Formal Verification of Advanced Families of Security Protocols: E-Voting and APIs

Ph.D. Defense
21st November 2014
LORIA, Nancy, France

Cyrille Wiedling

Supervisor:
Véronique Cortier
Introduction

Cryptographic protocols

Goal:

to secure communications and (secret) information
Introduction

Protocols need to be verified before being released to avoid leak of secrets and/or potential damages.

Two different approaches:

**Computational**
- More realistic model
- Strong guarantees
  - Attacker modeled by probabilistic polynomial-time Turing Machine
- Tedious proofs
  - Cryptographic primitives as polynomial algorithms

**Symbolic**
- Weaker guarantees
  - Cryptographic primitives as function symbols
- Abstract model
  - Easier proofs
  - Often automated
  - Attacker modeled by deduction rules
Introduction

- Horn clauses
- Systems
- Strand spaces
- Applied Pi-calculus

**Symbolic Models**

**Properties:**
- Trace Properties
  - Authentication, secrecy, …
- Equivalence Properties
  - Anonymity, ballot secrecy, …

**Tools:**
- aKiSs
- Tamarin
- APTE
- SPEC
- AVISPA
- Scyther
- ProVerif
Introduction: Two Families of Protocols

Application Programming Interfaces

Electronic Voting
Application Programming Interfaces

Part I

In collaboration with Graham Steel
Goal: To enforce security of data stored inside the trusted component, even when connected to untrusted host machines.

Many flaws found on PKCS #11 security tokens (BCFS’10).
Idea:
Use a hierarchy between keys

How to revoke keys?
(Especially Top keys)
Proposals for key management APIs with security proofs (*no revocation*) (CM’06, CC’09, CG’09…).

- **Use of long-term keys** implying **unrecoverable loss** of devices if keys are lost.

Proposals for key management APIs *with revocation* (EG’02, YR’07,…).

- **Still use long-term keys!**

Proposal of Sevecom using two root keys (Kar’09).

- **Attacked** by S. Möderschein & P. Modesti
  (solution proposed but no security proof)
API: Contributions

Specification of a new API design

- including revocation
- with a formal security proof
- and implemented
A clock assumed synchronized with a global clock (Keys come along with an expiration date.)

A blacklist of elements of the form \((l, t)\)

The API allows level max key management!

We have a hierarchy of levels for keys.
API: Presentation

Around **10 different** functions in the design.

Generation,

Encryption, Decryption,

Keys management (Update, Revocation, ...).

**UpdateMax** function to replace/revoke level max keys.

\[
\text{UpdateMax}(C, h_1, \ldots, h_n)
\]

Checks on \( h_1, \ldots, h_n \).

Let \((\text{updateMax}, k', v') = \text{dec}(k_1, \ldots, \text{dec}(k_n, C)) \) in

Checks on \( v' \)

Update of the memory of the TRD
Messages are represented by terms.

**Nonces, keys:**

\[ n, m, \ldots, k_1, k_2, \ldots \]

**Primitives:**

\[ \{m\}_k, \langle m_1, m_2 \rangle \]

**Modeling deduction rules:**

\[
\begin{align*}
\frac{x \ y}{\langle x, y \rangle} & \quad \frac{\langle x, y \rangle}{x} & \quad \frac{\langle x, y \rangle}{y} & \quad \frac{x \ y}{\{x\}_y} & \quad \frac{\{x\}_y \ y}{x}
\end{align*}
\]
We model the system using **global states**: \((P, I, M, N, K, t)\)

and several **transitions** modifying it.

\[(TIM) \ (P, I, M, N, K, t) \rightarrow (P, I, M, N, K, t') \quad (t' > t)\]

models the **time passing**...

\[(DED) \ (P, I, M, N, K, t) \rightarrow (P, I, M \cup \{m\}, N, K, t)\]

models the **deduction abilities** of the intruder. \((M \models m)\)
Intruder **controls** the network and host machines.

Only access TRD using API.

Can brute force **some** keys.

