Belenios: a simple private and verifiable electronic voting system

Authors : Véronique Cortier, Pierrick Gaudry, Stephane Glondu

CNRS, Inria, Univ. Lorraine, France

Arnaud Labourel (AMU, CNRS, LIS)

Distributed team DALGOrithms seminar

Introduction

It always takes longer than expected

Hofstadter's Law [Hofstadter 1979]

It always takes longer than you expect, even when you take into account Hofstadter's Law.

Previously on DALGO seminar

We have seen:

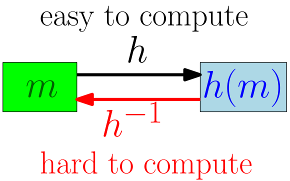
- the electronic voting system Belenios
- the global ideas behind it
- the properties guaranteed by the protocol: privacy and verifiability
- a cryptographic tool: ElGamal encryption scheme

On this DALGO seminar

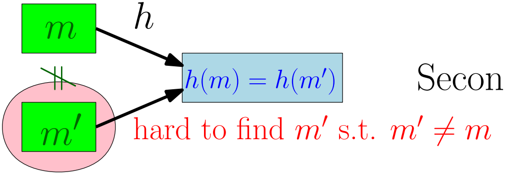
- Cryptographic hash function
  used for signatures and ZKP
- Non-interactive Zero-Knowledge Proofs (ZPK) used
  - by voters to prove validity of votes and avoid ballot stuffing
  - by trustees to prove the correct decryption of the election result
- Schnorr signature scheme
  used to sign the ballot
- Pedersen's threshold secret sharing scheme
  used by trustees s.t. no single authority has the private key of the election

Cryptographic hash function

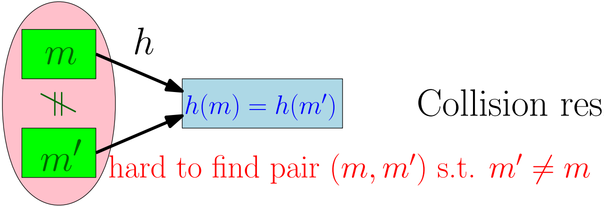
Cryptographic hash function



Pre-image resistance



Second pre-image resistance



Collision resistance

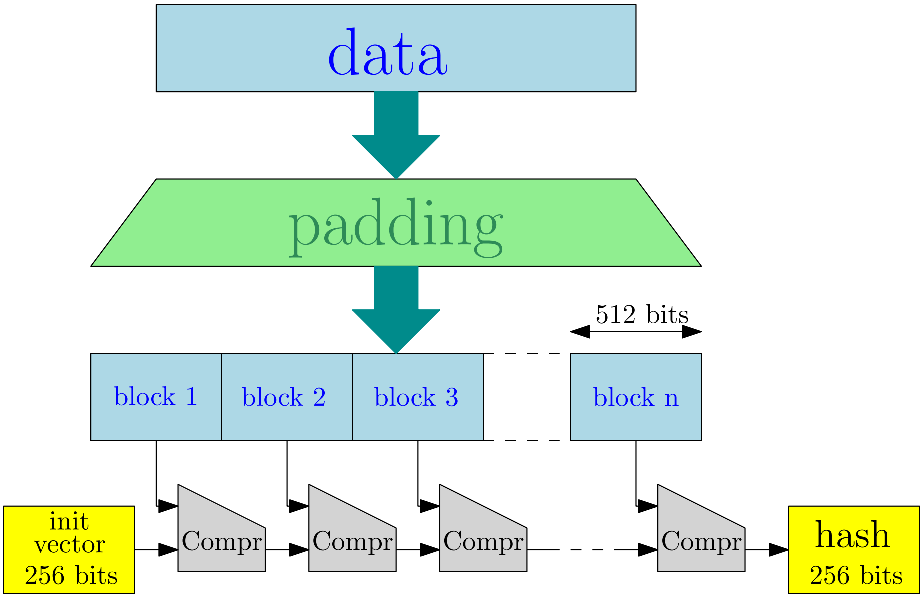
SHA256 used by Belenios

- Produce hash of 256 bits
- Published in 2001 by the NIST
- Based on Merkle–Damgård construction with Davies–Meyer compression function

