CRYE 6138 Security Models

Jorge L. Villar

UBa Cyber Crypto Center, Fall 2025

Computational Security Notions

Outline

Bounding the Adversary's Capabilities

Perfect Security vs. Computational Security

- **Perfect Security:** The adversary is unbounded, but only a few positive results can be obtained. (E.g., perfect one-time encryption.)

Bounding the Adversary's Capabilities

Perfect Security vs. Computational Security

- **Perfect Security:** The adversary is unbounded, but only a few positive results can be obtained. (E.g., perfect one-time encryption.)
- **Computational Security:** The adversary is bounded in
  - running time
  - memory usage
  - success probability

  Security is based on the assumption that some computational problem (family) is hard.

Bounding the Adversary's Capabilities

Perfect Security vs. Computational Security

- **Perfect Security:** The adversary is unbounded, but only a few positive results can be obtained. (E.g., perfect one-time encryption.)
- **Computational Security:** The adversary is bounded in
  - running time
  - memory usage
  - success probability

  Security is based on the assumption that some computational problem (family) is hard.

We need to define:

- A notion of problem hardness. (Asymptotic analysis.)
- A computation model. (Turing machine.)

Turing Machine

A model for mechanic computation, consisting of

- A finite automaton
- A storage device (“tape”)

Turing Machine

A model for mechanic computation, consisting of

- A finite automaton
  - state information $s \in S \cup \{\texttt{init}, \texttt{halt}\}$
  - transition function f
- A storage device (“tape”)

Turing Machine

A model for mechanic computation, consisting of

- A finite automaton
  - state information $s \in S \cup \{\texttt{init}, \texttt{halt}\}$
  - transition function f
- A storage device ("tape")
  - a "tape", or a sequence of cells containing '0', '1' or '`blank`'
  - a read/write moving head on a specific cell in the tape

Turing Machine

A model for mechanic computation, consisting of

- A finite automaton
  - state information $s \in S \cup \{\texttt{init}, \texttt{halt}\}$
  - transition function f
- A storage device ("tape")
  - a "tape", or a sequence of cells containing '0', '1' or '`blank`'
  - a read/write moving head on a specific cell in the tape

Computation step: $(s, wr, mv) \leftarrow f(s, c)$

$wr \in \{\texttt{write\_0}, \texttt{write\_1}, \texttt{erase}\}$

$mv \in \{\texttt{move\_left}, \texttt{move\_right}\}$

$c =$ contents of the current cell

Turing Machine

A model for mechanic computation, consisting of

- A finite automaton
  - state information $s \in S \cup \{\texttt{init}, \texttt{halt}\}$
  - transition function f
- A storage device ("tape")
  - a "tape", or a sequence of cells containing '0', '1' or '`blank`'
  - a read/write moving head on a specific cell in the tape

Computation step: $(s, wr, mv) \leftarrow f(s, c)$

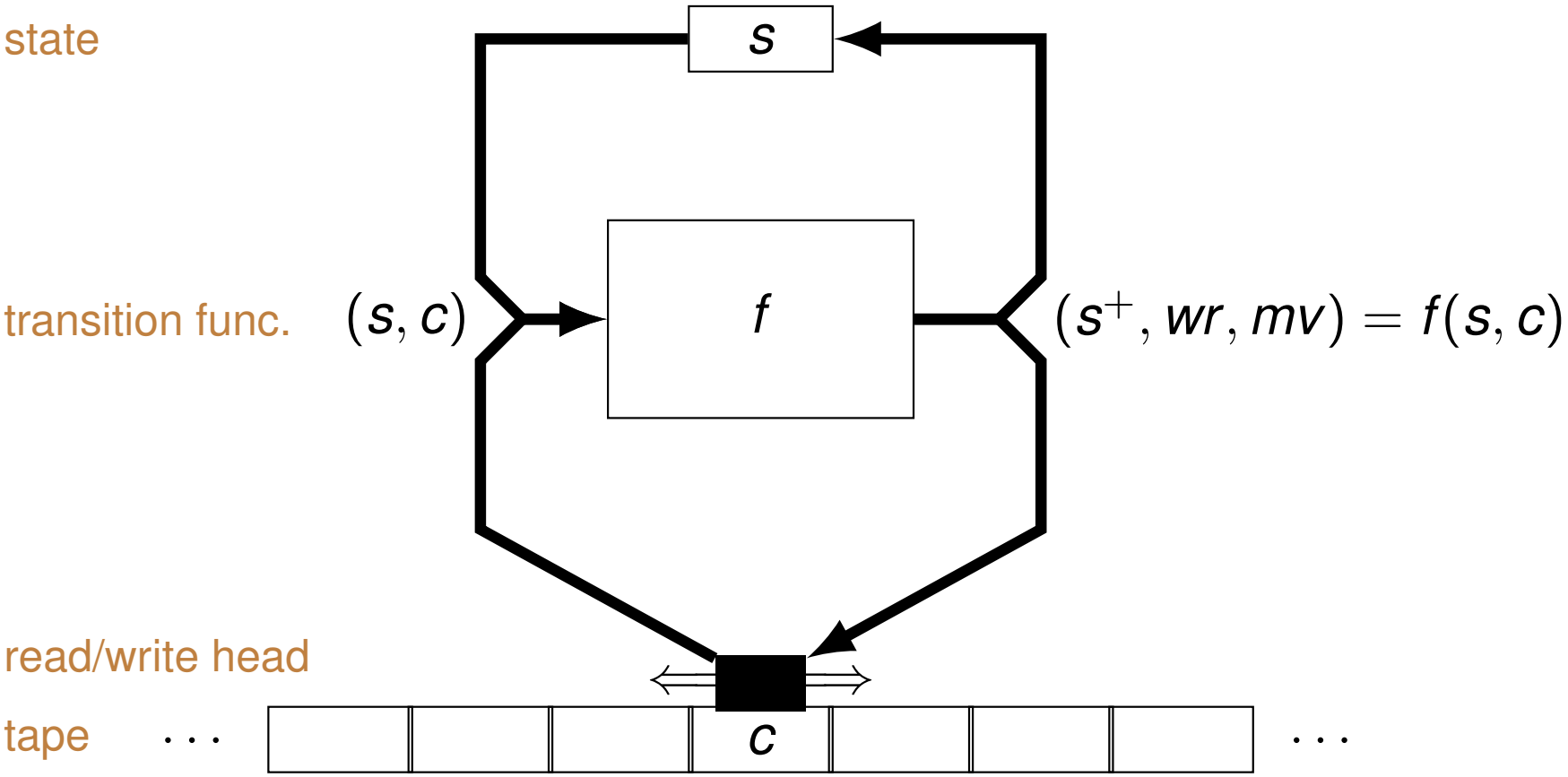
$wr \in \{\texttt{write\_0}, \texttt{write\_1}, \texttt{erase}\}$

