
Pascal Lafourcade

November 6, 2012
Habilitation à Diriger des Recherches
Nowadays Security is Everywhere!
What is cryptography based security?

**Cryptography:**
- Primitives: RSA, Elgamal, AES, DES, SHA-3 ...
- Protocols: Distributed Algorithms

**Properties:**
- Secrecy,
- Authentication,
- Privacy ...

**Intruders:**
- Passive
- Active
- CPA, CCA ...

Designing secure cryptographic protocols is difficult
Security of Cryptographic Protocols

How can we be convinced that a protocols is secure?

- Prove that there is no attack under some assumptions.
  - proving is a difficult task,
  - pencil-and-paper proofs are error-prone.

How can we be convinced that a proof is correct?

Formal Verification Approaches

Designer

Give a proof

Attacker

Find a flaw
Back to 1995

- **Cryptography**: Perfect Encryption hypothesis
- **Property**: Secrecy, Authentication
- **Intruder**:
  - Active
  - Controlling the network
  - Several sessions
Success Story of Symbolic Verification

Tools based on different theories for several properties

1995  Casper/FRD [Lowe]
2001  Proverif [Blanchet]
2003  Proof of certified email protocol with Proverif [AB]
      OFMC [BMV]
      Hermes [BLP]
      Flaw in Kerberos 5.0 with MSR 3.0 [BCJS]
2004  TA4SP [BHKO]
2005  SATMC [AC]
2006  CL-ATSE [Turuani]
2008  Scyther [Cremers]
      Flaw of Single Sign-On for Google Apps with SAT-MC [ACCCT]
      Proof of TLS using Proverif [BFCZ]
2010  TOOKAN [DDS] using SAT-MC for API
2012  Tamarin [BCM]
Main Contributions:

- Verification techniques for cryptography
  - Asymmetric Encryptions
  - Encryption Modes
  - Message Authentication Codes
- Properties for E-voting protocols
  - Taxonomy of privacy notions
  - Weighted votes
- Intruder models and algorithms for WSN
  - Neighbourhood Discovery Protocols
  - Independent Intruders
  - Routing Algorithms
Related Work

- **CryptoVerif [BP06]:**
  - tool that generates proofs by sequences of games
  - has automatic and manual modes

- **CIL [BDKL10]:** Computational Indistinguishability Logic for proving cryptographic primitives.

- **CertiCrypt [BGZB09] /EasyCrypt [BGHB11]:**
  - Framework for machine-checked cryptographic proofs in Coq
  - Improved by EasyCrypt: generates CertiCrypt proofs from proof sketches
Our Approach

Automatically proving security of cryptographic primitives

1. Defining a language
2. Modeling security properties
3. Building a Hoare Logic for proving the security

Correct but not complete.

- Asymmetric Encryption Schemes [CDELL’08, CDELL’10]
- Encryption Modes [GLLS’09]
- Message Authentication Codes (MACs) Submitted [GLL’13]
Examples of Asymmetric Encryptions

- **[BR’93]**: $f(r) || x \oplus G(r) || H(x || r)$

- **[SZ’93]**: $f(r) || G(r) \oplus (x || H(x))$

- **[BR’94] OAEP**: $f(s || r \oplus H(s))$
  where $s = x0^k \oplus G(r)$

- **[Shoup’02] OAEP+**: $f(s || r \oplus H(s))$
  where $s = x \oplus G(r) || H'(r || x)$.

- **[FO’99]**: $\mathcal{E}((x || r); H(x || r))$
  where $\mathcal{E}$ is IND-CPA.

$f$ is a one-way trapdoor permutation, $H$ and $G$ are hash functions and $r$ is a random seed.
Security Property: Indistinguishability

\[ \text{Indis}(x; V_1; V_2): \text{seeing } V_1 \text{ and } f(V_2). \]
Modelling: Generic Encryption Scheme

Grammar for Generic Encryption

```
cmd ::= x ← U | x := f(y) | x := H(y) |
     x := y ⊕ z | x := y∥z | cmd; cmd
```

A Generic Encryption Scheme

\[ \mathcal{E}(\text{in}_e, \text{out}_e) = \]

\[ c_1; \]
\[ c_2; \]
\[ \cdot \]
\[ c_n; \]

