

# Computational Indistinguishability Logic

(Work in Progress)

G. Barthe<sup>1</sup>   M. Daubignard<sup>2</sup>   B. Kapron<sup>3</sup>   Y. Lakhnech<sup>2</sup>

<sup>1</sup>IMDEA Software, Madrid   <sup>2</sup>University of Grenoble, CNRS - VERIMAG

<sup>3</sup>University of Victoria

April 16, 2010

# Introduction

Our goal:

Enable *Computer-aided* verification of cryptographic constructions in the computational model.

Ideally:

- ▶ For protocols and cryptographic primitives.
- ▶ In the concrete security framework.
- ▶ As automated as possible.

# Two approaches

For security protocols:

- ▶ Indirect approach:
  - ▶ A proof in a symbolic model.
  - ▶ A computational soundness proof.
- ▶ Direct approach: [Cryptoverif, Ceticrypt]
  - ▶ Security definitions and models are in the computational interpretation.
  - ▶ Proofs are reductionist.

# Indirect approach - advantages and drawbacks

Main advantages are:

- ▶ the level of abstraction,
- ▶ automated tools,

Drawbacks:

- ▶ The model has to be repeatedly adapted to guarantee computational soundness - new primitive  $\rightarrow$  new soundness proof.
- ▶ The decidability results and verification procedures are to be revisited, extended and often are difficult to obtain.
- ▶ Unlikely to get reasonable concrete security bounds.

# Our approach

A direct approach based on a logic (a formal system) for indistinguishability and event probabilities.

- ▶ Capture the main (may be eventually all) reasoning schema applied in cryptographic proofs.
- ▶ Explicitly distinguish between information-theoretic reasoning (no security loss) and reduction-based reasoning.
- ▶ Language-independent, i.e., can be instantiated by different languages to describe the system.
- ▶ Amenable to automation and proof certification.

# Proof Tree for the Probabilistic Signature Scheme of Bellare and Rogaway

Cf. The proof by D. Pointcheval in Advanced Courses CRM Barcelona, Spain – February 2004.

$$\begin{array}{c}
 \text{(UpTo)} \frac{\text{PSS1} \equiv_{\varphi} \text{PSS2}}{\text{(NegDET)} \frac{\text{PSS1} : \text{EF-CMA}_2}{\text{(UR)} \frac{\text{PSS} : \text{EF-CMA}_2}{\text{PSS} : \text{EF-CMA}}}}{\text{PSS2} : \diamond(\neg\varphi)} \quad \frac{\text{OW}}{\text{PSS2} : \text{EF-CMA}_2} \text{NegSUB} \quad \frac{\text{FAIL}}{\mathcal{O}_H} \\
 \text{NegSUB} \frac{\text{PSS} : \text{EF-CMA}_1}{\text{PSS} : \text{EF-CMA}}
 \end{array}$$

# CIL: Computational Indistinguishability Logic

Our starting point is a logic proposed by R. Impagliazzo and B. Kapron for indistinguishability [FOCS'03]. It is extended in several ways:

- ▶ Oracles: reasoning about oracles is crucial for many cryptographic constructions and protocols.
- ▶ Rules to reason about probabilities of events.
- ▶ A direct interpretation and soundness proof.
- ▶ Concrete security proofs.

It is also related to Yu Zhang's work based on Hofmann's Characterization of poly-time computations.

# Oracle Systems

A system is described by:

- ▶ A (countable) set of memories with an initial memory  $\bar{m}$ .
- ▶ A set of oracles, where each oracle is given by a name and an implementation:

$$O_o : \text{In}(o) \times M_o \rightarrow \mathcal{D}(\text{Out}(o) \times M_o)$$

- ▶ Distinguished oracles  $O_{\text{init}}$  for initialization and  $O_{\text{final}}$  for finalization, such that  $\text{In}(O_{\text{init}}) = \text{Out}(O_{\text{final}}) = \mathbf{1}$ .

# Oracle System Adversaries

An adversary  $\mathbb{A}$  (for an oracle system  $\mathbb{O}$ ) is given by:

- ▶ a (countable) set  $M_a$  of adversary memories and an initial memory  $\bar{m}_a$ ;
- ▶ a transition function, and an update function:

$$A \quad : \quad M_a \rightarrow \mathcal{D}(\text{Que} \times M_a)$$

$$A_{\downarrow} \quad : \quad M_a \times \text{QuAn} \rightarrow \mathcal{D}(M_a)$$

Que: the set of queries, a query is an oracle name and an input; and Ans is the union of all  $\text{Out}(o)$ .