$mv \in \{\texttt{move\_left}, \texttt{move\_right}\}$

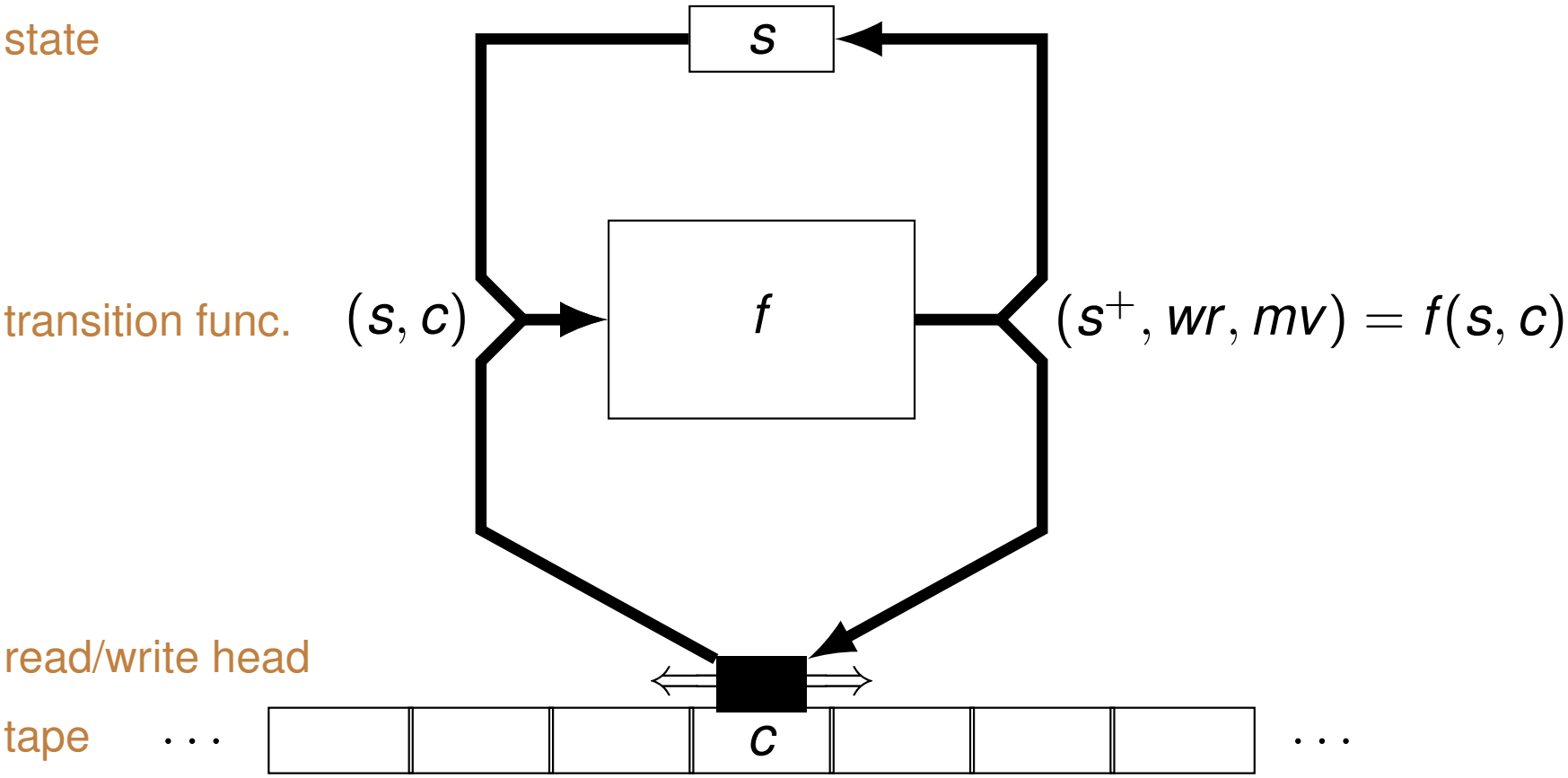
$c =$ contents of the current cell

Input/Output of the Turing Machine is the content of the tape in the `init`/`halt` state

