Systematic translation of Cryptographic Axioms by Bi-deduction
(Work in progress)

David Baelde, Adrien Koutsos, Justine Sauvage

Irisa, Inria Paris
Formal verification of cryptographic protocols.

Examples:
- Secure online payment (authentication)
- Secure messaging (privacy)
Squirrel and Cryptographic axioms

Protocol

hash, encryption . . .

Cryptographic axioms

→ Squirrel
(Logic and Proof assistant)
formula, axioms, proof system . . .

Justine Sauvage (Inria Paris)
This talk

- Insight of cryptographic axioms and Squirrel logic
- Capture cryptographic axioms into formulas, while ensuring their correctness.
Definition (Indistinguishability)

For any polynomial-time and randomized algorithms $A$,

$$| \Pr(A^{G_0} = 1) - \Pr(A^{G_1} = 1) |$$

is negligible (i.e., roughly exponentially small in the length of the keys).
Example: PRF games

Intuition: a pseudo random function is a function that “seams” random.

<table>
<thead>
<tr>
<th>Beispiel (PRF-Spiele)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spiel $G_0$</td>
</tr>
<tr>
<td><strong>Initialisierung</strong></td>
</tr>
<tr>
<td>$\text{sample}(k)$</td>
</tr>
<tr>
<td><strong>Hash</strong></td>
</tr>
<tr>
<td>$L := x$</td>
</tr>
<tr>
<td>$\text{return } h(x, k)$</td>
</tr>
<tr>
<td><strong>Herausforderung</strong></td>
</tr>
<tr>
<td>$\text{sample}(r)$</td>
</tr>
<tr>
<td>$L := x$</td>
</tr>
<tr>
<td>$\text{return } h(x, k)$</td>
</tr>
</tbody>
</table>

| Spiel $G_1$ |
| **Initialisierung** |
| $\text{sample}(k)$ |
| **Hash** |
| $L := x$ |
| $\text{return } h(x, k)$ |
| **Herausforderung** |
| $\text{sample}(r)$ |
| $L := x$ |
| $\text{return } r$ |
Example: PRF games

Intuition: a pseudo random function is a function that “seams” random.

<table>
<thead>
<tr>
<th>Game $G_0$</th>
<th>$Init$ :</th>
<th>$Challenge(x)$ :</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sample$(k)$;</td>
<td>$return h(x,k)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Game $G_1$</th>
<th>$Init$ :</th>
<th>$Challenge(x)$ :</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sample$(k)$;</td>
<td>$sample(r)$</td>
</tr>
</tbody>
</table>

$return r$
Example: PRF games

Intuition: a pseudo random function is a function that “seams” random.

Example (PRF games)

<table>
<thead>
<tr>
<th>Game $G_0$</th>
<th>$Init$ :</th>
<th>$Hash(x)$ :</th>
<th>$Challenge(x)$ :</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sample($k$);</td>
<td>$return \ h(x, k)$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Game $G_1$</th>
<th>$Init$ :</th>
<th>$Hash(x)$ :</th>
<th>$Challenge(x)$ :</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sample($k$);</td>
<td>$return \ h(x, k)$</td>
<td>sample($r$)</td>
</tr>
</tbody>
</table>
Example: PRF games

Intuition: a pseudo random function is a function that “seams” random.

<table>
<thead>
<tr>
<th>Game $G_0$</th>
<th>Init</th>
<th>Hash($x$)</th>
<th>Challenge($x$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\text{sample}(k)$; $L := x :: L$</td>
<td>$L := x :: L$; $\text{return } h(x, k)$</td>
<td>$\text{sample}(r)$; $\text{if } x \notin L$; $L := x :: L$; $\text{return } h(x, k)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Game $G_1$</th>
<th>Init</th>
<th>Hash($x$)</th>
<th>Challenge($x$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\text{sample}(k)$; $L := x :: L$</td>
<td>$L := x :: L$; $\text{return } h(x, k)$</td>
<td>$\text{sample}(r)$; $\text{if } x \notin L$; $L := x :: L$; $\text{return } r$</td>
</tr>
</tbody>
</table>
Example: PRF games

