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

David Baelde, Adrien Koutsos, Justine Sauvage

Irisa, Inria Paris

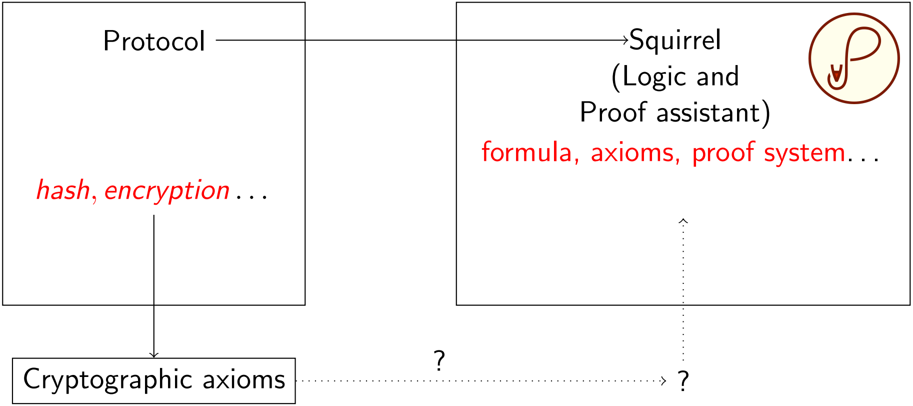
Context

Formal verification of cryptographic protocols.

Examples:

- Secure online payment (authentication)
- Secure messaging (privacy)

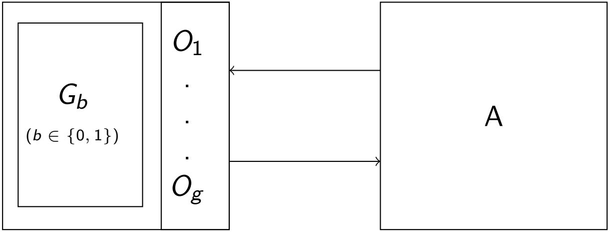
Squirrel and Cryptographic axioms



This talk

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

Cryptographic axiom



Definition (Indinstinguishability)

For any polynomial-time and randomized algorithms A,

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

is negligible (*i.e.*, roughtly exponentially small in the length of the keys).

Example: PRF games

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

Example (PRF games)

Game G_0

$Challenge(x)$:

$return\ \boxed{\mathsf{h}(\mathsf{x},\mathsf{k})}$

Game G_1

$Challenge(x)$:

sample(r)

$return\ \boxed{\mathsf{r}}$

Example: PRF games

Intuition: a pseudo random function is a function that "seams" random.

Example (PRF games)

Game G_0 $\quad$ *Init* :

sample(k);

Challenge(x) :

return $\boxed{h(x,k)}$

Game G_1 $\quad$ *Init* :

sample(k);

Challenge(x) :

sample(r)

return $\boxed{r}$

Example: PRF games

Intuition: a pseudo random function is a function that "seams" random.

Example (PRF games)

Game G_0	Init : sample(k);	$\mathit{Hash}(x)$: *return* $\mathsf{h}(x, \mathsf{k})$	$\mathit{Challenge}(x)$: *return* $\boxed{\mathsf{h}(\mathsf{x},\mathsf{k})}$
Game G_1	Init : sample(k);	$\mathit{Hash}(x)$: *return* $\mathsf{h}(x, \mathsf{k})$	$\mathit{Challenge}(x)$: sample(r) *return* $\boxed{\mathsf{r}}$

Example: PRF games

Intuition: a pseudo random function is a function that "seams" random.

Example (PRF games)

Game G_0	$Init$:	$Hash(x)$:	$Challenge(x)$:
	sample(k);	$L := x :: L$	sample(r)
	$L := []$;	$return\ \mathsf{h}(x, \mathsf{k})$	$if\ x \notin L$
			$L := x :: L$;
			$return\ \boxed{\mathsf{h}(\mathsf{x},\mathsf{k})}$

Game G_1	$Init$:	$Hash(x)$:	$Challenge(x)$:
	sample(k);	$L := x :: L$	sample(r)
	$L := []$;	$return\ \mathsf{h}(x, \mathsf{k})$	$if\ x \notin L$
			$L := x :: L$;
			$return\ \boxed{\mathsf{r}}$

