A new blockchain-based secure e-voting protocol

Chiara Spadafora
c.spadaf@libero.it
https://www.linkedin.com/in/chiara-spadafora/

Seminario De Cifris Athesis
May 29, 2020

Contents

Introduction

Introduction

Why e-voting?

- Somebody believes that traditional voting schemes cannot be trusted anymore.
- Possible solution to the problem of decreasing turn-out in elections.
- Reduction of the costs of running elections.

E-voting protocols should provide two properties:

- **Ballot Casting Assurance**: each voter gains personal assurance that their vote was correctly cast.
- **Universal Verifiability**: any observer can verify that all cast votes were properly tallied.

Historical Notes

- Introduction of democracy by the Athenians, 6^{th} century $B.C.$,
- Borda count, Jean-Charles de Borda, 1770,
- Chartist voting machine, Benjamin Jolly, 1838,
- First voting machine appropriate for use, Anthony Beranek, 1881,
- IBM voting machine, 1936,
- First e-voting protocol, Chaum, 1981.

Typical remote voting stages

- **Setup and Registration.** The system is initialized and the necessary information are made available.
- **Voting Phase.** In this stage voters can vote for the candidate they prefer.
- **Tallying.** In the last stage, tallies verify and validate ballots, count them and publish the results.

General requirements for remote voting systems

- **Transparency.** The voting system should be understandable in all its components.
- **Accuracy.** It is not possible for a casted vote to be altered nor for an invalid vote to be counted in the final tally.
- **Verifiability.** The correctness of elections results can be verified by all observers.

Requirements my protocol aimed to accomplish

A secure voting scheme should be *robust* and *resistant* to both coercion and vote-selling.

- **Coercion-Resistant**: voters can cast their ballots as they want, even if someone tries to actively force them to vote for a specific candidate.
- **Vote-Selling Resistant**: it is not possible to produce a document that undoubtedly demonstrates for which candidate a voter has voted.

State of the Art

- **Civitas** is the first electronic voting system that is coercion-resistant (in fact, a voter under coercion can vote with fake credentials so that in the tallying his vote is not counted), universally and voter verifiable, and suitable for remote voting.
- **Helios** uses homomorphic encryption to ensure ballot secrecy. Anyone can cast a ballot; however, for the final vote to be counted, the voter's identification must be verified.
- **Caveat Coercitor** is a remote voting scheme which proposes a change of perspective, replacing the requirement of coercion-resistance with a new requirement of coercion-evidence: there should be public evidence of the amount of coercion that has taken place during a particular execution of the voting system.
- **Bingo Voting** is a verifiable and coercion-free voting scheme, which is based on a trusted random number generator.

Our protocol I

This is a high level presentation of the new e-voting protocol that I have developed for my master degree thesis under the supervision of Prof. Massimiliano Sala and Prof. Riccardo Longo.

- Public but permissioned blockchain.
- Voting = spending a *v-token*.
- One *v-token* is valid, the others are fake.
- In the tally the fake *v-tokens* are erased.

A *v-tokens* is a blockchain token which is the representation of a vote.
The next slides (which were not part of the seminar) were added to give a schematic representation of the two-candidates protocol.

Our protocol II

The protocol has been designed to run on a public but permissioned blockchain (i.e. only allowed voters can take part in the election).
There are two authorities involved.

Setup:

- Authority $\mathcal{A}_1$ creates, for every voter v_i, the initial ballot $\bar{b}_i$, comprising of two *v-tokens* (one valid and one fake) and sends it to $\mathcal{A}_2$.
- $\mathcal{A}_1$ gives to each voter the information on which *v-token* is valid and which is fake.
- $\mathcal{A}_2$ creates the final ballot b_i and sends it to the respective voter v_i.
- Both $\mathcal{A}_1$ and $\mathcal{A}_2$ generate two numbers: one for the first candidate and one for the second candidate. These numbers will be used to mask the *v-tokens* in the voting phase. Eventually both $\mathcal{A}_1$ and $\mathcal{A}_2$ make a commitment on the values generated.

Our protocol III

Voting Phase:

- The voter v_i sends off chain to $\mathcal{A}_1$ his ballot b_i adding the information on which *v-token* is meant for the first candidate.
- $\mathcal{A}_1$ receives the ballot and masks the tokens with the candidate's mask. Then $\mathcal{A}_1$ sends the ballot to $\mathcal{A}_2$ which acts in the same way.
- Eventually $\mathcal{A}_2$ sends back the ballot to v_i which, with a blockchain transaction, sends his tokens to the respective candidates.
- A transaction is valid if and only if a voter voted with both of his tokens.
- A voter can abstain from voting without mining the integrity of the protocol.