*Ex: Side channel attacks, …*

**Specific transition:**

\[
(LST) \ (P, \mathcal{I}, M, N, K, t) \xrightarrow{\text{Lost}(k)} (P, \mathcal{I}, M \cup \{k\}, N, K, t)
\]

\[L_V = \{l \mid \exists \text{ Lost}(k) \text{ and Level}(k) = l\}\]

The intruder **has control** over whatever is **under a level** with a **lost** key.
API: Secrecy

Theorem 1

Let $E$ s.t. $E_0 \rightarrow^* E$, $L_V$ the set of lost levels and $k \in K$. 
\[ \forall k \text{ s.t. } \text{Level}(k) \not\in L_V, \ E \not\ni k. \]

Sketch of Proof

1. Find **invariant properties** of the system.

   \[ \forall k \in G_E^{L_V}, \forall m \in M \text{ s.t. } k \in \text{St}(m), \text{ then any occurrences of } k \text{ is:} \]
   
   (i) $\{m'\}_k$,

   (ii) $\{\ldots, k, \ldots\}_k'$ with $k' \in K$, $v_k \leq v_{k'} + \delta_k$ and $\text{Level}(k) < \text{Level}(k') \neq \text{Max}$,

   (iii) $\{\ldots, k, \ldots\}_{q_1 \ldots q_n} k'$ with $q_1, \ldots, q_n, k' \in K$ s.t., for $i \in [1, n]$, $\text{Max} \in \text{Level}(q_i)$, $\text{Max} \in \text{Level}(k')$, $k' \notin \mathcal{L}$ and $v_k \leq v_{k'} + \delta_k$,

2. **Prove** it.
API: Self-Repairing

Theorem 2 (Stated for one level)

Assume that all keys are secret a time $t$ except those under a level $l$. Then, at time $t + \Delta(l)$, all keys are secret except those under levels $l_1, \ldots, l_n$ such that $l_i < l$.

It assumes that, during time $\Delta(l)$, you do not lose a level higher than the one you «try» to repair.

Sketch of Proof

1. **Not losing a higher (or equal) level**
   
   The attacker has no other ‘entry point’ in the hierarchy.

2. **Time has passed enough**
   
   The corrupted keys are not valid anymore.
API: Blacklist

Blacklist command:

\[(l_3, t_3) \rightarrow \]

\[\begin{array}{c}
 l_1 \\
 l_2 \\
 l_3 \\
 l_4 \\
 l_5 \\
 l_6 \\
\end{array}\]

**Theorem 3** (Stated for one level)

Assume that all keys are secret a time \(t\) except those under a level \(l\).

If we blacklist level \(l\) on a TRD, then, **immediately**, all keys are secret.

*Only work* for TRDs that have received the command.

Time of blacklist must be long enough.

*Prevent* the attacker to operate on the TRD.

Design of a secure API for symmetric keys management with revocation and a formal security proof

Collaboration with M. Daubignard and the DGA

Prototype implementation in Javacard
- Working on simulators and real cards
- Supervision of students of Mines-Nancy
Part II
E-Voting: Introduction

Since 2000s, E-voting has spread worldwide.

Systems may be vulnerable to attacks:
- Diebold Machines in the U.S. (C. Hoke, 2008)
- Paperless EVM in India. (A. Halderman, R. Gonggrijp, 2010)

Some countries just decide to stop E-voting.
E-Voting: Limitations

We focused on equivalence-based properties like ballot secrecy.

Tools for automatic verification:

- ProVerif [Blanchet, CSFW, 2001]
- SPEC [Tiu and Dawson, CSF’10]
- aKiSS [Chadha, Ciobaca, Kremer, ESOP’12]

E-Voting protocols often include too many different cryptographic primitives!
E-Voting: Studied Systems

Norwegian Internet Voting Protocol

CNRS Boardroom Voting Protocol
E-Voting: Norwegian E-Voting Protocol

Developed by Scytl, source code available.

Used in legally binding elections in 2011 and 2013.

Around 30,000 and 70,000 users respectively.
E-Voting: Abstraction

Messages are represented by terms.

Nonces, keys:

\[ n, m, \ldots, k_1, k_2, \ldots \]

Primitives:

\[ \{m\}_{k}, \langle m_1, m_2 \rangle \]

Modeling deduction rules:

\[
\begin{align*}
\frac{x \ y}{\langle x, y \rangle} & \quad \frac{\langle x, y \rangle}{x} \\
\frac{\langle x, y \rangle}{y} & \quad \frac{x \ y}{\{x\}_y} & \quad \frac{\{x\}_y \ y}{x} \\
\end{align*}
\]

\[ P, Q, R ::= \]

0

\[ P \mid Q \]

