Zerocoin signatures for Belenios

Maxime Lalisse

October 29, 2024

1 Introduction

Belenios [1] is a secure and verifiable online voting system providing elibigity verifiability. Belenios relies on various types of zero-knowledges proofs [2][3], including those for ballot signatures. We aim to describe a signature scheme for Belenios that hides the credential used. This scheme [4], sometimes called a "zerocoin scheme", is similar to a ring signature scheme in that it remains unlinkable to a specific token. However, it also require disclosing a value *serial*, which is bound to a specific token (without revealing which one). As a result, re-voting is only possible with the same *serial*. This scheme requires another group generator h.

2 Credentials

From a secret *credential* c, two values are now derived: $s = \texttt{secret}(c)$ and $\texttt{serial} = \texttt{serial}(c)$. Public credentials are now pedersen commitments in the form $\mathsf{public}(c) = g^s \times h^{\texttt{serial}}$.

3 Zerocoin signatures

$$\texttt{zsignature} = \left\{ \begin{array}{ccc} \mathsf{hash} & : & \texttt{string} \\ \mathsf{serial} & : & \texttt{string} \\ \mathsf{proof} & : & \texttt{proof}^* \end{array} \right\}$$

One can reveal a $\texttt{serial}$ and prove that he knows a $\texttt{s}$ such that there exist a credential $c_n \in C$ in the form $g^s \times h^{\texttt{serial}}$, by proving that he knows $\texttt{s}$ such that there is ${c_n}' = c_n \times h^{-\texttt{serial}}$ in the form g^s, by creating a sequence of proofs $\pi_0, \dots, \pi_k$ with the following procedure, parameterised by a string S:

1. for $j \neq i$:
   (a) create π_j with a random $\mathsf{challenge}$ and a random $\mathsf{response}$
   (b) compute ${c_j}' = c_j \times h^{-\texttt{serial}}$
   (c) compute $A_i = g^{\mathsf{response}} \times {c_j}'^{\mathsf{challenge}}$
2. π_i is created as follows:
   (a) pick a random $w \in \mathbb{Z}_q$
   (b) compute $A_i = g^w$
   (c) $\mathsf{challenge}(\pi_i) = \mathcal{H}_{\mathsf{zsignature}}(S, \texttt{serial}, \texttt{hash}, A_0, \dots, A_k) - \sum_{j \neq i} \mathsf{challenge}(\pi_j) \mod q$
   (d) $\mathsf{response}(\pi_i) = w - \texttt{s} \times \mathsf{challenge}(\pi_i) \mod q$

In the above, $\mathcal{H}_{\mathsf{zsignature}}$ is computed as follows:

$$\mathcal{H}_{\mathsf{zsignature}}(S, \texttt{serial}, \texttt{hash}, A_0, \ldots, A_k) = \mathsf{SHA256}(\texttt{zsig|}S\texttt{|serial|hash|}A_0\texttt{,}\ldots\texttt{,}A_k) \mod q$$

where `zsig`, vertical bars and commas are verbatim. The result is interpreted as a 256-bit big-endian number.

The signature is verified as follows:

1. for $j \in [0 \ldots k]$, compute
$$c_j{}' = c_j \times h^{-\texttt{serial}}$$
$$A_j = g^{\mathsf{response}(\pi_j)} \times c_j{}'^{\mathsf{challenge}(\pi_j)}$$

2. check that
$$\mathcal{H}_{\mathsf{zsignature}}(S, \texttt{serial}, \texttt{hash}, A_0, \ldots, A_k) = \sum_{j \in [0 \ldots k]} \mathsf{challenge}(\pi_j) \mod q$$

References

[1] V. Cortier, P. Gaudry, and S. Glondu. Belenios: a simple private and verifiable electronic voting system. *Foundations of Security, Protocols, and Equational Reasoning: Essays Dedicated to Catherine A. Meadows*, pages 214–238, 2019.

[2] P. Gaudry. Some ZK security proofs for Belenios. working paper or preprint, 2017.

[3] S. Glondu. Belenios specification. *Version 0.1. http://www. belenios. org/specification. pdf*, 2013.

[4] J. Groth and M. Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 253–280. Springer, 2015.