Election Verifiability Revisited: Automated Security Proofs and Attacks on Helios and Belenios

Sevdenur Baloglu$^1$, Sergiu Bursuc$^1$, Sjouke Mauw$^2$, Jun Pang$^2$

$^1$ SnT, University of Luxembourg
$^2$ DCS, University of Luxembourg

E-mail: sevdenur.baloglu@uni.lu

IEEE CSF’21, June 22
## Introduction

| Traditional Voting                                                                 | Electronic Voting |
|----------------------------------------------------------------------------------|-------------------|
| 💌 paper ballots                                                                   | 🗺️ electronic ballots |
| 🏷️ voting booths                                                                  | 📱 voting platforms |
| 📥 ballot boxes                                                                   | 🗂️ voting server |
| 💀 adversary has limited corruption abilities                                     | 🧟‍♂️ adversary has extended corruption abilities |
Electronic voting protocols should be **verifiable**:
- by **voters** to ensure their vote is counted in the final tally,
- by **anybody** to ensure the final outcome reflects the intention of eligible voters.

A system satisfying those is called **end-to-end verifiable**.

**Our focus:** Formal verification of election verifiability
Previous Work on formal verification of election verifiability

- S. Kremer, M. Ryan, and B. Smyth, “Election verifiability in electronic voting protocols,” in ESORICS’10.

- V. Cortier, F. Eigner, S. Kremer, M. Maffei, and C. Wiedling, “Type-based verification of electronic voting protocols,” in POST’15.

- V. Cortier, A. Filipiak, and J. Lallemand, “BeleniosVS: Secrecy and verifiability against a corrupted voting device,” in IEEE CSF’19.

We propose an extension of Cortier et al. model: more general and covering new features desired in practice.
Limitations and Contributions

| Features                                      | Cortier et al. model | Our model |
|-----------------------------------------------|----------------------|-----------|
| automated verification                        | ✓                    | ✓         |
| soundness                                     | ✓                    | ✓         |
| general protocols and corruption scenarios    | ✗                    | ✓         |
| revoting                                      | ✗                    | ✓         |
| clash attacks                                 | ✗                    | ✓         |
| dynamic corruption                            | ✗                    | ✓         |
| quantum-resistant protocols                   | ✗                    | ✗         |

We apply our definition to various corruption scenarios and verification procedures in Helios and Belenios, and we find new attacks.
Modeling protocols in Tamarin

Tamarin is based on multiset rewriting rules:

\[
\text{rule : } [\text{Premise}] \rightarrow [\text{Action}] \rightarrow [\text{Conclusion}],
\]

where \text{Premise}, \text{Action} and \text{Conclusion} are sets of facts specifying terms.
Verifying protocols in Tamarin

Tamarin is based on multiset rewriting rules:

\[
\text{rule : } [ \text{Premise} ] \rightarrow [ \text{Conclusion} ],
\]

where \textbf{Premise}, \textbf{Action} and \textbf{Conclusion} are sets of facts.

- \( S \) : the set of rules specifying the protocol,
- \( \Phi \) : the formula specifying the desired property based on action facts.

\textbf{Verification: } S \models \Phi
E-voting Events as Action Facts

Tamarin is based on multiset rewriting rules:

\[
\text{rule : } \left[ \text{Premise} \right] - \left[ \text{Action} \right] \rightarrow \left[ \text{Conclusion} \right],
\]

where \text{Premise}, \text{Action} and \text{Conclusion} are sets of facts.

We use action facts to record events:

- \text{BBreg}(cr): eligible public credential
- \text{Vote}(id, cr, v): the vote is cast by the voter
- \text{Verified}(id, cr, v): the vote is verified by the voter
- \text{BBtally}(cr, b): the ballot is tallied for \( cr \)
- \text{Corr}(id, cr): corrupted voter
End-to-end Election Verifiability

\[ \Phi^\diamond_{E2E} = \Phi_{iv} \land \Phi_{eli} \land \Phi_{cl} \land \Phi^\diamond_{res} \]

- \( \Phi_{iv} \): Individual verification implies the corresponding vote is tallied.
- \( \Phi_{eli} \): Ballots correspond to eligible credentials.
- \( \Phi_{cl} \): Individual verification by distinct voters implies distinct public credentials.
- \( \Phi^\diamond_{res} \): Tallied ballots for honest voters cannot come from the adversary.
End-to-end Election Verifiability

$$\Phi_{E2E}^\diamondsuit = \Phi_{iv} \land \Phi_{eli} \land \Phi_{cl} \land \Phi_{res}$$

