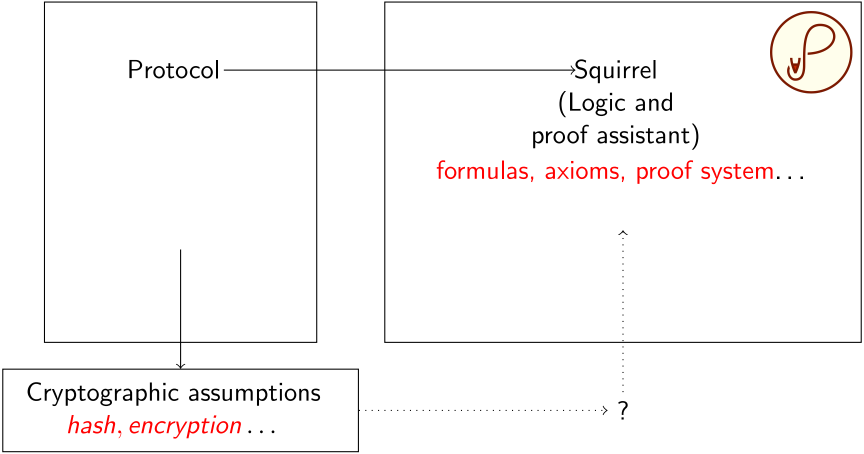
Cryptographic Reductions By Bi-Deduction

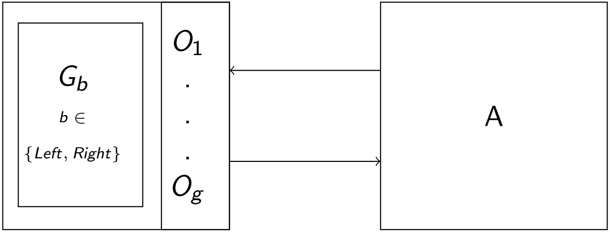
David Baelde, Adrien Koutsos, Justine Sauvage

Irisa, Inria Paris

Context: Squirrel and Cryptographic assumptions



Cryptographic assumptions



Definition (Indinstinguishability)

For any polynomial-time and randomized algorithm A,

$$|\Pr(A^{G_{Left}} = 1) - \Pr(A^{G_{Right}} = 1)|$$

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 "seems" random.

Example (PRF games)

Game G_{Left}

$Challenge(x)$:

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

Game G_{Righ}

$Challenge(x)$:

$r \xleftarrow{\$}$

return $\boxed{r}$

Example: PRF games

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

Example (PRF games)

Game G_{Left} *Init* :

$k \xleftarrow{\$}$

Challenge(x) :

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

Game G_{Righ} *Init* :

$k \xleftarrow{\$}$

Challenge(x) :

$r \xleftarrow{\$}$

return $\boxed{r}$

Example: PRF games

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

Example (PRF games)

Game G_{Left}	*Init* :	*Hash*(x) :	*Challenge*(x) :
	$k \xleftarrow{\$}$	
		return $h(x, k)$
			return $\boxed{h(x,k)}$

Game G_{Righ}	*Init* :	*Hash*(x) :	*Challenge*(x) :
	$k \xleftarrow{\$}$		$r \xleftarrow{\$}$
		return $h(x, k)$
			return $\boxed{r}$

Example: PRF games

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

Example (PRF games)

Game G_{Left}	$Init$:	$Hash(x)$:	$Challenge(x)$:
	$k \xleftarrow{\$}$	$log := x :: log$	$r \xleftarrow{\$}$
	$log := []$	$return\ h(x, k)$	$if\ x \notin log$
			$\quad log := x :: log$
			$\quad return\ \boxed{h(x,k)}$

Game G_{Righ}	$Init$:	$Hash(x)$:	$Challenge(x)$:
	$k \xleftarrow{\$}$	$log := x :: log$	$r \xleftarrow{\$}$
	$log := []$	$return\ h(x, k)$	$if\ x \notin log$
			$\quad log := x :: log$
			$\quad return\ \boxed{r}$

