Reversing, Breaking, and Fixing the French Legislative Election E-Voting Protocol

Alexandre Debant and Lucca Hirschi

Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

USENIX Security Symposium
August 11th 2023
Context

+1.5 millions legitimate voters (French citizens resident overseas only)

+500 000 ballots cast over the Internet (~77% of all the expressed votes)

11 deputies chosen for 5 years (11 constituencies split in ~200 consulates)

This protocol was based on a new protocol (FLEP), better be sure it is secure!
The different roles

- **Voter**: At home
- **Voting Client**: Javascript running in a browser
- **Voting Server**: @ French Ministry for Europe and Foreign Affairs
- **Decryption Trustees**: by representatives and officials
- **Third-Party**: by independent researchers
The different roles

Voter
At home

Voting Client
Javascript running in a browser

Voting Server

Decryption Trustees
by representatives and officials

Third-Party
by independent researchers

Available documentation was too lacunar to derive the workflow!
Contributions

First public and comprehensive specification of the protocol
➡️ by reversing the obfuscated voting client (Javascript & HTML)

Verifiability and vote secrecy can be attacked by a channel/server attacker:
▶️ design an implementation vulnerabilities
▶️ 6 attack variants

We proposed 6 fixes, most of them implemented for the 2023 elections

Lessons for the organisation of future e-voting elections
The workflow

Decryption Trustees

Voter

Voting Client

Voting Server

Third-Party
The workflow

Voter → Voting Client → Voting Server

Decryption Trustees

Third-Party
The workflow

Voter

Voting Client

Voting Server

Decryption Trustees

Third-Party

\[ v \rightarrow b := (\{v\}_{pkD}, ZKPs) \]
The workflow

Ballot Privacy: votes are encrypted
Voter

Voting Client

Voting Server

Decryption Trustees

Ballot Privacy: votes are encrypted

\[ \text{ballotBox} \text{ for each consular (~city)} \]

\[ b := (\{v\}^{pkD}, \text{ZKPs}) \]

1 per ballotBox

Third-Party
The workflow

ballotBox for each consular (~city)

Decryption Trustees

pkD

1 per ballotBox

Third-Party

Voter

Voting Client

Voting Server

Ballot Privacy: votes are encrypted

$\nu \rightarrow b := (\{v\}_{pkD}, \text{ZKPs})$

$H := h(b, \text{ballotBox}) \rightarrow \text{sign}(H)$
The workflow

ballotBox for each consular (~city)

Voter

Voting Client

Voting Server

Decryption Trustees

pkD

Ballot Privacy: votes are encrypted

1 per ballotBox

Third-Party

\[ b := (\{ v \}_{pkD}, \text{ZKPs}) \]

\[ H := h(b, \text{ballotBox}) \]

\[ \text{sign}(H) \]
The workflow

**ballotBox** for each consular (~city)

result per **ballotBox**

---

**Ballot Privacy:** votes are encrypted

Voter | Voting Client | Voting Server | Decryption Trustees | Third-Party

\[
v \xrightarrow{\text{Voter}} b := (\{v\}_{pkD}, \text{ZKPs}) \xrightarrow{\text{Voting Client}} H := h(b, \text{ballotBox}) \xrightarrow{\text{Voting Server}} \text{sign}(H) \xrightarrow{\text{Decryption Trustees}} \text{result per ballotBox} \xrightarrow{\text{Third-Party}}
\]
Voter

Voting Client

Voting Server

Decryption Trustees

Third-Party

ballotBox: for each consular (~city)

result per ballotBox

Ballot Privacy: votes are encrypted

\[ \begin{align*}
\text{Voter} & \quad \rightarrow \\
\text{Voting Client} & \quad b := \{(v)_{pkD}, ZKPs\} \\
\text{Voting Server} & \quad H := h(b, \text{ballotBox}) \quad \text{sign}(H) \\
\text{Decryption Trustees} & \quad \leftarrow 1 \text{ per ballotBox} \\
\text{Third-Party} & \quad \leftarrow
\end{align*} \]
The workflow

**ballotBox** for each consular (~city)

result per **ballotBox**

Ballot Privacy: votes are encrypted

Verifiability: act as verifiable receipts

**Voter**

Voting Client

Voting Server

Decryption Trustees

Third-Party

📊 ∈ 1 per ballotBox

📊 ballotBox for each consular (~city)

result per **ballotBox**
**Security goals and threat models**

**Vote secrecy** - “No one should know who I voted for”

**Verifiability** - “No one can modify the outcome of the election”

**Threat models — security expectations under**

- **Voter**
- **Voting Client**
- **Communication Channel**
- **Voting Server**
- **Decryption Trustees**
- **Third-Party**

**Ballot Privacy**

**Verifiability**
Security goals and threat models

**Vote secrecy** - “No one should know who I voted for”

**Verifiability** - “No one can modify the outcome of the election”

**Threat models** — security expectations under

- **Voter**
- **Voting Client**
- **Communication Channel**
- **Voting Server**
- **Decryption Trustees**
- **Third-Party**

**Ballot Privacy**

**Verifiability**
Security goals and threat models

**Vote secrecy** - “No one should know who I voted for”

**Verifiability** - “No one can modify the outcome of the election”

**Threat models – attacks under**

- **Voter**
- **Voting Client**
- **Communication Channel**
- **Voting Server**
- **Decryption Trustees**
- **Third-Party**

**Ballot Privacy**

**Verifiability**

*Cast-as-intended is acknowledge as not satisfied*
Attack against verifiability
(implementation bug…)

Voter

Voting Client

Voting Server

Third-Party

v ∈
Attack against verifiability
(implementation bug...)