- $\Phi_{iv}$: Individual verification implies the corresponding vote is tallied.
- $\Phi_{eli}$: Ballots correspond to eligible credentials.
- $\Phi_{cl}$: Individual verification by distinct voters implies distinct public credentials.
- $\Phi_{res}^\diamondsuit$: Tallied ballots for honest voters cannot come from the adversary.
  - $\Phi_{res}^\bullet$: honest voters $\equiv$ not corrupted
  - $\Phi_{res}^\circ$: honest voters $\equiv$ (not corrupted) $\land$ (verify their ballot)
Election Verifiability Definition

\[ \Phi_{E2E} = \Phi_{iv} \land \Phi_{eli} \land \Phi_{cl} \land \Phi_{\diamond res} \]

\( \Phi_{iv} \): Individual verification implies the corresponding vote is tallied:
\[
\text{Verified}(id, cr, v) \land \Omega(id, cr, v) \land \text{BBtally}(cr, b) \implies v = \text{open}(b)
\]
\( \Omega(id, cr, v) \): captures revoting policy.

\( \Phi_{eli} \): Ballots correspond to eligible credentials:
\[
\text{Verified}(id, cr, v) \lor \text{BBtally}(cr, b) \implies \text{BBreg}(cr)
\]

\( \Phi_{cl} \): Individual verification by distinct voters implies distinct public credentials:
\[
\text{Verified}(id, cr, v) \land \text{Verified}(id', cr, v') \implies id = id'
\]

\( \Phi_{\diamond res} \): Tallied ballots for \textbf{honest voters} cannot come from the \textbf{adversary}:
\[
\text{BBtally}(cr, b) \implies (\text{Vote}(id, cr, v) \land v = \text{open}(b)) \lor \Phi_{\diamond adv}(cr)
\]
Election Verifiability Definition

\[ \Phi_{E2E} = \Phi_{iv} \land \Phi_{eli} \land \Phi_{cl} \land \Phi_{\diamond res} \]

\(\Phi_{iv}\): Individual verification implies the corresponding vote is tallied:
\[\text{Verified}(id, cr, v) \land \Omega(id, cr, v) \land BBtally(cr, b) \implies v = \text{open}(b)\]
\(\Omega(id, cr, v)\): captures revoting policy.

\(\Phi_{eli}\): Ballots correspond to eligible credentials:
\[\text{Verified}(id, cr, v) \lor BBtally(cr, b) \implies BBreg(cr)\]

\(\Phi_{cl}\): Individual verification by distinct voters implies distinct public credentials:
\[\text{Verified}(id, cr, v) \land \text{Verified}(id', cr, v') \implies id = id'\]

\(\Phi_{\diamond res}\): Tallied ballots for honest voters cannot come from the adversary:
\[BBtally(cr, b) \implies (\text{Vote}(id, cr, v) \land v = \text{open}(b)) \lor \Phi_{\text{adv}}(cr)\]
Protocol Specification

An e-voting protocol specification $S = (P, V, A)$, where

- $P$: the voting protocol procedures,
- $V$: the individual verification procedures,
- $A$: the corruption abilities of adversary.

$S$ satisfies symbolic **election verifiability** if and only if $S \models \Phi_{E2E}$. 
Helios

Setup Phase

id → Server/Registrar

(id, pwd, cr)

BB: cr

Voting Phase

id

Voting Platform

id-pwd login

(cr, b₁)

(cr, b₂)

revoting

Server/Registrar

BB:

(cr, b₁)

(cr, b₂)

id

verification

Tally Phase

BB:

(cr, b₁)

Server/Registrar

(cr, b₂)

Trustees

Outcome
Helios and Belenios

• Main limitation of Helios:
  • Corrupted server can stuff ballots.

• Main addition of Belenios:
  • Belenios has additional signature mechanism.
  • Registrar generates a signature key pair for each eligible voter.
  • Trust is distributed between registrar and voting server.
Belenios

Setup Phase

Registrar
(id, pwd, (cr, skey))

Server
(id)

Vote Phase

id-pwd login

Voting Platform
(id)

Server
Log(id, cr)
Log(id, cr)

(b = (c, s, ZKP))

Revoting
(id)

BB:
(cr, b_1)
(cr, b_2)

Verification
(id)

Tally Phase

BB:
(cr, b_1)
(cr, b_2)

Server

BB:
(cr, b_2)

Trustees
Outcome
Individual Verification and Adversary

Individual Verifications

| $\mathcal{V}_1$ | last ballot anytime |
| $\mathcal{V}_2$ | all ballots anytime |
| $\mathcal{V}_3$ | last ballot in tally phase |
| $\mathcal{V}_4$ | empty ballot in tally phase |