# Semantics

The composition  $\mathbb{A} \mid \mathbb{O}$  of adversary  $\mathbb{A}$  and oracle system  $\mathbb{O}$  is a *probabilistic transition system*:

$$\text{step}_{\mathbb{A} \mid \mathbb{O}}(m_a, m_o) \stackrel{\text{def}}{=} \begin{array}{l} \text{let } ((o, q), m'_a) \leftarrow \mathbb{A}(m_a) \text{ in} \\ \text{let } (a, m'_o) \leftarrow \mathbb{O}_o(q, m_o) \text{ in} \\ \text{let } m''_a \leftarrow \mathbb{A}_\downarrow(m'_a, (o, q, a)) \text{ in} \\ \text{return } ((o, q, a), (m''_a, m'_o)) \end{array}$$

Only adversaries with bounded running time.

# Executions and Traces

- ▶ Executions:

$$(m_a^0, m_o^0) \xrightarrow{\langle o_1, q_1, a_1 \rangle} (m_a^1, m_o^1) \xrightarrow{\langle o_2, q_2, a_2 \rangle} \dots \xrightarrow{\langle o_k, q_k, a_k \rangle} (m_a^k, m_o^k)$$

such that  $o_1 = \mathcal{O}_{\text{init}}$  and  $o_k = \mathcal{O}_{\text{final}}$ .

- ▶ Traces are obtained by deleting the adversary's state:

$$m_o^0 \xrightarrow{\langle o_1, q_1, a_1 \rangle} m_o^1 \xrightarrow{\langle o_2, q_2, a_2 \rangle} \dots \xrightarrow{\langle o_k, q_k, a_k \rangle} m_o^k$$

$$\text{PR}[\mathbb{A} \mid \mathbb{O} = \eta] =$$

$$\prod_{i=0}^{k-1} \text{PR}[\text{step}_{\mathbb{A}|\mathbb{O}}(m_a^i, m_o^i) = (\langle o_i, q_i, a_i \rangle, (m_a^{i+1}, m_o^{i+1}))]$$

$$\text{PR}[\mathbb{A}|\mathbb{O} : \tau] = \text{PR}[\mathbb{A}|\mathbb{O} : \mathcal{T}^{-1}(\tau)]$$

# Statements of the logic

- ▶ Indistinguishability:  $\mathbb{O} \sim_\epsilon \mathbb{O}'$

$$\models \mathbb{O} \sim_\epsilon \mathbb{O}'$$

if for any adversary,

$$|\text{PR}[\mathbb{A} \mid \mathbb{O} : r = 1] - \text{PR}[\mathbb{A} \mid \mathbb{O}' : r = 1]| \leq \epsilon$$

- ▶ Event probability: An event  $E$  is a mapping from traces to Bool.  $\mathbb{O} :_\epsilon E$ :

$$\models \mathbb{O} :_\epsilon E$$

if for any adversary,

$$\text{PR}[\mathbb{A} \mid \mathbb{O} : E] \leq \epsilon$$

# CIL - The formal system

Reasoning in CIL consists in deriving a statement from a set of hypotheses that capture cryptographic assumptions by applying derivation rules.

The rules may have CIL statements as premisses or external statements:

- ▶ Valid logic formulae, e.g.  $A \Rightarrow A \vee B$
- ▶ Hoare triples for simple probabilistic programs.
- ▶ Equivalence of simple probabilistic programs.

There are two types of rules:

- ▶ Rules that do not rely on reduction arguments and do not cause loss in security.
- ▶ Rules that rely (their soundness is by) reduction and cause loss in security properties.

# Derivation Rules

The first set of rules supports equational and external reasoning:

$$\frac{}{\mathbb{O} \sim_0 \mathbb{O}} \quad \frac{\mathbb{O} \sim_\epsilon \mathbb{O}'}{\mathbb{O}' \sim_\epsilon \mathbb{O}} \quad \frac{\mathbb{O} \sim_\epsilon \mathbb{O}' \quad \mathbb{O}' \sim_{\epsilon'} \mathbb{O}''}{\mathbb{O} \sim_{\epsilon+\epsilon'} \mathbb{O}''}$$

$$\frac{\mathbb{O} :_{\epsilon_i} E_i \ (i \in I) \quad E \Rightarrow \bigvee_{i \in I} E_i}{\mathbb{O} :_{\sum_{i \in I} \epsilon_i} E} \text{ UR}$$

$$\frac{\forall o, \{\text{true}\} \mathbb{O} \{\text{PR}[\varphi] \leq \epsilon_o\}}{\mathbb{O} :_\epsilon \diamond \varphi} \text{ FAIL}$$

For  $\varphi : \text{QuAn} \times \text{M} \times \text{M} \rightarrow \text{Bool}$  and  $\epsilon = \sum_{o \in N_o} k_o \epsilon_o$ .

# Context

An  $\mathbb{O}$ -context  $\mathbb{C}$  is given by:

- ▶ sets  $M_c$  of context memories, an initial memory  $\bar{m}_c$  and  $N_c$  of procedures;
- ▶ for every  $c \in N_c$ , a query domain  $In_c(c)$ , an answer domain  $Out_c(c)$ , and forward and backwards implementations

$$C_c^{\rightarrow} : In_c(c) \times M_c \rightarrow \mathcal{D}(Que \times M_c)$$

$$C_c^{\leftarrow} : In_c(c) \times QuAn \times M_c \rightarrow \mathcal{D}(Out_c(c) \times M_c)$$

- ▶ distinguished initial and finalization procedures  $c_i$  and  $c_f$ .

