Trapdoor commitments in the SwissPost e-voting shuffle proof

Brief Summary

The implementation of the commitment scheme in the SwissPost-Scytl mixnet uses a trapdoor commitment scheme, which allows anyone who knows the trapdoor values to generate a shuffle proof transcript that passes verification but actually alters votes. This allows undetectable vote manipulation by an authority who implemented or administered a mix server.

Independent discoveries

Update: The same issue was independently discovered by Thomas Haines of NTNU, and also by Rolf Haenni of Bern University of Applied Sciences, whose report is here.

Verifiability, trust, and why this matters

Verifiability is a critical part of the trustworthiness of e-voting systems. Universal verifiability means that a proof of proper election conduct should be verifiable by any member of the public.

The SwissPost e-voting system, provided by Scytl, aims to offer a partial form of verifiability, called ``complete verifiability'', which resembles universal verifiability but adds the assumption that at least one of the components on the server-side, i.e., the computers running the voting system, behaves correctly. (Universal verifiability offers guarantees even if all server-side components are malicious.)

In the SwissPost system, encrypted electronic votes need to be shuffled to protect individual vote privacy. Each server who shuffles votes is supposed to prove that the set of input votes it received corresponds exactly to the differently-encrypted votes it output. This is intended to provide an electronic equivalent of the publicly observable use of a ballot box or glass urn.

We show that the mixnet specification and code recently made available for analysis does not meet the assumptions of a sound shuffle proof and hence does not provide universal or complete verifiability.

We give two examples of how an authority who implemented or administered a mix server could produce a perfectly-verifying transcript while actually - undetectably - manipulating votes.

Details

The problem derives from the use of a trapdoor commitment scheme in the shuffle proof. If a malicious authority knows the trapdoors for the cryptographic commitments, it can provide an apparently-valid proof, which passes verification, while actually having manipulated votes. There is no modification of the audit process that would make it possible to detect if a manipulation happened. Instead, the key generation process for the commitment scheme needs to be modified in such a way that it offers evidence that no trapdoor has been produced, and the audit process should include the verification of this new evidence.

The first cheating example requires knowing the randomness used to generate the vote ciphertexts that will be manipulated. There are several ways this could be achieved. For example, an attacker could compromise the clients used for voting. Weak randomness generation (such as that which affected the Norwegian Internet voting system) would allow the attack to be performed without explicit collusion.

The second cheating example does not require any extra information at all, though it does rely on the election parameters having been set up in a particular way, and on multiple selections being accepted as valid votes.

See our paper for a more detailed technical explanation, or Olivier Pereira's blog for an explanation in French, or Sarah Jamie Lewis's Twitter feed for a general analysis of the code.

We have not thoroughly tested or examined all other aspects of the system. Even if this issue is corrected, there may be other unnoticed problems. Analyses of other Internet voting systems in Washington D.C., Estonia, New South Wales and Western Australia have identified numerous serious issues affecting privacy, integrity or verifiability.

The cheating proofs

Download the complete cheating proof transcripts here.

Each pair of transcripts illustrates an example of how the commitment trapdoor could be used to produce an apparently-valid proof when actually manipulating votes. In each case, we have provided both a normal transcript with realistic encrypted votes, and a transcript with all randomness set to zero. The latter is not a normal run of the mix, but you can see immediately that the votes change.

You will need a running copy of the SwissPost-Scytl code in order to verify the cheating proofs. See the README for instructions.

Source of the problem

Nothing in our analysis suggests that this problem was introduced deliberately. It is entirely consistent with a naive implementation of a complex cryptographic protocol by well-intentioned people who lacked a full understanding of its security assumptions and other important details. Of course, if someone did want to introduce an opportunity for manipulation, the best method would be one that could be explained away as an accident if it was found. We simply do not see any evidence either way.

The bottom line

A problem in the implementation of the SwissPost-Scytl shuffle proof allows undetectable vote manipulation.