The Trash Attack

An Attack on Verifiable Voting Systems and a Simple Mitigation

Josh Benaloh
Microsoft Research

Eric Lazarus
DecisionSmith

Abstract

This short paper describes the *trash attack* which is effective against the majority of fully-verifiable election systems. The paper then offers a simple but counter-intuitive mitigation which can be incorporated within many such schemes to substantially reduce the effectiveness of the attack. This mitigation also offers additional benefits as it significantly improves the statistical properties of existing verifiable systems.

1. Introduction

Fully-verifiable (also known as end-to-end verifiable) election systems such as Scantegrity [CECCPSV08], Prêt à Voter [CRS05], VeriScan [Ben08], Helios [Adi08], and MarkPledge [Nef04,AdNe09] offer great promise. They enable the integrity of election results to be checked by anyone without having to trust election software, hardware, or personnel.

In traditional election systems, insider attacks can be perpetrated either by unscrupulous or subverted election officials or by the equipment they have chosen to trust. Verifiable election technologies prevent insider attacks by enabling voters themselves – as well as any observers – to verify the integrity of election results.

This paper introduces a previously unpublished attack that can be effective against most (perhaps even all) prior verifiable election systems. The attack is simple and practical, and it can be used to undetectably alter large numbers of votes.

Next, this paper describes a simple and effective mitigation which, although it conflicts with standard electoral doctrine, can be easily incorporated into many verifiable election systems. This mitigation makes the attack far more difficult and makes it nearly impossible to alter more than a small number of votes. The mitigation also offers additional benefits to many verifiable systems at minimal cost.

This paper does *not* propose a new verifiable election system. It instead describes a simple addition that can be incorporated into many existing systems which provides substantial benefits. A complete verifiable election system has many components including a voter interface, a back-end tallying process, a public verification process, and a dispute resolution process. The modification proposed in this paper affects both the voter interface and the public verification processes, but in most cases the actual changes are quite small. Other components are unaffected and remain as described in the cited references.

2. The Attack

Verifiable election systems give voters the ability to check that their votes have not been altered by providing them with receipts – usually in paper form – which voters can later check against a published list.[1] These receipts serve as an integral component of the integrity checking process. It is not assumed that all voters will check their receipts. Indeed, the expectation is that many – perhaps most – voters will not bother to check their receipts against the published election data. However it is not necessary that the majority of voters check their receipts. As long as election personnel and the systems they are charged with managing do not know which receipts will be checked, the checking of even a small fraction of receipts makes it very unlikely that election personnel or malicious vote management systems will be able to alter more than a few votes without detection.[2]

These statistics, however, only work if those who might have the ability to alter votes have no knowledge of which voters might be more likely to check their receipts. If there were a way for election workers or vote management systems to know with certainty or near certainty that particular voters would not check their receipts, they would be free to alter their corresponding votes. For this reason, it is essential to give receipts to *all* voters and not just offer receipts to those voters who request them.

The provision of receipts to voters who may not want them, however, suggests a very simple means by which election workers could find votes that are good candidates for alteration: poll workers could simply collect the contents of the nearest trash receptacles. Any receipts that have been discarded by voters would be strongly correlated with votes that could be altered without detection.[3] Active collection of receipts may also be viable through social engineering. Voters who can be induced to surrender their receipts have little protection against alteration of their votes by people or entities that have sufficient access to do so.

Although the most direct means of attack may be to collect receipts that have been separated from the voters to which they were issued, it may be sufficient to observe other characteristics of voters that make them appear to be less likely to check their receipts. Depending upon the verifiable election system in use, it may be possible for observers to associate voters with receipts and gain a statistical advantage at finding votes that can more likely be altered without detection.

3. The Mitigation

The attack described above takes advantage of the possibility of altering a single ballot or vote without disturbing others. This is generally possible in both traditional and verifiable election systems, and the common wisdom is that votes should be independent of each other and that the actions of one voter should not be manifested in any way that is observable by other voters. However, this principle seems overly broad. Indeed, it is this very independence of votes that facilitates the trash attack.

If altering the vote of a voter whose receipt is not verified would have an effect that would cause other voters' receipts to fail the verification process, then this attack would be substantially less effective, and it *is* possible to create such a dependency. There are many possible means by which dependencies could be created amongst voter receipts. An especially simple mechanism of creating such a dependency would be for each voter's receipt to include a cryptographic hash (e.g. SHA-256[4]) of the prior voter's receipt.[5] Since the receipt in many verifiable election systems already includes a cryptographic hash of ballot data, the mitigation could be

[1] Scantegrity II provides voters with a blank strip containing a ballot identifier onto which they can copy data revealed during the voting process.