# Context Application

The application of an  $\mathbb{O}$ -context  $\mathbb{C}$  to  $\mathbb{O}$  defines an oracle system  $\mathbb{C}[\mathbb{O}]$  such that:

- ▶ the set of memories is  $M_c \times M_o$ , and the initial memory is  $(\bar{m}_c, \bar{m}_o)$ ;
- ▶ the oracles are the procedures of  $\mathbb{C}$ , and their query and answer domains are given by  $\mathbb{C}$ . The initialization and finalization oracles are the initialization and finalization procedures of  $\mathbb{C}$ ;
- ▶ the implementation of an oracle  $c$  is:

```
 $\lambda q_c, (m_c, m_o).$   
  let  $((o, q_o), m'_c) \leftarrow C_c^{\rightarrow}(q_c, m_c)$  in  
  let  $(a_o, m'_o) \leftarrow O_o(q_o, m_o)$  in  
  let  $(a_c, m''_c) \leftarrow C_c^{\leftarrow}(q_c, (o, q_o, a_o), m'_c)$  in  
  return  $(a_c, (m''_c, m'_o))$ 
```

# Composition of Adversary with Context

- ▶ the set of memories is  $M_c \times M_a \times \text{Que}_c$ , and the initial memory is  $(\bar{m}_c, \bar{m}_a, -)$ ;
- ▶ the transition function is:

$$\begin{aligned} &\lambda(m_c, m_a, -). \\ &\quad \text{let } ((c, q_c), m'_a) \leftarrow A(m_a) \text{ in} \\ &\quad \text{let } ((o, q), m'_c) \leftarrow C_c^{\rightarrow}(q_c, m_c) \text{ in} \\ &\quad \text{return } ((o, q), (m'_c, m'_a, (o, q))) \end{aligned}$$

- ▶ the update function is:

$$\begin{aligned} &\lambda((m_c, m_a, (o_c, q_c)), (o_o, q_o, a_o)). \\ &\quad \text{let } (a_c, m'_c) \leftarrow C_c^{\leftarrow}(q_c, (o_o, q_o, a_o), m_c) \text{ in} \\ &\quad (m'_c, A_{\downarrow}(m_a, (o_c, q_c, a_c)), -) \end{aligned}$$

# SUB and NegSUB Rules

$$\frac{\mathbb{O} \sim_{\epsilon} \mathbb{O}'}{\mathbb{C}[\mathbb{O}] \sim_{\epsilon} \mathbb{C}[\mathbb{O}']} \text{ SUB} \qquad \frac{\mathbb{O} :_{\epsilon} E \circ \mathbb{C}}{\mathbb{C}[\mathbb{O}] :_{\epsilon} E} \text{ NegSUB}$$

The  $\mathbb{O}$ -event  $E \circ \mathbb{C}$  is defined as:

$$\lambda\tau. \exists\tau^{\text{mix}}. \pi_{\mathbb{O}}(\tau^{\text{mix}}) = \tau \wedge E(\pi_{\mathbb{C}[\mathbb{O}]}(\tau^{\text{mix}}))$$

## Proposition

Let  $\mathbb{O}, \mathbb{O}'$  be compatible oracle systems and  $\mathbb{C}$  be an  $\mathbb{O}$ -context.

- ▶ If  $\mathbb{C}$  is an indistinguishability context and  $\models \mathbb{O} \sim_{\epsilon} \mathbb{O}'$  then  $\models \mathbb{C}[\mathbb{O}] \sim_{\epsilon} \mathbb{C}[\mathbb{O}']$ .
- ▶ For every  $\mathbb{C}[\mathbb{O}]$ -event  $E$ , if  $\models \mathbb{O} :_{\epsilon} E \circ \mathbb{C}$  then  $\models \mathbb{C}[\mathbb{O}] :_{\epsilon} E$ .

# Equivalence of Oracle Systems

- ▶ Often, one needs to replace an oracle system by (an almost) equivalent one.
- ▶ Two probabilistic transition systems are equivalent, if they associate the same probability to each event.
- ▶ Two oracle systems  $\mathbb{O}$  and  $\mathbb{O}'$  are called *equivalent*, if for any adversary  $\mathbb{A}$ , the underlying probabilistic transition systems  $\mathbb{A} \mid \mathbb{O}$  and  $\mathbb{A} \mid \mathbb{O}'$  are equivalent.
- ▶ Bisimulations provide a powerful method for proving equivalence of probabilistic transition systems.
- ▶ We lift bisimulation-based reasoning to oracle systems. More specifically, we define a collection of conditions that ensure the existence of a bisimulation between  $\mathbb{A} \mid \mathbb{O}$  and  $\mathbb{A} \mid \mathbb{O}'$ , for every  $\mathbb{A}$ , and hence, the equivalence of  $\mathbb{O}$  and  $\mathbb{O}'$ .