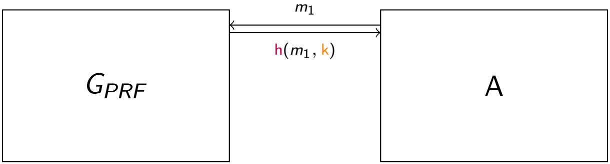
Example: PRF games

Example (PRF pair of games)

Game G_{PRF}

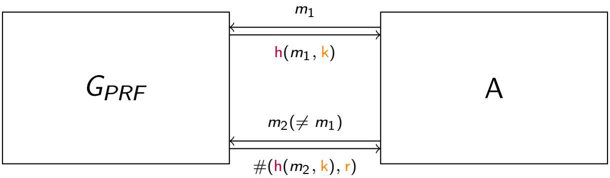
$Init$:	$Hash(x)$:	$Challenge(x)$:
sample(k);	$L := x :: L$	sample(r)
$l := []$;	$\mathsf{h}(x, \mathsf{k})$	$if\ x \notin L$
		$L := x :: L$;
		$\boxed{\#(\mathsf{h}(\mathsf{x},\mathsf{k}),\mathsf{r})}$

Playing with PRF: sequence of messages



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

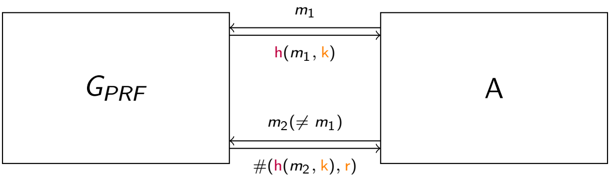
Playing with PRF: sequence of messages



$$m_1, h(m_1, k), m_2, \#(h(m_2, k), r)$$

$$:= (\ (m_1, h(m_1, k), m_2, h(m_2, k)),\\ (m_1, h(m_1, k), m_2, r)\ )$$

Playing with PRF: sequence of messages



$$\mathrm{equiv}((m_1, \mathrm{h}(m_1, \mathrm{k}), m_2, \#(\mathrm{h}(m_2, \mathrm{k}), \mathrm{r})))$$

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 represents messages

Semantics: interpreted as computation of a turing machine.

$$
\begin{aligned}
t := &\mid \mathsf{r} && \text{(sampling)}\\
&\mid f(t_1, \ldots, t_n) && \text{(function application)}\\
&\mid \#(t_0, t_1) && \text{(left/right difference)}
\end{aligned}
$$

Definition (Equivalence formulas)

$$\mathsf{equiv}(\vec{t})$$

PRF axiom schema

Question : is this formula always valid according to PRF ?

$$\mathsf{equiv}((m_1, \mathsf{h}(m_1, \mathsf{k}), m_2, \#(\mathsf{h}(m_2, \mathsf{k}), \mathsf{r})))$$

PRF axiom schema

Question : is this formula always valid according to PRF ?

$$\mathsf{equiv}((m_1, \mathsf{h}(m_1, \mathsf{k}), m_2, \#(\mathsf{h}(m_2, \mathsf{k}), \mathsf{r})))$$

- $m_1 = \mathsf{k}$: k adversary must not directly access the key.

PRF axiom schema

Question : is this formula always valid according to PRF ?

$$\mathsf{equiv}((m_1, \mathsf{h}(m_1, \mathsf{k}), m_2, \#(\mathsf{h}(m_2, \mathsf{k}), \mathsf{r})))$$

- $m_1 = \mathsf{k}$: 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 ?

$$\mathsf{equiv}((m_1, \mathsf{h}(m_1, \mathsf{k}), m_2, \#(\mathsf{h}(m_2, \mathsf{k}), \mathsf{r})))$$

- $m_1 = \mathsf{k}$: k adversary must not directly access the key.
- $m_1 = m_2$: forbidden by the game.
- $m_1 = \mathsf{r}$: r must be fresh.

PRF axiom schema

Question : is this formula always valid according to PRF ?