Turing Machine



Turing Machine

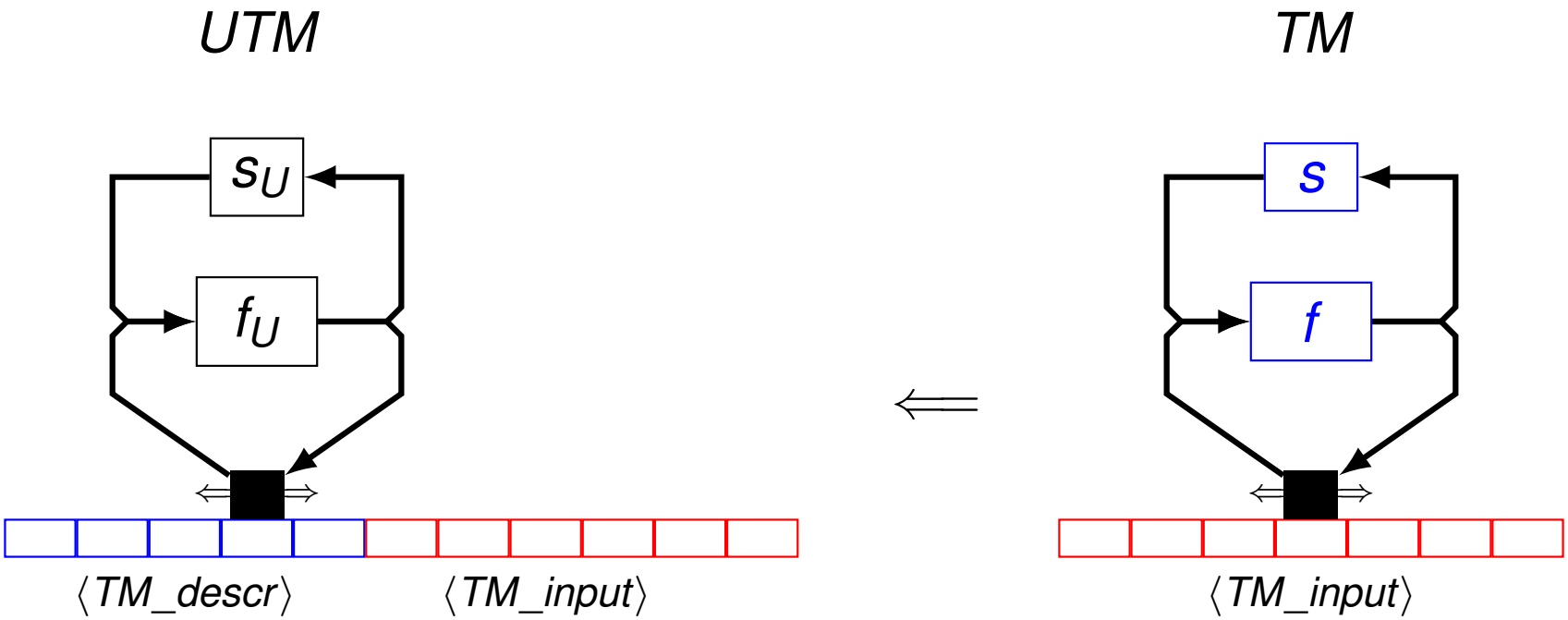


$s \in S \cup \{\texttt{init}, \texttt{halt}\}$

$wr \in \{\texttt{write\_0}, \texttt{write\_1}, \texttt{erase}\}$

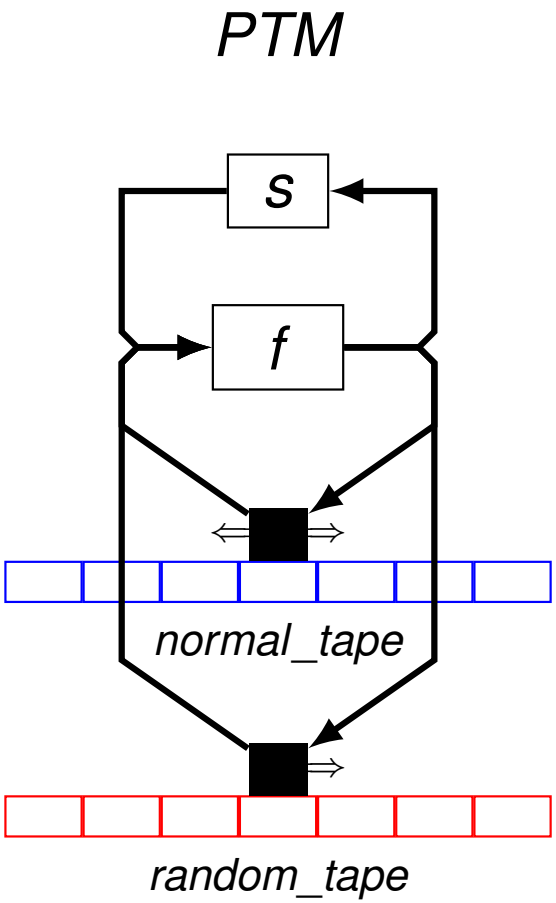
$mv \in \{\texttt{move\_left}, \texttt{move\_right}\}$

Universal Turing Machine



It can simulate any known (classical) computing device with reasonable efficiency

Probabilistic Turing Machine



A Turing Machine that take random decisions, from an additional input tape.

Computation step:
$(s, wr, mv, mvr) \leftarrow f(s, c, r)$

$wr \in \{\texttt{write\_0}, \texttt{write\_1}, \texttt{erase}\}$
$mv \in \{\texttt{move\_left}, \texttt{move\_right}\}$
$mvr \in \{\texttt{keep}, \texttt{move\_right}\}$

$c =$ contents of the current normal cell

$r =$ contents of the current random cell

Algorithmic Complexity

A running time limitation (number of steps of the Probabilistic Turing Machine) implies similar space and randomness limitations.
Every read/write operations takes one step.
Reading a random bit takes one step.

The running time typically depends on the size of the input (description of the problem to be solved).

Algorithmic Complexity

A running time limitation (number of steps of the Probabilistic Turing Machine) implies similar space and randomness limitations.
Every read/write operations takes one step.
Reading a random bit takes one step.

The running time typically depends on the size of the input (description of the problem to be solved).

Uniform approach: A single Turing Machine tries to solve problems of all sizes.

Algorithmic Complexity

A running time limitation (number of steps of the Probabilistic Turing Machine) implies similar space and randomness limitations.
Every read/write operations takes one step.
Reading a random bit takes one step.

The running time typically depends on the size of the input (description of the problem to be solved).

Uniform approach: A single Turing Machine tries to solve problems of all sizes.

Asymptotic analysis: We study the problem complexity by the type of growth of the running time as a function of the input size.

Hardness Notions

Easy computation: For a problem family $\mathcal{P}$, there exists a Turing Machine TM such that

- The running time is polynomial on the size of $P \in \mathcal{P}$.
- It always outputs a correct solution of any problem instance $P \in \mathcal{P}$

$$\exists T(\cdot) \in \textbf{poly} \quad \forall P \in \mathcal{P}$$
$$\text{time}(\text{TM}, P) \leq T(|P|) \quad \text{and} \quad \text{valid}(P, TM(P)) = 1$$

---

poly: The set of positive polynomials.

Hardness Notions

Easy computation: For a problem family $\mathcal{P}$, there exists a Turing Machine TM such that

- The running time is polynomial on the size of $P \in \mathcal{P}$.
- It always outputs a correct solution of any problem instance $P \in \mathcal{P}$

$$\exists T(\cdot) \in \textbf{poly} \quad \forall P \in \mathcal{P}$$
$$\text{time}(\text{TM}, P) \leq T(|P|) \quad \text{and} \quad \text{valid}(P, TM(P)) = 1$$

A model for normal operations performed by honest parties in a protocol.

---

poly: The set of positive polynomials.

Hardness Notions

Feasible computation: There exists a Probabilistic Turing Machine TM such that

- The running time is polynomial on the size of $P \in \mathcal{P}$.
- The probability that it gives a correct solution for any problem instance $P \in \mathcal{P}$ is not *negligible*.

$\exists T(\cdot), Q(\cdot) \in \textbf{poly} \quad \forall P \in \mathcal{P}$
$\text{time}(\text{TM}, P) \leq T(|P|)$ and $\Pr(\text{valid}(P, TM(P)) = 1) \geq 1/Q(|P|)$

---

negl: The set of functions $f(\cdot)$ such that for any $Q(\cdot) \in \textbf{poly}$ there exists n_0 s.t. $f(n) < 1/Q(n)$ for all $n > n_0$.