# Almost Equivalence of Oracle Systems

- ▶ Almost equivalence of oracle systems: this is when equivalence holds as long as the adversary does not do something bad.
- ▶ The bad is specified by:

$$\varphi \subseteq \text{Que} \times \text{Ans} \times (M_o + M'_o) \times (M_o + M'_o)$$

- ▶ Equivalence is ensured by: existence of an equivalence relation  $R \subseteq (M_o + M'_o) \times (M_o + M'_o)$  on memories that satisfies
  - ▶ *Initialization*:  $\bar{m}_o R \bar{m}_a$
  - ▶ *stability*: if  $m'_1 R m'_2$  then

$$\varphi((o, q, a), m_1, m'_1) \Leftrightarrow \varphi((o, q, a), m_2, m'_2)$$

- ▶ *compatibility*: if  $\varphi((o, q, a), m_1, m'_1)$ , then

$$\text{PR}[\hat{O}_o(q, m_1) \in (a, C)] = \text{PR}[\hat{O}_o(q, m_2) \in (a, C)]$$

where  $C$  is the equivalence class of  $m'_1$  under  $R$ .

Notation:  $\mathbb{O} \equiv_{R, \varphi} \mathbb{O}'$

## Proposition

If  $\mathbb{O} \equiv_{R, \varphi} \mathbb{O}'$  then  $\mathbb{A} \mid \mathbb{O} \sim_{-\varphi} \mathbb{A} \mid \mathbb{O}'$ .

# Oracle Rules

$$\frac{\mathbb{O} :_{\epsilon} E \wedge \Box(\varphi) \quad \mathbb{O} \equiv_{R,\varphi} \mathbb{O}' \quad E \in R^E}{\mathbb{O}' :_{\epsilon} E \wedge \Box(\varphi)} \text{NegOR}\forall$$

$$\frac{\mathbb{O} :_{\epsilon} \Diamond\neg\varphi \quad \mathbb{O} \equiv_{R,\varphi} \mathbb{O}'}{\mathbb{O}' :_{\epsilon} \Diamond\neg\varphi} \text{Neg}\Diamond$$

$$\frac{\mathbb{O} :_{\epsilon} \Diamond\neg\varphi \quad \mathbb{O} \equiv_{R,\varphi} \mathbb{O}'}{\mathbb{O} \sim_{\epsilon} \mathbb{O}'} \text{OR}$$

$$\frac{\mathbb{O} :_{\epsilon} (E \wedge \Box(\varphi))\mathcal{U}\neg\varphi \quad \mathbb{O} \equiv_{R,\varphi} \mathbb{O}' \quad E \in R^E}{\mathbb{O}' :_{\epsilon} (E \wedge \Box(\varphi))\mathcal{U}\neg\varphi} \text{NegOR}\exists$$

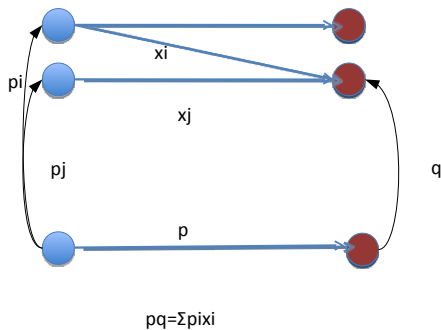
## Proposition

For all oracle systems  $\mathbb{O}$  and  $\mathbb{O}'$  and testable  $\varphi$  such that  $\mathbb{O} \equiv_{R,\varphi} \mathbb{O}'$ , every  $R$ -compatible events  $E$ , and adversary  $\mathbb{A}$ :

- ▶  $\text{PR}(\mathbb{A} \mid \mathbb{O} : E \wedge \Box(\varphi)) = \text{PR}(\mathbb{A} \mid \mathbb{O}' : E \wedge \Box(\varphi))$
- ▶  $\text{PR}(\mathbb{A} \mid \mathbb{O} : (E \wedge \Box(\varphi))\mathcal{U}\neg\varphi) = \text{PR}(\mathbb{A} \mid \mathbb{O} : (E \wedge \Box(\varphi))\mathcal{U}\neg\varphi)$

## Early-late sampling and Determinization

Bisimulation is in many cases too strong, in particular for moving samples earlier (later) in the execution.



