

An Elementary Tutorial on Provable Security

Jae Choon Cha

KMS-KIAS Summer Workshop on Cryptography 2007

June 29, 2007

Goal of this talk : Help you to prove the security of your own cryptosystem

We will : introduce necessary backgrounds
(for mathematics people)
discuss how to formalize the notion of security
give an example of "proof of security"

We will NOT : give a survey on known results
report nor announce new progresses

Aim : Prove that a cryptosystem is secure

we need technologies! we need definitions!

Roughly speaking, a cryptosystem is said to be secure if it is hard to break the cryptosystem i.e.,

there is no efficient algorithm breaking it. ... (*)

Issues : what do you mean by the terms in (*) ?

Are we really able to prove (*) ?

Outline

- Algorithm and model of computation
- Randomized computation
- Formalization of a cryptosystem and its security
 - security model : definition of security
- How to prove the security
 - reduction method, primitive problems and random oracle model
- A detailed example of security proof
 - Gap Diffie-Hellman signature scheme

Algorithm = a sequence of computational steps
transforming the input to the output
(The output is expected to be a solution of a given problem)

Correctness : is the output a solution ?

Efficiency : Time and space complexity input size
Asymptotic behaviour of running time $T(n)$
e.g. binary search $T(n) = O(\log n)$
i.e. $\exists C$ s.t. $T(n) \leq C \log n$
for all large n

Example : Computation of n^n
input = n , output = n^n

1. $r \leftarrow 1$
2. for $k \leftarrow 1$ to n do
 $r \leftarrow r \cdot k$
3. output r .

Running time $T(n) = O(n)$???

We need a precise def'n of running time !

Model of computation

Def : an algorithm = "Random Access Machine" (RAM)

- register
- memory = an array of integers

addr	content
0	...
1	...
2	...
⋮	⋮
- I/O tape 01101001... (binary data)
- program = a sequence of "instructions" ←

- register-memory transfer
 - arithmetic operations (+ - · /)
 - comparison (\leq \geq) and branching
 - tape I/O

Log-cost RAM : $\left(\begin{matrix} \text{running time} \\ \text{of each instruction} \end{matrix} \right) \propto \log(\text{operand})$
operand size

Total running time $T(n) = \sum (\text{running time of each instruction executed})$

where input size $n = \text{length of the input tape (data)}$
 $= \sum \log |x_i|$ if input = (x_1, x_2, \dots)

An algorithm is a polynomial time algorithm if
 $T(n) = O(n^c)$ for some const c .

Theorem : Polynomial time RAM \iff polynomial time TM

e.g. input = n
 $r \leftarrow 2$
 for $i \leftarrow 1$ to n do
 $r \leftarrow r \cdot r$
 output r

Unit-cost RAM: $T(n) = O(n)$

Log-cost RAM: $T(n) = \sum_{i=1}^n O(\log r_i) = \sum_{i=1}^n 2^i \geq 2^n$

Note that the output is $2^{2^{n-1}}$, which is too large for a log-cost RAM to construct within time $\leq n$.

Monte Carlo algorithm: even the output may vary,
 i.e. the output may be incorrect!

e.g. Random guessing: output \xleftarrow{R} {all possible outputs}

We think of: $\left\{ \begin{array}{l} \text{Success probability} = \Pr[\text{output is correct}] \\ \text{Advantage} = \Pr[\text{succ}] \\ \quad - (\text{random guessing's } \Pr[\text{succ}]) \end{array} \right.$

For a decision problem (Yes/No question),
 we usually normalize it as: $\text{Advantage} = 2\Pr[\text{succ}] - 1 \leq 1$

Randomized (Probabilistic) algorithm

= RAM + "unbiased coin toss" instruction

= RAM + a random tape 011011100 random numbers!

