive way.
Finally, we would like to emphasize E-Voting is not a stagnant area, so the landscape is bound to change rapidly in a matter of few years (as much in theoretic grounds, with new papers proposing new protocols and solving limitations of current ones, as in practical), and it may be rewarding to reevaluate the choice we have made for the present

4.5. Workings of a Blind Signature Voting Protocol

We next present a simplified version of a blind signature protocol.

For the sake of simplicity, we will skip the step of ballot distribution and the details of election setup. The reader may conceive a public catalog of current ongoing elections and the respective ballots, and that voter authentication is done only at the validation step. The anonymizing step is optional, and may be substituted by an anonymous submission channel from voter to counter. For more comprehensive descriptions and details of practical implementation issues in the case of REVS, please see [JZF03] or [Joa05].

---

[2] The author would be grateful to receive insight on other modern implementations over blind signatures or any other approach.

After election discovery and ballot provision, a voter's filled ballot is validated prior to submission by requiring the blind signature of at least a half plus one of all available administrators. Blinding is done with a random blinding factor that is generated for each blind signature operation (thus for every administrator and every election), and discarded as soon as the administrator's signature is retrieved and unblinded.
The validation of votes requires voter authentication. This could be done by (a) having voters obtaining certificates for their public signatures and presenting these certificates to administrators, or in alternative by (b) setting passwords for each voter-administrator pair. In the case of REVS, the latter option was adopted, and a password generation scheme was used to require the voter to memorize a single password, after which all the administrator-specific passwords are generated, through a digesting scheme.

When the election ends, submissions are gathered among all counters and decrypted. Two different approaches may be taken two ensure that counters can only open votes after the election ends. Either (a) a single entity may be trusted to only publish the elections private key (with which votes are encrypted) after the designated period for the election has terminated[3] or (b) by having the election key pair generate a shared private key[4]. Votes are then validated with respect to the number of administrator signatures they present and the authenticity of those signatures. Duplicate Votes are discarded, and results are published along with all the signatures involved.

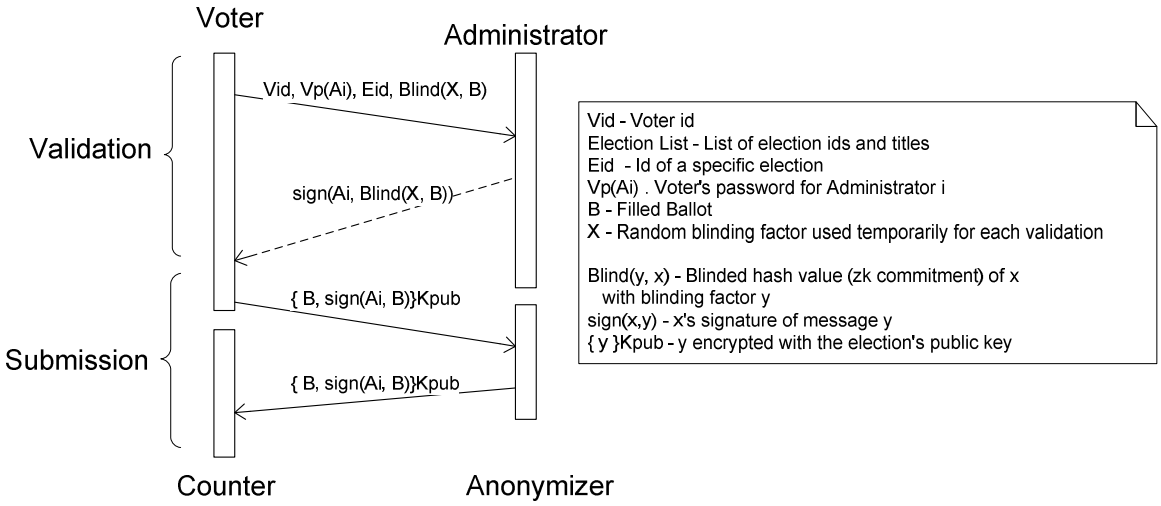

Figure 1 - Simplified diagram of a blind signature protocol

The above diagram attempts to summarize the basic flow of a blind signature voting protocol.

[3] This is the choice in REVS, and consequently in our traceable voting implementation, where the Commissioner is trusted with this responsibility.

4.6. Implementations of Blind Signature Protocols

4.6.1. FOO

The FOO protocol [FOO92] by Fujioka et al. proposed the use of blind signatures in e-voting to guarantee most of the core properties. One of its caveats was that every voter had to be involved in a tallying step of the election, as is discussed at length in [Dur99].

4.6.2. Sensus

Sensus [CC97] was the first real-world implementation of the FOO protocol with a Validator/Pollster/Tallier architecture to run for a real electorate (a college campus).

4.6.3. EVOX

The EVOX system [Her97] was a reimplementation of the same protocol, but removing the extra step necessary at tallying, allowing users to vote and walk away. It also introduced the Anonymizer entity as a middleman between Voter and Tallier.

4.6.4. EVOX-MA

To solve the excess of power placed on the Administrator module (previous systems allowed administrators to forge votes by voting as absent registered voters), Durette [Dur99] introduced with EVOX – Managed Administrators the decentralizing of validation in several parallel entities, requiring that voters obtain several signatures, so that forging votes would require collusion between administrators.

