helios

Trust the vote.

Seminar Cryptography and Data Security

Fall 2022

Dominik Frey

University of Bern

Table of Contents

Outline

- First web-based open-audit voting system
- Create, run and tally elections with only support of a web browser
- End-to-end verifiable elections at reach from just anyone who can access the Internet

- Suitable for **low-coercion** elections
    - High-stakes election typically insecure due to coercion risk
    - Truly private interaction needed for coercion resistance
    - Helios does **not** deal with coercion resistance

- Trust no one for integrity, trust Helios for privacy
    - *Privacy*: unauthorized user should not be able to intercept and read data
    - *Integrity*: unauthorized user should not be able to alter data

- If all election administrators are corrupt, only one of the two properties can be guaranteed
- *Integrity*: unlikely for them to convincingly fake a tally
- *Privacy*: recruit enough trustees and hope that min subset remains honest
- ensure **privacy**

Outline

Helios 1.0

Helios server acts as a single trustee
By Adida, 2008:

1. Helios server creates pair of keys for public key encryption scheme
2. Public key used by all voters to encrypt and submit their vote
3. Server performs shuffle to cut ballots from identity of voters
4. Server decrypts all shuffled ballots individually

Helios 2.0

By Adida, de Marneffe, Pereira and Quisquater, 2008:

- Replace shuffling with **homomorphic aggregation** of votes:
  - individual *encrypted* votes are aggregated to an encryption of the election outcome
  - decrypt only aggregated election outcome
- **Distributed** encryption scheme to strengthen privacy
  - no single entity would be in touch with enough keying material to decrypt individual ballots

Outline

- Protocols make use of multiplicative cyclic group $\mathbb{G}$ of prime order q in which the Decisional Diffie-Hellman problem is believed to be hard:

  $$\text{with } a, b, c \leftarrow \mathbb{Z}_q :$$
  $$(g^a, g^b, g^{ab}) \text{ is indistinguishable from } (g^a, g^b, g^c)$$

  where $\mathbb{Z}_q$ is a subgroup of $\mathbb{Z}_p^*$ with $q = 256$ bits prime and $p = 2048$ bits prime

- Encryption of votes is based on **ElGamal** public key encryption
  - **Secret** decryption key: $x \leftarrow \mathbb{Z}_q$
  - **Public** encryption key: $y = g^x$
  - Ciphertext: $(c_1, c_2) = (g^r, my^r)$ with $r \leftarrow \mathbb{Z}_q$ and $m \in \mathbb{G}$
  - Plaintext: $m = c_2 / c_1^x$

$\Rightarrow$ Ciphertexts of votes are indistinguishable (DDH)

- Use **homomorphic** property of ElGamal encryption scheme:

$$g^a \cdot g^b = g^{a+b}$$

- For a list of candidates, a voter encrypts either a ”0” (not supported) or a ”1” (supported) for each candidate

- **Encode** the plaintext before encrypting it:

$$m \xrightarrow{encode} g^m \xrightarrow{encrypt} g^m y^r$$

Candidate A

Candidate B

Voter 1

$$0 \rightarrow g^{0} \rightarrow g^{0} y^{r_{1A}} = g^{0} g^{x r_{1A}} = g^{x r_{1A}}$$
$$(c_1, c_2) = (g^{r_{1A}}, g^{x r_{1A}})$$

$$1 \rightarrow g^{1} \rightarrow g^{1} y^{r_{1B}} = g^{1} g^{x r_{1B}} = g^{1 + x r_{1B}}$$
$$(c_1, c_2) = (g^{r_{1B}}, g^{1 + x r_{1B}})$$

Voter 2

$$1 \rightarrow g^{1} \rightarrow g^{1} y^{r_{2A}} = g^{1} g^{x r_{2A}} = g^{1 + x r_{2A}}$$
$$(c_1, c_2) = (g^{r_{2A}}, g^{1 + x r_{2A}})$$

$$0 \rightarrow g^{0} \rightarrow g^{0} y^{r_{2B}} = g^{0} g^{x r_{2B}} = g^{x r_{2B}}$$
$$(c_1, c_2) = (g^{r_{2B}}, g^{x r_{2B}})$$

