

Automated Cryptographic Proofs for Asymmetric Encryption in the Random Oracle Model

J. Courant, M. Daubignard, C. Ene, Y. Lakhnech, P. Lafourcade

Université de Grenoble, CNRS
VERIMAG

19th May 2008
Alpine Verification Meeting

Outline

- 1 How Can One Prove Security ?
- 2 Designing a Hoare-logic-based Automatic Proof System For IND-CPA

Outline

- 1 How Can One Prove Security ?
- 2 Designing a Hoare-logic-based Automatic Proof System For IND-CPA

First Steps Towards A Scientific Approach For Security

- 1949: Shannon begins to link security with mathematics. He shows the need for compromise by defining perfect secrecy and its implications.

First Steps Towards A Scientific Approach For Security

- 1949: Shannon begins to link security with mathematics. He shows the need for compromise by defining perfect secrecy and its implications.
- 1976: Diffie and Hellmann found public-key cryptography. They figure out an essential type of primitive: one-way functions.

First Steps Towards A Scientific Approach For Security

- 1949: Shannon begins to link security with mathematics. He shows the need for compromise by defining perfect secrecy and its implications.
- 1976: Diffie and Hellmann found public-key cryptography. They figure out an essential type of primitive: one-way functions.
- First links between complexity theory and security.

First Steps Towards A Scientific Approach For Security

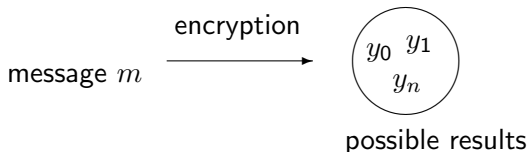
- 1949: Shannon begins to link security with mathematics. He shows the need for compromise by defining perfect secrecy and its implications.
- 1976: Diffie and Hellmann found public-key cryptography. They figure out an essential type of primitive: one-way functions.
- First links between complexity theory and security.
- 1979: Rabin proves secure a cryptosystem, providing the first reductionist proof.

Probabilistic Encryption

1984: Goldwasser and Micali start to work with probabilistic encryption.

Using probabilistic schemes implies:

- 1 If you encrypt m twice, you are unlikely to get the same result. Thus, these algorithms denote distributions on possible ciphertexts.

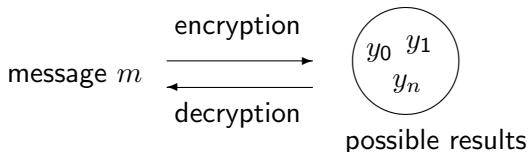


Probabilistic Encryption

1984: Goldwasser and Micali start to work with probabilistic encryption.

Using probabilistic schemes implies:

- 1 If you encrypt m twice, you are unlikely to get the same result. Thus, these algorithms denote distributions on possible ciphertexts.
- 2 But all possible results' decryptions lead to m !



Probabilistic Asymmetric Encryption Schemes: Formal Definition

- Definition : an asymmetric encryption scheme is a triple of
 - a key generation algorithm \mathcal{K} , that generates a pair $(pk, sk) \in \text{PKey} \times \text{SKey}$ of a public key and a secret key
 - a encryption algorithm $\mathcal{E} : \text{Ptext} \times \text{PKey} \rightarrow \text{Ctext}$
 - a decryption algorithm $\mathcal{D} : \text{Ctext} \times \text{SKey} \rightarrow \text{Ptext}$

- Functional correctness : $\mathcal{D}(\mathcal{E}(x, pk), sk) = x$

Hash functions

A hash function \mathcal{H} takes as input a bitstring and returns a corresponding 'digest' of fixed length.

Good hash functions are :

- collision-free: $\mathcal{H}(x) = \mathcal{H}(y) \Rightarrow x = y$

I do not have:

marion



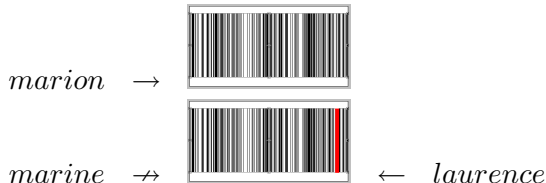
carine

Hash functions

A hash function \mathcal{H} takes as input a bitstring and returns a corresponding 'digest' of fixed length.

Good hash functions are :

- collision-free: $\mathcal{H}(x) = \mathcal{H}(y) \Rightarrow x = y$
- non-malleable: $x \mathcal{R} y \not\Rightarrow \mathcal{H}(x) \mathcal{R} \mathcal{H}(y)$



Oracles : What Can Your Adversary Do ?

- The **adversary** is a probabilistic poly-time Turing machine that can access oracles.

Oracles : What Can Your Adversary Do ?

- The **adversary** is a probabilistic poly-time Turing machine that can access oracles.
- A **random oracle** answers the following way:
 - 'seen-yet' query: the oracle answers the same bitstring as it has answered before,
 - 'fresh' query: the oracle picks its answer at random among the available values.
- We work in the *Random Oracle Model*: hash functions are random oracles and permanently available to adversaries.