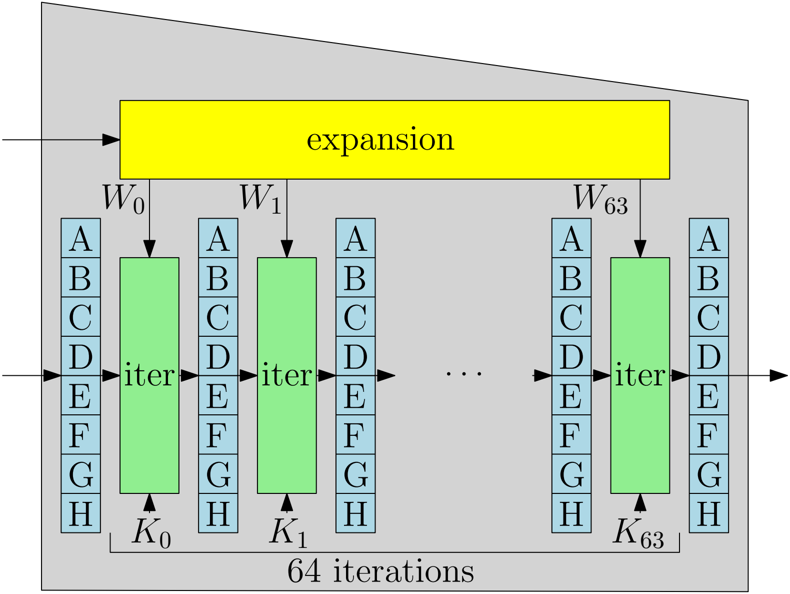
Rough ideas behind SHA256

- Iteratively use a compression function that outputs 256 bits from a block of 512 bits and the output of the previous computation
- The compression function is composed of numerous iterations of bitwise function on the block

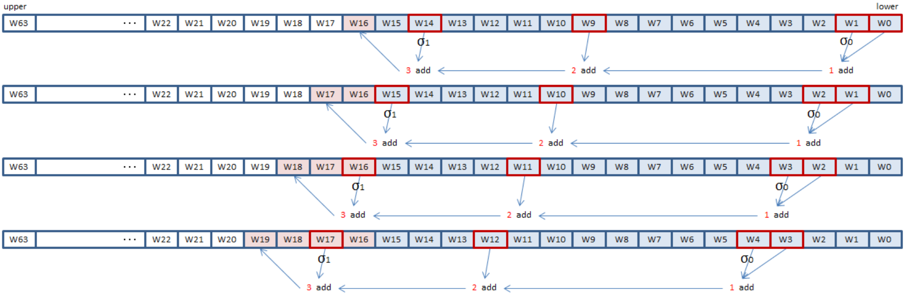
Merkle–Damgård construction



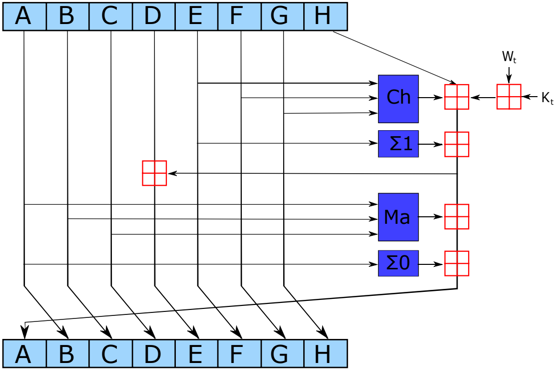
Compression function



Expansion function



Iterated function



$\boxplus$: addition mod 2^{32}

$$\mathrm{Ch}(E, F, G) = (E \wedge F) \oplus (\neg E \wedge G)$$
$$\mathrm{Ma}(A, B, C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$
$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

Non-interactive Zero-Knowledge Proofs

Non-interactive Zero-Knowledge Proofs (ZKP)

Used

- by voters to prove validity of votes (for instance prove that an encrypted ballot encode a value inside some specific set)
- by trustees to prove that they know their secret key and for the correct decryption of the election result

Three kind of proofs:

- that a discrete logarithm belongs to some finite set
- of knowledge a discrete algorithm
- of correct decryption

Intuition behind ZKP

- Victor is color-blind and Peggy wants to prove him that she can distinguish two different colored balls (which he cannot).
- Victor takes the ball and choose to switch them or not behind his back (without Peggy knowing) and then shows them to Peggy.
- Peggy has to guess if Victor has switched or not the balls and then repeat the process multiple times.
- The probability of randomly succeeded at guessing all switchs/non-switches approaches zero (soundness)
- Victor should become convinced (completeness) that the balls are indeed differently colored.