## Rules based on Determinization

$$\frac{\mathbb{O}' :_{\epsilon} E \circ \pi_{M_o}}{\mathbb{O} :_{\epsilon} E} \text{NegDET}$$

$$\frac{\mathbb{O} \leq_{\det(\gamma)} \mathbb{O}'}{\mathbb{O} \sim_0 \mathbb{O}'} \text{DET}$$

# Summary of the Rules

(REF), (SYM), (TRAN), (UR), (FAIL)

$$\frac{\mathbb{O} \sim_\epsilon \mathbb{O}'}{\mathbb{C}[\mathbb{O}] \sim_\epsilon \mathbb{C}[\mathbb{O}']} \text{ SUB} \qquad \frac{\mathbb{O} :_\epsilon E \circ \mathbb{C}}{\mathbb{C}[\mathbb{O}] :_\epsilon E} \text{ NegSUB}$$

$$\frac{\mathbb{O}' :_\epsilon E \circ \pi_{M_o}}{\mathbb{O} :_\epsilon E} \text{ NegDET} \qquad \frac{\mathbb{O} \leq_{\det(\gamma)} \mathbb{O}'}{\mathbb{O} \sim_0 \mathbb{O}'} \text{ DET}$$

$$\frac{\mathbb{O} :_\epsilon \diamond \neg \varphi \quad \mathbb{O} \equiv_{R,\varphi} \mathbb{O}'}{\mathbb{O}' :_\epsilon \diamond \neg \varphi} \text{ Neg}\diamond$$

$$\frac{\mathbb{O} :_\epsilon E \wedge \square(\varphi) \quad \mathbb{O} \equiv_{R,\varphi} \mathbb{O}' \quad E \in R^E}{\mathbb{O}' :_\epsilon E \wedge \square(\varphi)} \text{ NegOR}\forall$$

$$\frac{\mathbb{O} :_\epsilon \diamond \neg \varphi \quad \mathbb{O} \equiv_{R,\varphi} \mathbb{O}'}{\mathbb{O} \sim_\epsilon \mathbb{O}'} \text{ OR}$$

$$\frac{\mathbb{O} :_\epsilon (E \wedge \square(\varphi))\mathcal{U}\neg\varphi \quad \mathbb{O} \equiv_{R,\varphi} \mathbb{O}' \quad E \in R^E \quad \varphi \text{ testable}}{\mathbb{O}' :_\epsilon (E \wedge \square(\varphi))\mathcal{U}\neg\varphi} \text{ NegOR}\exists$$

# Timing Aspects - Discussion

Let  $\mathbb{O}$  be an oracle system with oracle names  $o_1, \dots, o_n$ . Then, in  $\mathbb{O} :_{\epsilon} E$ ,  $\epsilon$  is a function:

$$\epsilon : \mathbb{N}^{n+1} \rightarrow [0, 1]$$

and its interpretation is:

For every adversary  $\mathbb{A}$  that calls  $o_i$  at most  $q_i$ -times and terminates in time  $t + \sum_i q_i$ ,

$$\text{PR}[\mathbb{A} \mid \mathbb{O} : E] \leq \epsilon(t, q_1, \dots, q_n).$$

- ▶ “ $\mathbb{A}$  calls  $o_i$  at most  $q_i$ -times...” means

$$\text{PR}[\mathbb{A} \mid \mathbb{O} = \eta : \#\langle o_i, -, - \rangle \leq q_i] = 1$$

## NegSUB revisited

$$\frac{\mathbb{O} :_{\epsilon} E \circ \mathbb{C}}{\mathbb{C}[\mathbb{O}] :_{\epsilon_c} E} \text{NegSUB}$$

Assume  $\mathbb{C}$  has procedures  $c_1, \dots, c_m$ . Define

$\alpha : \{o_1, \dots, o_n\} \times \{c_1, \dots, c_m\} \rightarrow \{0, 1\}$  such that  $\alpha(o_i, c_j) = 1$  iff  $c_j$  may call  $o_i$ , i.e.

$$\exists m_c \in M_c \exists q \in \text{In}_c(c_j). \sum_{m'_c \in M_c} \text{PR}[\mathbb{C}^{\rightarrow}(c_j)(m_c, q) = (o_i, m'_c)] > 0$$

Then,

$$\epsilon_c(t, q'_1, \dots, q'_m) = \epsilon(t + T_C, \sum_{j=1}^m \alpha(o_1, c_j) q'_j, \dots, \sum_{j=1}^m \alpha(o_n, c_j) q'_j)$$

where  $T_C = \sum_{j=1}^m q'_j (T_{\mathbb{C}^{\rightarrow}(c_j)} + T_{\mathbb{C}^{\leftarrow}(c_j)})$ .

# Probabilistic Signature Scheme

Bellare&Rogaway'96.

- ▶ A signature scheme based on one-way trapdoor permutations.
- ▶ Part of the PKCS standard.