Example (PRF pair of games)

<table>
<thead>
<tr>
<th>Game $G_{PRF}$ Init</th>
<th>Hash$(x)$</th>
<th>Challenge$(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample$(k)$; $l := []$;</td>
<td>$L := x :: L$</td>
<td>sample$(r)$ if $x \not\in L$</td>
</tr>
<tr>
<td></td>
<td>$h(x, k)$</td>
<td>$L := x :: L$; $#(h(x, k), r)$</td>
</tr>
</tbody>
</table>
Playing with PRF: sequence of messages

$$G_{PRF}$$

$$m_1, h(m_1, k)$$

$$A$$

$$m_1, h(m_1, k)$$
Playing with PRF: sequence of messages

\[ G_{PRF} \]

\[ A \]

\[ m_1, h(m_1, k), m_2, \#(h(m_2, k), r) \]

\[ := ( (m_1, h(m_1, k), m_2, h(m_2, k)), (m_1, h(m_1, k), m_2, r) ) \]
Playing with PRF: sequence of messages

It there exists an adversary that can distinguish between this two sequences of messages, then PRF doesn’t holds.
Terms and formulas

Definition (Terms)

Intuition: terms represent messages
Semantics: interpreted as computation of a Turing machine.

\[ t := r \quad \text{(sampling)} \]
\[ | f(t_1, \ldots, t_n) \quad \text{(function application)} \]
\[ | \#(t_0, t_1) \quad \text{(left/right difference)} \]

Definition (Equivalence formulas)

\[ \text{equiv}(t) \]
PRF axiom schema

Question: is this formula always valid according to PRF?

\[
equiv((m_1, h(m_1, k), m_2, #(h(m_2, k), r)))
\]
PRF axiom schema

Question: is this formula always valid according to PRF?

\[
equiv((m_1, h(m_1, k), m_2, \#(h(m_2, k), r)))
\]

- \(m_1 = k\): k adversary must not directly access the key.
PRF axiom schema

Question: is this formula always valid according to PRF?

\[ \text{equiv}((m_1, h(m_1, k), m_2, \#(h(m_2, k), r))) \]

- \( m_1 = k \): k adversary must not directly access the key.
- \( m_1 = m_2 \): forbidden by the game.
PRF axiom schema

Question: is this formula always valid according to PRF?

\[
equiv((m_1, h(m_1, k), m_2, \#(h(m_2, k), r)))
\]

- \(m_1 = k\): k adversary must not directly access the key.
- \(m_1 = m_2\): forbidden by the game.
- \(m_1 = r\): r must be fresh.
**PRF axiom schema**

Question: is this formula always valid according to PRF?

\[
equiv((m_1, h(m_1, k), m_2, #h(m_2, k), r)))
\]

- \(m_1 = k\): k adversary must not directly access the key.
- \(m_1 = m_2\): forbidden by the game.
- \(m_1 = r\): r must be fresh.

**Definition (PRF axiom schema)**

For all terms \(\vec{t}\) and \(m\), samplings \(k\) and \(r\) such that

- \(k\) never appears in \(\vec{t}\) and \(m\) except: \(h(_, k)\).
- for all subterms \(h(m', k)\) of \(\vec{t}\) or \(m\): \(m\) and \(m'\) are never equal.
- \(r\) never appears in \(\vec{t}\) and \(m\)

\[
equiv((\vec{t}, #(h(m, k), r)))
\]
Problems and contributions

Problem with this method
Ad-hoc and manual work for each cryptographic axioms:

- Axiom schema design
- Correctness proof (understand the logic and its semantics)
- Implementation (understand the code)

Contributions

- A systematic way to prove that a formula is a consequence of a cryptographic axioms.
- Automation (WIP)
Changing point of view

Input:
\[ m_1, h(m_1, k), m_2, h(m_1, k), m_3, \#(h(m_3, k), r_{\text{fresh}}) \]

Question: Does there exist such \( A \)?
Construction of bi-deduction judgement: starting point

Intuition: there exists $A$ such that $A^{G_{PRF}}() = \vec{v}$.