[2] If, for example, only 5% of the receipts are checked, insiders would be unlikely to be able to alter more than about 20 votes without being detected.

[3] While it would be possible to create a ruse by making a copy of a receipt before discarding it, each individual discarded receipt would still be very unlikely to be checked.

[4] As defined in the National Institute of Standards and Technology Federal Information Processing Standard FIPS 180-3: Secure Hash Standard.

[5] Presumably, "prior voter" here would be the most recent prior voter using a particular voting device.

accomplished by simply adding the hash of the prior voter's receipt as one additional argument to the hash computation on a current voter's receipt. Such a change would be easy to incorporate within many verifiable election systems – including Prêt à Voter, VeriScan, and Helios, but more substantive changes would be required in a few systems such as Scantegrity where receipts are produced by the actions of each voter – without use of a vote recording or casting device.

The approach is quite simple. A verifiable voting device that produces a receipt should retain a copy of the last receipt produced (an arbitrary initial value can be used when the device is initialized). Each subsequent receipt should incorporate a cryptographic hash of the prior receipt with the current ballot receipt. By linking receipts in this way, each receipt that is checked serves not only as a verification that the associated vote has not been altered, it also serves as a verification that *none* of the votes previously cast on the same device have been subsequently altered.

The idea of a running hash is certainly not new. Hash chains are a common cryptographic tool and are found in many protocols. Indeed, the Merkle-Damgård construction of a hash function [Mer79, Mer89, Dam89] is defined by breaking the input into fixed-sized blocks and computing a running hash over the input blocks.

A running hash is not new to the election context either. Sandler and Wallach [SaWa07] suggest using a running hash to maintain the integrity of audit logs, and this has been incorporated into their VoteBox system [SDW08]. However, computing a running hash of actual voter receipts and incorporating this hash within each subsequently issued receipt appears to be a novel approach.

The actual construction would look very much like a typical Mergle-Damgård construction. At the beginning of an election, an initial value H_0, would be selected and published for each receipt-generating device.[6] This initial value H_0, should incorporate the date of the election and a unique identifier for the device.

An extent verifiable election system would produce a receipt r_i which is given to the ith voter. The proposal herein is that this receipt be modified to also include $H_i = H(r_i, H_{i-1})$ – where the function H is a cryptographically-strong one-way hash function such as SHA-256.

4. Ancillary Benefits

Besides thwarting the trash attack, a running hash, as described above, significantly improves the statistical properties of most – if not all – verifiable election systems. For example, suppose that it is known that 5% of voters are expected to verify their receipts in an election. With a standard design, an insider that randomly alters 10 ballots would escape detection about 60% of the time.

If a running hash were to be incorporated, this insider's options would be severely limited. If the insider had the ability to alter a ballot and a corresponding running hash value in real time (i.e. before the next voter uses a device), then the same 60% success rate could be achieved. But if the insider cannot mount a real-time attack, after-the-fact alteration of 10 ballots would only escape detection if they were all cast after the last ballot whose corresponding receipt was verified by a voter. Not only does this substantially restrict the pool of ballots available to the insider, but this threat can now be completely eliminated by a single diligent voter or observer recording the final running hash value at the end of the voting period.

5. Specific Instantiations

The details of how to incorporate a running hash of prior receipts into each new receipt varies

[6] Depending upon the specific verifiable election system, this device may also be used for voters to make their selections and/or cast their votes, or it may be a separate device whose principal purpose is generation of voter receipts.

somewhat depending on the verifiable election system that is used.

VeriScan, Helios, and MarkPledge all use a device which records a ballot and provides a distinct receipt to the voter. In these cases, the receipt already incorporates a hash of ballot contents, and it is a very simple matter to include the hash value from the prior receipt as an additional argument to the current hash value.

Prêt à Voter allows each voter to mark a paper ballot and then keep a copy of a portion of this marked ballot as a receipt. In current implementations, the copying is done with a scanner that also retains an electronic version of what is given to the voter. This scanner could be augmented to produce a hash of its electronic version of the ballot and to chain these hashes into a running hash that is printed onto the receipt that is given to each voter.

In Scantegrity, the receipt retained by a voter is simply a hand-written recording of codes on the paper ballot together with a (pre-printed) ballot identifier. This purely manual process requires a more substantial change to enable incorporation of a running hash value. The optical scanner that reads each Scantegrity ballot could be augmented to provide each voter with a complete receipt listing the ballot identifier and all codes on the ballot. Such a scanner-provided receipt could then easily be augmented to incorporate a running hash of all prior receipts produced by that optical scanner. This augmentation could also improve the direct verifiability of Scantegrity as voters would be able to do immediate checks of the integrity of their ballots and could potentially challenge discrepancies on the spot by instructing the scanner to return the contested ballot rather than drop it into the bin with other ballots. This augmentation would also address a frequent complaint of Scantegrity voters who find the manual creation of a receipt to be cumbersome.

