Cracking the Stateful Nut

Computational Proofs of Stateful Security Protocols using the Squirrel Proof Assistant

CSF’22

David Baelde  Univ Rennes, CNRS, IRISA
Stéphanie Delaune  Univ Rennes, CNRS, IRISA
Adrien Koutsos  Inria Paris
Solène Moreau  Univ Rennes, CNRS, IRISA

9 August 2022, Haifa, Israel
Security protocols are distributed programs which aim at providing some security properties.

Attacks against security protocols can be very damageable, e.g. theft or privacy breach.

Protocol design is though, and errors are hard to spot.

⇒ well-suited field for formal verification.
The Squirrel Prover

The Squirrel Prover:
- Tool for verification of security protocols in the computational model.
- Implements an indistinguishability logic.
  - Inference rules proved valid w.r.t. comp. attacker.
- Proof assistant:
  - Users prove goals using sequences of tactics.
  - Logical tactics: apply, rewrite, ...
  - Crypto. tactics: prf, euf, ...

Web-page:

https://squirrel-prover.github.io/
In **stateful protocols**, agents have a **mutable state** persistent between sessions.

Used in many **real-world protocols**, e.g.:

- using integer counters: Yubi-Key, \{3,4,5\}G-AKA, . . .
- using chains of hashes: OSK, YPLRK, . . .
- using ratcheting/key refresh: Signal, MLS . . .

**Problem:** **SQUIRREL** did not support mutable state, making stateful protocols out-of-scope.
Our Contributions

- Extend the indistinguishability logic with **mutable state**.
- New **generalized sequent calculus**.
  - Mix reachability and equivalence reasonings.
- **Proof automation**: design a proof system for **bi-deduction**.
  - **Intuition**: indistinguishability is **preserved by (public) computation**.
  - Allow for **automation** of some proof steps.
- **Implementation** in the **SQUIRREL** tool.
  - Main case-studies: **Yubi-Key, Yubi-HSM**.
Indistinguishability Logic
The OSK Protocol

$s_T$: mutable state of tag $T$

$s_R$: mutable state of reader $R$

$s_T$ and $s_R$ initial value: $n_s$

$n_s, k_H, k_G$: random samplings

$H, G$: keyed hash functions

The OSK protocol:

1: $T \rightarrow R$ :

$s_T := H(s_T, k_H)$;

out($G(s_T, k_G)$)

2: $R \rightarrow T$ :

in($x$);

if $x = G(H(s_R, k_H), k_G)$ then

out(ok);

$s_R := H(s_R, k_H)$
Terms represent probabilistic poly-time computations of bitstrings. Used to model both protocol and adversary computations.
**Indistinguishability Logic: Terms**

Terms represent probabilistic poly-time computations of bitstrings. Used to model both protocol and adversary computations.

- **Names** for random samplings of length \( \eta \) (security parameter):
  - \( n_s, k_H, k_G \)

- **Function symbols** for honest computations:
  - \( H(n_s, k_H) \)

- **Timestamps** for time-points of the protocol execution:
  - Protocol actions (\( \text{Tag}(i) \)), variables (e.g. \( \tau \)), predecessor \( \text{pred}(T) \)

- **Indices** for session identifiers:
  - Variable \( i \)

- **Macros** for protocol terms at a given time:
  - \( \text{input}@\tau, \text{output}@\tau, \text{frame}@\tau, s_T@\tau \)

- **Attacker function symbols** for adversary computations:
  - \( \text{att}(\text{frame}@\text{pred}(\tau)) \)
1: $T \rightarrow R:$ 

\[ s_T := H(s_T, k_H); \]

\[ \text{out}(G(s_T, k_G)) \]

2: $R \rightarrow T:$ 

\[ \text{in}(x); \]

\[ \text{if } x = G(H(s_R, k_H), k_G) \text{ then} \]

\[ \text{out}(\text{ok}); \]

\[ s_R := H(s_R, k_H) \]

Examples:

- **OSK tag $T$ state updates:**

  \[ s_T \odot \tau = H(s_T \odot \text{pred}(\tau), k_H) \]

- **Definition of $\text{input@}_\tau$:**

  \[ \text{att(frame@}\text{pred}(\tau)) \]
**Local formulas**: first-order formulas built over the atoms:

\[ t_1 = t_2, \, T_1 = T_2, \, T_1 \leq T_2, \, \text{happens}(T), \ldots \]

**Example**:

- **OSK tag T state updates**:

\[
\forall \tau. \left( \exists i. \, \tau = \text{Tag}(i) \wedge \text{happens}(\tau) \right) \rightarrow s_T @ \tau = H(s_T @ \text{pred}(\tau), k_H)
\]
Indistinguishability Logic: Local Formulas

- $\phi$ is valid w.r.t. $\mathcal{P}$ if it is true with overwhelming probability.