Las Vegas algorithm: the output is always correct but the running time may vary even for the same input

... we think of $\left\{ \begin{array}{l} \text{expected} \\ \text{worst-case} \end{array} \right\}$ running time
 (over all coin tosses & inputs)

Def A computational problem is feasible if:

\exists a polynomial time randomized algorithm solving the given problem with advantage "not too small".

(1) $\text{Adv} = 1$: Las Vegas

(2) $\text{Adv} \geq \frac{1}{2}$ if output = yes : RP (Randomized Poly. time)
 $= 1$ if output = no

(3) $\text{Adv} \geq \frac{1}{2}$: BPP (Bounded-error...)

(4) $\text{Adv} > 0$: PP (Prob. Poly. time)

(5) Adv is non-negligible (as a function of input size) \leftarrow

This is what we will use!

Def: $\nu(k)$ is negligible if \forall poly $P(k)$, $\nu(k) \leq \frac{1}{P(k)}$ asymptotically

Cryptosystem: a collection of algorithms

e.g. Symmetric Key Cryptosystem = $(\mathcal{E}, \mathcal{D})$

$$(K, M) \rightarrow \mathcal{E} \rightarrow \mathcal{E}(K, M) = C$$

$$\text{s.t. } \mathcal{D}(K, \mathcal{E}(K, M)) = M$$

$$(K, C) \rightarrow \mathcal{D} \rightarrow \mathcal{D}(K, C) = M$$

[Shannon] This can be secure only if the key size is as large as the message size

(assuming the adversary has unlimited computing power)

Modern Cryptography: based on complexity theory

• Security parameter $k \xrightarrow{\text{Setup}}$ Cryptosystem

• Said to be secure if: \nexists poly. time (in k) adversary which "breaks" the system.

We want to prove

To obtain meaningful results, we work in the rand. comp. model

e.g. Public Key Cryptosystem = $(\mathcal{K}, \mathcal{E}, \mathcal{D})$

$\mathcal{K}(1^k) = (PK, SK)$ PK = public key, SK = secret key

$\mathcal{E}(PK, M) = C$ s.t. $\mathcal{D}(SK, \mathcal{E}(PK, M)) = M$

$\mathcal{D}(SK, C) = M$

Note: C is not uniquely determined by (PK, M)

What do you mean by "breaking" a cryptosystem?

e.g. In case of PKC, requirements may be ...

• Recovery of SK is infeasible

.... what about $PK \leftarrow \text{random}$, $\mathcal{E}(PK, M) = C$,
 $\mathcal{D}(SK, C) = M$?

• Recovery of M (from PK, C) is infeasible

.... But, an adversary still can recover first bit of M

The answer is NOT straightforward at all!

To give a formal definition, we think of the goal and ability of an adversary:

e.g. Goal - indistinguishability

given m_0, m_1 , $y = \mathcal{E}(PK, m_b)$ where $b \in \{0, 1\}$ hidden, can an adversary find b ? (without knowing SK)

Ability - Chosen plaintext attack: $\mathcal{E}(PK, -)$ is available

Chosen ciphertext attack: $\mathcal{E}(PK, -)$, $\mathcal{D}(SK, -)$ before y is given

Adaptively chosen ciphertext attack:

$\mathcal{E}(PK, -)$, $\mathcal{D}(SK, -)$ anytime but $\neq y$

Def: IND-CCA2 security of (K, E, D)

An adversary A is a pair (A_1, A_2) of algorithms s.t.

$A_1(PK) = (x_0, x_1, s)$ where $x_i = \text{message}$,
 $s = \text{bitstring}$

$A_2(x_0, x_1, s, y) = d$ where $x_i, s = \text{as above}$
 $y = \text{ciphertext}, d \in \{0, 1\}$

Consider the following game (by a challenger \mathcal{C}):

1. $(PK, SK) \leftarrow K(1^k)$
2. $(x_0, x_1, s) \leftarrow A_1(PK)$ where A_1 can make $D(SK, -)$ and $E(PK, -)$ queries
3. $b \xleftarrow{R} \{0, 1\}; y \leftarrow E(PK, x_b)$
4. $d \leftarrow A_2(x_0, x_1, s, y)$ where A_2 can make $D(SK, -)$ and $E(PK, -)$ queries

We say A wins the game if $d = b$.