4.6.5. REVS

REVS [JZF03] is an E-Voting system first implemented in 2003 at IST that promises to consolidate the protocol used in EVOX-MA tackling real world operational issues for the Internet environment and hardening the system to both denial-of-service and intrinsic protocol attacks. It provides measures of availability and proofs of correctness. It separates in Distribution, Validation (Administration), Collection (Anonymizing) and Tallying

---

[4] With shared secret key encryption, only with the collaboration of a fraction n of the total t shares (1<n<=t) of the secret key may decryption occur. Systems that use shared secrets for collaborative decryption/singing are referred to as Threshold cryptosystems. For information on threshold implementation of RSA functions, please see [GJKR96b] or [Rab98].

modules. Like EVOX-MA, REVS supports distributed Administration, and introduces the distribution of the other modules (Distributors, Anonymizers and Talliers).

5. Traceability for Blind Signature voting

5.1. Incorporating Pseudonyms into a Blind Signature Voting Protocol

To achieve the traceability goal described earlier, the need arises to incorporate a special token in the protocol that should be published alongside each vote for every election, as it terminates.

To ensure that votes are consistently submitted with the correct pseudonym token, these tokens must necessarily constitute a part of validation, therefore also be blindly signed. To enforce **consistency**, administrators must keep a copy of the pseudonym blind signature requests so that they can refuse to sign a second pseudonym for the same voter. The blinding of pseudonyms for signing provides pseudonym **secrecy** in what regards administrators. To ensure secrecy with respect to the voter and prevent **coercion,** we must ensure that the voter runs in a trusted computing base that does not disclose the pseudonym when it is being operated upon. With this intent of avoiding pseudonym leakage to the voter, we design the whole protocol to not require pseudonym storage. The problem of pseudonym **collision** is addressed by generating the token as a random number within a very large interval[5]. We thus modify the administrator so that, apart from signing votes, it also signs pseudonyms, but only one for each voter. This entails the need for three extra interaction steps for the voter:

(a) a registration step, executed the first time a voter participates in an election - this step consists of registering a randomly chosen pseudonym with all administrators; the following is a diagram of this step:

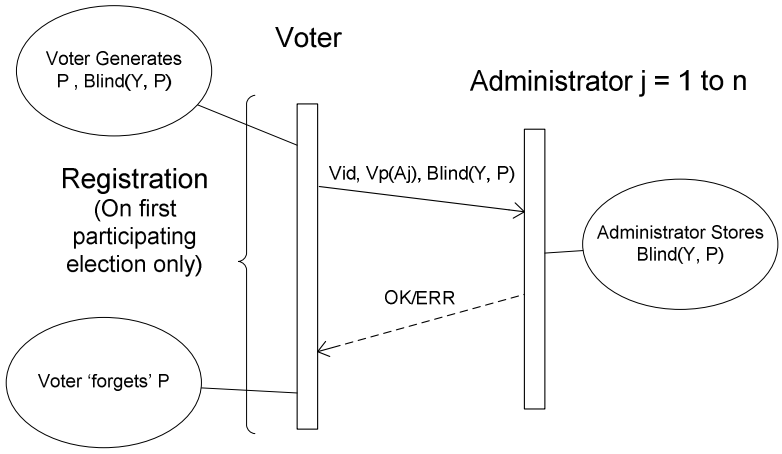

Figure 2 - Registration with an Administrator

[5] In our implementation, the random pseudonym is a number with 128 bits, thus with a random space of 2^128 possibilities.

(b) A pseudonym blind signature request step, executed once for each election and each administrator, whereby a voter obtains the pseudonym's blind signature to include in his vote submission along with the vote signature of that administrator;

(c) A blinded pseudonym request step, executed whenever the need arises to recover a pseudonym that has previously been created for an election. Through this step, the voter can obtain the unsigned, blinded pseudonym previously submitted, from which he can reconstruct the plain pseudonym[6] that must be part of the vote.

Step (c) serves a twofold purpose:
To recover voting pseudonyms after the first election, if the voter does not store the pseudonym, as may be required to prevent coercion.
To perform registration with an administrator by using a reconstructed pseudonym obtained from other (previously registered) administrators. This is only required when a new administrator registration is needed for a previously created pseudonym. This happens when the voter meets an administrator it could not register in the original pseudonym creation and registration step, either because that administrator was newly deployed, or because some availability problem prevented the voter to interact with that particular administrator before.

Finally, the blinding factors used when blinding pseudonyms differ from those used for vote signing in that they must not change for all elections, to maintain coherence with the blinded versions stored by administrators. To solve this challenge in practice, our prototypical implementation derived from REVS uses a second password that generates blind factors for all administrators that differ from each other, but are consistent across elections. The scheme used to generate the blind factors from this second password is the one used by revs to generate administrator passwords from a single password, and is described in [JZF03].

The following diagram portrays the complete voting sequence, apart from step (c) (which, if used, is only present after the first election).

[6] For this reason, instead of using the conventional technique substituting data to be signed by a zero-knowledge commitment of the data, such as a hash value, the pseudonym is itself submitted for signing, and afterwards signed.