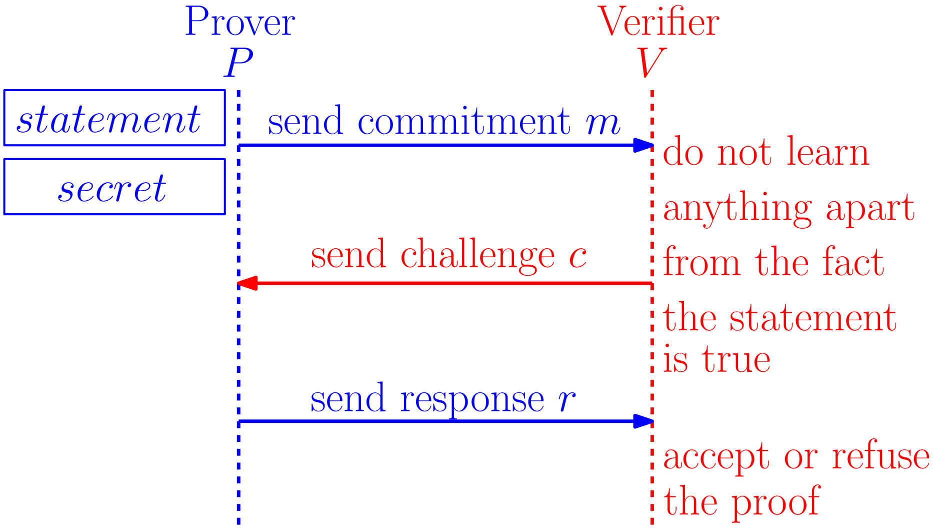
Interactive ZKP

A prover P must convince a verifier V that a statement is true. In order to prove the statement, it knows a secret but it does not want to divulge it.

Zero-knowledge proof of knowledge

Special case when the statement consists only of the fact that the prover P possesses the secret information.

Interactive ZKP



Desired properties of a ZKP protocol

- **Completeness**: V accept if P has the secret and follows the protocol
- **Zero-Knowledge**: V only learns that P knows the secret
- **Soundness**: if P does not know the secret then V must reject the proof with high probability

ZKP for knowledge of a logarithm

Public context

P and Q agrees on a cyclic group G of order q and the public key p of P

P must convince V that it knows its secret key s s.t. $p = g^s$

Three rounds of communications

1. $P \to V$: P pick a random n and sends $m = g^n$
2. $V \to P$: V pick a random $c \in [0, q-1]$ and sends c
3. $P \to V$: P computes $r = n + sc \pmod q$ and sends r

V accepts iff $m = g^r p^{-c}$

Proof of this ZPK

Proof of completeness:

If the protocol goes as expected, we have :

$g^r p^{-c} = g^{n+sc} g^{-sc} = g^n = m$

$\Rightarrow$ V accepts the proof

Proof of Zero-Knowledge:

Proved by simulation: a honest verifier can produce valid transcripts (with the same distribution for the challenge) that are indistinguishable from a real one (without knowing the secret).

Randomly pick c and r: the transcript $(m = g^r p^{-c}, c, r)$ is valid and indistinguishable from a real one since m and c are uniform random (m is a uniform random since its discrete log is uniform random)

Proof of this ZPK

Proof of soundness:
Sufficient to show a “special-soundness” property
Assuming two distinct valid transcripts with the same commitment, then we can deduce (in polynomial time) the secret from those two transcripts.

Let (m, c, r) and (m, c', r') be two distinct valid transcripts.
Then $m = g^r p^{-c} = g^{r'} p^{-c'}$, and

$$p^{c'-c} = g^{s(c'-c)} = g^{r'-r}$$

Since the transcripts are distinct, then $c \neq c'$ (otherwise we would also have $r = r'$), and we deduce
$s = (r' - r)/(c' - c) \mod q$.

Non-interactive ZKP

We can use the Fiat-Shamir technique to reduce the number of rounds.

Idea: replace the random challenge c from V by a value generated by a hash function agreed upon in advance.

P must convince V that it knows some secret s s.t. $p = g^s$

1. P pick a random n
   Then compute $c = h(g^s \parallel g^n)$ and $r = n - sc \mod q$ and sends (r, c) to P (with $\parallel$ the concatenation)
2. V accepts (r, c) with $p = g^s$ iff $c = h(p \parallel A)$ where $A = g^r p^c$.

Schnorr signature scheme

Schnorr signature scheme

Used in Belenios, by voters to sign their vote (proving the legitimacy of the ballot).

- designed by Schnorr in 1989.
- uses a group G for which the discrete logarithm is hard to solve
- uses a cryptographic hash function h
- ZKP of the knowledge of a discrete logarithm

Private signing key s

An integer s randomly chosen from $\{1, \ldots, q-1\}$.

Public verification key p

$p = g^s$

Schnorr signature scheme

Idea: Use a non-interactive ZKP of a discrete logarithm with a message as part of the input of the hash to obtain a digital signature scheme.