Hardness Notions

Feasible computation: There exists a Probabilistic Turing Machine TM such that

- The running time is polynomial on the size of $P \in \mathcal{P}$.
- The probability that it gives a correct solution for any problem instance $P \in \mathcal{P}$ is not *negligible*.

$\exists T(\cdot), Q(\cdot) \in \textbf{poly} \quad \forall P \in \mathcal{P}$
$\text{time}(\text{TM}, P) \leq T(|P|)$ and $\Pr(\text{valid}(P, TM(P)) = 1) \geq 1/Q(|P|)$

A model for a successful attack against a protocol.

---

negl: The set of functions $f(\cdot)$ such that for any $Q(\cdot) \in \textbf{poly}$ there exists n_0 s.t. $f(n) < 1/Q(n)$ for all $n > n_0$.

Hardness Notions

Infeasible computation: For all Probabilistic Turing Machines TM such that the running time is polynomial on the size of $P \in \mathcal{P}$, the probability that it gives a correct solution for $P \in \mathcal{P}$ is a negligible function of its length.

If $\exists T(\cdot) \in$ **poly** such that $\forall P \in \mathcal{P}$ time(TM, P) $\leq T(|P|)$ then $\Pr(\text{valid}(P, TM(P)) = 1) \in$ **negl**$(|P|)$.

PPTM: A Probabilistic Turing Machine running in Polynomial time.

Hardness Notions

Infeasible computation: For all Probabilistic Turing Machines TM such that the running time is polynomial on the size of $P \in \mathcal{P}$, the probability that it gives a correct solution for $P \in \mathcal{P}$ is a negligible function of its length.

If $\exists T(\cdot) \in$ **poly** such that $\forall P \in \mathcal{P}$ time(TM, P) $\leq T(|P|)$ then $\Pr(\text{valid}(P, TM(P)) = 1) \in$ **negl**$(|P|)$.

A model for a hard problem family.

PPTM: A Probabilistic Turing Machine running in Polynomial time.

Hardness Notions

Summary:

- **Easy:** There exists a TM that solves all problem instances in polynomial time.
- **Feasible:** There exists a PPTM that solves any problem instance with a non-negligible probability.
- **Infeasible:** All PPTM can only solve problem instances with a negligible probability.

In cryptography, the problem P is chosen at random among all problems with a specific size (e.g., the description of the problem instance contains a randomly generated public key).

Security as a Game

Security is modelled as a Game (random experiment) between a Challenger C and an Adversary A:

- C chooses a random problem instance of a given size
- A tries to find a solution in polynomial time
- A succeds only with a negligible probability

Here, the success probability is computed respect to

- The internal coin tosses of A
- The random choice of C

In general, the game can have more interaction steps.

Outline

One-Time Security for Symmetric Encryption

The adversary tries to decrypt a random ciphertext form scratch.

Challenger	$\leftarrow (\ell, \text{KeyGen}, \text{Enc}, \text{Dec}) \rightarrow$	Adversary
$k_* \leftarrow \text{KeyGen}(\ell)$		
$m_* \leftarrow \mathcal{M}$		
$c_* = \text{Enc}(k_*, m_*)$; **send**$(c_*)$		**receive**(c_*)
		???
receive(m')		**send**(m')
accept if $m' = m_*$		

It is too simplistic: the adversary does not know any message / ciphertext pair for the same key. (It is a one-time model.)

One-Time Security for Symmetric Encryption

The same, written in pseudocode style (we don't know what the adversary is really doing):

Experiment Exp-SE-OT$(\Pi, \mathcal{A}, \ell)$:
$k_* \leftarrow \mathsf{KeyGen}(\ell)$
$m_* \leftarrow \mathcal{M}$
$c_* \leftarrow \mathsf{Enc}(k_*, m_*)$
$m' \leftarrow \mathcal{A}(c_*)$
if $m' = m_*$ **output** 1; //$\mathcal{A}$ wins
else output 0;

One-Time Security for Symmetric Encryption

The same, written in pseudocode style (we don't know what the adversary is really doing):

Experiment Exp-SE-OT$(\Pi, \mathcal{A}, \ell)$:
$k_* \leftarrow \text{KeyGen}(\ell)$
$m_* \leftarrow \mathcal{M}$
$c_* \leftarrow \text{Enc}(k_*, m_*)$
$m' \leftarrow \mathcal{A}(c_*)$
if $m' = m_*$ **output** 1; //$\mathcal{A}$ wins
else output 0;

Definition (SE-OT)

The symmetric encryption scheme Π is SE-OT secure if for all PPTM $\mathcal{A}$

$$\Pr[\text{Exp-SE-OT}(\Pi, \mathcal{A}, \ell) = 1] \in \textbf{negl}(\ell)$$

One-Way Security for Public Key Encryption

Now, the adversary also receives the public key:
It can encrypt messages of its choice with the same target key (CPA model)

Challenger	$\leftarrow (\ell, \text{KeyGen}, \text{Enc}, \text{Dec}) \rightarrow$	Adversary
$(pk_*, sk_*) \leftarrow \text{KeyGen}(\ell)$		
$m_* \leftarrow \mathcal{M}$		
$c_* = \text{Enc}(pk_*, m_*)$; **send**$(pk_*, c_*)$		**receive**(pk_*, c_*)
		???
receive(m')		**send**(m')
accept if $m' = m_*$		

One-Way Security for Public Key Encryption

Experiment Exp-PKE-OW-CPA$(\Pi, \mathcal{A}, \ell)$:
$(pk_*, sk_*) \leftarrow \mathsf{KeyGen}(\ell)$
$m_* \leftarrow \mathcal{M}$
$c_* \leftarrow \mathsf{Enc}(pk_*, m_*)$
$m' \leftarrow \mathcal{A}(pk_*, c_*)$
if $m' = m_*$ **output** 1; //$\mathcal{A}$ wins
else output 0;

Definition (PKE-OW-CPA)

The public key encryption scheme Π is PKE-OW-CPA secure if for all PPTM $\mathcal{A}$

$$\Pr[\text{Exp-PKE-OW-CPA}(\Pi, \mathcal{A}, \ell) = 1] \in \mathbf{negl}(\ell)$$

One-Way Security for Public Key Encryption

Most proposed PKE schemes are conjectured to be OW-CPA secure, with a proper choice of the parameters.
E.g., ElGamal, RSA, Rabin, Paillier.

Regev's PKE is not OW-CPA secure because it encrypts a single bit (thus the adversary has easily a success probability of 1/2).