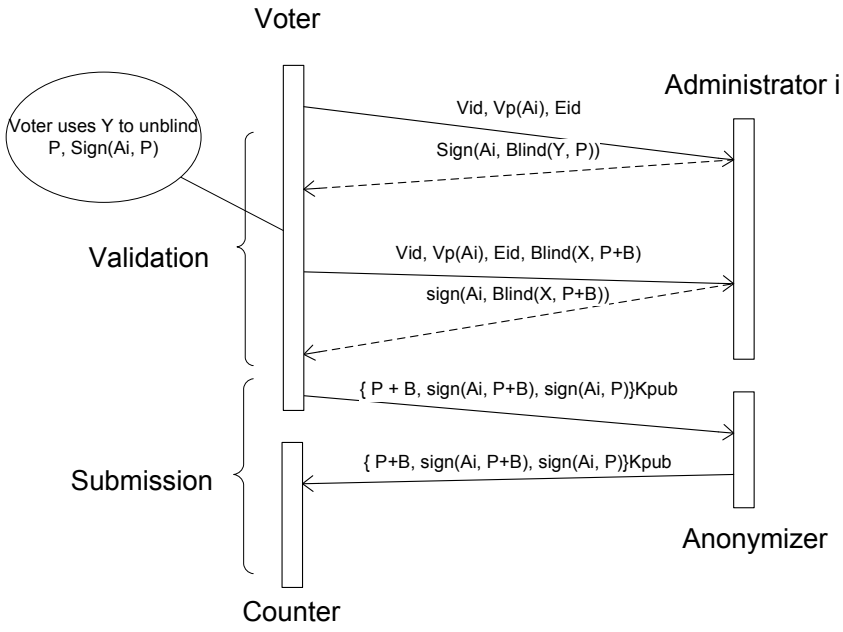

Figure 3 - Summary of the Traceable Voting Protocol

As we can more easily recognize in the above diagram, the final submission package includes everything required to validate the vote, the pseudonym, and both as a pair.

Packages are submitted as described earlier for a non-traceable protocol, and counting is done in a likely manner, with the exception that pseudonym signatures must also be validated, and in sufficient number. Also, and most importantly, the published results now include pseudonyms alongside votes, as we intended.

6. Traceable REVS: The REVS software package and changes in the Traceable version

This section very briefly walks through some implementation aspects of Traceable REVS, only insomuch as it is necessary to understand the effort of altering it to implement traceability.
For a detailed description of the inner workings and implementation choices in REVS, the reader is encouraged to refer to [Joa05].

6.1. Architecture

As required by the protocol, REVS modules run multiple instances of each kind of server that voters may access, according to protocol sequence. The architecture of a running complete REVS deployment is as follows:

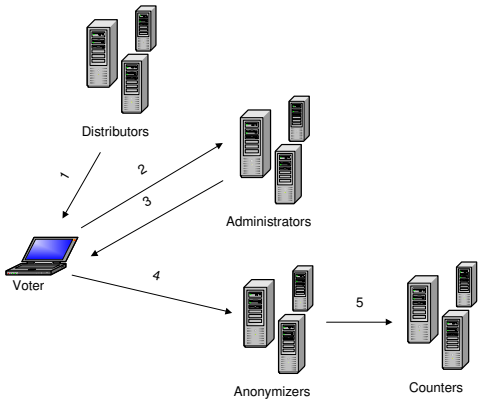

Figure 4 - The REVS Architecture

6.2. Technology

REVS is written in the Java Language, version 1.4.

- In addition to the common utility packages of the Java runtime API, the following software is used:
- **JBCL** – The JBuilder JavaBean Component Library, a set of GUI components built on top of swing
- The **logi.crypto** cryptographic library, used in RSA key manipulation.
- The **java.security** API, both for digesting and keystore functionality.
- **java.rmi** and **javax.net.ssl** for secure communication.
- The **Xerces** DOM Parser for XML manipulation.

To provide persistence, REVS relies on

- The **JDBC** API and the **JDBC MySQL driver** for database access from the application.
- The **MySQL** Database server for the persistence backend.

6.3. Software Design

6.3.1. Component Structure

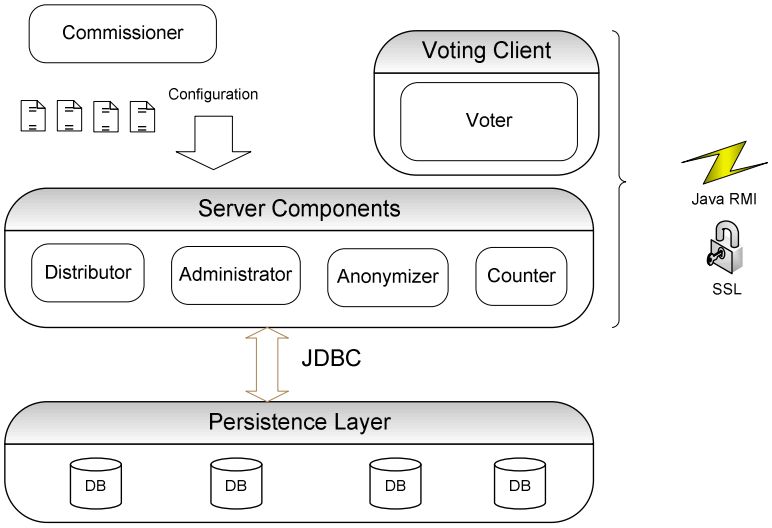

Figure 5 - REVS Component Structure

A clear distinction is drawn, in REVS, between the server and the client layer, as server components share much of their functionality. The database layer is also kept apart: each server instance accesses its own database repository through a common set of database abstraction code using **JDBC** to connect with the database management system (currently MySQL, but this should be flexible as JDBC is designed for database independence and REVS requires no vendor-specific functionality).

A **Commissioner** application is used to generate configuration files that regulate the behavior of all the other components.

6.4. Implementation and Changes on the Server Layer

As much of the functionality required for server operation is shared, the classes implementing the server layer follow a carefully designed hierarchic structure to promote code reuse.
The following class diagram of the server layer attempts to illustrate this hierarchy.

We include only the most meaningful methods for each class.