Advantage of $A := 2\Pr[d=b] - 1 \leq 1$

We say (K, E, D) is IND-CCA2 secure if

there is no polynomial time $A = (A_1, A_2)$
with non-negligible advantage

Remark: ① IND-CCA2 is the strongest known security notion for PKC

② There are known schemes which are IND-CCA2 secure (under certain assumptions!)

Signature Scheme = $(K, \mathcal{S}, \mathcal{V})$

$K(1^k) = (PK, SK)$ Public and secret key

$\mathcal{S}(SK, M) = \sigma$ Signature for the msg M

$\mathcal{V}(PK, \sigma, M) = \text{"valid" or "invalid"}$... Validity of σ

Def: Security against existential forgery (goal)
on adaptively chosen message attack (ability)

Adversary = an algorithm $A(PK) = (\sigma, M)$

- Game: 1. $(PK, SK) \leftarrow K(1^k)$
2. $(\sigma, M) \leftarrow A(PK)$ where A can make $\mathcal{S}(SK, -)$ queries

A wins if $\mathcal{V}(PK, \sigma, M) = \text{valid}$; Advantage := $\Pr[\text{win}]$

$(K, \mathcal{S}, \mathcal{V})$ is secure if \nexists poly. time A with non-neg. adv.

ID-Based signature scheme $(K, E, \mathcal{S}, \mathcal{V})$

$K(1^k) = (\text{group } G, \text{hash } H: \{0, 1\}^* \rightarrow G)$

$E(ID) = SK_{ID} \in G$

($PK_{ID} = H(ID)$ plays the role of a pub. key)

$\mathcal{S}(SK_{ID}, M) = \sigma$

$\mathcal{V}(PK_{ID}, M, \sigma) = \text{valid or invalid}$

- Game: 1. Run K
2. A makes queries to H, E, \mathcal{S}
3. A outputs (ID, M, σ) "chosen ID, chosen msg"

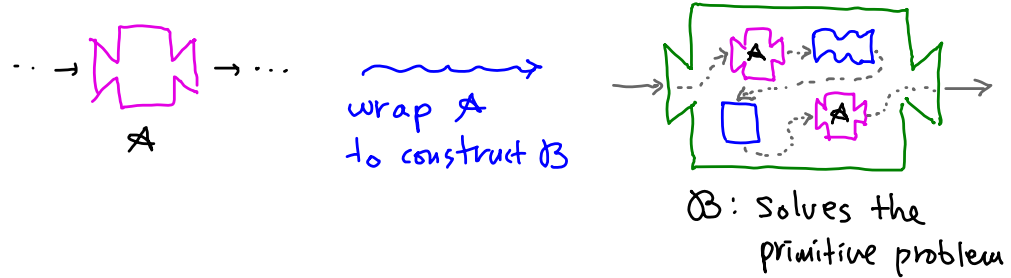
A wins if $\mathcal{V}(PK_{ID}, M, \sigma)$ is valid.

There is no known technology to prove the security without any additional assumption ... what can we do?

Primitive problem = a problem which $\left. \begin{array}{l} \text{is} \\ \text{seems} \\ \text{believed to be} \end{array} \right\}$ infeasible

- e.g. RSA inverting given $x^e \bmod n$, find x
- Factoring given $n=pg$, find p and g
- DLP given P and aP , find a ($a \in \mathbb{Z}, P \in G$)
- CDHP given P, aP, bP , find abP "finite abelian group"
- BDHP given aP, bP, cP , find $e(P,P)^{abc}$ "finite abelian group"
($e: G \times G \rightarrow G'$ bilinear)

Technique: Given an algorithm \mathcal{A} breaking the cryptosystem, Construct an algorithm solving the primitive problem using \mathcal{A} as a black-box tool!