There is a known security proof showing that Rabin is PKE-OW-CPA secure if and only if factoring an RSA modulus (with $p, q = 3 \bmod 4$) is hard.

One-Way Security for Public Key Encryption

Most proposed PKE schemes are conjectured to be OW-CPA secure, with a proper choice of the parameters.
E.g., ElGamal, RSA, Rabin, Paillier.

Regev's PKE is not OW-CPA secure because it encrypts a single bit (thus the adversary has easily a success probability of 1/2).

There is a known security proof showing that Rabin is PKE-OW-CPA secure if and only if factoring an RSA modulus (with $p, q = 3 \bmod 4$) is hard.

PKE-OW-CPA security is still unrealistic when the adversary has some a priori knowledge about the target message m_*.

Semantic Security

The adversary chooses some leakage function f for m_*:

Challenger $\leftarrow (\ell, \text{KeyGen}, \text{Enc}, \text{Dec}) \rightarrow$ Adversary

---

$b \leftarrow \{0,1\}$
$(pk_*, sk_*) \leftarrow \text{KeyGen}(\ell)$
send(pk_*)

receive(pk_*)
???
send$(f(\cdot))$
(keep the internal state st)

receive$(f(\cdot))$
$m_0, m_1 \leftarrow \mathcal{M}$; $h = f(m_0)$
$c_* = \text{Enc}(pk_*, m_b)$; **send**$(c_*, h)$

receive(c_*, h)
???
send(b')

receive(b')
accept if $b' = b$

This is a two-stage game (the Adversary is split in two stages).

Indistinguishability of Ciphertexts

Semantic Security game is shown to be equivalent to:

Challenger $\leftarrow (\ell, \text{KeyGen}, \text{Enc}, \text{Dec}) \rightarrow$ Adversary

---

$b \leftarrow \{0,1\}$
$(pk_*, sk_*) \leftarrow \text{KeyGen}(\ell)$
send(pk_*)

receive(m_0, m_1)
$c_* = \text{Enc}(pk_*, m_b)$
send(c_*)

receive(b')
accept if $b' = b$

receive(pk_*)
???
send(m_0, m_1)
(keep the internal state *st*)
receive(c_*)
???
send(b')

---

m_0, m_1 must be valid messages. (Otherwise, the Challenger rejects).

Indistinguishability of Ciphertexts

Experiment Exp-PKE-IND-CPA$(\Pi, \mathcal{A}_1, \mathcal{A}_2, \ell)$:
$(pk_*, sk_*) \leftarrow \text{KeyGen}(\ell)$
$(m_0, m_1, st) \leftarrow \mathcal{A}_1(pk_*)$ // st is the internal state passing from $\mathcal{A}_1$ to $\mathcal{A}_2$
$b \leftarrow \{0, 1\}$
$c_* \leftarrow \text{Enc}(pk_*, m_b)$
$b' \leftarrow \mathcal{A}_2(st, c_*)$
if $b' = b$ **output** 1; //$\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins
else output 0;

Definition (PKE-IND-CPA)

The public key encryption scheme Π is PKE-IND-CPA secure if for all PPTM $\mathcal{A}$

$$|\Pr[\text{Exp-PKE-IND-CPA}(\Pi, \mathcal{A}, \ell) = 1] - 1/2| \in \textbf{negl}(\ell)$$

$1/2$ is the success probability of a dummy adversary!

IND-CPA Security

Some well-known PKE schemes are conjectured to be IND-CPA secure, with a proper choice of the parameters.
E.g., ElGamal, Paillier, Regev.

But RSA and Rabin do not achieve IND-CPA security.

IND-CPA Security

Some well-known PKE schemes are conjectured to be IND-CPA secure, with a proper choice of the parameters.
E.g., ElGamal, Paillier, Regev.

But RSA and Rabin do not achieve IND-CPA security.

Lemma

No deterministic PKE can achieve IND-CPA security.

$\mathcal{A}_2$ can just encrypt m_0 and compare the result with c_*.

IND-CPA Security

Some well-known PKE schemes are conjectured to be IND-CPA secure, with a proper choice of the parameters.
E.g., ElGamal, Paillier, Regev.

But RSA and Rabin do not achieve IND-CPA security.

Lemma

No deterministic PKE can achieve IND-CPA security.

$\mathcal{A}_2$ can just encrypt m_0 and compare the result with c_*.

RSA and Rabin need some randomization (e.g., a randomized message padding scheme).

One-Way Functions

PKE implies the existence of function families easy-to-compute (encryption) but hard-to-invert (decryption)

One-Way Functions

PKE implies the existence of function families easy-to-compute (encryption) but hard-to-invert (decryption)

Definition

A function family $\mathcal{F} = \{\mathcal{F}_\ell\}_{\ell \in \mathbb{Z}^+}$, $\mathcal{F}_\ell = \{f_k : \mathcal{X}_k \to \mathcal{Y}_k\}_{k \in \mathcal{K}_\ell}$ is **one-way** if it is efficiently computable, but for all PPTM $\mathcal{A}$

$$\Pr[\mathcal{A}(1^\ell, k, y) \in f_k^{-1}(y) \ : \ k \leftarrow \mathcal{K}_\ell;\ x \leftarrow \mathcal{X}_k;\ y \leftarrow f_k(x)] \in \mathbf{negl}(\ell)$$

One-Way Functions

PKE implies the existence of function families easy-to-compute (encryption) but hard-to-invert (decryption)

Definition

A function family $\mathcal{F} = \{\mathcal{F}_\ell\}_{\ell \in \mathbb{Z}^+}$, $\mathcal{F}_\ell = \{f_k : \mathcal{X}_k \to \mathcal{Y}_k\}_{k \in \mathcal{K}_\ell}$ is **one-way** if it is efficiently computable, but for all PPTM $\mathcal{A}$

$$\Pr[\mathcal{A}(1^\ell, k, y) \in f_k^{-1}(y) \ : \ k \leftarrow \mathcal{K}_\ell;\ x \leftarrow \mathcal{X}_k;\ y \leftarrow f_k(x)] \in \textbf{negl}(\ell)$$

- 'Efficiently computable' means that there is a PPTM Eval such that $\text{Eval}(k, x) = f_k(x)$
- If f_k is one-way then f'_k defined by $f'_k(x, r) = (f_k(x), r)$ is also one-way
- The sets $\mathcal{X}_k$ and $\mathcal{Y}_k$ must be of size superpolynomial in ℓ

Injective Trapdoor One-Way Functions

PKE also requires the existence of a **trapdoor** which knowledge renders the decryption function easy to compute.