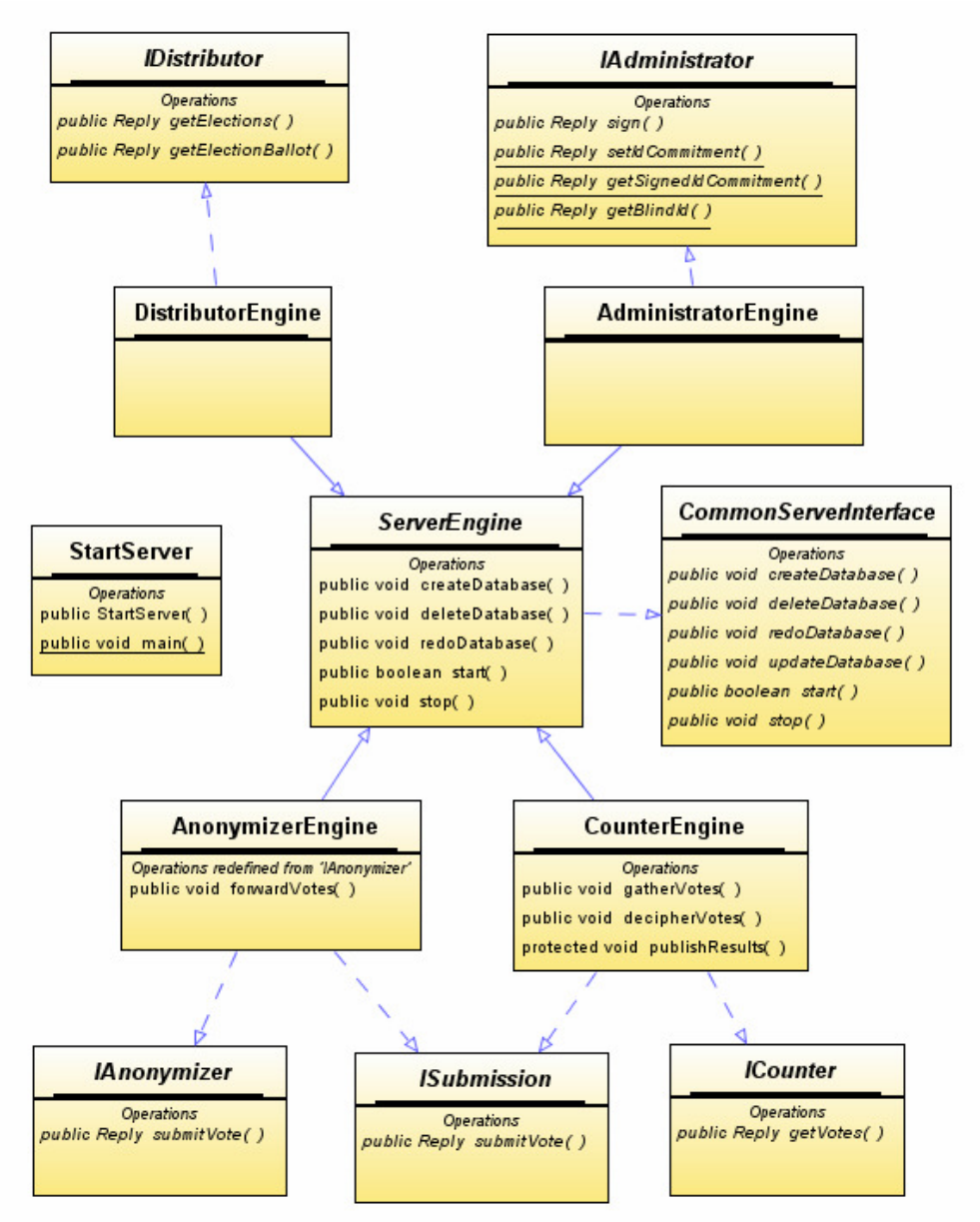

Figure 6 - REVS Server Layer Classes

6.4.1. Administrator Changes

The three underlined methods in the *IAdministrator* interface indicate the interface changes required to implement traceability, as described earlier.

6.4.2. Counter Changes

The Counter code also had to be changed, but without altering its method prototypes. Changes were applied to achieve:

- **Correct vote validation**: Validation now depends on the correctness and consistency of the pseudo-Id signatures for each administrator.
- **Pseudo-id publishing**: The published XML files with the answers from each voter now have to include the pseudo-Id of the corresponding voter. Thus, counter code was changed so that the pseudo-Id is appended as an attribute to the ballotAnswer node in Base64 encoded form.

6.5. Database Layer Changes

Changes are also needed in the database layer, as the administrator now is required to hold additional storage for each user's pseudonym.

The two methods in the DB_Administrator class are the methods added for our traceability goal – simply set or get the blind pseudonym from the voter row in the voter table of the administrator's database.

The DDL statements in the SQL code for the creation of the Administrator tables were also altered to reflect the added administrator column for the Id commitment.

In the following diagram we see the REVS classes responsible for handling database access (by using the JDBC API to send predefined queries to the database). It is interesting to note how there is a class of each of the different server types, and how two other classes are used to group database functionality: Anonymizers and counters don't have to hold voter information, whereas distributors, commissioners and administrators do.

