Traceable Electronic Voting

Paulo Ferreira

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

Abstract
Electronic voting technology enables institutions to deploy anonymous polling over wide networks in ways that secure voter privacy and election robustness. We take a modern e-voting system that makes use of blind signatures - REVS - and modify it with a scheme that permits observers to relate votes from the same voter across elections, without compromising voter anonymity. The resulting protocol is described, and its design very concisely explained.

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

1 Introduction

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 include speed, ease-of-use, the possibility of conducting elections over distributed communication networks and the promise of more transparent, verifiable election processes. Low-cost deployment of electronic voting opens new possibilities for democratic decision-making and for opinion polling, simultaneously delivering voter privacy and easy recollection/analysis of results, a combination that is difficult to achieve in conventional elections.

An approach to digital signing proposed by Chaum [Cha82] uses a blind signature mechanism that enables authorities to sign (validate) secrets without their disclosure, and in such a way that prevents signers from identifying published signed content with signature requests. This blind signature mechanism was applied to e-voting, resulting in a family of increasingly evolved polling protocols developed throughout the last two decades ([FOO92] [CC97], [Her97], [Dur99], [JZF03]).

This article presents a variation on one of the later protocols - REVS [JZF03] - that introduces the possibility of marking votes from the same voter in different elections with a common token that should be impossible to relate to the voter's identity. This token – a pseudonym - enriches election results by enabling statistical correlations of successive polls, establishing voting trends across votes. As a concrete example of this, let us imagine that a university decides to use electronic polling of students to evaluate faculty and course programs. In this case, apart from the ratings students give in their responses, the pseudonym token makes it possible for the institution to establish and take into account correlations between results for different courses (for instance, whether students tend to evaluate courses of the same department in a consistent manner).

2 A simplified overview of Blind Signature Voting

2.1. Blind Signature Schemes

A blind signature scheme is one that permits signature of blinded messages – that is, data may be blinded (altered) according to a blinding factor in such a way that: (a) It is not possible to recover the original message from blinded data without knowing the blinding factor chosen for blinding. (b) A signature of the data, when unblinded, yields a plain valid signature of the original unblinded message.

Thus, the steps to requesting and providing blind signatures go as follows:

1. 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.
2. The signer signs the blinded message, producing a blind signature: $s' = Sign(m')$
3. 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:

$$m' \equiv m \cdot (x^e)[\bmod N]\text{, for any random x such that } \gcd(x, N) = 1$$

$$s' \equiv (m \cdot (x^e))^d \equiv m^d \cdot x[\bmod N]$$

and finally

$$s \equiv x^{-1} \cdot s' \equiv m^d [\bmod N]$$

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].

2.2. Blind Signature Voting Protocols

We next present a very 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].

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 (for which votes are encrypted) after the designated period for the election has terminated[1] or (b) by having the election key pair generate a shared private key[2]. 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.

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

[2] 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].

The following diagram attempts to summarize the basic flow of a blind signature voting protocol:

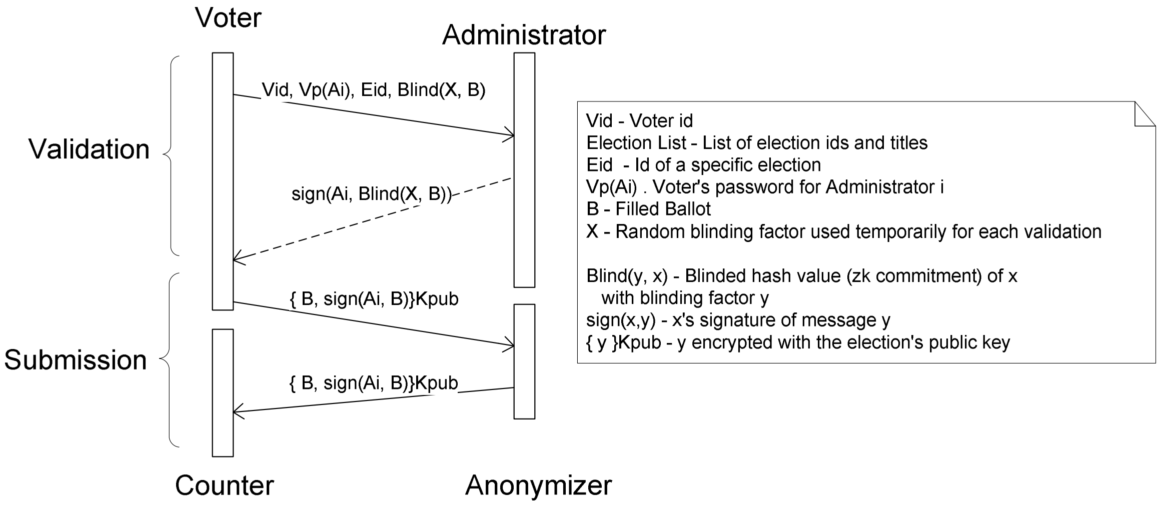
Voter
Administrator
Validation
Vid, Vp(Ai), Eid, Blind(X, B)
sign(Ai, Blind(X, B))
Submission
{ B, sign(Ai, B)}Kpub
{ B, sign(Ai, B)}Kpub
Counter
Anonymizer
Vid - Voter id
Election List - List of election ids and titles
Eid - Id of a specific election
Vp(Ai) . Voter's password for Administrator i
B - Filled Ballot
X - Random blinding factor used temporarily for each validation
Blind(y, x) - Blinded hash value (zk commitment) of x with blinding factor y
sign(x,y) - x's signature of message y
{ y }Kpub - y encrypted with the election's public key

Figure 1 - Simplified diagram of a blind signature protocol..