Our protocol IV

Tallying

- Both authorities decommit the values committed in the setup phase.
- Anyone can multiply the (value of the) tokens contained in each candidate's wallet. Then, with a brute force attack, the number of valid votes is retrieved. In this way the winner of the election is found.
- Anyone can check the correctness of the election thanks to a set of zero knowledge proofs.

Mathematical preliminaries

Decisional Diffie-Hellman Assumption

.

Definition (DDH Assumption)

Let a, b, $z \in \mathbb{Z}_p$ be chosen at random and g be a generator of the cyclic group $\mathbb{G}$ of prime order p. The decisional Diffie-Hellman assumption holds if no probabilistic polynomial-time algorithm $\mathbb{B}$ can efficiently distinguish between the tuples $\left(g, g^a, g^b, g^{ab}\right)$ and $\left(g, g^a, g^b, g^z\right)$.

Zero Knowledge Proofs

Protocol (Equality of discrete logarithms)

Let $\mathbb{G}$ be a cyclic group of prime order p, let u, h be generators of $\mathbb{G}$, and finally let $y, z \in \mathbb{G}$, $\omega \in \mathbb{Z}_p$. The prover knows ω and wants to convince the verifier that:

$$u^{\omega} = y \quad \text{and} \quad h^{\omega} = z, \tag{1}$$

without disclosing ω. The values of u, y, h and z are publicly known.

1. *The prover generates a random r and computes $t_1 = u^r$ and $t_2 = h^r$, then sends (t_1, t_2) to the verifier.*
2. *The verifier computes a random $c \in \{0, 1\}$ and sends it to the prover.*
3. *The prover creates a response $s = r + c \cdot \omega$ and sends s to the verifier.*
4. *The verifier checks that $u^s = y^c \cdot t_1$, $h^s = z^c \cdot t_2$. If the check fails the proof fails and the protocols aborts. The previous steps are repeated a polynomial number of times t.*

Zero Knowledge Proofs

Protocol (Equality of discrete logarithms)

Let $\mathbb{G}$ be a cyclic group of prime order p, let u, h be generators of $\mathbb{G}$, and finally let $y, z \in \mathbb{G}$, $\omega \in \mathbb{Z}_p$. The prover knows ω and wants to convince the verifier that:

$$u^{\omega} = y \quad \text{and} \quad h^{\omega} = z, \tag{1}$$

without disclosing ω. The values of u, y, h and z are publicly known.

1. *The prover generates a random r and computes $t_1 = u^r$ and $t_2 = h^r$, then sends (t_1, t_2) to the verifier.*
2. *The verifier computes a random $c \in \{0, 1\}$ and sends it to the prover.*
3. *The prover creates a response $s = r + c \cdot \omega$ and sends s to the verifier.*
4. *The verifier checks that $u^s = y^c \cdot t_1$, $h^s = z^c \cdot t_2$. If the check fails the proof fails and the protocols aborts. The previous steps are repeated a polynomial number of times t.*

Zero Knowledge Proofs

Protocol (Equality of discrete logarithms)

Let $\mathbb{G}$ be a cyclic group of prime order p, let u, h be generators of $\mathbb{G}$, and finally let $y, z \in \mathbb{G}$, $\omega \in \mathbb{Z}_p$. The prover knows ω and wants to convince the verifier that:

$$u^{\omega} = y \quad \text{and} \quad h^{\omega} = z, \tag{1}$$

without disclosing ω. The values of u, y, h and z are publicly known.

1. *The prover generates a random r and computes $t_1 = u^r$ and $t_2 = h^r$, then sends (t_1, t_2) to the verifier.*
2. *The verifier computes a random $c \in \{0, 1\}$ and sends it to the prover.*
3. *The prover creates a response $s = r + c \cdot \omega$ and sends s to the verifier.*
4. *The verifier checks that $u^s = y^c \cdot t_1$, $h^s = z^c \cdot t_2$. If the check fails the proof fails and the protocols aborts. The previous steps are repeated a polynomial number of times t.*

Zero Knowledge Proofs

Protocol (Equality of discrete logarithms)

Let $\mathbb{G}$ be a cyclic group of prime order p, let u, h be generators of $\mathbb{G}$, and finally let $y, z \in \mathbb{G}$, $\omega \in \mathbb{Z}_p$. The prover knows ω and wants to convince the verifier that:

$$u^{\omega} = y \quad \text{and} \quad h^{\omega} = z, \tag{1}$$

without disclosing ω. The values of u, y, h and z are publicly known.