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

- $m_1 = \text{k}$: k adversary must not directly access the key.
- $m_1 = m_2$: forbidden by the game.
- $m_1 = \text{r}$: 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

$$\frac{}{\text{equiv}((\vec{t}, \#(h(m, k), r)))}$$

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)

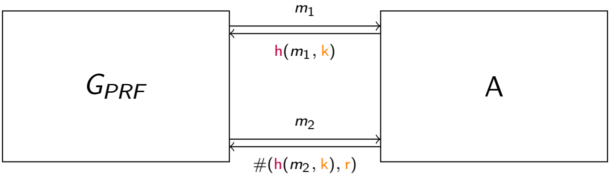
Contributions

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

Changing point of view

Input :

$$m_1, h(m_1, k), m_2, h(m_1, k), m_3, \#(h(m_3, k), r_{fresh})$$



Question: Does there exists such A ?

Bi-deduction

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 equiv$(\vec{v})$ holds.

$$\frac{\triangleright \vec{v}}{\mathsf{equiv}(\vec{v})}\ \textsc{Bi-deduction}$$

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

$$\mathrm{FA}\;\frac{\triangleright \vec{t}}{\triangleright f(\vec{t})}$$

Samplings

Example

$$\mathsf{h}(\mathsf{n}, \mathsf{s}), \mathsf{h}(\mathsf{h}(\mathsf{n}, \mathsf{s}), \mathsf{k})$$

adversary game

We need to keep track of the owner of each sampling.

Definition (Tags)

$$Tag = \{ T_a, T_{g,0}, T_{g,1}, \dots \}$$

$$\mathsf{n} \leftarrow T_a$$
$$\mathsf{s} \leftarrow T_a$$
$$\mathsf{k} \leftarrow T_{g,key}$$

Extending bi-deduction with constraints

Adding sampling tagging

C records who sampled what:

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

Definition (Adversary samplings)

$$\textsc{Adv sampling}\ \frac{C : \triangleright \vec{v}}{C, < \mathsf{n}, T_a >: \triangleright \mathsf{n}, \vec{v}}$$

$$\frac{\overline{\emptyset : \triangleright \emptyset}}{}$$

$$: \triangleright \mathsf{h}(\mathsf{n}, \mathsf{s})$$

Extending bi-deduction with constraints

Adding sampling tagging

C records who sampled what:

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

Definition (Adversary samplings)

$$\textsc{Adv sampling}\ \frac{C : \triangleright \vec{v}}{C, < \mathsf{n}, T_a >: \triangleright \mathsf{n}, \vec{v}}$$

$$\cfrac{\cfrac{\cfrac{}{\emptyset : \triangleright \emptyset}}{< \mathsf{s}, T_a >: \triangleright \mathsf{s}}\textsc{Adv sampling}}{\cfrac{}{: \triangleright \mathsf{h}(\mathsf{n}, \mathsf{s})}}$$

Extending bi-deduction with constraints

Adding sampling tagging

C records who sampled what:

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

Definition (Adversary samplings)

$$\frac{C : \rhd \vec{v}}{C, < \mathsf{n}, T_a >: \rhd \mathsf{n}, \vec{v}}\ \textsc{Adv sampling}$$

$$\dfrac{\dfrac{\dfrac{}{\emptyset : \rhd \emptyset}}{< \mathsf{s}, T_a >: \rhd \mathsf{s}}\textsc{Adv sampling}}{\dfrac{< \mathsf{n}, T_a >, < \mathsf{s}, T_a >: \rhd \mathsf{n}, \mathsf{s}}{: \rhd \mathsf{h}(\mathsf{n}, \mathsf{s})}}\textsc{Adv sampling}$$

Extending bi-deduction with constraints

Adding sampling tagging

C records who sampled what:

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

Definition (Adversary samplings)

$$\textsc{Adv sampling}\ \frac{C : \rhd \vec{v}}{C, < \mathsf{n}, T_a >: \rhd \mathsf{n}, \vec{v}}$$

$$\cfrac{\cfrac{\cfrac{}{\emptyset : \rhd \emptyset}}{< \mathsf{s}, T_a >: \rhd \mathsf{s}}\textsc{Adv sampling}}{\cfrac{< \mathsf{n}, T_a >, < \mathsf{s}, T_a >: \rhd \mathsf{n}, \mathsf{s}}{< \mathsf{n}, T_a >, < \mathsf{s}, T_a > \ : \rhd \mathsf{h}(\mathsf{n}, \mathsf{s})}\mathrm{FA}}\textsc{Adv sampling}$$

Oracle calls on example

Definition (Oracle rule : instenciated for hash oracle)

$$\textsc{Hash}\ \frac{C : \triangleright m, \vec{v}}{C, < \mathsf{k}, T_{g,key} >: \triangleright \mathsf{h}(m, \mathsf{k}), \vec{v}}$$