• There are 4 versions of $v$ with various consistency checks in the JavaScript voting client
There are 4 versions of $\mathcal{v}$ with various consistency checks in the JavaScript voting client.

- **Implementation vulnerability** $\Rightarrow$ the $\mathcal{v}$ actually displayed to the voter can be attacker-controlled.
Attack against verifiability
(implementation bug…)

• There are 4 versions of with various consistency checks in the JavaScript voting client
• **Implementation vulnerability** ⇒ the actually displayed to the voter can be attacker-controlled

**Impact:** channel or server attacker can **stealthily modify the outcome by replacing or dropping ballots**
Attack against vote privacy
(design vulnerability…)

• Design vulnerability $\Rightarrow$ ballots ZKPs do not bind ballotBox
Attack against vote privacy
(design vulnerability…)

- **Design vulnerability** $\Rightarrow$ ballots ZKPs do not bind ballotBox
Attack against vote privacy
(design vulnerability…)

- **Design vulnerability** $\Rightarrow$ ballots ZKPs do not bind ballotBox
Attack against vote privacy
(design vulnerability…)

- **Design vulnerability** ⇒ ballots ZKPs do not bind ballotBox
Attack against vote privacy
(design vulnerability…)

- **Design vulnerability** ⇒ ballots ZKPs do not bind ballotBox
Attack against vote privacy
(design vulnerability…)

- **Design vulnerability** $\Rightarrow$ ballots ZKPs do not bind ballotBox
Attack against vote privacy
(design vulnerability…)

- **Design vulnerability** ⇒ ballots ZKPs do not bind ballotBox
Attack against vote privacy
(design vulnerability…)

• Design vulnerability ⇒ ballots ZKPs do not bind ballotBox
Attack against vote privacy
(design vulnerability…)

- **Design vulnerability** $\Rightarrow$ ballots ZKPs do not bind ballotBox

Impact: channel or server attacker can **stealthily learn some target voters’ vote** (and perform remote coercion)
Fixes for future elections

We proposed 6 fixes and notably:

1. Display and check 🟢 instead of 🟥
2. Binds ballotBox to the ballot ZKPs
3. Third-Party checks ballotBox

(Attacks and fixes were responsibly disclosed to the vendor and stakeholders.)
Special thanks to the ANSSI who have been proactive in this process.
Fixes for future elections

We proposed 6 fixes and notably:

1. Display and check ✋ instead of ⌌ ✔/✘ partially done for 2023 election
2. Binds ballotBox to the ballot ZKPs ✔ already implemented for 2023
3. Third-Party checks ballotBox ✔ already implemented for 2023

(Attacks and fixes were responsibly disclosed to the vendor and stakeholders.)
Special thanks to the ANSSI who have been proactive in this process.
Lessons learned
(recommendations and research questions)
Lessons learned
(recommendations and research questions)

State-of-art protocol
affected by none of the attacks

Adapt the design

FLEP Protocol
Implement, Deploy, Audit

2022 Election
FLEP 2022
affected by 6 attacks
+ other concerns
not discussed here
Lessons learned
(recommendations and research questions)

BELENIOS

Adapt the design

FLEP Protocol

Implement, Deploy, Audit

2022 Election

State-of-art protocol
affected by none of the attacks

FLEP 2022
affected by 6 attacks
+ other concerns
not discussed here

How can this happen? 😕
Lessons learned
(recommendations and research questions)

1: Adapt the design:

➡ state-of-the art solutions lack features
  • multi-ballot-box for announcing fine-grain results
  • downloadable receipts

➡ state-of-the-art solutions propose unpractical features
  • voters authentication currently relies on a single-point-of-trust
Lessons learned
(recommendations and research questions)

1: Adapt the design:

- state-of-the-art solutions lack features
  - multi-ballot-box for announcing fine-grain results
  - downloadable receipts

- state-of-the-art solutions propose unpractical features
  - voters authentication currently relies on a single-point-of-trust

Academic papers should take into account operational constraints
Lessons learned
(recommandations and research questions)

1: Adapt the design:

➡ state-of-the art solutions lack features
   • multi-ballot-box for announcing fine-grain results
   • downloadable receipts

➡ state-of-the-art solutions propose unpractical features
   • voters authentication currently relies on a single-point-of-trust

2: Implement, deploy, audit

➡ transparency and openness
   • clear security objectives and threat models
   • open specification, promote public scrutiny (e.g. as in Switzerland)

➡ identify the (most) critical components, e.g. Voting client > Server
   • make it auditable (specification, open source, etc)
   • make it monitorable

Academic papers should take into account operational constraints
Lessons learned
(recommandations and research questions)

1: Adapt the design:

➤ state-of-the art solutions lack features
  • multi-ballot-box for announcing fine-grain results
  • downloadable receipts

➤ state-of-the-art solutions propose unpractical features
  • voters authentication currently relies on a single-point-of-trust

2: Implement, deploy, audit

➤ transparency and openness
  • clear security objectives and threat models
  • open specification, promote public scrutiny (e.g. as in Switzerland)

➤ identify the (most) critical components, e.g. Voting client > Server
  • make it auditable (specification, open source, etc)
  • make it monitorable

Academic papers should take into account operational constraints

Any component that needs to be trusted is critical
Conclusion

First public and comprehensive specification of the protocol

Verifiability and vote secrecy can be attacked by a channel/server attacker:

- design an implementation vulnerabilities
- 6 attack variants

We proposed 6 fixes, most of them implemented for the 2023 elections

Lessons for future e-voting elections

https://eprint.iacr.org/2022/1653