1. *The prover generates a random r and computes $t_1 = u^r \quad \text{and} \quad t_2 = h^r$, then sends (t_1, t_2) to the verifier.*
2. *The verifier computes a random $c \in \{0, 1\}$ and sends it to the prover.*
3. *The prover creates a response $s = r + c \cdot \omega$ and sends s to the verifier.*
4. *The verifier checks that $u^s = y^c \cdot t_1$, $h^s = z^c \cdot t_2$. If the check fails the proof fails and the protocols aborts. The previous steps are repeated a polynomial number of times t.*

Two-Candidates Protocol: technical description

Key components

The key components involved in the protocol are:

1. A finite set of voters $V = \{v_1, \dots, v_N\}$ with $N \in \mathbb{N}$ the number of eligible voters.
2. Two distinct candidates named *Alpha* and *Beta*.
3. Two different trusted authorities $\mathcal{A}_1$ and $\mathcal{A}_2$.
4. One ballot b_i comprising two *v-tokens* for $i \in \{1 \dots N\}$, i.e. one for each eligible voter.

Setup I

- $\mathcal{A}_1$ chooses uniformly at random the values r, k, λ in $\mathbb{Z}_p$.
- $\mathcal{A}_1$ chooses uniformly at random for every voter v_i the values $x_i, y_i' \in \mathbb{Z}_p$, and for every $i \in \{1 \dots N\}$ it commits to the couples (v_i, g^{rx_i}), $(v_i, g^{y_i'})$, $(v_i, g^{x_i y_i'})$.

Remark

$\mathcal{A}_1$ knows that the *v-tokens* computed using k at the exponent are valid, while the ones computed using λ are fake, but this information is kept secret.

Setup II

- For both candidates $\mathcal{A}_1$ and $\mathcal{A}_2$ choose at random a value in $\mathbb{Z}_p$: α', β' and α'', β'' respectively.
- $\mathcal{A}_1$ commits to the values g^r, g^k, g^λ, $g^{\alpha'}$, $g^{\beta'}$, $g^{\alpha' k}$, $g^{\beta' k}$, $g^{\alpha' \lambda}$, $g^{\beta' \lambda}$.
- $\mathcal{A}_2$ commits to the values $g^{\alpha''}$, $g^{\beta''}$.

Setup III

- For every voter v_i $\mathcal{A}_1$ chooses uniformly at random $\pi_i \in \{1, 2\}$ and creates the preliminary ballot

$$\bar{b}_i = \left(g^{y_i'(x_i+\sigma_{i,\mathbf{1}})}, g^{y_i'(x_i+\sigma_{i,\mathbf{2}})}\right) \tag{2}$$

where:

$$\sigma_{i,j} := \begin{cases} k \iff \pi_i = j & \text{i.e. } \bar{b}_{i,j} \text{ is real} \\ \lambda \text{ otherwise,} & \text{i.e. } \bar{b}_{i,j} \text{ is fake} \end{cases} \tag{3}$$

In this notation, i represents the voter while $j = 1, 2$ is the *v-token* position in the couple.

- π_i, i.e. the information about which token is real, is communicated by $\mathcal{A}_1$ to v_i in a safe and controlled environment (e.g. a police station).

Setup IV

- Eventually $\mathcal{A}_2$ picks at random $y_i'' \in \mathbb{Z}_p$ for all $i \neq i' \in \{1 \dots N\}$. In this way it creates the final ballot for every voter v_i:

$$b_i = \bar{b}_i^{y_i''} = \left(g^{y_i\left(x_i+\sigma_{i,\mathbf{1}}\right)}, g^{y_i\left(x_i+\sigma_{i,\mathbf{2}}\right)}\right), \quad \text{with} \quad y_i := y_i' \cdot y_i''. \qquad (4)$$

Then for every $i \in \{1 \dots N\}$ it commits to the couple $(v_i, g^{y_i''})$.

Voting Phase I

- Since the *v-tokens* can be reordered, the voter orders them so that the first *v-token* is meant for *Alpha* and the second for *Beta*.
- $\mathcal{A}_1$ computes:

$$\bar{b}_{i_\alpha} = b_{i,1}^{\frac{\alpha'}{y_i'}} = \left(g^{y_i\left(x_i+\sigma_{i,\mathbf{1}}\right)}\right)^{\frac{\alpha'}{y_i'}} = \left(g^{\alpha'\cdot y_i''\left(x_i+\sigma_{i,\mathbf{1}}\right)}\right), \tag{5}$$

$$\bar{b}_{i_\beta} = b_{i,2}^{\frac{\beta'}{y_i'}} = \left(g^{y_i\left(x_i+\sigma_{i,\mathbf{2}}\right)}\right)^{\frac{\beta'}{y_i'}} = \left(g^{\beta'\cdot y_i''\left(x_i+\sigma_{i,\mathbf{2}}\right)}\right), \tag{6}$$

and sends off-chain the couple $\bar{b}_{i_{\alpha\beta}} = (\bar{b}_{i_\alpha}, \bar{b}_{i_\beta})$ to $\mathcal{A}_2$.