Reduction method

If we are lucky, then we can show:

\exists an algorithm breaking the cryptosystem
 $\Rightarrow \exists$ an algorithm solving an associated primitive prob.

(algorithms are polynomial time and of non-negligible advantage)

In this case we say: **the cryptosystem is provably secure**
 (under the assumption that the primitive problem is hard)

Remark: Usually, the converse is true.

Hash functions play an important role in modern cryptography

A typical usage:

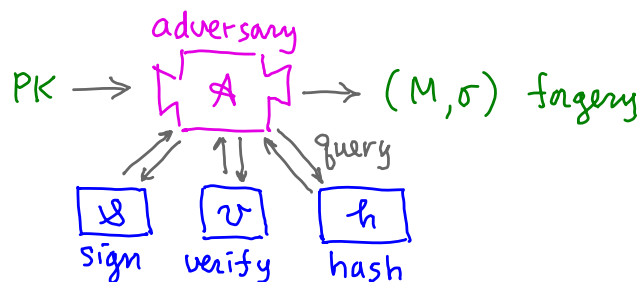
$$\left\{ \begin{array}{l} \text{long messages} \\ \text{(bitstrings)} \end{array} \right\} \xrightarrow{h} G = \left(\begin{array}{l} \text{algebraic object} \\ \text{where the primitive} \\ \text{problem is defined} \end{array} \right)$$

Usually (a pointer to) h is available to all parties as a system parameter so that queries to h can be made:

Random Oracle Model:

assume that h is a random function, i.e., for each (new) query, h generates a random output.

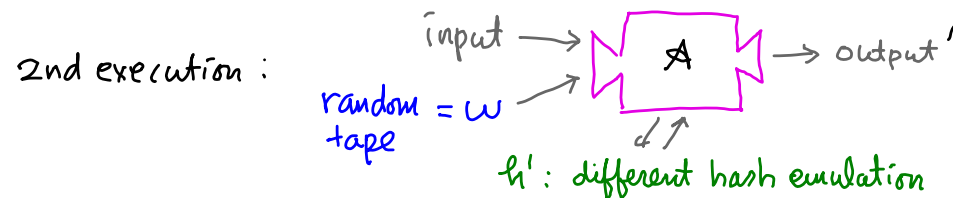
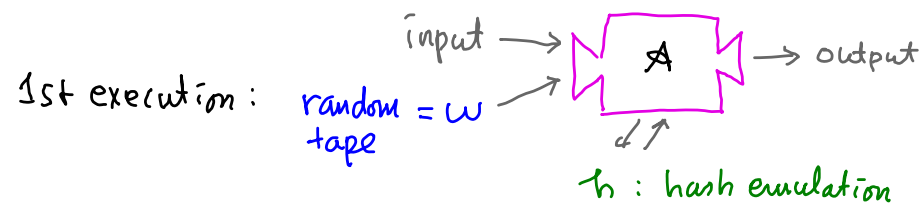
e.g. Existential forgery for a signature scheme on adaptively chosen message attack:



In a security proof, one may "emulate" h (as well as S and V) for one's own purpose, provided that its output is a random distribution.

Oracle Replay Attack [Pointcheval-Stern '96]

Idea: "replay" the adversary A with the same random tape but with a differently controlled (emulation of) hash functions



An example of security proof: GDH-signature scheme [Bohler, Lynn, Shacham]

G = cyclic group with generator g , $|G| = p$ prime

- Assume:
- ① Decision Diffie-Hellman is feasible
i.e. given (g, g^a, g^b, g^c) , we can verify if $c = ab$ (valid DH-tuple)
 - ② Computational Diffie-Hellman is infeasible
i.e., given (g, g^a, g^b) , computing g^{ab} is hard.

Such a group G is called a gap Diffie-Hellman (GDH) group.