!P

\[ \nu n. P \]

if \( \phi \) then \( P \) else \( Q \)

\[ u(x). P \]

\[ \overline{u}(M). P \]

\[ A, B, C ::= \]

extended processes

plain process

parallel composition

replication

name restriction

conditional

message input

message output

Applied-Pi Calculus
by M. Abadi and C. Fournet
E-Voting: Modeling & Challenges

We provide a **model** of the Norwegian e-voting protocol.

\[ \Sigma_{\text{sign}} = \{ \text{ok, fst, hash, p, pk, s, snd, vk, blind, d, dec, } +, *, \circ, \diamond, \text{ pair, renc, sign, unblind, checkpfk}_1, \text{checkpfk}_2, \text{checksign, penc, pfk}_1, \text{pfk}_2 \} \]

\[ \text{penc}(x_1, r_1, k_p) \circ \text{penc}(x_2, r_2, k_p) = \text{penc}(x_1 \diamond x_2, r_1 \ast r_2, k_p) \]

\[ \text{renc}(\text{penc}(x, r, \text{pk}(k_1)), k_2) = \text{penc}(x, r, \text{pk}(k_1 + k_2)) \]

**AC symbols and properties** !

**Not in the scope** of current tools for automatic verification !
E-Voting: Ballot Secrecy

Definition 1 (DKR’09)

A protocol $P$ ensures ballot secrecy if:

$$P[V_1(a) \mid V_2(b)] \simeq_l P[V_1(b) \mid V_2(a)]$$

**Labelled Bisimilarity**
(defines indistinguishability)

$$A' \equiv B'$$

$$A \equiv B$$

Same visible behavior

Static equivalence of messages

$$\forall \text{ (message)} = \text{ (message)}$$
E-Voting: Proof of Ballot-Secrecy

Sketch of Proof

1. **Exhibit** a relation and **prove** that it is a bisimulation.

   ![Diagram](Image)

   Find *all possible successors* of the two processes.

   End with **around 45 relations** between processes.
2. **Show** static equivalence of the outputs of the two processes.

\[
\begin{align*}
\theta_{\text{init}} &= \{\frac{vk(id_k)}{id_p_k}, \frac{s(id_k)}{s_k} \mid k = 1..n\} \cup \{\frac{vk(id_R)}{id_p_R}\} \cup \{\frac{pk(a_k)}{g_k} \mid k = 1..3\}, \\
\theta_0 &= \theta_{\text{init}} \cup \{\frac{\text{penc}(v_k,r_k,g_1)}{e_k}, \frac{\text{pfk}_1(id_k,t_k,v_k,e_k)}{p_k}, \frac{\text{sign}(\langle e_k,p_k\rangle, id_k)}{s_{i_k}} \mid k = 1..2\}, \\
\theta_k &= \theta_{k-1} \cup \{\frac{\text{sign}(\text{hash}(\Pi_1(M)), id_R)}{s_{r_k}}, \frac{d(p(id_k), \text{dec}(\Pi_2(M), a_3))}{r_{e_k}}\}, \\
\theta_{\delta} &= \theta_n \cup \{\frac{\text{dec}(U, a_1)}{r_{e_k}} \mid k = 1..n\}. \\
\end{align*}
\]

**Proposition 1**

Let \(\delta\) is a substitution of \([1, n]\) and \(t\delta = \delta \circ [1 \mapsto 2, 2 \mapsto 1]\).

Then we have: \(\forall M, \forall U\)

\[
\nu \tilde{\omega} \theta_{\delta \cdot M,U}^L \sigma_L \approx_S \nu \tilde{\omega} \theta_{t\delta \cdot M,U}^R \sigma_R,
\]

with \(\sigma_L = \{a / v_1, b / v_2\}\) and \(\sigma_R = \{b / v_1, a / v_2\}\).
### E-Voting: Norwegian Results

<table>
<thead>
<tr>
<th>Corr. Voters</th>
<th>0</th>
<th>1</th>
<th>4</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>✓ (ProVerif)</td>
<td>✓ (ProVerif)</td>
<td>✓ (ProVerif)</td>
<td>? (&gt;1h)</td>
<td>? (&gt;1h)</td>
</tr>
<tr>
<td>A</td>
<td>✓ (ProVerif)</td>
<td>✓ (ProVerif)</td>
<td>✓ (ProVerif)</td>
<td>✓ (ProVerif)</td>
<td>? (&gt;1h)</td>
</tr>
<tr>
<td>D + *</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B + R</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B + A</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R + A</td>
<td>✓ (ProVerif)</td>
<td>✓ (ProVerif)</td>
<td>✓ (ProVerif)</td>
<td>✓ (ProVerif)</td>
<td>? (&gt;1h)</td>
</tr>
</tbody>
</table>

*Corr.* stands for Corrupted, *Players* stands for Players.