Roughly : What Is a Distinguisher ?

Consider the following **IND-CPA** game against a given scheme:

- 1 The adversary computes anything he wants, and chooses two plaintexts (x_0, x_1) he gives me.

calls to \mathcal{E} ←
calls to oracles ←



x_0, x_1
→

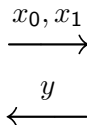


Roughly : What Is a Distinguisher ?

Consider the following **IND-CPA** game against a given scheme:

- ① The adversary computes anything he wants, and chooses two plaintexts (x_0, x_1) he gives me.
- ② I choose one of them x_b and replies $y = \mathcal{E}(x_b)$.

calls to \mathcal{E} ←
calls to oracles ←



Roughly : What Is a Distinguisher ?

Consider the following **IND-CPA** game against a given scheme:

- ① The adversary computes anything he wants, and chooses two plaintexts (x_0, x_1) he gives me.
- ② I choose one of them x_b and replies $y = \mathcal{E}(x_b)$.
- ③ The adversary has to tell which plaintext I chose to cipher.

calls to \mathcal{E} ←
calls to oracles ←



x_0, x_1
→
 y
←
Value of b ?
→



Roughly : What Is a Distinguisher ?

Consider the following **IND-CPA** game against a given scheme:

- 1 The adversary computes anything he wants, and chooses two plaintexts (x_0, x_1) he gives me.
- 2 I choose one of them x_b and replies $y = \mathcal{E}(x_b)$.
- 3 The adversary has to tell which plaintext I chose to cipher.



Advantage of an adversary

$$\text{Adv}(\mathcal{A}) = Pr[\mathcal{A}(\mathcal{E}(x_0)) = 0] - Pr[\mathcal{A}(\mathcal{E}(x_0)) = 1]$$

Motivation

Now we know what to prove, but...

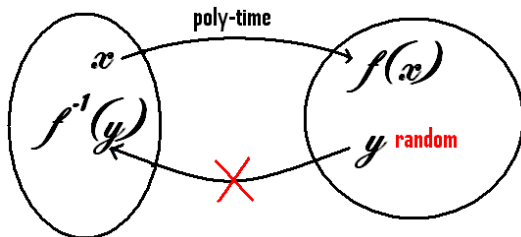
- Problem : nowadays, one scheme = one proof
- Our long-term goal is to prove cryptographic systems secure by enabling
 - Computer-Aided Cryptographic Proofs*at the level of abstract constructions and their implementations.
- Subgoal : find similarities between proofs and essential moves that allow to conclude.

Outline

- 1 How Can One Prove Security ?
- 2 Designing a Hoare-logic-based Automatic Proof System For IND-CPA

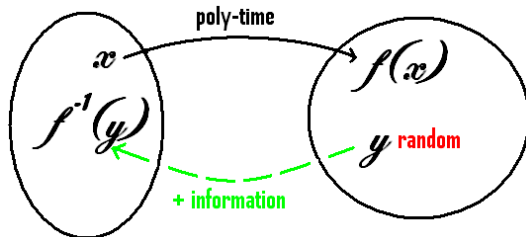
Inside the Ciphering Algorithm: One-way Functions

- **One-way function** f : $f(x)$ is easy to compute, but there is no probabilistic poly-time algorithm that can compute $f^{-1}(y)$ given a randomly sampled y .



Inside the Ciphering Algorithm: One-way Functions

- **One-way function** f : $f(x)$ is easy to compute, but there is no probabilistic poly-time algorithm that can compute $f^{-1}(y)$ given a randomly sampled y .



- **Trapdoor permutation** f : bijective function easy to compute in one way, hard to invert without additional information (e.g. RSA); replaces the choice of (pk, sk) .

A Simple Programming Language

Command $c ::= x \stackrel{r}{\leftarrow} \mathcal{U} \mid x := f(y) \mid x := f^{-1}(y)$
 $\mid x := H(y) \mid x := y \oplus z$
 $\mid x := y || z \mid \text{if } x = y \text{ then } c1 \text{ else } c2 \text{ fi} \mid c; c$

Oracle declaration $\mathcal{O} ::= \mathcal{N}(in, out) : c$

Semantics $\llbracket c \rrbracket : S \mapsto (S, Pr[S])$, lifted as a function from and onto distributions on states.

Three cornerstones: (1) Indis...

- if *world 1* and *world 2* are **indistinguishable**, the adversary cannot say in which *world* I picked my argument.

Three cornerstones: (1) Indis...

- if *world 1* and *world 2* are **indistinguishable**, the adversary cannot say in which *world* I picked my argument.