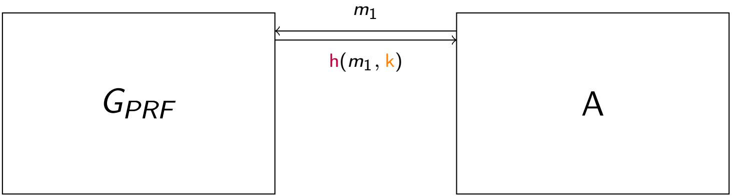
Example: PRF games

Example (PRF pair of games)

Game G_{PRF}

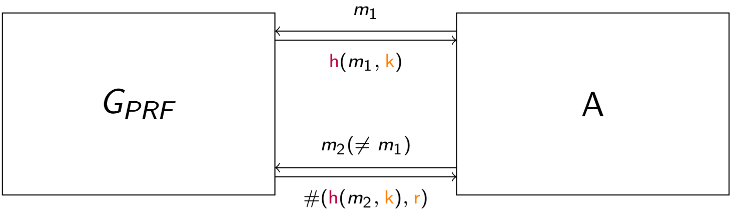
$Init$:	$Hash(x)$:	$Challenge(x)$:
$\mathsf{k} \xleftarrow{\$}$;	$L := x :: L$	$\mathsf{r} \xleftarrow{\$}$
$l := []$;	$\mathsf{h}(x, \mathsf{k})$	$if\ x \notin L$
		$L := x :: L$;
		$\boxed{\#(\mathsf{h}(\mathsf{x},\mathsf{k}),\mathsf{r})}$

Playing with PRF: sequence of messages



$$m_1, \mathsf{h}(m_1, \mathsf{k})$$

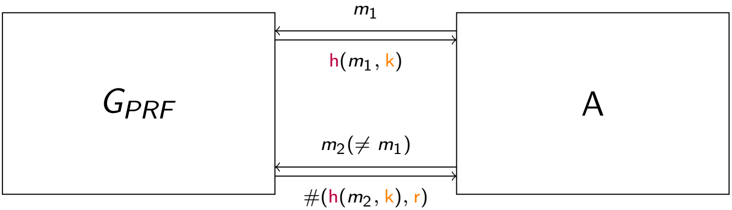
Playing with PRF: sequence of messages



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

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

Playing with PRF: sequence of messages



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

If there exists an adversary that can distinguish between this two sequences of messages, then the PRF assumption doesn't hold.

Terms and formulas

Definition (Terms)

Intuition: terms represent messages

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

Definition (Equivalence formulas)

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

PRF axiom schema

Question: is this formula a consequence of PRF assumption?

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

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

Definition (PRF axiom schema)

For all terms $\vec{t}$ verifying specific syntactic properties:

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

Problems

Problems with this method

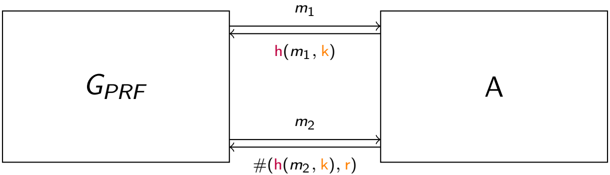
Ad hoc and manual work for each cryptographic axioms:

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

Changing point of view

Input:

$$m_1, h(m_1, k), m_2, h(m_1, k)$$



Question: does there exists such an A?

Contributions

- Theoritical framework to reduce equivalences to cryptographic assumption: extended notion of bi-deduction [BDKM22].
- Proof system for bi-deduction
- Application: implementation of SQUIRREL tactic crypto

Bi-deduction

Construction of then bi-deduction judgement: simulator

$$\triangleright \vec{v}$$

means that there exists an adversary S such that $S^G() = \vec{v}$.

Link between Bi-deduction and Equivalence

If an adversary can compute $\vec{v}$ then the formula $\mathsf{equiv}(\vec{v})$ holds.

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

What do we need ?

Goal: Framework for bi-deduction and associated proof system

Adversaries' capabilities ?

An adversary can:

- compute deterministic functions
- draw samplings
- interact with the game: oracles calls.

What do we need ?

Goal: Framework for bi-deduction and associated proof system

Adversaries' capabilities ?

