One-out-of-many signatures for Belenios

Maxime Lalisse

October 6, 2024

1 Introduction

Belenios [1][3] is a secure and verifiable online voting system that provide elibigity verifiability. We aim to describe a one-out-of-many signature scheme for Belenios. A one-out-of-many signature [4] is a signature-of-knowledge, a special type of proof-of-knowledge in which data is authenticated by including it as part of the public information used by the Fiat-Shamir heuristic. This type of signature is similar to a ring signature in that it remains unlinkable. However, we also require disclosing a value *serial*, which is bound to one's identity using a Pedersen commitment. This scheme allows re-voting, but only with the same *serial*, preventing a voter's votes from being counted multiple times.

2 Protocol

Public context: Group $G = \langle g \rangle$, with $\#G = q$. h is an independant generator of G.

Setting: Prover $\mathcal{P}$ has a pedersen commitment of a public key $c = g^x \times h^{\mathtt{serial}}$, where x is his secret key and `serial` open the commitment. The set of credentials C is made public. $\mathcal{P}$ wants to convince $\mathcal{V}$ that he knows x such that $\exists i \, : \, C_i = g^x h^{\mathtt{serial}}$.

Commitment $\mathcal{P} \rightarrow \mathcal{V}$.
Let $i \in [0, k]$ be such that $c = C_i$. For all $j \neq i$, the prover $\mathcal{P}$ picks random (σ_j, ρ_j) in $\mathbb{Z}_q$ and computes $A_j = g^{\rho_j} \times (c_j \times h^{-\mathtt{serial}})^{-\sigma_j}$.
He also picks a random element w in $\mathbb{Z}_q$ and computes $A_i = g^w$.
The prover $\mathcal{P}$ sends all the A_j for $j \in [0, k]$. The prover $\mathcal{P}$ also sends `serial`.

Challenge $\mathcal{V} \rightarrow \mathcal{P}$. Verifier $\mathcal{V}$ picks e at random in $\mathbb{Z}_q$ and sends e.

Response $\mathcal{P} \rightarrow \mathcal{V}$: Prover $\mathcal{P}$ computes $\sigma_i = e - \sum_{j \neq i} \sigma_j \mod q$ and $\rho_i = w + r\sigma_i \mod q$. He sends all the pairs (σ_j, ρ_j) for $j \in [0, k]$.

Result. $\mathcal{V}$ checks the following equalities and accepts if and only if all of them hold. First, she checks that $\sum \sigma_j = e$, and then for each $j \in [0, k]$ that $A_j = g^{\rho_j} \times (c_i \times h^{-\mathtt{serial}})^{-\sigma_j}$.

3 Implementation

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

Credentials are now pedersen commitments with public $g^s \times h^{\mathtt{serial}}$. and secret $(s, \mathtt{serial})$. One can reveal a `serial` and prove that he knows a `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 $\pi_0, \ldots, \pi_k$ 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) $\mathsf{challenge}(\pi_i) = \mathcal{H}_{\mathsf{mproof}}(S, \texttt{serial}, \texttt{hash}, A_0, \ldots, A_k) - \sum_{j \neq i} \mathsf{challenge}(\pi_j) \mod q$
   (d) $\mathsf{response}(\pi_i) = w - \texttt{s} \times \mathsf{challenge}(\pi_i) \mod q$

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

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

where $\texttt{msig}$, 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
$$ {c_j}' = c_j \times h^{-\texttt{serial}} $$
2. for $j \in [0 \ldots k]$, compute
$$ A_j = g^{\mathsf{response}(\pi_j)} \times {c_j}'^{\mathsf{challenge}(\pi_j)} $$
3. check that
$$ \mathcal{H}_{\mathsf{mproof}}(S, \texttt{serial}, \texttt{hash}, A_0, \ldots, A_k) = \sum_{j \in [0 \ldots k]} \mathsf{challenge}(\pi_j) \mod q $$

4 Security proof

Similarly to other ZK security proofs for Belenios detailed in [2], we need to prove three properties: completeness, zero-knowledge and soundness.

Completeness. By construction.

Zero-knowlege. By simulation. For any challenge e, $\mathcal{P}$ can first generate σ_j, ρ_j and then valid commitments $A_j = g^{\rho_j}(c_j h^{-\texttt{serial}})^{-\sigma_j}$ without knowing x.

Special-soudness. TODO. If there is two different valid transcripts, there exists j such that there is two different (σ_j, ρ_j) from which the secret x can be extracted.

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.