- **Indistinguishability**, expressed by predicate Indis:

$X \models \text{Indis}(\nu x, V_1, V_2)$ iff

$[S \stackrel{r}{\leftarrow} X : S(V_1), f(S(V_2))]$ \approx

$[S \stackrel{r}{\leftarrow} X ; u \stackrel{r}{\leftarrow} \mathcal{U}; S' := S\{x \mapsto u\} : (S'(V_1), f(S'(V_2)))]$

- the adversary **sees** V_1 and $f(V_2)$ to help him choose.

Indistinguishability - Example

Example

Let $X = [x \xleftarrow{r} \{0, 1\}^n; y := H(x)]$.

We have $X \models \text{Indis}(\nu y; y; x)$

Indistinguishability - Example

Example

Let $X = [x \xleftarrow{r} \{0, 1\}^n; y := H(x)]$.

We have $X \models \text{Indis}(\nu y; y; x)$

but we do not have $X \models \text{Indis}(\nu y; \{x, y\}; \emptyset)$.

WHY ?

Indistinguishability - Example

Example

Let $X = [x \xleftarrow{r} \{0, 1\}^n; y := H(x)]$.

We have $X \models \text{Indis}(\nu y; y; x)$

but we do not have $X \models \text{Indis}(\nu y; \{x, y\}; \emptyset)$.

WHY ? Because the adversary can query H on x and compare it to y !

Three cornerstones: (2) $H(H, e) \dots$

- **ROM hypothesis:** Hash values must be asked for.
- Idea: the adversary cannot query hash oracles on randomized values.

Three cornerstones: (2) $H(H, e) \dots$

- **ROM hypothesis:** Hash values must be asked for.
- Idea: the adversary cannot query hash oracles on randomized values.
- **Not-Hashed-Yet**, expressed by predicate $H(H, e)$:
 $X \models H(H, e)$ iff
 $\Pr[S \stackrel{r}{\leftarrow} X : S(e) \in \text{Arg}(H)]$ is negligible,
where $\text{Arg}(H)$ are arguments on which the hash oracle was queried.

Three cornerstones : (3) WS...

- **ROM hypothesis**: Hash values must be asked for.
- Many systems' security rely on the inability of an adversary to compute **the right argument** on which the oracle should be queried!

Three cornerstones : (3) WS...

- **ROM hypothesis**: Hash values must be asked for.
- Many systems' security rely on the inability of an adversary to compute **the right argument** on which the oracle should be queried!
- **Weak Secrecy**, expressed by predicate WS:
 $X \models \text{WS}(x; V)$ iff for any adversary \mathcal{A} ,
 $\Pr[S \stackrel{r}{\leftarrow} X : \mathcal{A}(S(V)) = S(x)]$ is negligible.

...And A Few Rules...

- $\{true\} x \stackrel{r}{\leftarrow} \mathcal{U} \{H(H, x)\},$

...And A Few Rules...

- $\{true\} x \stackrel{r}{\leftarrow} \mathcal{U} \{H(H, x)\},$
- $\{\text{Indis}(\nu y; V \cup \{y\}; \emptyset)\} x := f(y) \{\text{WS}(y; V \cup \{x\})\}$
if $y \notin V \cup \{x\}$

...And A Few Rules...

- $\{true\} x \stackrel{r}{\leftarrow} \mathcal{U} \{H(H, x)\},$
- $\{Indis(\nu y; V \cup \{y\}; \emptyset)\} x := f(y) \{WS(y; V \cup \{x\})\}$
if $y \notin V \cup \{x\}$
- $\{WS(y; V) \wedge H(H, y)\} x := H(y) \{Indis(\nu x; V \cup \{x\}; \emptyset)\}$
etc...

...To Build Secure Schemes!

Proposition

For every rule $\{\varphi\}c\{\varphi'\}$, we have

$$X \models \varphi \text{ implies } \llbracket c \rrbracket X \models \varphi'.$$

Theorem

Let $GE = (\mathbb{F}, \mathcal{E}(in_e, out_e) : c, \mathcal{D}(in_d, out_d) : c')$ be an asymmetric encryption scheme.

If true \xrightarrow{c} $Indis(\nu out_e; in_e, out_e; \emptyset)$ then \mathcal{E} is IND-CPA.

Schemes Proven In The Logic

- BR'93:

$$f(r) || x \oplus G(r) || H(x || r)$$

- ZS':

$$f(r) || G(r) \oplus (x || H(x))$$

- OAEP₊:

$$f(s || r \oplus H(s))$$

where $s = x \oplus G(r) || H'(r || x)$.

- Fujisaki & Okamoto:

$$\mathcal{E}((x || r); H(x || r))$$

where \mathcal{E} is IND-CPA.

Conclusion

- **AUTOMATED proof** of IND-CPA in the random-oracle model.
- **The method has been implemented in CamL.**
- The logic has been extended to prove IND-CCA, for which we have a semantic criterion and a syntactic one.
- Possible extensions to other criteria and to specific key-events that bound the advantage of a distinguisher.