Corruption Scenarios

| $\mathcal{A}_1$ | trustees and voters |
| $\mathcal{A}_2$ | trustees, voters and server |
| $\mathcal{A}_3$ | trustees, voters and registrar |
| $\mathcal{A}_4$ | trustees, voters, server and registrar |
| $\mathcal{A}_5$ | trustees, voters, server, registrar and voting platform |
## Verification Results obtained by Tamarin

### Helios

| $V_i/A_j$ | $A_1$ | $A_2$ | $A_4$ | $A_5$ |
|-----------|-------|-------|-------|-------|
| $V_1$     | ✓     | ✗     | ✗     | ✗     |
| $V_3$     | ✓     | ✓     | ✓     | ✗     |

### Belenios

| $V_i/A_j$ | $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ |
|-----------|-------|-------|-------|-------|-------|
| $V_1$     | ✗     | ✗     | ✗     | ✗     |       |
| $V_3$     | ✓     | ✓     | ✓     | ✓     | ✗     |

### Individual Verifications

| $V_i$ | **Description**                           |
|-------|-------------------------------------------|
| $V_1$ | last ballot anytime                       |
| $V_2$ | all ballots anytime                       |
| $V_3$ | last ballot in tally phase                |
| $V_4$ | empty ballot in tally phase               |
Verification Results obtained by Tamarin

### Helios

| $\nu_i / A_j$ | $A_1$ | $A_2$ | $A_4$ | $A_5$ |
|--------------|------|------|------|------|
| $\nu_1$      | ✓    | X    | X    | X    |
| $\nu_3$      | ✓    | ✓    | ✓    | X    |

### Belenios

| $\nu_i / A_j$ | $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ |
|--------------|------|------|------|------|------|
| $\nu_1$      | X    | X    | X    | X    | ?*   |
| $\nu_3$      | ✓    | ✓    | ✓    | ✓    | ✓    |

### Individual Verifications

| $\nu_i$ | Description                                      |
|---------|--------------------------------------------------|
| $\nu_1$ | last ballot anytime                             |
| $\nu_2$ | all ballots anytime                              |
| $\nu_3$ | last ballot in tally phase                      |
| $\nu_4$ | empty ballot in tally phase                     |
Clash Attacks against Helios

Clash attacks proposed by:

- R. Küsters, T. Truderung, and A. Vogt, “Clash attacks on the verifiability of e-voting systems,” in IEEE S&P'12.

is mounted by clash on public credentials when the server and voting platforms are corrupted.

We show that it is not necessary to corrupt voting platforms, if

- revoting is allowed,
- voters verify their ballots anytime.

We discover this based on the property $\Phi_{cl}$ with Tamarin.
Clash Attack by corrupted server against Helios

**Setup Phase**
- \(id_1, id_2\)
- **Server/Registrar**
  - \(id_1\)
  - \(id_2\)
- **BB**: \(cr_1, cr_2\)

**Voting Phase**
- **BB**: \(cr_1, cr_1\)
- Server/Registrar
  - \(id_1\) → \((cr_1, b_1)\)
  - \(id_2\) → \((cr_1, b_2)\)
  - \((cr_1, b_1)\) → **id_1**
  - \((cr_1, b_2)\) → **id_2**
- Verification

**Tally Phase**
- \((cr_1, b_1), (cr_1, b_2)\)
- Server/Registrar
  - \((cr_1, b_2)\) → **id_1**
  - \((cr_2, b_A)\) → **id_1**
- \(b_2\) is tallied for \(cr_1\) even though \(b_1\) is verified by \(id_1\).
Belenios should be secure against ballot stuffing with **corrupted registrar**: 

- The voting server verifies passwords, and checks the consistency on the logs.

However, we find clash attack ($\Phi_{cl}$) and attacks against individual verifiability ($\Phi_{iv}$) and result integrity ($\Phi_{res}^\circ$, $\Phi_{res}$):

$$\Phi_{iv} \; : \; \text{If voters verify the last ballot they cast, it should be counted.}$$
Attack on Individual Verifiability against Belenios

**Setup Phase**

- id₁, id₂

  **Registrar**

  - id₁
  - id₂

  **BB:** cr₁, cr₂

**Voting Phase**

- id₁ → (cr₁, b₁)
- id₂ → (cr₁, b₁)
- id₂ → (cr₁, b₂)

  **Server**

  - Log(id₂, cr₁)
  - Log(id₂, cr₁)

  **BB:**

  - (cr₁, b₁) → id₁

  **Verification**

**Tally Phase**

- (cr₁, b₁), (cr₁, b₂) → **Server**

  **Server** → (cr₁, b₂)

  **BB:**

  - b₂ is tallied for cr₁ even though b₁ is verified by id₁.
Bridging the gap between practice and theory
Thank you for listening!