An adversary can:

- compute deterministic functions (done)
- draw samplings
- interact with the game: oracles calls.

Definition

Function application: inference rule

$$\text{FA}\ \frac{\triangleright \vec{t} \quad adv(f)}{\triangleright f(\vec{t})}$$

S :

$\vec{x_t} := S_t()$

$y := f(\vec{x_t})$

return y

Samplings

Example

$$h(n, s), h(h(n, s), k)$$

simulator ← (n, s, n, s) | (k) → game

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

Definition (Tags)

$$Tag = \{\mathtt{T}_S, \mathtt{T}^{\mathsf{glob}}_{\mathsf{G},key}, \ldots\}$$

$$n \leftarrow \mathtt{T}_S$$

$$s \leftarrow \mathtt{T}_S$$

$$k \leftarrow \mathtt{T}^{\mathsf{glob}}_{\mathsf{G},key}$$

Extending bi-deduction with constraints

Adding sampling tagging

C records who sampled what: $C : \triangleright \vec{v}$

Definition (Adversary samplings)

$$\frac{C : \triangleright \vec{v}}{C, \langle \mathsf{n}, \mathtt{T}_S \rangle : \triangleright \mathsf{n}, \vec{v}}\ \textsc{Adv sampling}$$

S :

$\vec{y} := S_v()$

$x := \$$

return $x, \vec{y}$

$$\dfrac{\dfrac{\dfrac{}{\emptyset : \triangleright \emptyset}}{\langle \mathsf{s}, \mathtt{T}_S \rangle : \triangleright \mathsf{s}}\ \textsc{Adv sampling}}{\dfrac{\langle \mathsf{n}, \mathtt{T}_S \rangle, \langle \mathsf{s}, \mathtt{T}_S \rangle : \triangleright \mathsf{n}, \mathsf{s}}{\langle \mathsf{n}, \mathtt{T}_S \rangle, \langle \mathsf{s}, \mathtt{T}_S \rangle : \triangleright \mathsf{h}(\mathsf{n}, \mathsf{s})}\ \mathrm{FA}}\ \textsc{Adv sampling}$$

Oracle calls on example

Definition (Oracle rule: instantiated for hash oracle)

$$\textsc{Hash}\ \frac{C : \rhd m, \vec{v}}{C, \langle \mathsf{k}, \mathrm{T}^{\mathsf{glob}}_{\mathsf{G},key} \rangle : \rhd \mathsf{h}(m, \mathsf{k}), \vec{v}}$$

S :

$x_m, \vec{x_v} := S_{m,v}()$

$x := \mathcal{O}_{Hash}(x_m)$

return $x, \vec{x_v}$

Example

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

$$\dfrac{\dfrac{\vdots}{C : \rhd \mathsf{h}(\mathsf{n}, \mathsf{s})}}{C, \langle \mathsf{k}, \mathrm{T}^{\mathsf{glob}}_{\mathsf{G},key} \rangle : \rhd \mathsf{h}(\mathsf{h}(\mathsf{n}, \mathsf{s}), \mathsf{k})}\ \textsc{Hash}$$

Oracle rule: Challenge

Oracle rule instantiated for challenge

$$\frac{C : \rhd m, \vec{v}}{C, \langle \mathsf{r}, \mathtt{T}^{\mathsf{loc}}_{\mathsf{G}} \rangle, \langle \mathsf{k}, \mathtt{T}^{\mathsf{glob}}_{\mathsf{G}, key} \rangle : \rhd \#(\mathsf{h}(m, \mathsf{k}), \mathsf{r}), \vec{v}}$$

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

Oracle rule: Challenge

Oracle rule instantiated for challenge

$$\frac{\theta, \varphi, C : \rhd m, \vec{v} \qquad \{\varphi\}\mathcal{O}_{Challenge}(m)\{\psi\}}{\theta, \psi, C, \langle \mathsf{r}, \mathtt{T}^{\mathsf{loc}}_{\mathsf{G}} \rangle, \langle \mathsf{k}, \mathtt{T}^{\mathsf{glob}}_{\mathsf{G}, key} \rangle : \rhd \#(\mathsf{h}(m, \mathsf{k}), \mathsf{r}), \vec{v}}$$

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

$$log = [m] \qquad log = [m, m]?$$

Adding pre and post conditions

$$\varphi, \psi; C : \rhd \vec{v}$$

Consistency of taggings

C: registers randomness usage.