Public information

- a group G of order q
- a cryptographic hash function h

Signing a message M

1. Choose a random integer n from $\{1, \ldots, q-1\}$
2. Compute $c = h(M \parallel g^n)$ with $\parallel$ the concatenation
3. Produce the signature of M:
   $\mathrm{sign}(M) = (n - sc \pmod{q}, c)$

Schnorr signature scheme

Verifying a signed message M

Given a message M, a signature (r, c) and verifying key p :

1. compute $a = g^r p^c$
2. if $c = h(M \parallel a)$ then accept the signature

A correctly signed message will verify correctly.
Recall that $r = n - sc \pmod{q}$.
We have $a = g^r v^c = g^{n-sc \pmod{q}}(g^s)^c = g^n$.

Assumptions for security

- intractability of discrete logarithm
- h is collision resistant

Pedersen's threshold secret sharing scheme

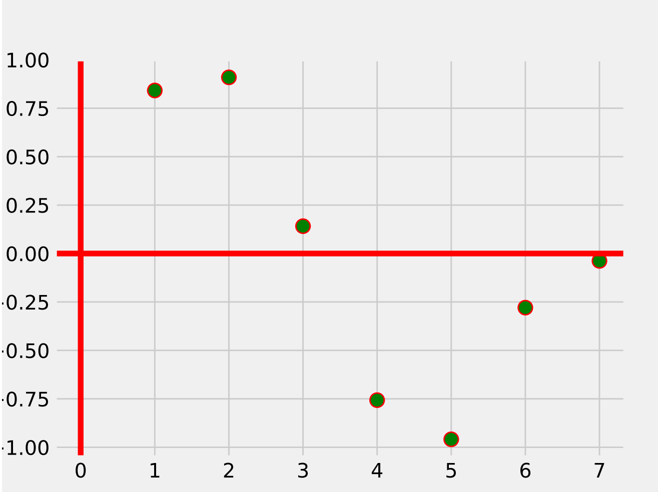
Pedersen's threshold sharing scheme

Used to share the private key of the election between several trustees s.t. the key is safe if few trustees are compromised.

Rough idea of the scheme

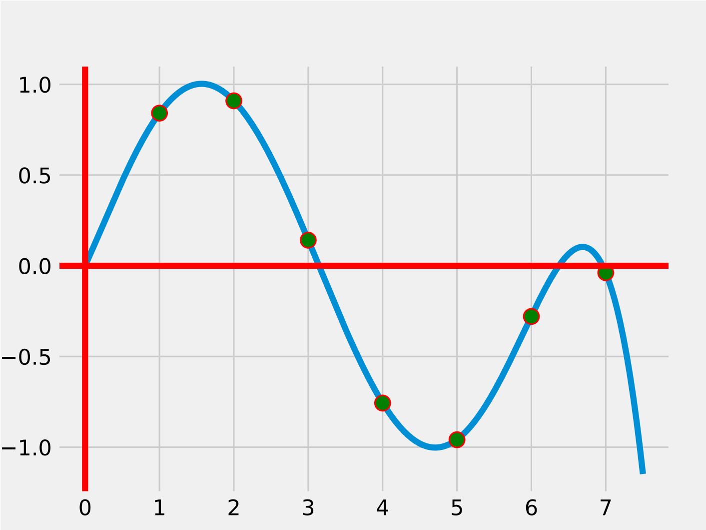
- Each trustee generate a secret (intuitively its part of the private decryption key)
- Each trustee will generate a polynomial of degree t which has a value equal to the secret for $x = 0$
- Each trustee share a distinct point of the polynomial to each other trustee
- With $t + 1$ trustees ($t + 1$ points for each polynomial), it is possible to decrypt

Sharing a secret with a polynomial



Goal: finding a polynomial of degree t passing through $t+1$ points.

Sharing a secret with a polynomial



The secret is the value of the polynomial for $x = 0$

Pedersen's threshold secret sharing scheme

- Each trustee T_i (for $1 \leq i \leq n$) has:
  - a secret s_i
  - its part of the public encryption key $e_i = g^{s_i}$
- The public global encryption key is $E = \prod_{i=1}^{n} e_i$
- The virtual decryption key is $D = \sum_{i=1}^{n} s_i$ but only $t + 1$ trustees are needed to decrypt.