3 Traceability and Pseudonym Tokens

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.

Requirements of Pseudonyms

(1) **Consistency**: A voter should not be able to submit votes with different pseudonyms in different elections.
(2) **Collision-free**: Pseudonyms from different voters should not collide.
(3) **Secrecy**: Pseudonyms must not be stored or accessible alongside data that may reveal voter identity. This means it must never be available to administrators.
(4) **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.

Incorporating Pseudonyms into the Voting Protocol

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[3]. This is not an absolute solution in the sense that it does not completely remove the possibility of collision, but instead reduces that possibility to a value without practical relevance.

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:

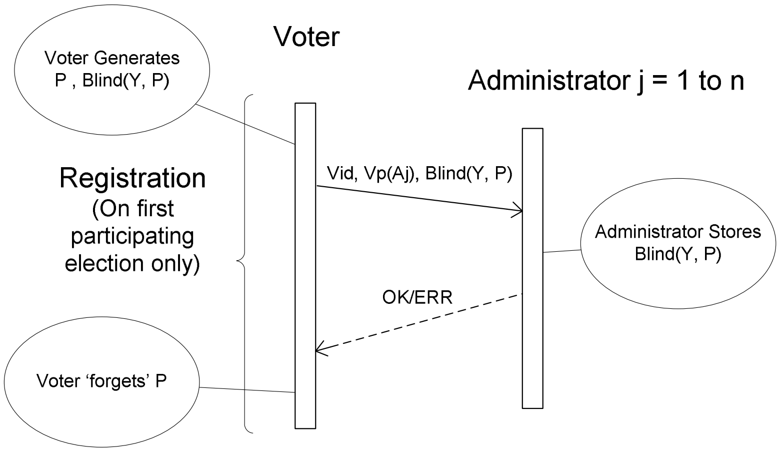
Voter Generates P , Blind(Y, P)
Voter
Administrator j = 1 to n
Registration (On first participating election only)
Vid, Vp(Aj), Blind(Y, P)
Administrator Stores Blind(Y, P)
OK/ERR
Voter 'forgets' P

Figure 2 - Registration with an Administrator

(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[4] that must be part of the vote.

Step (c) serves a twofold purpose:

1. To recover voting pseudonyms after the first election, if the voter does not store the pseudonym, as may be required to prevent coercion.
2. 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.

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

[4] 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.

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).

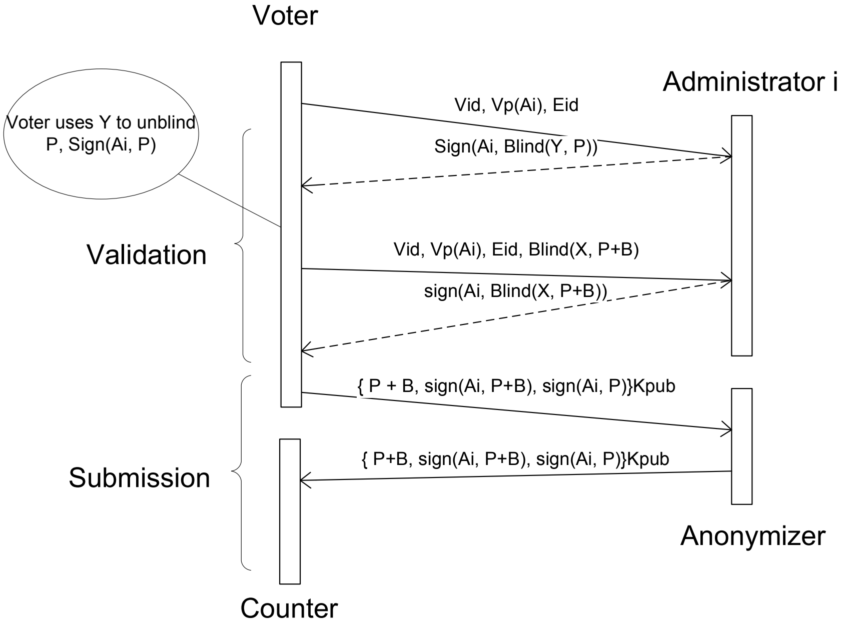
Voter
Administrator i
Voter uses Y to unblind P, Sign(Ai, P)
Vid, Vp(Ai), Eid
Sign(Ai, Blind(Y, P))
Validation
Vid, Vp(Ai), Eid, Blind(X, P+B)
sign(Ai, Blind(X, P+B))
{ P + B, sign(Ai, P+B), sign(Ai, P)}Kpub
Submission
{ P+B, sign(Ai, P+B), sign(Ai, P)}Kpub
Anonymizer
Counter

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.

4 Limitations

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.

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.[5]

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. This is however expected to improve as new versions and variations of the implementation are produced.

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.

Possibilities of further development

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:

(1) traceable and non-traceable elections, according to a per-election configuration option
(2) 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.

[5] 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.

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 it is done now.

Conclusions

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.

References

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

[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.

[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.

[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.

[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.