$\triangleright \vec{v}$

Definition (Link between Bi-deduction and Equivalence)

Intuition: if an adversary can compute $\vec{v}$ then the formula $\text{equiv}(\vec{v})$ holds.

\[
\begin{align*}
\text{Bi-deduction} & \\
\triangleright \vec{v} & \\
\text{equiv}(\vec{v}) &
\end{align*}
\]
Proof system

Goal: Proof system for this bi-deduction judgement.

What can compute an adversary?

- An adversary is a program: function applications
- An adversary can draw samples: samplings.
- Interaction with the games: oracles calls
**Proof system**

**Goal:** Proof system for this bi-deduction judgement.

What can compute an adversary?

- An adversary is a program: function applications *(done)*
- An adversary can draw samples: samplings.
- Interaction with the games: oracles calls

**Definition**

Function application inference rule

\[
\begin{array}{c}
\text{FA} \\
\triangleright \vec{t} \\
\Rightarrow f(\vec{t})
\end{array}
\]
We need to keep track of the owner of each sampling.

**Definition (Tags)**

\[ Tag = \{ T_a, T_{g,0}, T_{g,1}, \ldots \} \]

\[ n \leftarrow T_a \]

\[ s \leftarrow T_a \]

\[ k \leftarrow T_{g,\text{key}} \]
Extending bi-deduction with constraints

Adding sampling tagging

$C$ records who sampled what:

$$C : \triangleright \vec{v}$$

Definition (Adversary samplings)

$$\text{ADV SAMPLING}
\begin{align*}
C : \triangleright \vec{v} \\
C, < n, T_a > : \triangleright n, \vec{v}
\end{align*}$$

$$\emptyset : \triangleright \emptyset$$

$$\emptyset : \triangleright \emptyset$$

$$\triangleright h(n, s)$$
Extending bi-deduction with constraints

Adding sampling tagging

$C$ records who sampled what:

$C : \triangleright \vec{v}$

Definition (Adversary samplings)

\[
\begin{align*}
\text{ADV SAMPLING} \\
C : \triangleright \vec{v} \\
\hline
C, < n, T_a > : \triangleright n, \vec{v}
\end{align*}
\]

\[
\begin{align*}
\emptyset : \triangleright \emptyset \\
\hline
< s, T_a > : \triangleright s
\end{align*}
\]

\[
\begin{align*}
\text{ADV SAMPLING} \\
\hline
: \triangleright h(n, s)
\end{align*}
\]
Extending bi-deduction with constraints

Adding sampling tagging

$C$ records who sampled what:

$C : \triangleright \vec{v}$

Definition (Adversary samplings)

\[
\text{ADV SAMPLING}
\]

\[
C : \triangleright \vec{v}
\]

\[
C, < n, T_a > : \triangleright n, \vec{v}
\]

\[
\emptyset : \triangleright \emptyset
\]

\[
< s, T_a > : \triangleright s
\]

\[
< n, T_a >, < s, T_a > : \triangleright n, s
\]

\[
: \triangleright h(n, s)
\]
Extending bi-deduction with constraints

Adding sampling tagging

$C$ records who sampled what:

$$C : \triangleright \vec{v}$$

Definition (Adversary samplings)

\[
\begin{align*}
\text{ADV SAMPLING} \\
C : \triangleright \vec{v} \\
\hline
C, < n, T_a > : \triangleright n, \vec{v}
\end{align*}
\]

\[
\begin{align*}
\emptyset : \triangleright \emptyset \\
\hline
< s, T_a > : \triangleright s & \text{ADV SAMPLING} \\
\hline
< n, T_a >, < s, T_a > : \triangleright n, s & \text{ADV SAMPLING} \\
\hline
< n, T_a >, < s, T_a > : \triangleright h(n, s) & \text{FA}
\end{align*}
\]
**Oracle calls on example**