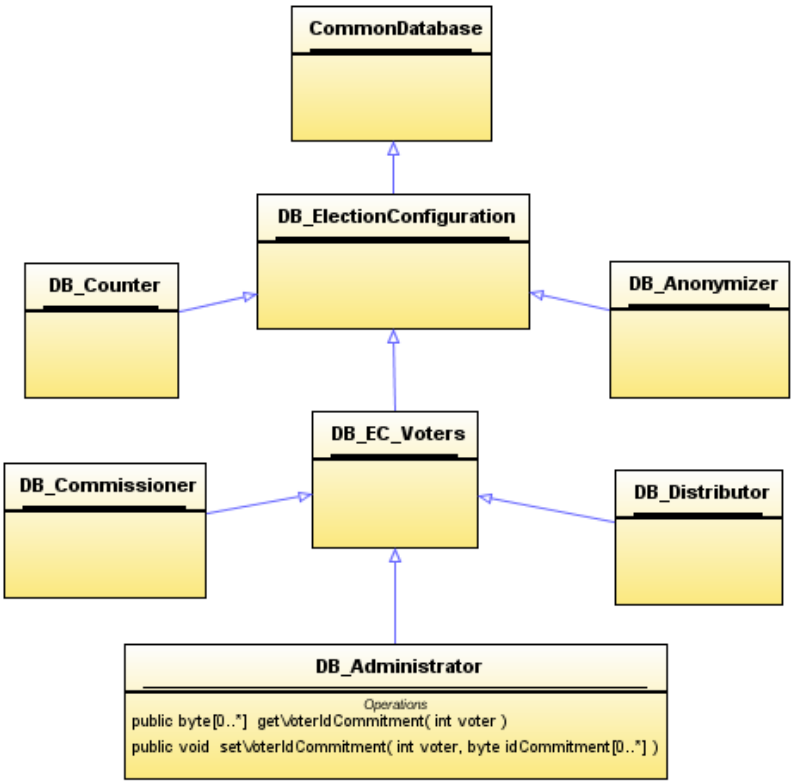

Figure 7 - The REVS Database Layer

In the future, is traceability is to be incorporated in the main REVS package, then the database layer will also have to be modified so that the election configuration data model also includes information on whether elections are to be traced, and which.

6.6. Changes in the client layer

6.6.1. Commissioner

No changes were required on the commissioner. The module is kept intact from non-traceable REVS.

6.6.2. Voter

The voter modules required most of the modification work.

The following diagram attempts to describe the voting process, and how pseudonym management code is introduced.

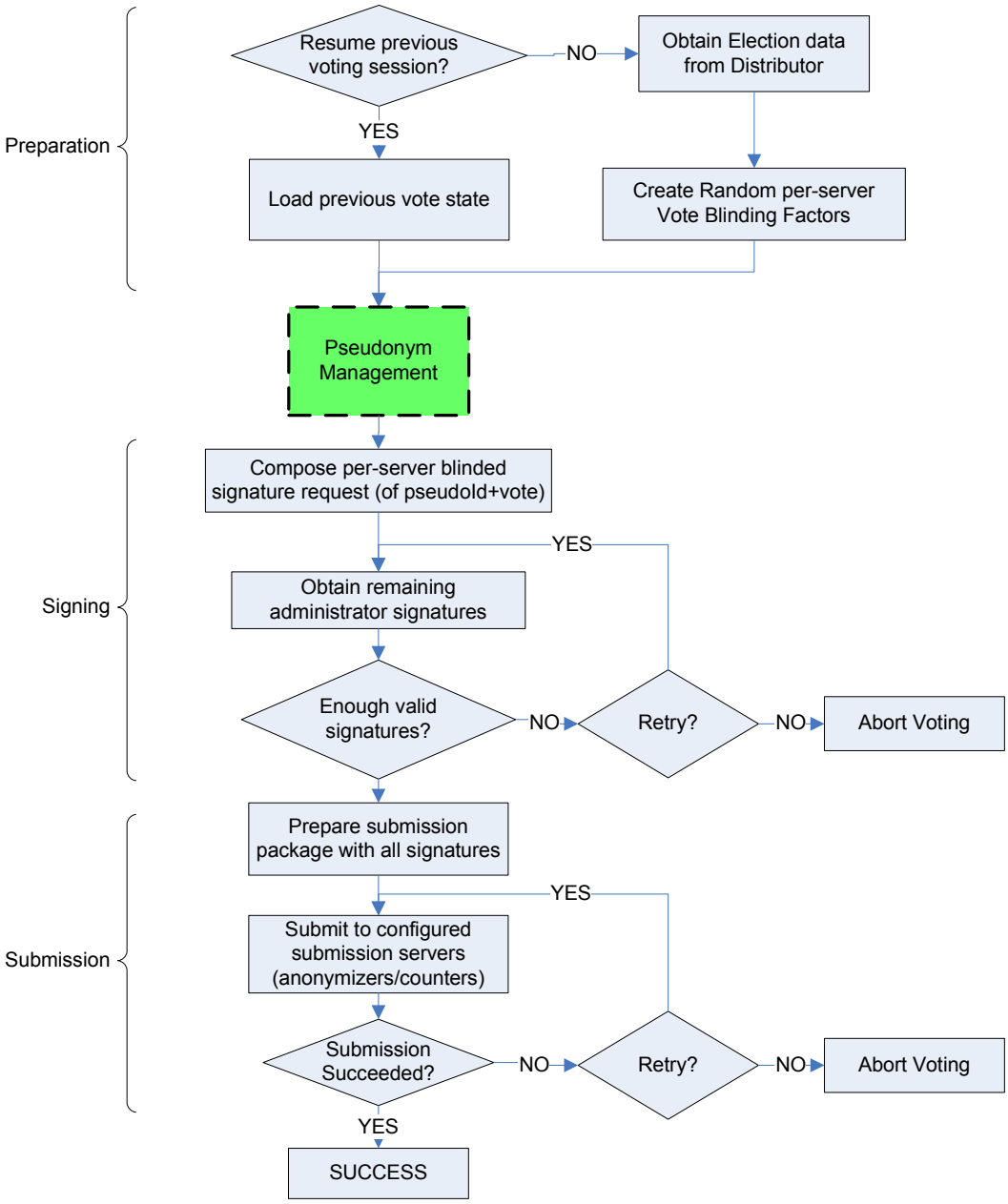

Figure 8 - The voting client process flow

The preparation and submission parts of the voting code were essentially kept intact, with minor changes as to the contents of the messages exchanged.