GDH-signature scheme

Setup \mathcal{K} : choose G ($\log |G| \propto$ security parameter k)
choose a hash function $h: \{0,1\}^* \rightarrow G^*$
 $x \leftarrow^R \mathbb{Z}_p^*$
Output: $PK = g^x$, $SK = x$.

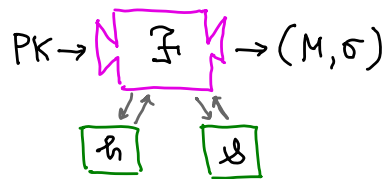
Sign \mathcal{S} : given (x, M) ,
output $\sigma = h(M)^x$

Verify \mathcal{V} : given (PK, M, σ) ,
output = valid $\iff (g, PK, h(M), \sigma)$
is a valid DH-tuple.
(If $\sigma = h(M)^x$, then it holds!)

Proof of security

Suppose there is a forgery algorithm \mathcal{F} with

running time = t
 # of h -queries = q_H
 # of \mathcal{S} -queries = q_S
 advantage = ϵ



$$\text{i.e. } \epsilon = \Pr \left[\mathcal{V}(\text{PK}, M, \sigma) = \text{valid} \mid \begin{array}{l} (\text{PK}, \text{SK}) \leftarrow \mathcal{K} \\ (M, \sigma) \leftarrow \mathcal{F}(\text{PK}) \end{array} \right]$$

Goal: construct an algorithm solving CDHP
 with running time \leq polynomial in t, q_H, q_S
 advantage $\geq \epsilon \cdot \frac{1}{\text{polynomial}}$.

We start with a very simple challenger algorithm,
 and proceed to more sophisticated ones...

Notation: $B_\zeta =$ distribution on $\{0, 1\}$
 given by $\Pr[1] = \zeta, \Pr[0] = 1 - \zeta$ ($0 \leq \zeta \leq 1$)
 (ζ will be specified later)

$M_i =$ input (message) to the i^{th} h -query.

We may assume: \mathcal{F} requests $h(M)$ before $\mathcal{S}(\text{SK}, M)$.
 \mathcal{F} requests h for its output.

Algorithm A_1 : input = (g, g^a, g^b)

$\text{PK} \leftarrow g^a$

For $i=1, 2, \dots, q_H$, pick $s_i \xleftarrow{R} B_\zeta, r_i \xleftarrow{R} \mathbb{Z}_p^*$
 set $h_i \leftarrow g^{r_i}, \sigma_i \leftarrow (g^a)^{r_i}$.

Run $\mathcal{F}(\text{PK})$; to the i^{th} h -query, return h_i
 to \mathcal{S} -query for M_i , return σ_i .

Let (M^*, σ^*) be the output of \mathcal{F} ($M^* = M_{i^*}$ for some i^*)

If \mathcal{F} succeeds, output "success"

otherwise, output "fail".

Analysis: (M_i, σ_i) is a valid signature

\mathcal{F} cannot distinguish A_1 from the standard
 challenger

$$\therefore \text{Adv}(A_1) = \text{Adv}(\mathcal{F}) = \epsilon.$$

$$\begin{pmatrix} g, \text{PK}, h(M_i), \sigma_i \\ \parallel \parallel \parallel \\ g^a, g^{r_i}, g^{ar_i} \end{pmatrix}$$

Algorithm A_2

The same as A_1 except the following:

If \mathcal{F} succeeds and $s_{i^*} = 1$, then output "success".
 otherwise, output "fail".