Voting Phase II

- $\mathcal{A}_2$ then computes the final vote (let $\alpha := \alpha' \cdot \alpha''$ and $\beta := \beta' \cdot \beta''$):

$$b_{i_\alpha} = \bar{b}_{i_\alpha}^{\frac{\alpha''}{y_i''}} = \left(g^{\alpha' \cdot y_i''(x_i + \sigma_{i,1})}\right)^{\frac{\alpha''}{y_i''}} = \left(g^{\alpha(x_i + \sigma_{i,1})}\right), \quad (7)$$

$$b_{i_\beta} = \bar{b}_{i_\beta}^{\frac{\beta''}{y_i''}} = \left(g^{\beta' \cdot y_i''(x_i + \sigma_{i,2})}\right)^{\frac{\beta''}{y_i''}} = \left(g^{\beta(x_i + \sigma_{i,2})}\right), \quad (8)$$

  and sends off-chain the couple $b_{i_{\alpha\beta}} = (b_{i_\alpha}, b_{i_\beta})$ back to the voter.
- The two masked *v-tokens* are sent with a transaction on the blockchain to the respective candidates. The voter receives the receipt of its vote.

Tallying I

- Suppose the first $T \leq N$ voters voted.
- $\mathcal{A}_1$ and $\mathcal{A}_2$ publish the decommitments, excluding the couples $(v_i, g^{y_i'})$, $(v_i, g^{x_i y_i'})$, $(v_i, g^{y_i''})$ $\forall i$.
- $\mathcal{A}_1$ computes $g^{\alpha} = (g^{\alpha''})^{\alpha'}$, and similarly computes g^{β}, $g^{\alpha k}$, $g^{\beta k}$, $g^{\alpha\lambda}$, $g^{\beta\lambda}$, then publishes all these values.
- $\mathcal{A}_1$ computes $\mathtt{sum} = \sum_{s=1}^{T} x_s$, then it publishes $g^{\alpha \cdot \mathtt{sum}}$, $g^{\beta \cdot \mathtt{sum}}$, $g^{r \cdot \mathtt{sum}}$.

Tallying II

- Multiplying all *v-tokens* in *Alpha*'s wallet and dividing by $g^{\alpha \cdot \mathtt{sum}}$ anyone can compute:

$$\left(g^{\alpha \cdot \mathtt{sum}}\right)^{-1} \prod_{i=1}^{T} g^{\alpha\left(x_i + \sigma_{i,j}\right)} = \left(g^{\alpha k}\right)^{\mathtt{valid}_\alpha} \left(g^{\alpha \lambda}\right)^{\mathtt{fake}_\alpha} \tag{9}$$

where $j = 1$ or 2 depending on the *v-token* used.

Comparing the number of valid votes for each candidate, the winner of the elections is found.

Tallying III

Remark

- $\texttt{valid}_\alpha, \texttt{valid}_\beta \geq 1$.
- $\texttt{valid}_\alpha + \texttt{fake}_\alpha = T$.
- $\texttt{valid}_\alpha = \texttt{fake}_\beta$ and $\texttt{fake}_\alpha = \texttt{valid}_\beta$.

Correctness Check I

As first thing a ZKP is needed to assure that votes have been masked correctly. In other words, that the authorities computed

$$g^{y_i(x_i+k)} \to g^{\alpha(x_i+k)} \tag{10}$$

without messing with the exponents.
$\mathcal{A}_1$ and $\mathcal{A}_2$ decommit to the voter the values of $(v_i, g^{y_i'})$ and $(v_i, g^{y_i''})$ respectively. Then $\mathcal{A}_1$ computes g^{y_i} and proves that the result is correct using:

$$\omega = y_i', \qquad u = g, \qquad y = g^{y_i'}, \qquad h = g^{y_i''}, \qquad z = g^{y_i}. \tag{11}$$

Then $\mathcal{A}_1$ can prove the correctness of the mask setting:

$$\omega = (x_i + k), \quad u = g^{y_i}, \quad y = g^{y_i(x_i+k)}, \quad h = g^{\alpha}, \quad z = g^{\alpha(x_i+k)}. \tag{12}$$

Multi-Candidates Protocol: brief description

Multiple candidates protocol: brief overview I

The key components involved in the protocol are:

- A finite set of voters $V = \{v_1, \dots, v_N\}$ with $N \in \mathbb{N}$ the number of eligible voters.
- $\mathcal{C} > 2$ distinct candidates named $(C_1, \dots, C_{\mathcal{C}})$.
- Three different trusted authorities $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$.
- One voting sheet (ballot) comprising $\mathcal{C}$ *v-tokens*.