Injective Trapdoor One-Way Functions

PKE also requires the existence of a **trapdoor** which knowledge renders the decryption function easy to compute.

Definition

An injective one-way function family $\mathcal{F} = \{\mathcal{F}_\ell\}_{\ell \in \mathbb{Z}^+}$, $\mathcal{F}_\ell = \{f_k : \mathcal{X}_k \to \mathcal{Y}_k\}_{k \in \mathcal{K}_\ell}$ is called **trapdoor one-way** if there exists a family of trapdoors $\mathcal{T} = \{\mathcal{T}_\ell\}_{\ell \in \mathbb{Z}^+}$, and two PPTM Sample and Inv such that

$$\Pr[\mathsf{Inv}(1^\ell, t, f_k(x)) = x \ : \ (k, t) \leftarrow \mathsf{Sample}(1^\ell);\ x \leftarrow \mathcal{X}_k] = 1$$

and $(k, t) \leftarrow \mathsf{Sample}(1^\ell)$ samples the uniform distribution in $\mathcal{K}_\ell$, and $t \in \mathcal{T}_\ell$

PKE From Injective TOW Functions

Let $\mathcal{F}$ be an injective trapdoor one-way function family.

KeyGen(ℓ) :
$(k, t) \leftarrow$ Sample(ℓ);
output (k, t);

Enc(k, m) :
output Eval(k, m);

Dec(t, c) :
output Inv(t, c);

PKE From Injective TOW Functions

Let $\mathcal{F}$ be an injective trapdoor one-way function family.

KeyGen(ℓ) :
$(k, t) \leftarrow$ Sample(ℓ);
output (k, t);

Enc(k, m) :
output Eval(k, m);

Dec(t, c) :
output Inv(t, c);

It is PKE-OW-CPA secure but not PKE-IND-CPA secure (because the encryption function is deterministic)

Hardcore Predicates of a One-Way Function

Let $\mathcal{F}$ be an injective one-way function family between the set families $\mathcal{X}$ and $\mathcal{Y}$. A family of predicates $\mathcal{H}$ (functions taking binary values) on $\mathcal{X}$ is hardcore for $\mathcal{F}$ if computing $h_k(x)$ from $f_k(x)$ is as hard as computing x from $f_k(x)$.

Hardcore Predicates of a One-Way Function

Let $\mathcal{F}$ be an injective one-way function family between the set families $\mathcal{X}$ and $\mathcal{Y}$. A family of predicates $\mathcal{H}$ (functions taking binary values) on $\mathcal{X}$ is hardcore for $\mathcal{F}$ if computing $h_k(x)$ from $f_k(x)$ is as hard as computing x from $f_k(x)$.

Examples:

- For an RSA public key $(n = pq, e)$, computing $LSB(x)$ from $x^e \mod n$ is as hard as computing x from $x^e \mod n$
- **Goldreich-Levin predicate:**

$$h(x, r) = x_1 r_1 + \ldots + x_n r_n \mod 2$$

is a hardcore predicate for $f'_k(x, r) = (f_k(x), r)$

PKE From Hardcore Predicates

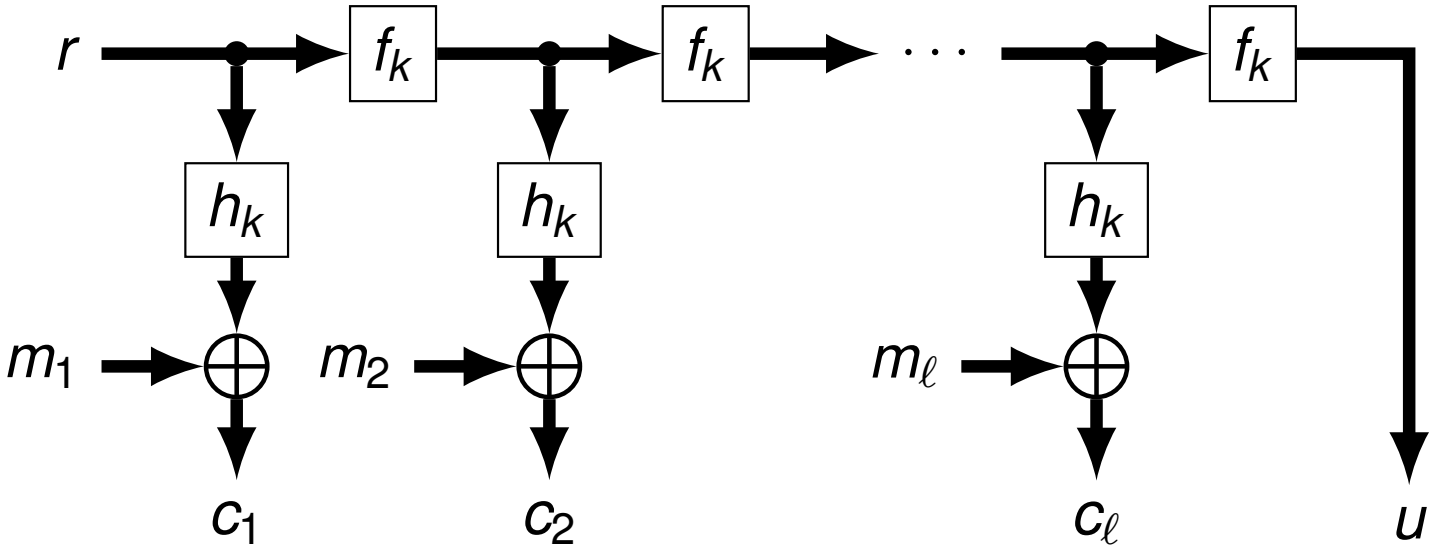
Trapdoor One-Way Permutation (TOWP) Family: A trapdoor one-way family of bijections $f_k : \mathcal{X}_k \to \mathcal{X}_k$
$\mathcal{H}$ hardcore predicate family for $\mathcal{F}$

KeyGen(ℓ) :
$(k, t) \leftarrow$ Sample($\kappa(\ell)$);
output (k, t);