Informal description:

- ▶ The scheme uses 3 hash functions in the random oracle model.
- ▶ The signature of  $M$  is

$$f^{-1}(H(M|r)|G(H(M|r)) \oplus r|F(H(M|r)))$$

where  $r$  is sampled in  $\{0, 1\}^{k_1}$ ,  $H : \{0, 1\}^{k_m+k_1} \rightarrow \{0, 1\}^{k_2}$ ,  
 $G : \{0, 1\}^{k_2} \rightarrow \{0, 1\}^{k_1}$  and  $F : \{0, 1\}^{k_2} \rightarrow \{0, 1\}^{k_0}$ .

- ▶ To verify  $(M, x)$ , let  $y = f(x)$ ,  $w_1|w_2|w_3 = y$ ,  
 $r = G(w_1) \oplus w_2$  and verify

$$w_1 = H(M|r) \wedge w_3 = F(w_1)$$

## PSS as Oracle system

- ▶  $\mathcal{O}_{\text{init}}$  samples  $(f, f^{-1})$  and returns  $(f, (f, f^{-1}, L_H : [], L_F : [], L_G : []))$ .



$$\begin{aligned} \mathcal{O}_H : \lambda x, m. \quad & (x, r) \in L_H \rightarrow \text{return } (r, m) \\ & \top \rightarrow \text{let } r \leftarrow \{0, 1\}^{k_2} \text{ in} \\ & \quad \text{return } (r, m[L_H := L_H :: (x, r)]) \end{aligned}$$

- ▶ Similarly for  $\mathcal{O}_F$  and  $\mathcal{O}_G$ .



$$\begin{aligned} \mathcal{O}_S : \lambda M, m. \quad & \text{let } r \leftarrow \{0, 1\}^{k_1} \text{ in} \\ & \text{let } (w_1, m_1) \leftarrow \mathcal{O}_H(M|r, m) \text{ in} \\ & \text{let } (w_2, m_2) \leftarrow \mathcal{O}_G(w_1, m_1) \text{ in} \\ & \text{let } (w_3, m_3) \leftarrow \mathcal{O}_F(w_1, m_2) \text{ in} \\ & \text{return } (f^{-1}(w_1|w_2 \oplus r|w_3), m_3) \end{aligned}$$

# The Security Property

An adversary breaks PSS, if she is able to provide a message and a signature of that message that has not been produced by  $\mathcal{O}_S$ .

$$\text{EF-CMA} = \diamond [(\langle \mathcal{O}_{\text{final}}, R_1 | R_2, \mathbf{1} \rangle, m_1, m_2) \wedge V(R_1, R_2, m_2)] \wedge \\ \square \neg (\langle \mathcal{O}_S, R_1, R_2 \rangle, -, -)$$

$V : \lambda M, x, m. \text{ let } w_1 | w_2 | w_3 = f(x) \text{ in}$   
 $\text{ let } (w_1, g) \in m.L_G \text{ in}$   
 $\text{ let } r = w_2 \oplus g \text{ in}$   
 $\text{ return } (M | r, w_1) \in m.L_H \wedge (w_1, w_3) \in m.L_F$

Notice that  $r$  is uniquely defined. Denote it  $r(R_2)$ .

# One-way Permutation

- ▶ Memory:  $(y, pk, sk, b)$ ,  $y \in \{0, 1\}^k$ ,  $pk, sk$  public and secret keys of the One-way permutation. Initial memory  $b = \text{true}$ .
- ▶  $\mathcal{O}_{\text{init}}$  :

$\lambda x, m \quad b \rightarrow \text{let } y \leftarrow \{0, 1\}^k \text{ in}$   
 $\text{let } (pk, sk) \leftarrow \mathcal{K} \text{ in}$   
 $\text{return } ((pk, y), (y, pk, sk, \text{false}))$   
 $\top \rightarrow ((pk, y), (y, pk, sk, \text{false}))$

- ▶  $\mathcal{O}_{\text{final}}$  :  $\lambda x, m. \text{return } (\mathbf{1}, m)$



$V_{\text{ow}} : \lambda \tau, m. \text{let } \tau = \tau' \xrightarrow{\langle \mathcal{O}_{\text{final}}, x, \mathbf{1} \rangle} m \text{ in}$   
 $f(m.pk, x) = m.y$



$\mathbb{O}_{\text{ow}} : \epsilon_{\text{ow}}(t) V_{\text{ow}}$

## Two cases

$$\text{EF-CMA} = \text{EF-CMA}_1 \vee \text{EF-CMA}_2$$

where

$$\text{EF-CMA}_1 = \text{EF-CMA}_1 \wedge R_1 | r(R_2) \notin L_H$$

$$\text{EF-CMA}_2 = \text{EF-CMA}_1 \wedge R_1 | r(R_2) \in L_H$$

$$(\text{UR}) \frac{\text{PSS } :_{2-k_2} \text{EF-CMA}_1 \quad \text{PSS } :_{\epsilon-2-k_2} \text{EF-CMA}_2}{\text{PSS } :_{\epsilon} \text{EF-CMA}}$$