Multiple candidates protocol: brief overview II

- **Setup.** $\mathcal{A}_1$ creates for every voter v_i the ballot

$$b_i = \left(g^{y_{i_1}\left(x_i+\sigma_{i,1}\right)}, g^{y_{1_2}\left(x_i+\sigma_{i,2}\right)}, g^{y_{i_3}\left(x_i+\sigma_{i,3}\right)}, \ldots, g^{y_{i_\mathcal{C}}\left(x_i+\sigma_{i,\mathcal{C}}\right)}\right).$$

- **Voting Phase.** Since the *v-tokens* cannot be reordered, the voter reorders the list of the candidates to let it match the way it wants to vote.
- **Tallying.** For every candidate C_u with its associated mask ω_u, the product of the *v-tokens* in its wallet is

$$\left(g^{\omega_u\left(x_1+\sigma_{1,j_1}\right)}\right)\cdots\left(g^{\omega_u\left(x_T+\sigma_{T,j_T}\right)}\right) = \left(g^{\omega_u\left((\texttt{valid}_{\omega_u})\cdot k+(\texttt{fake}_{\omega_u})\cdot\lambda+\sum_{s=1}^{T} x_s\right)}\right).$$

Proof of security of the Two-Candidates Protocol

Basic aspects of a proof of security

Formally, the security of a scheme relies on an *assumption*. In a formal proof of security, there are two parties involved:

- **Challenger** $\mathcal{C}$ which runs the algorithms of the protocol.
- **Adversary** $\mathcal{A}$ which tries to break the scheme making queries to $\mathcal{C}$.

Security Game

Definition (Security Game)

The security game for a two-candidates protocol proceeds as follows:

- **Init** The adversary $\mathcal{A}$ chooses $N-2$ users that it will control.
- **Setup** The *Challenger* controls both authorities and the other two voters.
- **Phase 0** The adversary may request to see the *v-tokens* of any voter.
- **Phase 1** The adversary may request either to see some v-tokens or ask to some voters it controls, to vote and see the receipt.
- **Challenge** The *Challenger* votes randomly with the two voters it controls (for different candidates).
- **Phase 2** Phase 1 is repeated.
- **Phase 3** The voting phase ends and the values committed by the authorities are decommitted. The votes are counted and the adversary can request some ZKP of the correctness of the results.
- **Guess** The adversary outputs a guess on the challenge.

Security Game

Definition (Security Game)

The security game for a two-candidates protocol proceeds as follows:

- **Init** The adversary $\mathcal{A}$ chooses $N-2$ users that it will control.
- **Setup** The *Challenger* controls both authorities and the other two voters.
- **Phase 0** The adversary may request to see the *v-tokens* of any voter.
- **Phase 1** The adversary may request either to see some v-tokens or ask to some voters it controls, to vote and see the receipt.
- **Challenge** The *Challenger* votes randomly with the two voters it controls (for different candidates).
- **Phase 2** Phase 1 is repeated.
- **Phase 3** The voting phase ends and the values committed by the authorities are decommitted. The votes are counted and the adversary can request some ZKP of the correctness of the results.
- **Guess** The adversary outputs a guess on the challenge.

Security Game

Definition (Security Game)

The security game for a two-candidates protocol proceeds as follows:

- **Init** The adversary $\mathcal{A}$ chooses $N-2$ users that it will control.
- **Setup** The *Challenger* controls both authorities and the other two voters.
- **Phase 0** The adversary may request to see the *v-tokens* of any voter.
- **Phase 1** The adversary may request either to see some v-tokens or ask to some voters it controls, to vote and see the receipt.
- **Challenge** The *Challenger* votes randomly with the two voters it controls (for different candidates).
- **Phase 2** Phase 1 is repeated.
- **Phase 3** The voting phase ends and the values committed by the authorities are decommitted. The votes are counted and the adversary can request some ZKP of the correctness of the results.
- **Guess** The adversary outputs a guess on the challenge.

Security Game

Definition (Security Game)

The security game for a two-candidates protocol proceeds as follows:

- **Init** The adversary $\mathcal{A}$ chooses $N-2$ users that it will control.
- **Setup** The *Challenger* controls both authorities and the other two voters.
- **Phase 0** The adversary may request to see the *v-tokens* of any voter.
- **Phase 1** The adversary may request either to see some v-tokens or ask to some voters it controls, to vote and see the receipt.
- **Challenge** The *Challenger* votes randomly with the two voters it controls (for different candidates).
- **Phase 2** Phase 1 is repeated.
- **Phase 3** The voting phase ends and the values committed by the authorities are decommitted. The votes are counted and the adversary can request some ZKP of the correctness of the results.
- **Guess** The adversary outputs a guess on the challenge.