Analysis: The choice of s_i is independent of \mathcal{F}

$$\therefore \text{Adv}(A_2) = \Pr[s_{i^*} = 1] \cdot \text{Adv}(A_1) = \zeta \cdot \epsilon.$$

Algorithm A₃

The same as A₂ except:

If \mathcal{F} succeeds and $s_i^* = 1$ and

\mathcal{F} asked for signatures only for M_i s.t. $s_i = 0$,
then output "success"

otherwise, output "fail"

Analysis: Suppose the j^{th} \mathcal{S} -query is on M_j .

$$\begin{aligned} \text{Then } \text{Adv}(A_3) &= \left(\prod_j \Pr[s_j = 0] \right) \cdot \text{Adv}(A_2) \\ &= (1-\zeta)^{\delta_S} \cdot \zeta \cdot \varepsilon. \end{aligned}$$

Algorithm A₄

The same as A₃ except:

If \mathcal{F} requests the sign of M_i for which $s_i = 1$,
then output "fail" and exits immediately.

Analysis: Compare runs of A₃ and A₄ with the same
input and random tape.

If $\mathcal{S}(\text{sk}, M_i)$ is requested and $s_i = 1$ in A₃,
then so is in A₄, and both output fail.

Otherwise, the output of A₃ and A₄ are the same

$$\therefore \text{Adv}(A_4) = \text{Adv}(A_3) = (1-\zeta)^{\delta_S} \cdot \zeta \cdot \varepsilon.$$

Algorithm A₅

Modify A₄ as follows:

In the setup step, let

$$h_i \leftarrow g^b g^{r_i}, \quad \sigma_i \leftarrow "*" \quad \text{if } s_i = 1$$

$$h_i \leftarrow g^{r_i}, \quad \sigma_i \leftarrow (g^b)^{r_i} \quad \text{if } s_i = 0 \quad (\text{as before})$$

Analysis: The choice of h_i is still random.

Since we don't need σ_i for $s_i = 1$,

the behaviour of \mathcal{F} remains the same.

$$\therefore \text{Adv}(A_5) = \text{Adv}(A_4) = (1-\zeta)^{\delta_S} \cdot \zeta \cdot \varepsilon.$$

Algorithm A₆ Modify A₅ as follows:

When A₅ outputs "success",

A₆ outputs "success" and $\sigma^* / (g^a)^{r_i^*}$.

Analysis: Obviously $\text{Adv}(A_6) = \text{Adv}(A_5) = (1-\zeta)^{\delta_S} \cdot \zeta \cdot \varepsilon$.

Since $s_{i^*} = 1$, $h_{i^*} = g^b \cdot g^{r_{i^*}}$ by our choice

Since $(g, g^a, h_{i^*}, \sigma^*)$ is a valid DH-tuple,

$$\text{we have } \sigma^* = (h_{i^*})^a = g^{ab} \cdot g^{a r_{i^*}}$$

\therefore output $\sigma^* / (g^a)^{r_{i^*}} = g^{ab}$ as desired !!!

Final Analysis :

$$Adv = (1-\zeta)^{g_s} \cdot \zeta \cdot \varepsilon \text{ is maximized at } \zeta = \frac{1}{g_s+1}$$

$$\left(\begin{aligned} \frac{d}{d\zeta} Adv &= -g_s(1-\zeta)^{g_s-1} \zeta \varepsilon + (1-\zeta)^{g_s} \cdot \varepsilon \\ &= (1-\zeta)^{g_s-1} \cdot \varepsilon \cdot (1-\zeta - g_s \zeta) = 0 \\ &= 0 \text{ when } \zeta = \frac{1}{g_s+1}. \end{aligned} \right)$$

$$\text{At } \zeta = \frac{1}{g_s+1}, \quad Adv = \frac{1}{g_s} \underbrace{\left(1 - \frac{1}{g_s+1}\right)^{g_s+1}}_{\rightarrow \frac{1}{e} \text{ as } g_s \rightarrow \infty} \cdot \varepsilon$$

$$\begin{aligned} \text{Running time} &= \text{Running time of } \mathcal{F}_1 + (\text{const}) \\ &+ \underbrace{(\text{exponentiation computing time for } h_i, \sigma_i)}_{c \cdot (g_H + g_S)} \end{aligned}$$

Thank You!