# PSS :<sub>2-k<sub>2</sub></sub> EF-CMA<sub>1</sub>

$$\mathbb{O} = \mathcal{O}_{\text{init}}, \mathcal{O}_H, \mathcal{O}_{\text{final}}$$

$$\mathcal{O}_{\mathcal{O}_{\text{init}}} : \lambda x, m. (\mathbf{1}, m[L_H := []])$$

$$\mathcal{O}_{\mathcal{O}_H} : \lambda x, m. (x, r) \in L_H \rightarrow \text{return } (r, m)$$

$$\top \rightarrow \text{let } r \leftarrow \{0, 1\}^{k_2} \text{ in}$$

$$\text{return } (r, m[L_H := L_H :: (x, r)])$$

$$\mathcal{O}_{\mathcal{O}_{\text{final}}} = \mathcal{O}_{\mathcal{O}_H}$$

$$\text{GuessHash} = \diamond[(\langle \mathcal{O}_{\text{final}}, R_1 | R_2, \mathbf{1} \rangle, m_1, m_2) \wedge m_2.L_H(R_1) = R_2] \wedge R_1 \notin m_1.L_H$$

$$\frac{\text{FAIL}}{\mathbb{O}_H :_{2-k_2} \text{GuessHash}} \\ \frac{}{\mathbb{C}[\mathbb{O}_H] :_{2-k_2} \text{EF-CMA}_1}$$

$\mathbb{C}$ -memory:  $(f, f^{-1}, L_F : [], L_G : [])$

# PSS1

Compute  $F(h)$  and  $G(h)$  as soon as  $h$  is produced by  $H$ . To keep the system consistent with PSS, we need some book keeping using lists  $L'_F$  and  $L'_G$ .

$$\begin{aligned} \mathcal{O}_{\mathcal{O}_G} &:: \lambda x, m. \quad (x, w_2) \in L_G \rightarrow (w_2, m) \\ &\quad (x, w_2) \in L'_G \rightarrow (w_2, m[L'_G := L'_G \setminus (x, w_2), \\ &\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad L_G := L_G :: (x, w_2)]) \\ &\quad \top \rightarrow \text{let } w_2 \leftarrow \{0, 1\}^{k_1} \text{ in} \\ &\quad \quad (w_2, m[L_G := L_G :: (x, w_2)]) \\ &\quad \text{return } (w_2, m_3) \\ \mathcal{O}_{\mathcal{O}_H} &: \lambda x, m. \quad \text{let } (w_1, m_1) \leftarrow \mathcal{O}_H(x, m) \text{ in} \\ &\quad \text{let } (w_2, m_2) \leftarrow \mathcal{O}'_{\mathcal{O}_G}(w_1, m_1) \text{ in} \\ &\quad \text{let } (w_3, m_3) \leftarrow \mathcal{O}'_{\mathcal{O}_F}(w_1, m_2) \text{ in} \\ &\quad \text{return } (w_1, m_3) \end{aligned}$$

$$\begin{aligned}
 \mathcal{O}'_{\mathcal{O}_G} &:: \lambda x, m. (x, w_2) \in L_G, L'_G \rightarrow (w_2, m) \\
 &\top \rightarrow \text{let } w_2 \leftarrow \{0, 1\}^{k_1} \text{ in} \\
 &\quad (w_2, m[L'_G := L'_G :: (x, w_2)]) \\
 &\text{return } (w_2, m_3)
 \end{aligned}$$

The implementation of  $\mathcal{O}_S$  is as before.

# Equivalence of PSS and PSS1

Equivalence of PSS and PSS1 is shown using determinization.

- ▶ Consider a memory  $m$  of PSS1. Let  
 $X(m) = (\text{rng } m.L_H) \setminus (\text{dom } m.L'_G)$  and  
 $Y(m) = (\text{rng } m.L_H) \setminus (\text{dom } m.L'_F)$ .
- ▶ Define  $\gamma(m)$  as the uniform distribution over all pairs  $(L'_F, L'_G)$  such that  $\text{dom } L'_F = X$  and  $\text{dom } L'_G = Y$  and typing is respected.
- ▶ Then,

$$\text{PSS} \leq_{\text{det}, \gamma} \text{PSS1}$$

Memory: as before +  $y, L_u$ .

Initialization:  $y$  is sampled randomly and  $L_u := []$ .