Voter 3

$$1 \rightarrow g^{1} \rightarrow g^{1} y^{r_{3A}} = g^{1} g^{x r_{3A}} = g^{1 + x r_{3A}}$$
$$(c_1, c_2) = (g^{r_{3A}}, g^{1 + x r_{3A}})$$

$$0 \rightarrow g^{0} \rightarrow g^{0} y^{r_{3B}} = g^{0} g^{x r_{3B}} = g^{x r_{3B}}$$
$$(c_1, c_2) = (g^{r_{3B}}, g^{x r_{3B}})$$

- The votes are then aggregated for each candidate by multiplying the ciphertexts:
  - Candidate A

$$
\begin{aligned}
&(g^{r_{1A}}, g^{xr_{1A}}) \cdot (g^{r_{2A}}, g^{1+xr_{2A}}) \cdot (g^{r_{3A}}, g^{1+xr_{3A}}) \\
&= (g^{r_{1A}} g^{r_{2A}} g^{r_{3A}}, g^{xr_{1A}} g^{1+xr_{2A}} g^{1+xr_{3A}}) \\
&= (g^{r_{1A}+r_{2A}+r_{3A}}, g^{2+x(r_{1A}+r_{2A}+r_{3A})})
\end{aligned}
$$

  - Candidate B

$$
\begin{aligned}
&(g^{r_{1B}}, g^{1+xr_{1B}}) \cdot (g^{r_{2B}}, g^{xr_{2B}}) \cdot (g^{r_{3B}}, g^{xr_{3B}}) \\
&= (g^{r_{1B}} g^{r_{2B}} g^{r_{3B}}, g^{1+xr_{1B}} g^{xr_{2B}} g^{xr_{3B}}) \\
&= (g^{r_{1B}+r_{2B}+r_{3B}}, g^{1+x(r_{1B}+r_{2B}+r_{3B})})
\end{aligned}
$$

- Only the **aggregated** ciphertext will then be **decrypted** (Candidate A):

$$
\begin{aligned}
c_2/c_1^{x} &= \frac{g^{2+x(r_{1A}+r_{2A}+r_{3A})}}{\left(g^{r_{1A}+r_{2A}+r_{3A}}\right)^{x}} \\
&= g^{2+x(r_{1A}+r_{2A}+r_{3A})-(r_{1A}+r_{2A}+r_{3A})x} = g^{2} = g^{m}
\end{aligned}
$$

- $\log_g(g^m)$?
  - m upper bounded by # voters
  - efficient algos to calculate 40 bit discrete log

- **Distribute** key generation and distribution among set of trustees $T_1, \ldots, T_n$
  - Key pair for each T_i:

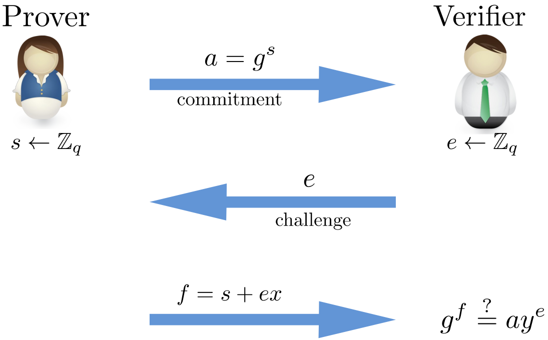
$$(x_i, y_i = g^{x_i}) \rightarrow g^x = \prod g^{x_i}$$

  - Decryption factor for each T_i:

$$d_i = c_1^{x_i} \rightarrow g^m = c_2 / \prod d_i$$

Zero-Knowledge proofs

1. T_i need to show that they know x_i (Schnorr protocol)



Prover

Verifier

$a = g^s$

commitment

$s \leftarrow \mathbb{Z}_q$

$e \leftarrow \mathbb{Z}_q$

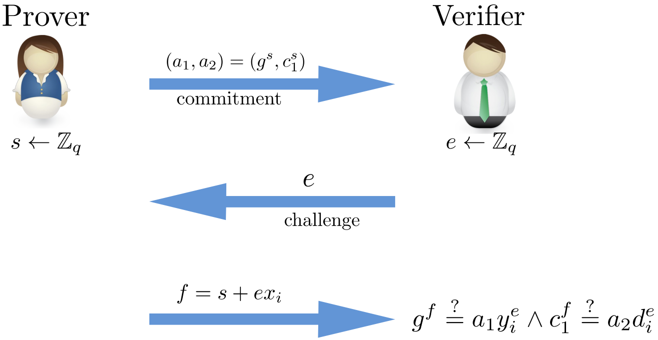
e

challenge

$f = s + ex$

$g^f \stackrel{?}{=} ay^e$

2. T_i need to show that their decryption factor is computed correctly as $d_i = c_1^{x_i}$ (Chaum-Pedersen protocol)



Prover

Verifier

$(a_1, a_2) = (g^s, c_1^s)$

commitment

$s \leftarrow \mathbb{Z}_q$

$e \leftarrow \mathbb{Z}_q$

e

challenge

$f = s + ex_i$

$g^f \overset{?}{=} a_1 y_i^e \wedge c_1^f \overset{?}{=} a_2 d_i^e$

3. Ballot validity: *disjunctive* Chaum-Pedersen with a **real** and a **simulated** proof

$$\begin{aligned} c_1 &\to y \\ d_i &\to c_2/g^v,\ v \in \{0,1\} \\ x_i &\to r \end{aligned}$$

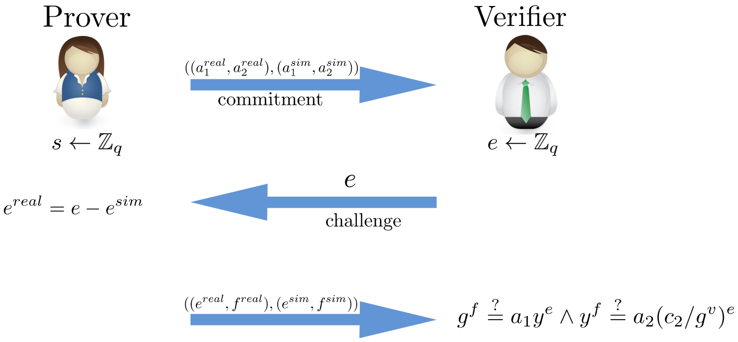
Prover creates the following tuples:

$$(e^{sim} \leftarrow \mathbb{Z}_q, f^{sim} \leftarrow \mathbb{Z}_q)$$

$$\Rightarrow (a_1^{sim}, a_2^{sim}) \begin{cases} g^{f^{sim}} = a_1^{sim} y^{e^{sim}} \\ y^{f^{sim}} = a_2^{sim} (\frac{c_2}{g^v})^{e^{sim}} \end{cases}$$

and

$$(a_1^{real}, a_2^{real}) = (g^s, y^s)$$



Prover

Verifier

$((a_1^{real}, a_2^{real}), (a_1^{sim}, a_2^{sim}))$

commitment

$s \leftarrow \mathbb{Z}_q$

$e \leftarrow \mathbb{Z}_q$

e

challenge

$e^{real} = e - e^{sim}$

$((e^{real}, f^{real}), (e^{sim}, f^{sim}))$

$g^f \overset{?}{=} a_1 y^e \wedge y^f \overset{?}{=} a_2(c_2/g^v)^e$

- Implementation based on **non-interactive** ZKP via Fiat-Shamir transformation

$$H : \{0,1\}^* \to \mathbb{Z}_q$$

E.g.: parameters of a single-choice proof in Helios

```
{
    "A": "3bZcd35GAS",
    "B": "7bXcd352sd",
    "choice_num": 0,
    "ciphertext":
        {
            "alpha": "6BtdxuEwbcs+dfs3",
            "beta": "nC345Xbadw3235SD"
        },
    "election_hash": "Nz1fWLvVLH3eY3Ox7u5hxfLZPdw",
    "question_num": 2
}
```

Outline

Outline

- Wikström and Smyth-Cortier, December 2010
  - An attacker can tamper a vote via a browser, copy it and cast it as its own
- Estehghari-Desmedt, August 2010
  - Corrupt firefox extension via Helios-specific rootkit to alter content displayed to the voter
  - User's vote is flipped and goes undetected as all verification channels are corrupted in that browser

Outline

References I

- Ben Adida, *Helios documentation: Attacks and defenses*, `https://documentation.heliosvoting.org/attacks-and-defenses`, Online; last accessed 07 December 2022.
- ———, *Helios official website*, `https://vote.heliosvoting.org/`, Online; last accessed 07 December 2022.
- ———, *Helios: Web-based open-audit voting*, USENIX Security Symposium (2008), 335–348.
- IBM, *Data integrity and privacy*, 2021, Last accessed 22 November 2022.

References II

Olivier Pereira, *Internet voting with helios*, Feng Hao and Peter Y. A. Ryan (Eds.), Real-World Electronic Voting: Design, Analysis and Deployment, CRC Press, 2016, p. 279–310.