The pseudonym management block is next presented in more detail.

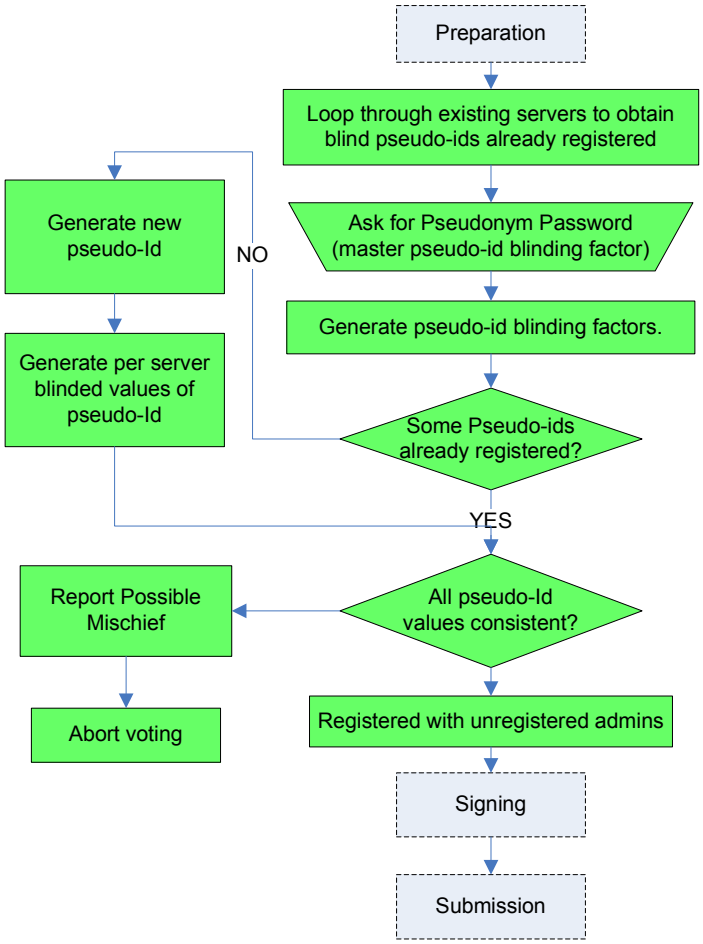

Figure 9 - The pseudonym management process required in the client code

In the above diagram we see the basic flow followed in the voting modules to handle pseudo-Ids. We use the terms “pseudonym” and pseudo-Id to distinguish the pseudonym perceived by the user from the actual pseudonym token published with the election results. Here, “pseudonym” is actually the pseudonym password used to generate the per-server pseudo-Id blinding factor.

7. Requirement Evaluation

We now attempt to review the protocol with respect to the voting requirements stated earlier, both those for regular voting and those for traceability. We try to mention, for each requirement, how it could be attacked (or generally how it could fail), and the effort for that attack.

7.1. Voting Requirements

Accuracy

(1) No legitimate vote can be ignored in the final tally.

To have a vote ignored, an attacker must obtain the collusion of all counters (or of the majority of administrators). One thing that changes in this respect is that a new attack is made possible by traceability. If an attacker has the voter's password before the first election, he can not only vote for the victim but also, by registering his own pseudonym, prevent the victim from using the administrator in the future.

(2) No illegitimate vote can be counted as valid; For a vote to be valid, it requires the signatures of more than half of the available administrators. As such, unless that many administrators cooperate in foul play, it is vote that is impossible for an attacker to submit and have counted a vote without obtaining all the signatures, which can be made only once per election per voter.

(3) Votes can not be tampered with (changed) after submission. If an attacker (a counter) tampers with a vote, he is invalidating the administrator signatures, and thus invalidating the vote. We then fall in the first of the two situations described above in (1) – without the cooperation of all counters, the attack fails.

Democracy

(1) All legitimate voters can vote. Attacking this requirement can only be made by either (a) Hindering availability of administrators, anonymizers or counters or (b) have those servers conspire to deny voting to a group of voters (or all). As with REVS and EVOX-MA, the number of servers can be arbitrarily enlarged, and the servers spread on different points of the network, so both denial-of-service attack damage and the risk of collusion can be minimized as need be.

*(2) Legitimate votes are counted only once (all votes have equal weight).*By the design of REVS, repeated votes are discarded, not counted twice, so vote weighting is not possible through vote injection. The only possibility of a voter to vote by someone else is to hold their private password and pseudonym password.

1. Privacy

(1) No party can associate a vote with the originating voter (voting is anonymous)

As long as submission is done through an anonymous channel (i.e., there are no identifiable traces in the networking from voter to anonymizer/counter), the only way to identify voters is by registering the order by which votes are cast, and only for sparse elections (that is, where votes are separated in time), can anonymity be at risk, and only by a collusion of administrator, anonymizer and counter. To perform such an attack, anonymizer and counter must restore the order of submissions and compare it to the signing order of a colluding administrator.

A second aspect is to be considered though, as traceability is introduced – that of voter narrowing. For elections sets where all elections are universal (i.e. all voters can vote), privacy is fully kept. For elections where voters are grouped into groups that are publicly known, and that control which election voters participate in, an attacker could analyze pseudonym information and census data and narrow the voter identity possibilities for a particular set of votes. We will mention this again ahead in section 8.2 (voter narrowing).

(2) Voters may not prove how they have voted.

REVS is not receipt-free, so users may proof their votes by taking hold of either (a) the blinding factors or (b) their pseudonyms, both of which can be used as proof of vote. As mentioned above, to avoid this requires that the voter module run in a trusted computing base that prevents the voter from modifying software behavior, and revokes him access to both pseudonym and blinding factors.