Bellare & Rogaway’93:

\[ f(r)∥\text{in}_e ⊕ G(r)∥H(\text{in}_e∥r) \]

\[ \mathcal{E}_{BR93}(\text{in}_e, \text{out}_e) = \]

\[ r \leftarrow \mathcal{U}; \]
\[ a := f(r); \]
\[ g := G(r); \]
\[ b := \text{in}_e ⊕ g; \]
\[ t := \text{in}_e∥r; \]
\[ c := H(t); \]
\[ \text{out}_e := a∥b∥c \]
Only Three Predicates in the ROM

<table>
<thead>
<tr>
<th>Predicates</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \psi ) ::= ( H(G, e) )</td>
</tr>
<tr>
<td>( \varphi ) ::= true</td>
</tr>
</tbody>
</table>

- **\( H(G, e) \):** Not-Hashed-Yet
  \[
  \Pr[S \leftarrow X : S(e) \in S(T_H).\text{dom}] \text{ is negligible.}
  \]
- **\( WS(x; V) \):** cannot to compute some “hidden” value.
  \[
  \Pr[S \leftarrow X : A(S) = S(x)] \text{ is negligible.}
  \]
- **\( Indis(x; V_1; V_2) \):** seeing \( V_1 \) and \( f(V_2) \).

But more than 30 rules
Verification Technique: Hoare Logic

Set of rules \( (R_i) : \{ P \} \text{ cmd } \{ Q \} \)

\( (R_5) \{ P_0 \} \ c_1 \ { Q_0 } \)
\( (R_2) \{ P_1 \} \ c_2 \ { Q_2 } \), where \( P_1 \subseteq Q_0 \)

...\( (R_8) \{ P_n \} \ c_n \ \{ \text{Indis(out}_e) \} \) ?

Examples of rules:

\( (X2) : \{ \text{Indis}(w; V_1, y, z; V_2) \} \ x := y \oplus z \ \{ \text{Indis}(w; V_1, x, y, z; V_2) \} \)

\( (H6) : \{ \text{WS}(y; V_1; V_2, y) \land H(H, y) \} \ x := H(y) \ \{ \text{WS}(y; V_1, x; V_2, y) \} \)
Example: Bellare & Rogaway's 1993

\[
\begin{align*}
  r & \leftarrow \{0, 1\}^{n_0} & \text{Indis}(r) \land H(G, r) \land H(H, h \parallel r) \\
  a & := f(r) & \text{Indis}(a; \text{Var} - r) \land WS(r; \text{Var} - r) \land H(H, h \parallel r) \\
  g & := G(r) & \text{Indis}(a; \text{Var} - r) \land \text{Indis}(g; \text{Var} - r) \land WS(r; \text{Var} - r) \land H(H, h \parallel r) \\
  e & := h \oplus g & \text{Indis}(a; \text{Var} - r) \land \text{Indis}(e) \land WS(r; \text{Var} - r) \land H(H, h \parallel r) \\
  d & := h \parallel r & \text{Indis}(a) \land \text{Indis}(e) \land WS(r; \text{Var} - r) \land H(H, d) \land WS(d) \\
  c & := H(d) & \text{Indis}(a) \land \text{Indis}(e) \land \text{Indis}(c) \\
  \text{out}_e & := a \parallel e \parallel c & \text{Indis}(\text{out}_e; \{\text{in}_e, \text{out}_e\})
\end{align*}
\]
Conclusion: Hoare Logics for proving

- **Asymmetric Encryption Schemes**
  - An OCAML prototype of our 30 rules
  - Extensions done for proving IND-CCA using IND-CPA + Plaintext Awareness
  - Exact Security

- **Symmetric Encryption Modes**
  - Counters
  - FOR loops
  - Exact Security
  - An OCAML prototype of our 21 rules

- **Message Authentication Codes (MACs)**
  - Different property: Unforgeability
  - Almost-universal Hash function
  - Keep track of possible collisions
  - FOR loops
  - An OCAML prototype of our 44 rules
Revisited [Benaloh’94] Homomorphic Encryption

\[
\begin{align*}
\{0\}_{pk_S} & \quad \{1\}_{pk_S} \\
\prod_{i=1}^{n} \{v_i\}_{pk_S} &= \sum_{i=1}^{n} \{v_i\}_{pk_S}
\end{align*}
\]

Result [FLA’11]

- Original Benaloh’s scheme is ambiguous (33%):
  \[
  \text{dec(\text{enc}(14, pk_S), sk_S)} = 14 \mod 15 \text{ or } 14 \mod 5 = 4
  \]
- Proposition of corrected version
- Proof using Kristian Gjosteen result