We want to ensure:

- Not two samples for one "role" (*e.g.*, $k \leftarrow \mathtt{T}^{\mathsf{glob}}_{\mathsf{G},key}, k' \leftarrow \mathtt{T}^{\mathsf{glob}}_{\mathsf{G},key}$)
- No sample owned by both the adversary and the oracles
- Freshness of local random sampling (*e.g.*, r)

Definition (validity of C)

$\mathsf{Valid}(C) \stackrel{def}{=} \mathsf{Uniqueness}(C) \wedge \mathsf{Ownership}(C) \wedge \mathsf{Freshness}(C)$

Coming back to the judgment

Bideduction judgmenent

For all conditions φ, ψ, constraints C, term $\vec{v}$:

$$\varphi, \psi; C : \triangleright \vec{v} \text{ iff}$$

if Valid(C) there exists a G-adversary S such that
for all memory μ satisfying φ, $(\!| S |\!)_\mu() = \mu'$

- μ' satisfy ψ
- $\mu'[res] = \vec{v}$

Semantics

Denotational semantic for adversaries and terms with early-random samplings.

$$(\!|S|\!)^{\eta}_{\mu} = X^{\text{left}}_S, X^{\text{right}}_S$$
$$[\![\vec{v}]\!]^{\eta} = X^{\text{left}}_v, X^{\text{right}}_v$$

Different randomness sources:

$$X^{\text{left/right}}_S : p \mapsto \cdots$$
$$X^{\text{left/right}}_v : \rho \mapsto \cdots$$

Equality of distribution: $\Pr_p(X^b_S = x) = \Pr_\rho(X^b_s = x)$

Coupling

Lifting through couplings:

$$
\begin{array}{ccc}
X_S^{\text{left}} & \approx & X_S^{\text{right}} \\
| & & | \\
=_d & & =_d \\
| & & | \\
X_v^{\text{left}} & \sim & X_v^{\text{right}}
\end{array}
$$

(proof: coupling built from C)

Implementation : game declarations

A language for games :

```
game PRF = {
  rnd key : kty;
  var lhash : mset = empty_set; var lchal : mset = empty_set;
  oracle ohash (x:message) : message = {
    lhash := add x lhash; return if mem x lchal then zero else h (x, key)
  }
  oracle challenge (x:message) : message = {
    rnd r : message;
    var old_lchal : mset = lchal;
    lchal := add x lchal;
    return if (mem x old_lchal || mem x lhash) then zero
           else diff(r, h (x, key))
  }
}
```

Implementation: tactic crypto

- Goal-directed proof-search procedure, based on the proof system.
- Langage to describe game memory.
- Ad hoc handling of induction for recursive terms.

Case studies

Protocol	Hypothesis	Properties
Basic Hash	EUF-MAC and PRF	Unlinkability
Hash Lock	PRF	Strong secrecy
Private Authentification	$\mathrm{CCA}_{\$}$	Anonymity
NSL proof step	CCA2	Strong secrecy

Conclusion

Contributions:

- Formal framework linking games, adversaries, and formulas
- Bi-deduction judgment to capture adversaries interacting with a game
- Proof system for this judgment
- Implementation: proof search automation and SQUIRREL tactics
- Validation through various case studies

Future work:

- Larger case study: FOO e-voting protocol
- Limitation in the theory.
- Improve tactic heuristics (time insentive invariant).

Conclusion

Contributions:

- Formal framework linking games, adversaries, and formulas
- Bi-deduction judgment to capture adversaries interacting with a game
- Proof system for this judgment
- Implementation: proof search automation and SQUIRREL tactics
- Validation through various case studies

Future work:

- Larger case study: FOO e-voting protocol
- Limitation in the theory.
- Improve tactic heuristics (time insentive invariant).

(Submitted work to CSF'24.)

References

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)