Swiss Post Public Intrusion Test

Undetectable Attack Against Vote Integrity and Secrecy

Rolf Haenni

Bern University of Applied Sciences

1 Theoretical Background

Commitment schemes are usually *perfectly hiding* and *computationally binding*. This means that no information about the message can be derived from the commitment and that the commitment can not be opened to a message other than the original one. The Pedersen commitment scheme achieves these properties by computing $c = G^m H^r$ in a multiplicative cyclic group, for which the discrete logarithm assumption (DL) is believed to hold. This scheme is perfectly binding, because a randomization $r' \neq r$ exists for any other message $m' \neq m$ such that $c = G^{m'} H^{r'}$. This means that c could potentially be opened to all q messages from $\mathbb{Z}_q$, but this requires the computation of the discrete logarithm. The Swiss Post voting protocol works with a subgroup $G_q \subseteq \mathbb{Z}_p^*$ of integers modulo p, where $q = |G_q|$ denotes the prime order of the subgroup, $m \in \mathbb{Z}_q$ the message, and $r \in \mathbb{Z}_q$ the randomization. Both G and H are elements of G_q.

A pre-condition for the scheme to be computationally binding is the *independence* of the two values $G, H \in G_q \setminus \{1\}$ (in a group of prime order q, both values are generators of G_q). Independence means that respective discrete logarithms $h = \log_G H$ and $g = \log_H G$ are unknown to everyone. Otherwise, for example if $h = \log_G H$ is known to the person who created c, then c can be rewritten as

$$c = G^m H^r = G^m (G^h)^r = G^{m+hr \bmod q}.$$

Therefore, to open c to a different message $m' \neq m$, the adversary can easily solve

$$m + hr \equiv m' + hr' \pmod q$$

to find the matching randomization $r' = (m - m')h^{-1} + r \bmod q$. As a consequence, the binding property of the commitment scheme is completely broken in that case.

2 Problem and Attack Description

Commitment schemes are important building blocks in other cryptographic primitives, for example in zero-knowledge proofs. The Bayer-Groth proof system, which is used to prove that the encrypted votes have been shuffled correctly, makes intensive use of Pedersen commitments. Often, the commitment involves multiple messages $m_1, \dots, m_n$, where n denotes the number of encrypted votes. The Pedersen commitment can be extended easily to this general case by

$$c = G_1^{m_1} \dots G_n^{m_n} H^r,$$

where $G_1, \dots, G_n, H$ are all pairwise independent in the way described above. If independence is violated between H and one single value G_i, then the extended commitment c can be open for any vector of alternative messages $m'_1, \dots, m'_n$. If this happens, then the whole shuffle proof argument collapses, i.e., it is possible to construct a fake proof for an incorrect shuffle. This can be exploited by the mixing component in at least two different ways:

1. Manipulation of the Election Result:
   Instead of re-encrypting the shuffled encrypted votes, the malicious mixing component manipulates the input encryption list in an arbitrary way, for example such that the preferred candidate wins the election. Then a fake proof is constructed, which links the manipulated output encryption list to the correct input encryption list. The verification of the proof will succeed and the attack remains undetected.

2. Breaking Vote Privacy:
   The first mixing component adds a marking to each encrypted vote of the input list. This can be done easily due to the homomorphic property of the ElGamal encryption scheme. As above, a fake proof is then constructed, which links the manipulated output encryption list to the correct input encryption list. Again, the attack remains completely undetected at this point. After decryption, the markings will become visible in the result cleartext votes, which exposes the links between votes and voters to the malicious first mixing component.

Both attacks are easily implementable in the Swiss Post voting system by a malicious mixing component, because of a critical flaw in the generation of the values $(G_1, \dots, G_n, H)$. In [2, Page 128], this vector of size $n+1$ is called *commitment key* ck.[1] According to [2, Section 9.1.7], each of these values is generated using the so-called *Group Element Generation Primitive*. This primitive is specified in [2, Section 9.1.26], and there is the critical flaw. As already reported in Issue #274, the generation of random group element should not be done according to the proposed algorithm, because it reveals the discrete logarithm $r = \log_g R$ of the generated random value $R \in G_q$. If two supposedly independent generators $H_1, H_2 \in G_q$ are generated in this way, then knowledge of $h_1 = \log_g H_1 \bmod p$ and $h_2 = \log_g H_2 \bmod p$ reveals $\log_{H_1} H_2 \bmod p = \frac{h_2}{h_1} \bmod q$, i.e., the pre-condition of the above attack against the commitment scheme and the shuffle-proof system is given in a trivial way.

[1] No keys are involved in generating and opening commitments, so calling the values $ck = (G_1, \dots, G_n, H)$ a *key* is a rather inappropriate choice.

3 Counter-Measures

Generally, it is important to distinguish the problems of generating random group elements and independent generators. Best practices for the former problem have been discussed in the submitted report registered as Issue #274. The latter problem, however, is more delicate, but there are international standards such as FIPS PUB 186-4 [1, Appendix A.2.3], which are not difficult to implement. The key feature of such methods is making the generation of independent generators publicly verifiable. More general methods have been developed for producing *verifiably random numbers*. The resulting verification is important for both parties opening a commitment and parties verifying a shuffle proof. Without such a verification, neither the commitment nor the shuffle proof is convincing.

As a counter-measure to avoid such attacks, we suggest to implement standard algorithms like the one mentioned above. In addition to this, an additional verification step for checking the independence of the generators needs to be implemented into the verification procedure of the mixing steps.

4 Conclusion

This encountered problem in the generation of independent generators fully undermines the security of the whole mix-net, which opens doors for arbitrarily scalable vote integrity and vote secrecy attacks by malicious mixing components. Most of these attacks are undetectable both during and in the aftermath of an election. On the other hand, fixing the problem is relatively simple using existing methods.

References

[1] Digital signature standard (DSS). FIPS PUB 186-4, National Institute of Standards and Technology (NIST), 2013.

[2] Scytl sVote – Protocol Specifications. Technical report, Scytl Secure Electronic Voting, Barcelona, Spain, 2018.