Impact on an election: Result can change (either 14 or 4)
Security Properties of E-Voting Protocols

- Fairness
- Eligibility
- Individual Verifiability
- Universal Verifiability
- Correctness
- Receipt-Freeness
- Privacy
- Robustness
- Coercion-Resistance
Motivation

Existing several models for Privacy, but they

- designed for a specific type of protocol
- often cannot be applied to other protocols

Our Contributions:

- Define **fine-grained** Privacy definitions to **compare** protocols
- Analyze **weighted votes** protocols
- **One coercer is enough**
4 Dimensions for Privacy [DLL’12a, DLL’11]

Modeling in Applied π-Calculus

1. Communication btwn the attacker & the targeted voter

[DKR09]
Vote-Privacy (VP) Receipt-Freeness (RF) Coercion-Resistance (CR)

2. Intruder is controlling another voter

Outsider (O) Insider (I)

3. Secure against Forced-Abstention: (FA) or not (PO)

4. Honest voters behavior:
Relations without $\exists$ and $\forall$

\[
\begin{align*}
CR^O,FA &\rightarrow CR^I,FA^* \\
\star CR^O,PO \quad [LBD^+04] &\rightarrow CR^I,PO \quad [BMQR07] \\
RF^O,FA &\rightarrow RF^I,FA^* \\
\star RF^O,PO &\rightarrow RF^I,PO \quad [Oka96] \\
VP^O,FA &\rightarrow VP^I,FA^* \\
\star VP^O,PO &\rightarrow VP^I,PO \quad [FOO92]
\end{align*}
\]
All relations among the notions
Privacy for Weighted Votes [DLL’12b]

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>66%</td>
<td>34%</td>
<td>≈ 1</td>
</tr>
</tbody>
</table>

Vote: Red, Blue

Result: 66%, 34%

Vote: Blue, Red

Result: 34%, 66%
Privacy for Weighted Votes [DLL’12b]

Still: Some privacy is possible!

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
<th>Carol</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>50%</td>
<td>25%</td>
<td>25%</td>
<td>50%, 50%</td>
</tr>
</tbody>
</table>

Vote:  
≈ /  

Vote:  
50%, 50%
Definition of Vote-Privacy (VP) for weighted votes

Idea: Two instances with the same result should be bi-similar

Vote 1: $V_A^1$ $V_B^1$ $\cdots$ $\approx I$ $\cdashrightarrow$ Result 1

Vote 2: $V_A^2$ $V_A^2$ $\cdots$ $\Rightarrow$ $\equiv$ $\Rightarrow$ Result 2
### Single-Voter Receipt Freeness (SRF)

If a protocol respects (EQ), then (SRF) and (SwRF) are equivalent.
Multi-Voter Receipt Freeness (MRF)

(MRF) implies (SRF) and (MCR) implies (SCR).
One Coerced Voter is enough!

Unique decomposition of processes in the applied $\pi$-calculus.
Challenges in WSNs

Nodes

- Broadcast communication
- Low computation power
- Battery

- **Cryptography:** Lightweight, energy- and resource-aware ...
- **Properties:** \((k)\)-neighborhood, routing ...
- **Intruders:** Black-hole, wormhole, Byzantine, independent ...
Our Contributions

- (k)-Neighbourhood Verification [JL’12]
- Independent Intruders [KL’12]
- Analysis of non-backtracking random walk [ADGL’12]
- Resilient routing algorithm [ADJL]
Usual Intruders

Dolev-Yao’s Intruder [83]
Intruder Model in WSNs

Several intruders with sharing [ACD12]
Independent Intruder Model

Independent intruders without sharing
Usual Constraints System

- **Intruder knowledge monotonicity:**
  \[ T_1 \subseteq \cdots \subseteq T_n. \]

- **Variable origination:** if \( x \) occurs in \( \text{vars}(T_i) \) for certain \( T_i \) then there exists \( k < i \) such that \( x \in \text{vars}(u_k) \).
Partially Well-Formed Constraint System

\[ \mathcal{C} = T_1^l \models u_1 \land \cdots \land T_n^q \models u_n \]