Security Game

Definition (Security Game)

The security game for a two-candidates protocol proceeds as follows:

- **Init** The adversary $\mathcal{A}$ chooses $N-2$ users that it will control.
- **Setup** The *Challenger* controls both authorities and the other two voters.
- **Phase 0** The adversary may request to see the *v-tokens* of any voter.
- **Phase 1** The adversary may request either to see some v-tokens or ask to some voters it controls, to vote and see the receipt.
- **Challenge** The *Challenger* votes randomly with the two voters it controls (for different candidates).
- **Phase 2** Phase 1 is repeated.
- **Phase 3** The voting phase ends and the values committed by the authorities are decommitted. The votes are counted and the adversary can request some ZKP of the correctness of the results.
- **Guess** The adversary outputs a guess on the challenge.

Security Game

Definition (Security Game)

The security game for a two-candidates protocol proceeds as follows:

- **Init** The adversary $\mathcal{A}$ chooses $N - 2$ users that it will control.
- **Setup** The *Challenger* controls both authorities and the other two voters.
- **Phase 0** The adversary may request to see the *v-tokens* of any voter.
- **Phase 1** The adversary may request either to see some v-tokens or ask to some voters it controls, to vote and see the receipt.
- **Challenge** The *Challenger* votes randomly with the two voters it controls (for different candidates).
- **Phase 2** Phase 1 is repeated.
- **Phase 3** The voting phase ends and the values committed by the authorities are decommitted. The votes are counted and the adversary can request some ZKP of the correctness of the results.
- **Guess** The adversary outputs a guess on the challenge.

Security Game

Definition (Security Game)

The security game for a two-candidates protocol proceeds as follows:

- **Init** The adversary $\mathcal{A}$ chooses $N-2$ users that it will control.
- **Setup** The *Challenger* controls both authorities and the other two voters.
- **Phase 0** The adversary may request to see the *v-tokens* of any voter.
- **Phase 1** The adversary may request either to see some v-tokens or ask to some voters it controls, to vote and see the receipt.
- **Challenge** The *Challenger* votes randomly with the two voters it controls (for different candidates).
- **Phase 2** Phase 1 is repeated.
- **Phase 3** The voting phase ends and the values committed by the authorities are decommitted. The votes are counted and the adversary can request some ZKP of the correctness of the results.
- **Guess** The adversary outputs a guess on the challenge.

Security Game

Definition (Security Game)

The security game for a two-candidates protocol proceeds as follows:

- **Init** The adversary $\mathcal{A}$ chooses $N-2$ users that it will control.
- **Setup** The *Challenger* controls both authorities and the other two voters.
- **Phase 0** The adversary may request to see the *v-tokens* of any voter.
- **Phase 1** The adversary may request either to see some v-tokens or ask to some voters it controls, to vote and see the receipt.
- **Challenge** The *Challenger* votes randomly with the two voters it controls (for different candidates).
- **Phase 2** Phase 1 is repeated.
- **Phase 3** The voting phase ends and the values committed by the authorities are decommitted. The votes are counted and the adversary can request some ZKP of the correctness of the results.
- **Guess** The adversary outputs a guess on the challenge.

Vote Indistinguishability

Definition (Vote Indistiguishability)

A Two-Candidates Protocol with security parameter ξ is VI-secure if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function ϕ such that:

$$\mathbb{P}[\mathcal{A} \text{ wins}] \leq \frac{1}{2} + \phi(\xi). \tag{13}$$

The protocol is proven VI-secure under the DDH assumption in the security game defined above.

Proof of security I

Theorem

Suppose that the commitment scheme is perfectly hiding and computationally binding. If an adaptive distinguisher adversary can break the scheme, then a simulator can be constructed to play the decisional Diffie-Hellman game with non-negligible advantage.

Corollary

Suppose that the Decisional Diffie-Hellman problem is an hard mathematical problem, then no adaptive distinguisher adversary can break the scheme.

Remark

The security of the protocol will be proven in terms of two authorities, $\mathcal{A}_1$ honest and $\mathcal{A}_2$ honest but *leaky*.

Proof of security II

Recall that *ballot* is composed of:

$$b_i = \left(g^{y_i(x_i+k)}, g^{y_i(x_i+\lambda)}\right).$$

The simulator takes in a DDH challenge $(g, A = g^a, B = g^b, T)$ with

$$T = g^{ab} \quad \text{or} \quad T = R = g^z.$$

Setup. The two uncontrolled voters are v_1 and v_2. The simulator implicitly sets:

$$y_1' = \frac{\bar{y}_1}{a}, \qquad y_2' = \frac{\bar{y}_2}{a}, \qquad k = \bar{k} \cdot a, \qquad \lambda = \bar{\lambda} \cdot a.$$

while all the other parameters are chosen uniformly at random.
The parameters of $\mathcal{A}_2$ are **leaked** to the adversary and the required commitments are done.