$O_{\mathcal{O}_H}$  computes  $F(w_1)$  and  $G(w_1)$  even if they have been already computed.

$$\begin{aligned}
 O_{\mathcal{O}_H} : & \lambda x, m. (x, w_1) \in L_H \rightarrow \text{return } (w_1, m) \\
 & \top \rightarrow \text{let } u \leftarrow \{0, 1\}^k \text{ in} \\
 & \quad \text{let } w_1 | w_2 | w_3 = f(u) \otimes y \text{ in} \\
 & \quad \text{let } x = x' | r \text{ in} \\
 & \quad \quad \text{return } (w_1, m[L_H := L_H :: (x, w_1), \\
 & \quad \quad \quad L'_F := L'_F :: (w_1, w_3), \\
 & \quad \quad \quad L'_G := L'_G :: (w_1, w_2 \oplus r) \\
 & \quad \quad \quad L_u := L_u :: (x', r', u, w_1)])
 \end{aligned}$$

$\mathcal{O}_S : \lambda M, m.$  let  $r \leftarrow \{0, 1\}^{k_1}$  in  
let  $u \leftarrow \{0, 1\}^k$  in  
let  $w_1|w_2|w_3 = f(u)$  in  
return  $(u, m[L_H := L_H :: (M|r, w_1),$   
 $L_F := L_F :: (w_1, w_3),$   
 $L_G := L_G :: (w_1, w_2 \oplus r)])$

# Equivalence of PSS1 and PSS2

$$\text{PSS1} \equiv_{R, \varphi} \text{PSS2}$$

where

$R m m'$  iff  $m$  and  $m'$  agree on  $L_H, L_F, L'_F, L_G, L'_G, f, f^{-1}$

and

$$\begin{aligned} \varphi((o, q, a), m_1, m_2) = & \quad o \neq \mathcal{O}_H \wedge o \neq \mathcal{O}_S \rightarrow \text{return true} \\ & \quad o = \mathcal{O}_H \rightarrow \\ & \quad \quad \text{return } (q \in m_1.L_H) \vee \\ & \quad \quad \quad a \notin m_1.L_F, m_1.L'_F, m_1.L_G, m_1.L'_G \\ & \quad o = \mathcal{O}_S \rightarrow \\ & \quad \quad \text{let } w_1 = f(a)[k_2] \text{ in} \\ & \quad \quad \text{let } w'_2 = f(a)[k_2 + 1, k_1] \text{ in} \\ & \quad \quad \quad \text{return } w_1 \notin (m_1.L_F, m_1.L'_F, m_1.L_G, m_1.L'_G) \\ & \quad \quad \quad \wedge \forall (w_1, g) \in m_2.L_G. (M|w'_2 \oplus w_2) \notin m_1.L_H \end{aligned}$$

Assuming the invariant that the length of the list  $L_F, L'_F, L_G, L'_G$ , is bounded by  $q_F + q_G + q_H + q_S$ ,

$$\text{PR}[\mathcal{O}; \neg\varphi] \leq 2^{-k_2}(q_F + q_G + q_H + q_S) + (q_H + q_S)2^{-k_1}$$

$$(\text{UpTo}) \frac{\text{PSS1} \equiv_{\varphi} \text{PSS2} \quad \frac{(\text{FAIL})}{\text{PSS2} :_{\epsilon_2} \diamond(\neg\varphi)} \quad \text{PSS2} :_{\epsilon_3} \text{EF-CMA}_2}{\text{PSS1} :_{\epsilon_2 + \epsilon_3} \text{EF-CMA}_2}$$

$$\epsilon_2 = (q_H + q_S)(2^{-k_2}(q_F + q_G + q_H + q_S) + (q_H + q_S)2^{-k_1})$$

PSS2 : $\epsilon_3$  EF-CMA<sub>2</sub> by reduction to the one-way permutation hypothesis.

$$\epsilon_3 = \text{succ}^{\text{ow}}(t + (q_H + q_S)T_f + \text{overhead})$$

In summary,

$$\epsilon = \text{succ}^{\text{ow}}(t + (q_H + q_S)T_f + \text{overhead}) + (q_H + q_S)(2^{-k_2}(q_F + q_G + q_H + q_S) + (q_H + q_S)2^{-k_1}) + 2^{-k_2}$$

# Conclusions

Examples treated with CIL:

- ▶ ElGamal, hashed ElGamal.
- ▶ FDH, PSS.
- ▶ OAEP: IND-CPA, IND-CCA.

Cross fertilization with certicrypt and possibly other tools.

- ▶ IND-CCA/OAEP: The insight gained from the proof in CIL enabled the proof in certicrypt (submitted).
- ▶ Certicrypt can be used to discharge some of the external reasoning of CIL.

Formalization in a theorem prover:

- ▶ Formalization in Coq almost complete (P. Corbineau and help from Ch. Paulin).
- ▶ Our main goal concerning this is the soundness of the logic, not necessarily having Coq-certified proofs for cryptosystems.

## Further work

- ▶ A thorough treatment for the timing aspects (do not care about poly-time, aim at concrete security). Formalization in Coq is under development.
- ▶ External reasoning:
  - ▶ Reasoning about events:  $E \Rightarrow E'$ .
  - ▶  $\{\text{true}\} \mathbb{O} \{\text{PR}[\varphi] \leq \epsilon\}$ .
  - ▶  $\mathbb{O} \equiv_{R, \varphi} \mathbb{O}'$ .
  - ▶  $\mathbb{O} \leq_{\text{det}, \gamma} \mathbb{O}'$