- **Global Origination.**
- **Partial monotonicity:**
  \[ T_k^j \subseteq T_i^j \text{ for every } j \in \{1, 2, \ldots, m\} \text{ such that } k < i. \]
Quasi-Solved Form

\[ R_{ax} : C \land T_i^j \models u_i \rightsquigarrow C \]
\[ R_{unif} : C \rightsquigarrow_{\sigma} C\sigma \]
\[ R'_{unif} : C \land T_i^j \models u_i \rightsquigarrow_{\sigma} C\sigma \land T_i^j \sigma \models u_i\sigma \]
\[ R_f : C \land T_i^j \models f(u, v) \rightsquigarrow C \land T_i^j \models u \land T_i^j \models v \]
\[ R_{fail} : C \land T_i^j \models u_i \rightsquigarrow \bot \]

if \( T_i^j \cup \{x | T_k^j \models x \in C, k < i\} \not\models u_i \)
\[ \sigma = \text{mgu}(t_1, t_2), \ t_1, t_2 \in \text{st}(C) \]
\[ \sigma = \text{mgu}(t, f(t_1, t_2)), f \in \{\langle-, -\}, - :: -\}, \ t \in \text{vars}(u_i), t_1, t_2 \in \text{st}(T_k^j), \text{ where } k \leq i \]
if \( f \in \{\text{senc}, \text{aenc}, \langle-, -\}, - :: -, \text{hmac}, \text{sig}\} \)
if \( T_i^j = \emptyset \), or \( \text{vars}(T_i^j \cup \{u_i\}) = \emptyset \)
and \( T_i^j \not\models u_i \)

Soundness, completeness and termination.

Example of Quasi-Solved Form:

\[ T_1^1 = \{a, b\} \models x \]
\[ T_2^2 = \{x\} \models a \]
\[ T_3^3 = \{x\} \models b \]

Procedure for finding a solution to a quasi-solved form.
Resilient Routing Algorithms

Even with “perturbation” a resilient protocol should work “well”

- Perturbation: abnormal behavior, node destruction, battery ...
- Well: Hitting time, average delivery rate...

Existing protocols

Probabilistic vs Deterministic
Random walk GBR, GFG

Our Goal: Design an efficient resilient routing algorithm using a reputation mechanism
Our Resilient Algorithm: TLCNS [ADJL]

Shared symmetric key $K_{OS}$ between the sink and all nodes $O$.

- Each node $O$ sends: $\{Data, N_O\}_{K_{OS}}, H(N_O), O, F$
- Sink S acknowledges: $N_O, O$

3 lists for each node:

- $M_{ack} = [(H(N_O), A), (H(N_B), C)]]$: List of hashed nonces and sender identity.
- $M_{Queue} = [(N^1_O, A), (N^2_O, B)]]$: List of messages sent
- $L_{Routing} = [A, B, C]]$: List of “preferred” first hops (FIFO)

Why does it work?

- Each node prefers **preferred** next hop
- **All** neighbours are possible
Scenario for testing the Resilience

- Simulation using SINALGO
- $|L_{Routing}| = 10$, $|M_{Queue}| = 5$ and $|M_{ack}| = 3$
- 200 nodes, 1 sink

Intruders:
- Black Holes: Node not forwarding any message
- Worm Holes: False link in the topology

Scenario in 2 phases:
- Static: 10 Black holes + 10 Wormholes
- Dynamic: 20 Black holes
  (Wormholes → Black Holes)
Results
## Summary

### Automatic proofs of programs (Hoare Logic)
- Generic Asymmetric Encryption \([\text{CDELL}'08, \text{CDELL}'10]\)
- Generic Encryption Mode: counter + For loop \([\text{GLL}'09]\)
- Generic MAC: Double execution + For loop \([\text{GLL}'13]\)

### Cryptography & Process Algebra (Applied π-Calculus)
- Revisited Benaloh’s encryption scheme \([\text{FLA}'11]\)
- Privacy notions \([\text{DLL}'12a, \text{DLL}'11]\)
- Weighted votes \([\text{DLL}'12b]\)

### Constraints Solving & Randomized Algorithms
- Neighbourhood Discovery Verification \([\text{JL}'12]\)
- Independent Intruders \([\text{KL}'12]\)
- Design of routing algorithms \([\text{AGDL}'12, \text{ADLP}'11]\)
Future Work

- **Computer-Aided Cryptography:**
  - Hoare Logic for other primitives: Pairing, E-Stream ...
  - How to prove Benaloh’ scheme?
  - Using verification for the synthesis of new schemes

- **Properties:**
  - E-auctions: Non cancellation, Non repudiation, Privacy ...
  - Non-functional properties for WSNs: energy consumption

- **Intruder Model:**
  - With a battery
  - Mobility
Conclusion

Thank you for your attention.

Questions?