Proof of security III

Setup. The *v-tokens* of the uncontrolled voters are constructed in this way:

$$b_1 = \left(g^{y_1'' \bar{y}_1 (d+\bar{k}-\bar{\lambda})} \cdot B^{y_1'' \bar{y}_1}, B^{y_1'' \bar{y}_1} \cdot g^{y_1'' \bar{y}_1 d}\right),$$

$$b_2 = \left(B^{-y_2'' \bar{y}_2} \cdot g^{y_2'' \bar{y}_2 e}, g^{y_2'' \bar{y}_2 (e-\bar{k}+\bar{\lambda})} \cdot B^{-y_2'' \bar{y}_2}\right),$$

implicitly setting

$$\begin{aligned} x_1 + k &= (d + \bar{k} - \bar{\lambda})a + ab, \\ x_1 + \lambda &= ab + da, \\ x_2 + k &= -ab + ea, \\ x_2 + \lambda &= (e - \bar{k} + \bar{\lambda})a - ab. \end{aligned}$$

so that the DDH challenge appears only in the votes.

Proof of security IV

Phase 1 The adversary chooses some voters u, $3 \leq u \leq N$

$$V_u = \left(g^{\alpha(x_u+k)}, g^{\beta(x_u+\lambda)}\right) = \left(g^{\alpha x_u} \cdot A^{\bar{k}\alpha}, g^{\beta x_u} \cdot A^{\bar{\lambda}\beta}\right). \tag{14}$$

Challenge The votes are constructed as:

$$V_1 = \left(T^{\alpha} \cdot A^{(d+\bar{k}-\bar{\lambda})\alpha}, T^{\beta} \cdot A^{\beta \cdot d}\right), \tag{15}$$

$$V_2 = \left(T^{-\beta} \cdot A^{\beta \cdot e}, T^{-\alpha} \cdot A^{\alpha(e-\bar{k}+\bar{\lambda})}\right). \tag{16}$$

Proof of security V

Note that:

$$V_{1,\alpha} \cdot V_{2,\alpha} \cdot \prod_{i=3}^{N} V_{i,\alpha} = A^{(d+e)\alpha} \cdot \prod_{i=3}^{N} V_{i,\alpha}, \tag{17}$$

$$V_{1,\beta} \cdot V_{2,\beta} \cdot \prod_{i=3}^{N} V_{i,\beta} = A^{(d+e)\beta} \cdot \prod_{i=3}^{N} V_{i,\beta}, \tag{18}$$

so the product of all the votes received by both of the candidates does not contain the value of the challenge T.

Proof of security VI

Guess

- The adversary outputs a guess on the challenge.
- The simulator outputs 0 to guess that $T = g^{ab}$ if the guess of $\mathcal{A}$ was correct, otherwise it outputs 1 to indicate that T is random.
- If T is not random the simulator $\mathcal{S}$ gives a perfect simulation:

$$V_1 = \left(T^{\alpha} \cdot A^{(d+\bar{k}-\bar{\lambda})\alpha}, T^{\beta} \cdot A^{\beta \cdot d}\right) = \left(g^{\alpha(x_1+k)}, g^{\beta(x_1+\lambda)}\right), \tag{19}$$

$$V_2 = \left(T^{-\beta} \cdot A^{\beta \cdot e}, T^{-\alpha} \cdot A^{\alpha(e-\bar{k}+\bar{\lambda})}\right) = \left(g^{\beta(x_2+k)}, g^{\alpha(x_2+\lambda)}\right). \tag{20}$$

Proof of security VII

- This means that the advantage is preserved and so it holds that:

$$\mathbb{P}[\mathcal{S}(g, A, B, T = g^{ab}) = 0] = \frac{1}{2} + \varepsilon. \tag{21}$$

- On the contrary when T is a random element $R \in \mathbb{G}$ the votes are completely random values from the adversary point of view, so:

$$\mathbb{P}[\mathcal{S}(g, A, B, T = R) = 0] = \frac{1}{2}. \tag{22}$$

Therefore, $\mathcal{S}$ can play the DDH game with non-negligible advantage $\frac{\varepsilon}{2}$.

Conclusions

Articles:

- Two-Candidates Protocol (concluded),
- $\mathcal{C}$-Candidates Protocol.

Implementation:

- Hyperledger,
- Quadrans.

Thank you for your attention!

Appendix

This section was not part of the seminar but has been added to clarify some presented concepts.

