Compensating for a lack of transparency

Berry Schoenmakers
Department of Mathematics and Computing Science,
Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
berry@win.tue.nl

Paper-ballot elections may be a bit cumbersome yet many people will agree that the procedures to be followed are sufficiently transparent to all parties involved. The principles for fair elections are clear, and common sense tells people what to look for to detect potential fraud. The process is transparent.

For computerized elections the principles for fair elections are not all that obvious anymore. In particular, the internet turns out to be a highway fraught with accidents, where breaches of security and privacy have become all too common. News items reporting on such incidents get to the general public on an almost daily basis. For elections to be considered fair in such hostile environments, new rules and standards need to be set in order to compensate for the lack of transparency.

It is helpful to start looking at one of the problems to be solved by any voting system. Namely, the fundamental problem of achieving accuracy and ballot secrecy at the same time-assuming it is agreed that ballot secrecy is an essential ingredient for achieving democracy. Accuracy of the final tally means that it corresponds exactly to the ballots cast by legitimate voters. For instance, when voting by hand-raising it is easy to convince everybody of the correctness of the tally, but then there is no privacy.

A common way for internet-based elections these days is to put up a secure web server (plus database) for collecting the votes. Assuming the web server somehow ascertains the identity of voters and allows only one vote per voter, the idea is that the web server will honestly process each ballot by keeping a running total and*forgetting* who exactly voted what. At the end of the day, a final tally may then be produced.

Now suppose someone is not happy with the result, or rather, suppose not everybody believes it is the correct result, and a *re-count* is demanded. What are we supposed to do? We could show them the records representing the ballots again, and have them check the total. But these records need not have to do anything with the ballots cast by the legitimate voters: to protect the voters' privacy all identifying information has been removed. In fact, any audit trail for this type of system would violate ballot secrecy.

A natural reaction is to introduce (computerized) observers who will monitor the entire process to ensure its correct functioning. But this requires all the observers to be forgetful too! And how can we be sure the observers will spot every manipulation attempt; do we need to observe them too? Similarly, one could think of sending copies of the voter's ballot to multiple servers and somehow take the majority to obtain the final tally. This will fail hopelessly once voters start sending 'different copies' to these servers (or no copies to some servers, and so on). Also, we then have to trust each of these servers for being forgetful.

There is a powerful approach to avoid this kind of dilemmas, namely by making the process *verifiable*. If a process is verifiable, we do not need to trust the computers and election officers for following the protocols. The key element for achieving verifiability is a special type of encryption for the ballots, called *homomorphic* encryption. Concretely, homomorphic encryption of the ballots allows us to multiply them together, and obtain an encryption (of the same type) containing the sum of the votes contained in these ballots. By forming the product of all ballots cast we thus obtain an encryption containing the final tally.

The two main stages of an election then are:

1. Voters post their homomorphically-encrypted ballots, accompanied by a digital signature. Anyone is allowed to see the ballots cast, and anyone may verify the signatures for these ballots.

2. The product of all encrypted ballots is formed and only this product is decrypted, which yields the final tally. The product as well as its decryption (and hence the final tally) can be verified by anyone.

Following this principle, voting systems can be build for which monitoring is not necessary. Copies of all signed and encrypted ballots can simply be kept, as well as a copy of the product of all ballots and its decryption, all of which can be verified for consistency by any interested party.

This captures the idea behind a verifiable election. Of course, the details for solid implementations of this type of voting systems are not trivial. For example, off-the-shelf encryption methods cannot be used, as these are not homomorphic. Rather, an engine needs to be build which efficiently implements the cryptographic protocols designed for verifiable elections. For further reading on the topic of (universally) verifiable elections we have included some references at the end of this article.

The approach described above follows the paradigm introduced by Benaloh *et al.* (e.g., see [CF85,BY86,Ben87]); the scheme of [CGS97] uses novel cryptographic tools to obtain a secure and efficient implementation of such a scheme. We used a variant of this scheme as the voting engine for the *InternetStem* project, a small-scale 'shadow election' held during the Dutch national elections in May 1998. The Seattle-based company VoteHere.net has also been using homomorphic techniques in all of their trials since October 1999, including the Alaskan Republican Straw Poll for US President, which is the first binding internet election (see http://votehere.net). In case of InternetStem, the voting clients were implemented as Java applets, downloaded through SSL to a browser on a PC. Of course, voting clients running on smart cards, PDAs, mobile phones, and set-top boxes are also possible. On top of the voting engine one may then put different mechanisms for identifying voters, one may use different methods for storing the encrypted ballots in a redundant way (such that elections do not fail if some voting servers crash), one may distribute the computational work over different places, and so on: the particular way these subproblems are handled does not affect the secrecy of the ballots nor the accuracy of the final tally.

References

[Abe98] Masayuki Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. In *Advances in Cryptology-EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 437-447, Berlin, 1998. Springer-Verlag.

[Ben87] J. Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Yale University, Department of Computer Science Department, New Haven, CT, September 1987.

[BT94] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *Proc. 26th Symposium on Theory of Computing (STOC '94)*, pages 544-553, New York, 1994. A.C.M.

[BY86] J. Benaloh and M. Yung. Distributing the power of a government to enhance the privacy of voters. In *Proc. 5th ACM Symposium on Principles of Distributed Computing (PODC '86)*, pages 52-62, New York, 1986. A.C.M.

[CF85] J. Cohen and M. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proc. 26th IEEE Symposium on Foundations of Computer Science (FOCS '85)*, pages 372-382. IEEE Computer Society, 1985.

[CFSY96] R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret ballot elections with linear work. In *Advances in Cryptology-EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 72-83, Berlin, 1996. Springer-Verlag.

[CGS97] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology-EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103-118, Berlin, 1997. Springer-Verlag.

[Sch99] B. Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *Advances in Cryptology-CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 148-164, Berlin, 1999. Springer-Verlag.

[SK94] K. Sako and J. Kilian. Secure voting using partially compatible homomorphisms. In *Advances in Cryptology-CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 411-424, Berlin, 1994. Springer-Verlag.

[SK95] K. Sako and J. Kilian. Receipt-free mix-type voting scheme-a practical solution to the implementation of a voting booth. In *Advances in Cryptology-EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 393-403, Berlin, 1995. Springer-Verlag.