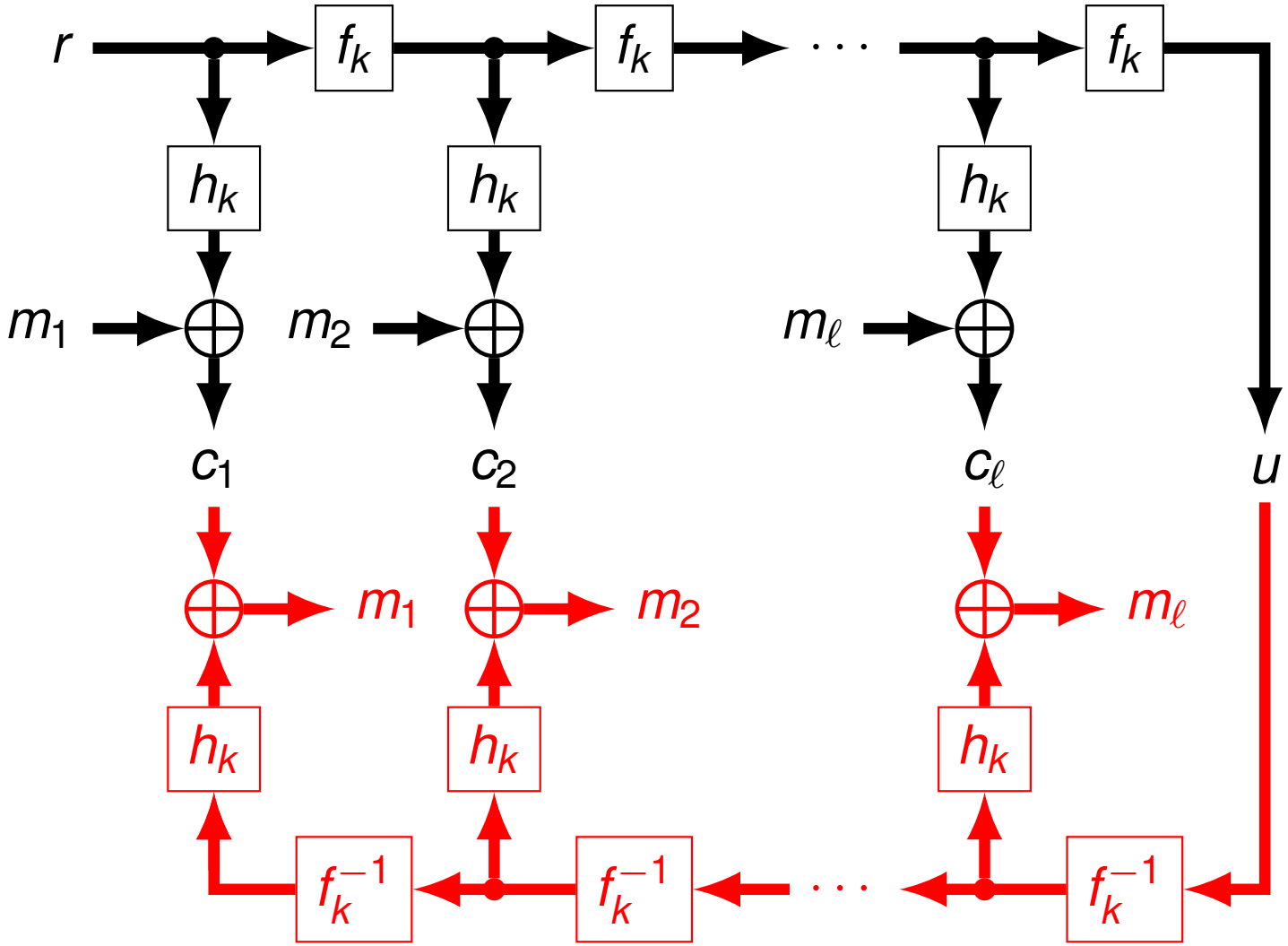
Enc(k, m) :
$(m_1, \ldots, m_\ell) = m$;
$r \leftarrow \mathcal{X}_k$;
$c_i \leftarrow m_i \oplus h_k(f_k^{i-1}(r)); \quad i = 1, \ldots, \ell$
output $(c_1, \ldots, c_\ell, f_k^\ell(r))$;

Dec(t, c) :
$(c_1, \ldots, c_\ell, u) = c$;
$m_i \leftarrow c_i \oplus h_k(f_k^{-(\ell-i+1)}(u)); \quad i = 1, \ldots, \ell$
output $m = (m_1, \ldots, m_\ell)$;

PKE From Hardcore Predicates



PKE From Hardcore Predicates



Beyond IND-CPA Security

Some realistic attacks fall outside the IND-CPA model.

The adversary has limited extra access to:

- An oracle that tells whether a (possibly manipulated) ciphertext is valid or not.
- An oracle that decrypts a (possibly manipulated) ciphertext.
- An oracle that decrypts a ciphertext related to the target one c_*.

Beyond IND-CPA Security

Some realistic attacks fall outside the IND-CPA model.

The adversary has limited extra access to:

- An oracle that tells whether a (possibly manipulated) ciphertext is valid or not.
- An oracle that decrypts a (possibly manipulated) ciphertext.
- An oracle that decrypts a ciphertext related to the target one c_*.

A maximal notion of security is defined: IND-CCA (for Chosen Ciphertext Attack).

The adversary can ask for decryptions of any possible ciphertext except for c_*.

IND-CCA Security

Challenger	$\leftarrow (\ell, \text{KeyGen}, \text{Enc}, \text{Dec}) \rightarrow$	Adversary
$(pk_*, sk_*) \leftarrow \text{KeyGen}(\ell)$ **send**(pk_*)		**receive**(pk_*)
	(many times)	???
receive(c_i) $m_i = \text{Dec}(sk_*, c_i)$; **send**$(m_i)$		**send**(c_i) // oracle call **receive**(m_i)
		???
receive(m_0, m_1) $b \leftarrow \{0,1\}$; $c_* = \text{Enc}(pk_*, m_b)$; **send**$(c_*)$		**send**(m_0, m_1) **receive**(c_*)
	(many times)	???
receive(c_i) **if** $c_i \neq c_*$; $m_i = \text{Dec}(sk_*, c_i)$; **send**$(m_i)$ **else abort**		**send**(c_i) // oracle call **receive**(m_i)
		???
receive(b') **accept if** $b' = b$		**send**(b')

IND-CCA Security

Experiment Exp-PKE-IND-CCA$(\Pi, \mathcal{A}_1, \mathcal{A}_2, \ell)$:
$(pk_*, sk_*) \leftarrow \text{KeyGen}(\ell)$
$(m_0, m_1, st) \leftarrow \mathcal{A}_1^{\mathcal{O}_1(\cdot)}(pk_*)$
$b \leftarrow \{0,1\}$
$c_* \leftarrow \text{Enc}(pk_*, m_b)$
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}_2(\cdot)}(st, c_*)$
if $b' = b$ **output** 1; // $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins
else output 0;
Oracle $\mathcal{O}_1(c)$ // Decryption oracle
return $\text{Dec}(sk_*, c)$
Oracle $\mathcal{O}_2(c)$ // Restricted decryption oracle
if $c \neq c_*$ **return** $\text{Dec}(sk_*, c)$ **else abort**

Definition (PKE-IND-CCA)

The PKE Π is PKE-IND-CCA secure if for all PPTM $\mathcal{A}$

$$|\Pr[\text{Exp-PKE-IND-CCA}(\Pi, \mathcal{A}, \ell) = 1] - 1/2| \in \textbf{negl}(\ell)$$

Known CCA Attacks

None of the previous PKE examples achieve IND-CCA security:

Known CCA Attacks

None of the previous PKE examples achieve IND-CCA security:

- **RSA** or **Rabin:** They are not even IND-CPA secure.