On the honesty of $\mathcal{A}_1$ I

Finally, a voter can ask (always in a safe and authenticated environment) for a proof that in the registration phase the authority $\mathcal{A}_1$ correctly computed and identified the *v-tokens*, i.e. that the *v-token* identified as `valid` by $\mathcal{A}_1$ was the one containing k.
Recall that a *v-token* is:

$$b_{i,j} = g^{y_i(x_i+\sigma_{i,j})} = g^{y_i \cdot x_i} \cdot g^{y_i \cdot \sigma_{i,j}}, \quad \text{with} \quad \sigma_{i,j} \in \{k, \lambda\}. \tag{23}$$

$\mathcal{A}_1$ starts by decommitting $(v_i, g^{y_i' x_i})$ to the voter, then computes $g^{y_i x_i}$ from $g^{y_i''}$ setting:

$$\omega = y_i' x_i, \qquad u = g, \qquad y = g^{y_i' x_i}, \qquad h = g^{y_i''}, \qquad z = g^{y_i \cdot x_i}. \tag{24}$$

On the honesty of $\mathcal{A}_1$ II

Then the voter knows g, g^{y_i}, g^k, g^λ, $g^{r\cdot x_i}$ and g^r so first $\mathcal{A}_1$ can prove the validity of the factor $g^{y_i\cdot x_i}$ by setting:

$$\omega = x_i, \qquad u = g^r, \qquad y = g^{r\cdot x_i}, \qquad h = g^{y_i}, \qquad z = g^{y_i\cdot x_i}. \tag{25}$$

To conclude $\mathcal{A}_1$ can prove than the `valid` coin contains k while the `fake` contains λ by setting:

$$\omega = k, \qquad u = g, \qquad y = g^k, \qquad h = g^{y_i}, \qquad z = g^{y_i\cdot k}, \tag{26}$$

and:

$$\omega = \lambda, \qquad u = g, \qquad y = g^\lambda, \qquad h = g^{y_i}, \qquad z = g^{y_i\cdot \lambda}, \tag{27}$$

where the values of z can be derived by the voter dividing the *v-tokens* by $g^{y_i\cdot x_i}$ that has been proved correct in the previous step.

Security Considerations (of the ZK protocol) I

If the DDH assumption holds then:

- **Completeness.** To show that this protocol is correct, it suffices to verify that the equations of steps 3 and 4 hold when s is computed correctly.
- **Soundness.** To show soundness first note that the prover can guess all t values of the challenges c only with probability 2^{-t} which is negligible. Therefore if the prover manages to complete a proof with more than negligible probability then there has to be a repetition in which the prover does not fail even when guessing wrong, i.e. it can answer both possible challenges correctly.

Security Considerations II

- **Zero-knowledge.** To show that we use a simulator $\mathcal{S}$ that takes in input (u, y, h, z) and can interact with a (possibly malicious) verifier V producing a view that is indistinguishable from a real one, as follows:
  1. $\mathcal{S}$ initialises the verifier V with u, y, h, z and $i = 0$;
  2. $\mathcal{S}$ selects $c' \in \{0, 1\}$ at random;
  3. $\mathcal{S}$ selects $s \in \mathbb{Z}_p$ at random and sets $t_1 = u^s \cdot y^{-c'}$, $t_2 = h^s \cdot z^{-c'}$;
  4. $\mathcal{S}$ gives (t_1, t_2) and gets the challenge c;
  5. If $c \neq c'$ $\mathcal{S}$ rewinds V and goes back to step 2 with the same i, otherwise it proceeds;
  6. $\mathcal{S}$ gives s to V, since the check succeeds, if $i = t$ the proof successfully completes, otherwise $\mathcal{S}$ sets $i = i + 1$ and proceeds with the simulation repeating from step 2.

  If there exists a V that can distinguish this simulation from a real protocol interaction then we can break DDH assumption.

Commitment Scheme

A commitment scheme is composed by two algorithms:

- $\texttt{Commit}(m, r)$: takes the message m to commit with some random value r as input and outputs the commitment c and an opening value d.
- $\texttt{Verify}(c, m, d)$: takes the commitment c, the message m and the decommitment value d and outputs true if the verification succeeds, false otherwise.

A commitment scheme must have the following two properties:

- **Binding:** it is infeasible to find $m' \neq m$ and d, d' such that $\texttt{Verify}(c, m, d) = \texttt{Verify}(c, m', d') = \texttt{true}$.
- **Hiding:** Let $[c_1, d_1] = \texttt{Commit}(m_1, r_1)$ and $[c_2, d_2] = \texttt{Commit}(m_2, r_2)$ with $m_1 \neq m_2$, then it is infeasible for an attacker having only c_1, c_2, m_1 and m_2 to distinguish which c_i corresponds to which m_i.

DDHA specifications

Examples of groups for which the Decisional Diffe Hellman is believed to hold are:

- $\mathbb{Q}_p$ the subgroup of quadratic residues in $\mathbb{Z}_p$ with p safe prime.
- Cyclic groups of order $(p-1)(q-1)$ with p and q safe primes.
- Some prime order elliptic curves over $GF(p)$.