(A Theorem 2)

(A Theorem 1)

(ProVerif)

(Attacks)
E-Voting: CNRS Protocol & Results

Developed for CNRS boardroom meetings, by an internal committee.

Results displayed on a screen for verifiability.

**As simple as possible** (no cryptography).

We provide a **full analysis** on three slightly different versions regarding **Ballot Secrecy** and **Correctness**.

<table>
<thead>
<tr>
<th>Results</th>
<th>Ballot secrecy</th>
<th>Correctness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>None</td>
<td>Ballot Box</td>
</tr>
<tr>
<td>System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corr. Players</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2FV¹</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>F2FV²</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>F2FV³</td>
<td>✓</td>
<td>×</td>
</tr>
</tbody>
</table>

In collaboration with Mathilde Arnaud

*Analysis of a Boardroom Voting Protocol, Vote-ID’13.*
Analysis of two implemented and used protocols in different interesting cases of corruption.

Useful intermediate lemmas for further analyses.
Types Systems: Automation of E-Voting Proofs

Part III

In collaboration with Fabienne Eigner, Steve Kremer and Matteo Maffei
Types Systems: Introduction

**Fully Automatic Tools**

- ProVerif
- SPEC
- APTE
- aKiSs

Do not support well AC-properties

Accurate but not really scalable with complex protocols

**Type Systems**

- They do!

- Less precision but better scalability!

- Term-based abstraction

- Code-based
Ballot Secrecy is an **equivalence-based property**.

Designed by G. Barthe, C. Fournet et al., POPL’14.

Extension of existing type-system F*.

Can be used to ensure unlinkability, integrity and privacy properties.

**E-Passport**

(Unlinkability)

**Smart meter billing**

(Integrity & Privacy)
Indistinguishability properties are proved by type-checking two executions of the protocol.

If protocol type-checks then executions are observationally equivalent.

It also allows logical assumptions and assertions.

We can use it for authorization policy based properties, like verifiability.
Types Systems: RF* Approach

Homomorphobic encryption scheme:

\[ \text{enc}(x, k) \ast \text{enc}(y, k) = \text{enc}(x + y, k) \]

Approach in RF*:

« eq types »: type eq bytes = x:bytes{\( L \ x = R \ x \)}

\((L \ x)/(R \ x)\) are values of variable \( x \) in first and second execution resp.

Theorem 1

Let \( M : x : \text{bytes}\{|L \ x = v_1 \land R \ x = v_2\}\), \( M' : x : \text{bytes}\{|L \ x = v_2 \land R \ x = v_1\}\).

If \( \emptyset \vdash P(M, M') \rightsquigarrow I_{\text{eq}} \) then \( P \) satisfies ballot secrecy.
Types Systems: Modeling Helios

Modeling the exchanged messages...

cipher = eq bytes

L ca = enc(mLa) and R ca = enc(mRa)

i.e. L ca = R ca

Providing an interface...

type hom_fun_t = c1:cipher -> c2:cipher -> c:cipher {
  forall (mL1:bytes) (mL2:bytes) (mR1:bytes) (mR2:bytes).
  (Encryptedboth mL1 mR1 (L c1) (R c1)) && (Encryptedboth mL2 mR2 (L c2) (R c2)) =>
  (exists (mL:bytes) (mR:bytes).
    (Encryptedboth mL mR (L c) (R c)) && mL = (mL1+mL2) && mR = (mR1+mR2))
}

Result is an eq bytes!

SMT Solver

a + b \equiv b + a
Cryptographic library in RF* for type-checking Helios

Ballot secrecy for Helios

Verifiability for Helios

Typed-Based Verification of Electronic Voting Protocols, Submitted to POST’14.
Conclusion

Full analysis of implemented e-voting protocols

Specification and proof of a secure API design for key management with revocation

Implementation

Type-based Library for automatic verification of e-voting protocols
Future Work

Weaken assumptions and extend to asymmetric.

Adapt the result to possible clock skew between TRDs.

Automation of the relation step.

Study of different properties like receipt-freeness, coercion-resistance, verifiability…
Future Work

Apply this work on different e-voting protocols.

Extend to different security properties.
That's all folks!

Questions?