First step of the algorithm

- Each trustee T_i randomly generates a polynomial of degree t in $\mathbb{Z}_q$
  $P_i(x) = c_{i,0} + c_{i,1}x + c_{i,2}x^2 + \cdots + c_{i,t}x^t$
- The secret of T_i is $s_i := c_{i,0} = P_i(0)$
- The share of the secret key of T_j is $d_j := \sum_{i \in Q} P_i(j)$
- T_i broadcasts $a_{i,j} = g^{c_{i,j}}$ for $1 \leq j \leq t$ to everyone
- T_i sends $P_i(j)$ to T_j.

$\Rightarrow$ Each T_j can check values sent by T_i with:

$$g^{P_i(j)} = g^{\sum_{k=0}^{t} c_{i,k}.j^k} = \prod_{k=0}^{t} a_{i,k}^{j^k}$$

Malicious trustees (sending values that does not check out or falsely complaining about a valid trustee) are black listed (Q set of indexes of non-black listed trustees).

Verification keys/encryption

Verification keys

Each T_j shares its verification key:

$$v_j = \prod_{i \in Q} g^{P_i(j)} = g^{\sum_{i \in Q} P_i(j)} = g^{d_j}$$

Somehow proves that it knows the values $P_i(j)$ that can be checked by everyone knowing the $a_{i,k}$ since:

$$v_j = \prod_{i \in Q} g^{P_i(j)} = \prod_{i \in Q} \prod_{k=0}^{t} a_{i,k}^{j^k}$$

Encryption

Each message is encrypted with key $E = \prod_{i \in Q} e_i$

Encrypted vote m: $(R := g^r, S := E^r.m)$ with random r

Decryption

For an encrypted message $(R := g^r, S := E^r.g^v)$, T_i outputs a decryption share:
$(i, D_i := R^{d_i})$ with $d_i := \sum_{i \in Q} P_i(j)$
For decryption, we assume that we have:

- an encrypted message (R, S)
- $t + 1$ decryption shares (j, D_j) for $j \in I := \{i_1, \ldots, i_{t+1}\}$

The algorithms outputs:

$$m = S.\left(\prod_{j \in I} D_j^{\ell_j}\right)^{-1}$$

Lagrange coefficients

Lagrange coeffients

$$\ell_j := \prod_{k=0,k\neq j}^{k=t} \frac{x_k}{x_k - x_j}$$

Lagrange interpolation

Given $t+1$ points (x_i, y_i) of a polynomial curve P, we can compute:

$$P(0) = \sum_{j=0}^{t} y_j.\ell_j$$

Lagrange coefficients

We compute Lagrange coefficients for points $(i, P_j(i))$:

$$\ell_j := \prod_{k \in I \setminus \{j\}} \frac{k}{k-j}$$

For any polynomials P, we have:

$$P(0) = \sum_{j=0}^{t} P(j).\ell_j$$

Completeness of the scheme

Consider the encrypted message $(R, S) = (g^r, E^r.m)$

We have:

$$\sum_{j\in I} \ell_j d_j = \sum_{j\in I} \ell_j \left(\sum_{i\in Q} P_i(j) \right) = \sum_{i\in Q} \left(\sum_{j\in I} \ell_j P_i(j) \right) = \sum_{i\in Q} P_i(0)$$

$$\prod_{j\in I} D_j^{\ell_j} = \prod_{j\in I} (R^{d_j})^{\ell_j} = R^{\sum_{j\in I} \ell_j d_j} = R^D$$

Hence the algorithm outputs

$$S. \left(\prod_{j\in I} D_j^{\ell_j} \right)^{-1} = S.R^{-D} = g^{Dr}.m.g^{-Dr} = m$$

Vote result

In belenios, we compute the result of the election which is the product of the encrypted ballots:

$$\begin{aligned} \mathrm{res} &= \left(\prod_{i=1}^{n} g^{r_i}, \prod_{i=1}^{n} e^{r_i} g^{v_i}\right) \\ &= \left(g^{\sum_{i=1}^{n} r_i}, e^{\sum_{i=1}^{n} r_i} g^{\sum_{i=1}^{n} v_i}\right) \\ &= \mathrm{enc}_e\left(\sum_{i=1}^{n} v_i, \sum_{i=1}^{n} r_i\right) \end{aligned}$$

After decryption, we obtain $g^{\sum_{i=1}^{n} v_i}$ and we can compute $\sum_{i=1}^{n} v_i$ since the discrete logarithm is tractable for small values.

Conclusion ?

Conclusion ?

We have seen all the cryptographic tools used by Belenios.

- Cryptographic hash function
- Non-interactive Zero-Knowledge Proofs (ZPK)
- Schnorr signature scheme
- Pedersen's threshold secret sharing scheme

Maybe on a next DALGO seminar, we will see more details on how it is implemented.