**Definition (Oracle rule: instanciated for hash oracle)**

\[
\text{HASH} \\
C : \diamondsuit m, \vec{v} \\
C, < k, T_{g, key} > : \lozenge h(m, k), \vec{v}
\]

**Example**

\[
h(n, s), h(h(n, s), k)
\]

\[
* \\
C : \lozenge h(n, s)
\]
Oracle calls on example

**Definition (Oracle rule : instenciated for hash oracle)**

\[
\text{HASH} \\
\begin{array}{c}
C : \triangleright m, \vec{v} \\
\hline
C, < k, T_{g, key} > : \triangleright h(m, k), \vec{v}
\end{array}
\]

**Example**

\[
h(n, s), h(h(n, s), k)
\]

\[
* \\
\begin{array}{c}
C : \triangleright h(n, s) \\
\hline
C, < k, T_{g, key} > : \triangleright h(h(n, s), k)
\end{array}
\]
Consistency of tagging

\[ \text{BI-Deduction} \]
\[ C : \not\exists \vec{t} \quad \vdash \text{Valid}(C) \]
\[ \quad \text{equiv}(\vec{t}) \]

\textbf{Valid}(C) ensures:

- Not two samples for one “role” (e.g., \( k \leftarrow T_{g,\text{key}}, k' \leftarrow T_{g,\text{key}} \))
- No sample owned by both the adversary and the oracles
Oracle rule : Challenge

Oracle rule instanciated for challenge

\[ C : \triangleright m, \vec{v} \]
\[ C, < r, T_g >, < k, T_{g, key} > : \triangleright \#(h(m, k), r), \vec{v} \]

\[ m \rightarrow h(m, k) \rightarrow \#(h(m, k), r) \]
Oracle rule: Challenge

Oracle rule instantiated for challenge

\[
C : \triangleright m, \vec{v}
\]

\[
C, < r, T_g >, < k, T_{g, key} > : \triangleright \#(h(m, k), r), \vec{v}
\]

\[
m \rightarrow h(m, k) \rightarrow \#(h(m, k), r)
\]

\[
L = [m] \quad L = [m, m]?
\]
Hoare’s style oracle triplets

**Definition (Hoare’s triples for an oracle o)**

Let $\phi$ and $\psi$ be pre and post conditions.

\[
\{ \phi \} c_o[\vec{t}, \vec{s}] \{ \psi \}
\]

is correct iff:
- When $\phi$ holds the oracle $o$ return $c_o[\vec{t}, \vec{s}]$ on input $\vec{t}$ and samplings $\vec{s}$.
- $\psi$ holds after $o$ call.

When $L = lis \implies m \notin L$ then

\[
\{ L := lis \} \#(h(m, k), r) \{ L := m :: lis \}
\]

is correct.

**Adding pre and post conditions**

$\phi, \psi; C : \triangleright \vec{v}$
Definition (Oracle rule)

**Oracle**

\[
\frac{\phi, \psi; C : \triangleright \vec{s}, \vec{t}}{\phi, \theta; C, < \vec{s}, T_g, \ldots > : \triangleright c_o[\vec{s}, \vec{t}], \vec{v}}
\]
Induction and abstract semantics

Challenges following:

Recursive terms: induction in the proof system

→ Approximate pre and post condition

- Abstract representation of the game memory.
- Adapt the rule and/or the proof of soundness regarding these approximation.
Conclusion

- Formal framework linking games, adversaries, and formulas
- Bi-deduction judgment to capture adversaries interacting with a game
- Proof system for this judgment
- Over-approximation of pre and post conditions (WIP)
- Implementation
  - Automation of proof search (WIP)
  - How to generate/check Hoare triples? (Future Work)
Questions?


David Baelde, Stéphanie Delaune, Adrien Koutsos, and Solène Moreau, Cracking the stateful nut, 2022.

(justine.sauvage@inria.fr)