Universal verifiability

Voters, administrative entities and third-parties can verify each vote was properly authorized and counted for. This is held valid. At the end of an election, the results are published, and for each vote anyone must be able to see the pseudonym, the pseudonym signature, the vote and the signature of the compound

$$p, b, s_i(p+b), s_i(p)$$

with the sufficient number of signatures, one for each administrator i.

Flexibility

We understand the resulting system to be flexible, as it imposes no particular restrictions on the format or content of ballots.

7.2. Usability Requirements

Ease-of-use

Though voting - where usability is most important - is relatively straightforward for voters in REVS, election configuration and setup is still an effort-consuming step, and some technical understanding is required to start an election

Mobility

The system, as REVS, is inherently mobile, as it is conceived to permit voting over arbitrarily large networks, and poses no restrictions on voter position on the network. It relies on the networking layers to provide this, so voter mobility is fundamentally the network mobility available.

7.3. Operational Requirements and Robustness

Availability

(1) How a system maintains functionality and stability from the beginning to the end of an election. This is provided, as servers may be replicated arbitrarily to prevent denial of service attacks, and there is no central vulnerable target to halt or bog down the complete protocol.
(2) The ability of a system to receive votes from all voters equally and at any time when the election is open. We see no discriminating vulnerability that could reduce system availability to some users.

One issue that should be mentioned at this point, and kept in mind when deploying in elections in wide (or otherwise unsupervised) networks, is that the network itself is vulnerable to attacks, so an attack on user connectivity may lock out individual voters or whole voter subnetworks. We do not consider this as a limitation or particularity of the protocol, but instead as a fundamental consequence of network use to security-sensitive computing that should be always kept in mind.

Collusion Resistance

Collusion resistance can be made arbitrarily high by enlarging the number of servers involved, and the number of administrator signatures required accordingly.

Scalability

We consider REVS, and our variant, scalable with respect to the size of the voting population. In what regards servers, however, we believe scalability could prove problematic as the number of administrators enlarging increases the number of interactions necessary to vote, and thus hinders availability and the affects the time required to vote.

8. Limitations and possible attacks

8.1. Prevented Attacks

Administrator Collusion

Obtaining control of a majority of administrators (over half), an attacker could deny voting, by either not responding to disfavored voters or returning false blinded pseudonyms (so the voter cannot retrieve a previously registered pseudonym

8.2. Voter-narrowing in non-universal elections

There is a theoretical, fundamental caveat to traceability for scenarios where voters are not required to vote in every election: just by analyzing the results of successive elections, and data on who participated in what elections, pseudonyms can be successively narrowed down to smaller sets of possible voters, and ultimately to a single voter.

This is especially problematic in the frequently occurring scenario where voters are grouped in publicly known subsets, and these subsets determine which elections voters are part of. Consider, as a concrete example, the student course evaluation setting described in the introduction. If students participate only in courses they take, and course enrollment is free to a significant degree (i.e., students choose their own curricula), it might likely be the case that there is only one student with a particular enrollment history. If that history is public, anyone may compare it with election results and identify the student with the set of votes. As an alternative, you may imagine a nation-wide election system where citizens vote for state/district elections, and people move often. Election registration, if public, may be compared with results throughout the years, narrowing down voter identity.

8.3. Cryptographic limitations

Any cryptographic protocol is subject to the limitations of the cryptographic schemes it uses. In the case of blind signature voting this means special attention should be given to the choice of hash, signing and encryption functions to use. REVS choices (1024bit RSA, SHA1) are at present trusted enough for their widespread use in

security critical industry applications to continue, but new implementations should choose up to date standard functions.[7]

8.4. Ease of use

We find the ease of use should be significantly improved. As with REVS election deployment involves a number of steps that could be made easier with a configuration/setup script. At the moment, a number of command line scripts have to be run with different concerns in order to get REVS running. This is however expected to fade away as new versions and variations of the implementation are produced.

8.5. Vote resuming versus Pseudonym storage

REVS currently supports the possibility of vote saving and resuming allowing the voter to collect parts of the signatures in distinct moments in time. This has been kept active in the current implementation of the traceable REVS, but we must bear in mind that it may lead to pseudonym disclosure to the voter (and eventually vote proof) if the voter modules run under a non-trusted computing base that gives the voter access to the saved voting state. This is somehow inevitable, as the voting state must include administrator signatures that are unique to the voter's pseudonym. In the end, a choice must be made between:
(1) providing a completely trusted computing base for the voter module (where the saved vote is inaccessible to a malicious voter), which permits secure vote saving and eradicates proof of vote
(2) yielding the proof of vote to voters who either recover the pseudonym from the saved state or who inspect voter execution and locate the pseudonym (and the vote blinding factor) used.

[7] In fact, SHA1 remains safe for practical use in 2007, the strong attacks on SHA0 and the recent discovery of an attack faster than brute force (with 2^63 hash operations) on SHA1 [WYH05] has made several agencies recommend that new applications choose the new SHA2 succeeding variants in detriment of SHA1.

9. Possibilities of further development

9.1. Merging with other REVS projects

The most immediate advance to expect is the integration of our traceable version of REVS into REVS projects (REVS, MobileREVS and other work being developed), so that the same software could produce:
(a) Traceable and non-traceable elections, according to a per-election configuration option;
(b) Elections with optional tracing (where voters could either include their pseudonym or omit it).
We expect this merge to hold no great effort or theoretical challenges.