In all of these cases, the remainders of the processes are as described in the respective systems. No attempt is made here to describe a complete verifiable election system which would require a fully-specified voter interface, precise roles for election officials, a back-end verifiable tallying mechanism, a dispute-resolution process, and much more. Instead, this work describes one very simple augmentation that can be incorporated within otherwise complete verifiable systems to both mitigate a previously unrecognized vulnerability and to improve the statistical properties of the respective existing systems.

6. Conclusions

Although the details may vary between systems, it is clear that the simple inclusion of a running hash within voter receipts mitigates a serious vulnerability that may occur when insiders or others, who may have the ability to change votes after they have been cast, can use external information to tell which voters are more likely to check their receipts against published lists. This mitigation is very simple to incorporate into many verifiable election systems, but the effect can be profound. Without the mitigation, verifiable election systems may in practice be vulnerable to some well-known attacks that plague traditional election systems, but with the mitigation in place, these systems can not only achieve the properties they were previously thought to hold, but they can actually achieve even better properties with fewer voter checks providing greater confidence in the integrity of elections.

References

[Adi08] **Adida, B.** *Helios: Web-Based Open-Audit Voting.* Proceedings of 17th Usenix Security Symposium. San Jose, CA, 2008.

[AdNe09] **Adida, B. and Neff, C.A.** *Efficient Receipt-Free Ballot Casting Resistant to Covert Channels.* Proceedings of EVT/WOTE – 4th Electronic Voting Technology Workshop / Workshop on Trustworthy Elections. Montreal, PQ, 2009.

[Anon09] **Authors names withheld to preserve anonymity.** *A Serious Threat with a Simple Solution.* Rump session of EVT/WOTE – 4th Electronic Voting Technology Workshop / Workshop on Trustworthy Elections. Montreal, PQ, 2009.

[Ben08] **Benaloh, J.** *Public and Administrative Verifiability: Can We Have Both?* Proceeding of 3rd Electronic Voting Technology Workshop. San Jose, CA, 2008.

[CECCPSV08] **Chaum, D., Essex, A., Carback, R., Clark, J., Popoveniuc, S., Sherman, A., Vora, P.** *Scantegrity: End-to-End Voter-Verifiable Optical-Scan Voting.* IEEE Secruity and Privacy Magazine, Vol. 6, no. 4, pp. 40-46. May/June 2008.

[CRS05] **Chaum, D., Ryan, P.Y.A., and Schneider, S.** *A Practical Voter-Verifiable Election Scheme.* 2005. Proceedings of ESORICS 2005, 10th European Symposium on Research in Computer Security. pp. 118-139.

[Dam89] **Damgård, I.** *A Design Principle for Hash Functions.* Advances in Cryptology -- Crypto '89 Proceedings. [ed.] G. Brassard.: Springer-Verlag, Lecture Notes in Computer Science, Vol. 435, pp. 416-427. Santa Barbara, CA, 1989.

[Lam81] **Lamport, L.** *Password Authentication with Insecure Communication.* Association for Computing Machinery, Communications of the ACM, Vol. 24, no. 11, pp. 770-772. November 1981.

[Mer79] **Merkle, R.C.** *Security, Authentication, and Publik Key Systems.* Stanford University, Ph.D. thesis, 1979.

[Mer89] **Merkle, R.C.** *A Certified Digital Sigrature* Advances in Cryptology -- Crypto '89 Proceedings. [ed.] G. Brassard.: Springer-Verlag, Lecture Notes in Computer Science, Vol. 435, pp. 218-238. Santa Barbara, CA, 1989..

[Nef04] **Neff, C.A.** Practical High Certainty Intent Verification for Encrypted Votes. *VoteHere.* [Online] http://voterhere.net/vhti/documentation/vsv-2.0.3638.pdf.

[SaWa07] **Sandler, D. and Wallach, D.S.** *Casting Votes in the Auditorium.* Proceedings of 2nd Electronic Voting Technology Workshop. Boston, MA, 2007.

[SDW07] **Sandler, D., Derr, Kyle, and Wallach, D.S.** *VoteBox: A Tamper-evident, Verifiable Electronic Voting System.* Proceedings of 17th Usenix Security Symposium. San Jose, CA, 2008.