A privacy attack on the SwissPost e-voting system

Véronque Cortier, Alexandre Debant, Pierrick Gaudry (speaker)

The SwissPost e-voting system is currently proposed under the scrutiny of the community, before being deployed in 2022 for political elections in several Swiss Cantons. We explain how real world constraints led to shortcomings that allowed a privacy attack to be mounted. More precisely, dishonest authorities can learn the vote of several voters of their choice, without being detected, even when the requested threshold of honest authorities act as prescribed.

Context

In Spring 2020, the Swiss Post company has acquired from Scytl the rights necessary for independently developing their e-voting system. Since then, the protocol and the source code have been modified, fixing the issues discovered in 2019, and greatly improving the general structure of the code and of the documentation. While the system is now different from any previously published one, it still carries the heritage of its Scytl ancestor, which is well known by the e-voting community.

During 2021, the specification and the source code are gradually revealed for being studied by the community[1]. A private bug bounty program has been launched and a public bug bounty will follow. The goal is to obtain feedback in order for the system to be ready to be used by the Swiss Cantons in the course of 2022.

Protocol overview

The protocol is a return-code scheme. For sake of simplicity, we omit the description of the return-codes, mainly used to ensure verifiability. Instead, we focus on the relevant key-points for vote privacy.

Before the election day, an actor called Print Office is designated to generate cryptographic material. Each voter receives, by classical postal mail, everything that is needed for voting, printed on paper. Then, the voter connects to a Voting Server, via the Internet. Her choice is encrypted with an election public key. At the end of the election day, the encrypted ballots are mixed and decrypted by actors called Mixing Control Components (CCM). The decryption key is shared in 4 parts. The first 3 parts are held directly by CCM_1, CCM_2, CCM_3. The last part is stored by an Election Board, and will be given to the CCM_4, just when it needs it, and only after verifications by (trusted) Auditors.

Security goals

In Switzerland, e-voting is regulated by the Federal Chancellery[2]. Security goals in terms of privacy and verifiability are precisely defined, with respect to an explicit threat model. In particular, vote privacy must hold as soon as the Print Office is honest, and at least one of the 4 CCMs is honest. The Election Board includes another level of secret sharing and is therefore assumed to be honest (as a whole).

Security analyses and proofs have been conducted in the computational and symbolic models in order to demonstrate that the system matches these requirements. Unfortunately, there exists a gap between the theoretical scenarios they consider (e.g. unique election with a unique ballot-box) and the practical ones (possibly multiple elections and ballot-boxes in parallel). Our attack takes place there.

Description of the attack

The decryption phase is done in a single round. First CCM_1 mixes and partially decrypts the list of ballots. Then it sends it to CCM_2 who mixes and decrypts as well, and so on with CCM_3 and finally CCM_4. The Auditors step in at several stages. They first verify (and sign) the ballot box received as

[1] `https://gitlab.com/swisspost-evoting/documentation`

[2] `https://www.bk.admin.ch/bk/en/home/politische-rechte/e-voting/versuchsbedingungen.html`

input by CCM$_1$. They also check the validity of the data before and after CCM$_4$ performs its share of the work and finally reveals the tally. If at least one of the CCMs is honest, the secret of the global mixing is preserved, and therefore it is expected that voting privacy follows.

Our attack exploits two peculiarities that are driven by practical concerns. First, in order to keep the CCMs simple and efficient, they do not perform any verification themselves, relying on the Auditors. In particular, the zero-knowledge proofs (ZKP) of previous mixing are not checked. Since the Auditors operate only before CCM$_1$ and CCM$_4$, this opens opportunities to fool CCM$_2$ and CCM$_3$. Second, the actors actually handle several ballot boxes during the same election event. This is due to the fact that results must be computed and published at a scale much smaller than the whole electorate of the Canton.

The attack proceeds as follows. When CCM$_1$ (malicious) receives the k ballot-boxes that it has to decrypt, it can produce an additional one, B_{k+1}, that contains only Alice's ballot. CCM$_1$ then proceeds normally on the $k+1$ ballot-boxes and submits them to CCM$_2$. CCM$_2$ (honest) will then mix and decrypt the $k + 1$ ballot-boxes and send the results to CCM$_3$ which will do the same. Before sending the data to the Auditors, CCM$_3$ (malicious) removes the elements corresponding to B_{k+1}. Everything looks correct from the Auditors' point-of-view and the Electoral Board secret key is revealed to CCM$_4$. Finally, CCM$_4$ (malicious) is able to collude with CCM$_1$ and CCM$_3$ to complete the decryption of Alice's ballot. Note that CCM$_4$ can decrypt the k "valid" ballot-boxes to complete the election without being detected.

Our attack enables three colluding dishonest CCMs and a dishonest Voting Server to learn the vote of Alice, hence of actually any voter of their choice since the Voting Server knows the correspondence between ballots and voters. This breaks the security goals of the Swiss Chancellery that requests that vote privacy is guaranteed as soon as one of the CCMs is honest (here CCM2). The attack can easily be extended to learn the vote of several voters. It suffices to add as many (malicious) ballot boxes as desired, each balot box containing the ballot of one targeted voter. This would remain undetected unless the number of added ballot boxes eventually renders the process too slow. CCM3 could also be honest (with CCM2) since actually all communications go through the Voting Server (dishonest).

Discussion

A first remark is that while privacy and verifiability of an e-voting system are properties that are considered well understood in theory, the design of a system that provides both proves to be surprisingly hard in a practical and large scale setting. In this particular case, the general structure of the system is sound, but a few minor modifications, dictated by real world constraints, opened the possibility for a privacy breach. The fact that several ballot boxes exist in the same election event was the key to allow a replay attack. And the pressure to keep the CCMs simple and efficient, pushed towards delegating to others the verification of the ZKP, which is also a dangerous behaviour.

Another aspect is that attacks of this kind are often not covered by (computational or symbolic) security proofs. E-voting protocols are peculiar in that they involved numerous actors playing different roles, and there are various properties to check according to possibly different trust assumptions. This leads to long and complex proofs and a first simplification that is often done is to study a unique election with a unique ballot box to tally. A classical argument for this is that all the data is bound to a unique election identifier. Our attack scenario is an illustration that this is not necessarily enough.

Epilogue

Our attack was sent to Swiss Post as a confidential issue on their GitLab repository, that has then been acknowledged, made public and the system will be patched to prevent the problem[3]. We also obtained a generous reward from the bug bounty program (40 k€). A robust way to fix the protocol would be to make the CCMs verify themselves that what they are asked to decrypt is indeed a transformation of the board by valid mixes.

[3]`https://gitlab.com/swisspost-evoting/documentation/-/issues/11`