- **Example of valid formula:** w.r.t. any protocol $\mathcal{P}$
  - Random samplings freshness:
    
    $$n_1 \neq n_2$$
Local formulas can capture reachability security properties.

Example:

- **Authentication** of the OSK protocol:

  \[
  \forall \tau. \; \phi^R_{\text{accept}}[\tau] \rightarrow \exists i. \; \text{Tag}(i) \leq \tau \land \text{input} @ \tau = \text{output} @ \text{Tag}(i)
  \]
Global formulas: **first-order** logic formulas $\Phi$ over the atoms:

- $[\phi]_P$ where $\phi$ is a [local formula](#). *Valid* if the [local formula](#) $\phi$ is valid w.r.t. $P$.

- $[\vec{u} \sim \vec{v}]_{P_1, P_2}$ where $\vec{u}, \vec{v}$ are same-length sequences of terms. *Valid* if no PPTM $A$ can distinguish between $\vec{u}$ and $\vec{v}$. (w.r.t., respectively, $P_1$ and $P_2$)

Notations: $\tilde{\forall}, \tilde{\exists}$... to distinguish from local logic constructs.
Global formulas can capture equivalence security properties.

Example:

- Strong secrecy of the OSK state: (\(\mathcal{P} = \text{OSK}\))
  \[\forall \tau. [\text{happens}(\tau)]_{\mathcal{P}_1} \Rightarrow [\text{frame}\@\tau, s_T\@\tau \sim \text{frame}\@\tau, \text{n}_{\text{fresh}}]_{\mathcal{P}, \mathcal{P}}\]
Example of a valid global formula:

\[ [s = t]_{P_1} \Rightarrow [\vec{u}[s] \sim \vec{v}]_{P_1, P_2} \Rightarrow [\vec{u}[t] \sim \vec{v}]_{P_1, P_2} \]
Example of a valid global formula:

\[ [s = t]_{\mathcal{P}_1} \Rightarrow [\vec{u}[s] \sim \vec{v}]_{\mathcal{P}_1,\mathcal{P}_2} \Rightarrow [\vec{u}[t] \sim \vec{v}]_{\mathcal{P}_1,\mathcal{P}_2} \]

**Global formulas** allow to mix reachability and equivalence properties.
Sequents and Proof Systems
Local and Global Sequents

\[ \Sigma; \Theta : \Gamma \vdash \phi \quad \text{and} \quad \Sigma; \Theta \vdash \Phi \]

local formulas \hspace{2cm} \text{global formulas}

\[ \Sigma: \text{universally quantified variables} \]

Semantics

\[ \Sigma; \Theta \vdash \Phi \rightsquigarrow \forall \Sigma. (\tilde{\tilde{\Theta}} \Rightarrow \Phi) \]
\[ \Sigma; \Theta : \Gamma \vdash \phi \rightsquigarrow \forall \Sigma. (\tilde{\tilde{\Theta}} \Rightarrow [\land \Gamma \Rightarrow \phi]) \]
Classical FO inference rules are sound:

- Purely local (local seq.):

\[
\begin{align*}
\Sigma; \Theta : \Gamma, \phi_1 & \vdash \psi & \Sigma; \Theta : \Gamma, \phi_2 & \vdash \psi \\
\Sigma; \Theta : \Gamma, \phi_1 \lor \phi_2 & \vdash \psi
\end{align*}
\]
Proof System: Classical Reasoning

Classical FO inference rules are sound:

- **Purely local (local seq.):**

  \[ \Sigma; \Theta : \Gamma, \phi_1 \vdash \psi \quad \Sigma; \Theta : \Gamma, \phi_2 \vdash \psi \]

  \[ \Sigma; \Theta : \Gamma, \phi_1 \lor \phi_2 \vdash \psi \]

- **Purely global (local and global seq.):**

  \[ \Sigma; \Theta, \Phi_1 : \Gamma \vdash \psi \quad \Sigma; \Theta, \Phi_2 : \Gamma \vdash \psi \]

  \[ \Sigma; \Theta, \Phi_1 \lor \Phi_2 : \Gamma \vdash \psi \]

  \[ \Sigma; \Theta, \Phi_1 \lor \Phi_2 \vdash \psi \]

  \[ \Sigma; \Theta, \Phi_1 \lor \Phi_2 \vdash \psi \]
Selected *inference rules* involving *mixed kinds of sequents*:

**Global-Local**

\[
\begin{align*}
\Sigma; \Theta \vdash [\phi]_P \\
\Sigma; \Theta : \Gamma \vdash \phi
\end{align*}
\]

**Local-Global**

\[
\begin{align*}
\Sigma; \Theta : \Gamma \vdash P \phi \\
\Sigma; \Theta \vdash [\phi]_P
\end{align*}
\]

**Rewrite-Equiv**

\[
\begin{align*}
\Sigma; \Theta \vdash [\phi \sim \psi]_{P,P'} \\
\Sigma; \Theta : \Gamma \vdash P' \psi \\
\Sigma; \Theta : \Gamma \vdash P \phi
\end{align*}
\]
Selected **inference rules** involving **mixed kinds of sequents**:

**Global-Local**

\[
\begin{align*}
\Sigma; \Theta & \vdash \left[ \phi \right]_P \\
\Sigma; \Theta & : \vdash \neg_P \phi
\end{align*}
\]

**Local-Global**

\[
\begin{align*}
\Sigma; \Theta & : \vdash P \phi \\
\Sigma; \Theta & \vdash \left[ \phi \right]_P
\end{align*}
\]

**Rewrite-Equiv**

\[
\begin{align*}
\Sigma; \Theta & \vdash \left[ (\Gamma \Rightarrow \phi) \sim (\Delta \Rightarrow \psi) \right]_{P,P'} \\
\Sigma; \Theta & : \Delta \vdash \neg_{P'} \psi \\
\Sigma; \Theta & : \Gamma \vdash \neg_P \phi
\end{align*}
\]
Example: Strong Secrecy $\rightarrow$ Weak Secrecy

Example:

Strong secrecy of a state value $s_T$:

$$
\Phi_S \overset{\text{def}}{=} [\text{frame}@T, s_T@T \sim \text{frame}@T, n_{\text{fresh}}]_P,P
$$

implies weak secrecy of $s_T$:

$$
\text{input}@T \neq s_T@T
$$
Example: Strong Secrecy $\rightarrow$ Weak Secrecy

Example:

**Strong secrecy** of a state value $s_\tau$:

$$\Phi_S \overset{\text{def}}{=} [\text{frame}@\tau, s_\tau @ \tau \sim \text{frame}@\tau, n_{\text{fresh}}]_{\mathcal{P}, \mathcal{P}}$$

implies **weak secrecy** of $s_\tau$:

$$\text{input}@\tau \neq s_\tau @ \tau$$

Proof:

$\tau; \Phi_{\text{hap}}, \Phi_S \vdash [\text{input}@\tau \neq s_\tau @ \tau] \sim [\text{input}@\tau \neq n_{\text{fresh}}]_{\mathcal{P}, \mathcal{P}}$

$$\tau; \Phi_{\text{hap}}, \Phi_S : \vdash_{\mathcal{P}} \text{input}@\tau \neq n_{\text{fresh}}$$

$$\tau; \Phi_{\text{hap}}, \Phi_S : \vdash_{\mathcal{P}} \text{input}@\tau \neq s_\tau @ \tau$$

(\text{where } \Phi_{\text{hap}} \text{ is } [\text{happens}(\tau)]_{\mathcal{P}})
Example: Strong Secrecy $\rightarrow$ Weak Secrecy

Example:

**Strong secrecy** of a state value $s_T$:

$$\Phi_S \overset{\text{def}}{=} [\text{frame} @ \tau, s_T @ \tau \sim \text{frame} @ \tau, n_{\text{fresh}}] \mathcal{P}, \mathcal{P}$$

implies **weak secrecy** of $s_T$:

$$\text{input} @ \tau \neq s_T @ \tau$$

**Proof:**

$$\tau; \Phi_{\text{hap}}, \Phi_S \vdash [(\text{input} @ \tau \neq s_T @ \tau) \sim (\text{input} @ \tau \neq n_{\text{fresh}})] \mathcal{P}, \mathcal{P}$$

$$\tau; \Phi_{\text{hap}}, \Phi_S : \vdash_\mathcal{P} \text{input} @ \tau \neq n_{\text{fresh}}$$

$$\tau; \Phi_{\text{hap}}, \Phi_S : \vdash_\mathcal{P} \text{input} @ \tau \neq s_T @ \tau$$

- 2\textsuperscript{nd} premise: consequence of $n_{\text{fresh}}$ freshness
- 1\textsuperscript{st} premise: RHS can be (bi)-deduced from $\Phi_S$!