Example

$$\mathsf{h}(\mathsf{n}, \mathsf{s}), \mathsf{h}(\mathsf{h}(\mathsf{n}, \mathsf{s}), \mathsf{k})$$

$$\dfrac{\dfrac{*}{C : \triangleright \mathsf{h}(\mathsf{n}, \mathsf{s})}}{}$$

Oracle calls on example

Definition (Oracle rule : instenciated for hash oracle)

$$\textsc{Hash}\quad \frac{C : \rhd m, \vec{v}}{C, < \mathsf{k}, T_{g,key} >: \rhd \mathsf{h}(m, \mathsf{k}), \vec{v}}$$

Example

$$\mathsf{h}(\mathsf{n}, \mathsf{s}), \mathsf{h}(\mathsf{h}(\mathsf{n}, \mathsf{s}), \mathsf{k})$$

$$\frac{\dfrac{*}{C : \rhd \mathsf{h}(\mathsf{n}, \mathsf{s})}}{C, < \mathsf{k}, T_{g,key} >: \rhd \mathsf{h}(\mathsf{h}(\mathsf{n}, \mathsf{s}), \mathsf{k})}\,\textsc{Hash}$$

Consistency of tagging

$$\frac{\textsc{Bi-deduction}\quad C : \triangleright \vec{t} \qquad \vdash \mathit{Valid}(C)}{\mathrm{equiv}(\vec{t})}$$

$\mathit{Valid}(C)$ ensures :

- Not two samples for one "role" ($e.g.$, $k \leftarrow T_{g,key}, k' \leftarrow T_{g,key}$)
- No sample owned by both the adversary and the oracles

Oracle rule : Challenge

Oracle rule instenciated for challenge

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

$$m \longrightarrow \mathsf{h}(m, \mathsf{k}) \longrightarrow \#(\mathsf{h}(m, \mathsf{k}), \mathsf{r})$$

Oracle rule : Challenge

Oracle rule instenciated for challenge

$$\frac{C : \rhd m, \vec{v}}{C, < \mathsf{r}, T_g >, < \mathsf{k}, T_{g,key} >: \rhd \#(\mathsf{h}(m, \mathsf{k}), \mathsf{r}), \vec{v}}$$

$$m \xrightarrow{\hspace{3cm}} \mathsf{h}(m, \mathsf{k}) \xrightarrow{\hspace{2cm}} \#(\mathsf{h}(m, \mathsf{k}), \mathsf{r})$$

$L = [m]$ $\quad\quad\quad$ $L = [m, m]$?

Hoare's style oracle triplets

Definition (Hoare's triples for an oracle o)

Let ϕ and ψ be pre and post conditions.

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

is correct iff:

- When ϕ holds the oracle o return $c_o[\vec{t},\vec{s}]$ on input $\vec{t}$ and samplings $\vec{s}$.
- ψ holds after o call.

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

$$\{L := lis\}\#(\mathsf{h}(m,\mathsf{k}),\mathsf{r})\{L := m :: lis\}$$

is correct.

Adding pre and post conditions

$$\phi, \psi; C : \triangleright \vec{v}$$

Oracle rule

Definition (Oracle rule)

$$\textsc{Oracle}\ \frac{\phi, \psi; C : \triangleright \vec{t}, \vec{v} \qquad \{\psi\} c_o[\vec{s}, \vec{t}]\{\theta\}}{\phi, \theta; C, < \vec{s}, T_{g,\dots} >: \triangleright c_o[\vec{s}, \vec{t}], \vec{v}}$$

Induction and abstract semantics

Chalenges following:

Recursive terms : induction in the proof system

$\rightarrow$ Approximate pre and post condition

- Abstract representation of the game memory.
- Adapt the rule and/or the proof of soundness ragarding 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 ?

Gergei Bana and Hubert Comon-Lundh, A computationally complete symbolic attacker for equivalence properties, Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014 (Gail-Joon Ahn, Moti Yung, and Ninghui Li, eds.), ACM, 2014, pp. 609–620.

David Baelde, Stéphanie Delaune, Charlie Jacomme, Adrien Koutsos, and Solène Moreau, An interactive prover for protocol verification in the computational model, 42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021, IEEE, 2021, pp. 537–554.

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

(justine.sauvage@inria.fr)