9.2. Pseudonym storage

If either proof of vote is allowed or a safe pseudonym storage mechanism is available, a cumbersome part of the protocol – retrieval of pseudonym from blinded versions stored at administrators - could be skipped. Though it would serve no great purpose to completely exclude from the software the possibility of storageless traceable voting, having the option of pseudonym storage could avoid leaving to the user the responsibility of deciding whether to generate a pseudonym (first-time process), as is done now.

10. Ending Note

We find to have successfully developed a viable e-voting traceability mechanism without significant shortcomings (for as long as the cryptosystem may, as at present, be made arbitrarily secure), and expect it to be applicable to a large number of election and survey scenarios, where voter narrowing is not a major problem. In those cases, we expect that vote tracing will bring added value to statistics, and through that enhance the role of electronic voting in democratic communities and community-concerned institutions.

11. References

[RSA78] Rivest, R. L., Shamir, A., and Adleman, L. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (Feb. 1978), 120-126.

[Cha81] Chaum, D. L. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* 24, 2 (Feb. 1981), 84-90.

[Cha82] D. Chaum. Blind signatures for untraceable payments. In *Crypto'82*, pp. 199-203. New York: Plenum Press, 1983.

[CF85] Josh D. Cohen (Benaloh) and Michael J. Fischer. A robust and verifiable cryptographically secure election scheme. In Proc. 26th IEEE Symposium on the Foundations of Computer Science (FOCS), pp. 372{382. IEEE, 1985.

[GMR85] Goldwasser, S., Micali, S., and Rackoff, C. 1985. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on theory of Computing* (Providence, Rhode Island, United States, May 06 - 08, 1985). STOC '85. ACM Press, New York, NY, 291-304.

[BCR87] Brassard, G., Crépeau, C., and Robert, J. 1987. All-or-nothing disclosure of secrets. In *Proceedings on Advances in cryptology---CRYPTO '86* (Santa Barbara, California, United States). A. M. Odlyzko, Ed. Springer-Verlag, London, 234-238.

[FOO92] Fujioka, A., Okamoto, T., and Ohta, K. 1993. A Practical Secret Voting Scheme for Large Scale Elections. In *Proceedings of the Workshop on the theory and Application of Cryptographic Techniques: Advances in Cryptology* (December 13 - 16, 1992). J. Seberry and Y. Zheng, Eds. Lecture Notes In Computer Science, vol. 718. Springer-Verlag, London, 244-251.

[PIK93] Park, C., Itoh, K., and Kurosawa, K. 1994. Efficient anonymous channel and all/nothing election scheme. In *Workshop on the theory and Application of Cryptographic Techniques on Advances in Cryptology* (Lofthus, Norway). T. Helleseth, Ed. Springer-Verlag New York, Secaucus, NJ, 248-259.

[SK95] K. Sako, J. Killian, Receipt-free mix-type voting scheme-a practical solution to the implementation of a voting booth. In *Advances in Cryptology-EUROCRYPT'95*, Springer-Verlag, Berlin, 1995, pp. 393-403.

[Reed96] Reed, M. G., Syverson, P. F., and Goldschlag, D. M. 1996. Proxies For Anonymous Routing. In *Proceedings of the 12th Annual Computer Security Applications Conference* (December 09 - 13, 1996). ACSAC. IEEE Computer Society, Washington, DC, 95.

[GJKR96b] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. "Robust and Efficient Sharing of RSA Functions "; Appeared in CRYPTO'96.

[CC97] Cranor, L. and Cytron, R., 1997. Sensus: A Security-Conscious Electronic Polling System for the Internet. Proc. of the Hawaii International Conference on System Sciences. Wailea, Hawaii.

[Her97] Herschberg, M., 1997. Secure Electronic Voting Using the World Wide Web. MIT Ms.C thesis.

[OKST97] Ogata, W., Kurosawa, K., Sako, K., and Takatani, K. 1997. Fault tolerant anonymous channel. In *Proceedings of the First international Conference on information and Communication Security* (November 11 - 14, 1997). Y. Han, T. Okamoto, and S. Qing, Eds. Lecture Notes In Computer Science, vol. 1334. Springer-Verlag, London, 440-444.

[Rab98] Rabin, T. 1998. A Simplified Approach to Threshold and Proactive RSA. In *Proceedings of the 18th Annual international Cryptology Conference on Advances in Cryptology* (August 23 - 27, 1998). H. Krawczyk, Ed. Lecture Notes In Computer Science, vol. 1462. Springer-Verlag, London, 89-104.

[Dur99] DuRette, B., 1999. Multiple Administrators for Electronic Voting. MIT Bs.C thesis.

[Naor99] M. Naor, Y. Naor, O. Reingold, Applied Kid Cryptography or How To Convince Your Children You Are Not Cheating, Weizmann Institute of Israel Web, 1999

[HS00] M. Hirt and K. Sako. Efficient Receipt--Free Voting Based on Homomorphic Encryption. In EuroCrypt'00, 2000, 539--556. Springer-Verlag. LNCS Vol. 1807.

[JJR02] M. Jakobsson, A. Juels and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking. 2002. In *Proceedings of the 11th USENIX Security Symposium.*

[VT02] I. Rosner, G. Rosner. 2002. Electronic Voting Protocols and Schemes. Advanced Topics in Computer Security. Hebrew University of Israel.

[JZF03] R. Joaquim, A. Zúquete and P. Ferreira. REVS – A Robust Electronic Voting System. IADIS International Journal of WWW/Internet - IADIS Press, ISSN 1645-7641, December 2003.

[Joa05] R. Joaquim. A Fault Tolerant Voting System for the Internet. 2005. Masters Thesis. IST

[WYH05] X. Wang, Y. L. Yin, and H. Yu. *Finding collisions in the full SHA*-1. In Crypto 2005.