(where $\Phi_{\text{hap}}$ is $[\text{happens}(@\tau)]_{\mathcal{P}}$)
Bi-Deduction
Indistinguishability is preserved by (public) computation:

\[
\text{if } [\vec{u}_1 \sim \vec{u}_2] \text{ then } \forall B. \ [B(\vec{u}_1) \sim B(\vec{u}_2)]
\]

As a pseudo-inference rule:

\[
\exists B \text{ s.t. } B \text{ computes } \vec{v}_i \text{ from } \vec{u}_i
\]

\[
\Sigma; \ \Theta, [\vec{u}_1 \sim \vec{u}_2] \vdash [\vec{v}_1 \sim \vec{v}_2]
\]
\( \exists B \text{ s.t. } B \text{ computes } \vec{v}_i \text{ from } \vec{u}_i \)

\[
\Sigma; \Theta, [\vec{u}_1 \sim \vec{u}_2] \vdash [\vec{v}_1 \sim \vec{v}_2]
\]

**Example:**

\( \tau; \Phi_{\text{hap}}, [\text{frame}@\tau, s_\tau@\tau \sim \text{frame}@\tau, n_{\text{fresh}}] \vdash [\text{(input}@\tau \neq s_\tau@\tau) \sim (\text{input}@\tau \neq n_{\text{fresh}})] \)

Proved by bi-deduction with:

\[
B(\text{frame}@\tau, x) \overset{\text{def}}{=} (\text{att}(\text{frame}@\text{pred}(\tau)) = x)
\]
The bi-deduction rule:

\[
\text{Bi-Deduce} \\
\Sigma; \#(\vec{u}_1, \vec{u}_2) \triangleright \#(\vec{v}_1, \vec{v}_2) \\\n\Sigma; \Theta, [\vec{u}_1 \sim \vec{u}_2] \vdash [\vec{v}_1 \sim \vec{v}_2]
\]

We designed a proof system for bi-deduction, e.g.:

\[
\text{FA} \\
\Sigma; \#(\vec{u}_1, \vec{u}_2) \triangleright \#(\vec{v}_1, \vec{v}_2) \\\n\Sigma; \#(\vec{u}_1, \vec{u}_2) \triangleright \#(\text{f}(\vec{v}_1), \text{f}(\vec{v}_2))
\]
Fully-automated procedure for bi-deduction implemented in Squirrel:

- soundness follows from our bi-deduction proof system;
- integrated in the apply tactic (for global sequents);
- extension with fully-automated inductive reasoning using abstract interpretation.
Case-Studies
Security analysis of the Yubi-Key protocol (used for 2FA).

- Yubi-Keys are physical authentication devices with a single button, which generated an OTP (one-time password).
- Uses counters for protection against replay-attack:
  - OTPs include the encrypted Yubi-Key counter;
  - the counter is incremented after each session.
- We prove injective authentication:
  - successful login must be preceded by a button press;
  - each counter value is accepted at most once.
Also studied the Yubi-HSM protocol:

- Yubi-HSM = Yubi-Key + keys stored in a HSM (server side).
- We prove injective authentication
Case-Study: Yubi-HSM

Also studied the **Yubi-HSM** protocol:

- **Yubi-HSM** = **Yubi-Key** + keys stored in a HSM (server side).
- We prove **injective authentication** in two steps:
  - equivalence of **Yubi-HSM** with an idealized version;
  - proof of injective authentication, using **Rewrite-Equiv** to switch from the real to the ideal protocol.
Conclusion
Conclusion

Our Contributions

- Extend the indistinguishability logic with mutable state.
- Generalized sequent calculus.
  - Mix reachability and equivalence reasonings.
- Proof automation: design a proof system for bi-deduction.
  - Allow for automation of some proof steps.
- Implementation in SQUIRREL + case-studies: Yubi-{Key,HSM}.

Future Works

- More complex protocols and security properties.
- More automation, e.g. using SMT solvers.
- Systematic translation of crypto. assumptions as inference rules.
Thank you for your attention
Proof System: Local $\neq$ Global

Local hypothesis $\neq$ global hypothesis:

- **Global hypothesis**: property of a bitstring distribution
- **Local hypothesis**: property of a bitstring

Global hyp. are stronger than local hyp.:

\[
\Sigma; \Theta : \phi, \Gamma \vdash \psi \quad \Rightarrow \quad \phi \rightarrow \psi \text{ true with overwh. prob.}
\]

\[
\Sigma; [\phi]_\mathcal{P}, \Theta : \Gamma \vdash \psi \quad \Rightarrow \quad \phi \text{ true with overwh. prob.}
\]
implies
\[
\psi \text{ true with overwh. prob.}
\]

But the converse does not generally hold.

**Counter-example:**

- $n = 0 \rightarrow n = 1$ not valid
- $[n = 0] \Leftrightarrow [n = 1]$ valid