Known CCA Attacks

None of the previous PKE examples achieve IND-CCA security:

- **RSA** or **Rabin:** They are not even IND-CPA secure.
- **ElGamal:** The adversary can modify c_* in a number of ways and submit the result to the decryption oracle. E.g., a rerandomization of c_* will be accepted by the oracle, and it will answer m_b.

Known CCA Attacks

None of the previous PKE examples achieve IND-CCA security:

- **RSA** or **Rabin:** They are not even IND-CPA secure.
- **ElGamal:** The adversary can modify c_* in a number of ways and submit the result to the decryption oracle. E.g., a rerandomization of c_* will be accepted by the oracle, and it will answer m_b.
- **Paillier:** The same attack also applies.

Known CCA Attacks

None of the previous PKE examples achieve IND-CCA security:

- **RSA** or **Rabin:** They are not even IND-CPA secure.
- **ElGamal:** The adversary can modify c_* in a number of ways and submit the result to the decryption oracle. E.g., a rerandomization of c_* will be accepted by the oracle, and it will answer m_b.
- **Paillier:** The same attack also applies.
- **Hashed ElGamal:** XORing the second component of c_* with any mask z makes the oracle answer $m_b \oplus z$.

Known CCA Attacks

None of the previous PKE examples achieve IND-CCA security:

- **RSA** or **Rabin:** They are not even IND-CPA secure.
- **ElGamal:** The adversary can modify c_* in a number of ways and submit the result to the decryption oracle. E.g., a rerandomization of c_* will be accepted by the oracle, and it will answer m_b.
- **Paillier:** The same attack also applies.
- **Hashed ElGamal:** XORing the second component of c_* with any mask z makes the oracle answer $m_b \oplus z$.
- **Regev:** Simply adding $(q-1)/2$ to the second component of c_* makes the oracle answer $1 - m_b$.

Known CCA Attacks

With similar attacks, it is shown that:

Lemma

Any homomorphic PKE is IND-CCA insecure.

There is a tradeoff between security and functionality.

Known CCA Attacks

With similar attacks, it is shown that:

Lemma

Any homomorphic PKE is IND-CCA insecure.

There is a tradeoff between security and functionality.

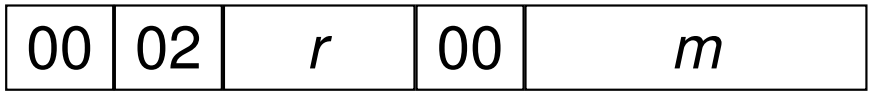
New methods are necessary to upgrade existing PKE to the IND-CCA security level.

- Randomized paddings, or Fujisaki-Okamoto Transforms (in the Random Oracle Model)
- New cryptographic tools (Hash Proof Systems, Canetti-Halevi-Katz Transform, . . .)

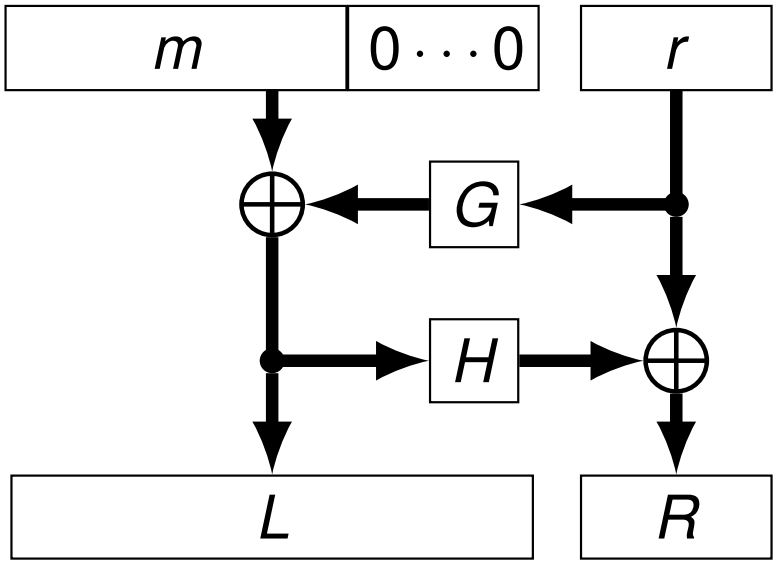
E.g., Fixing RSA PKE

- Plain RSA PKE is conjectured to be OW-CPA secure, but it is neither IND-CPA nor OW-CCA secure.
- A simple randomized message padding can give IND-CPA security, but not OW-CCA.
- Bleichenbacher's attack against PKCS#1 v1.5 shows a practical CCA attack against a padded version of RSA.
- RSA-OAEP (PKCS#1 v2) fixes the attack and provides IND-CCA security for RSA (in the Random Oracle Model).

PKCS#1 v1.5 vs. v2.0 Message Encodings



PKCS#1 v1.5



PKCS#1 v2.0

Outline

Universal Unforgeability (UF)

The basic security notion for signatures: forge a valid signature for any given message only from the public key:

Challenger	$\leftarrow (\ell, \text{KeyGen}, \text{Sig}, \text{Ver}) \rightarrow$	Adversary
$(pk_*, sk_*) \leftarrow \text{KeyGen}(\ell)$		
$m_* \leftarrow \mathcal{M}$		
send(pk_*, m_*)		**receive**(pk_*, m_*)
		???
receive(s_*)		**send**(s_*)
accept if $\text{Ver}(pk_*, m_*, s_*) = 1$		

Universal Unforgeability (UF)

The basic security notion for signatures: forge a valid signature for any given message only from the public key:

Challenger	$\leftarrow (\ell, \text{KeyGen}, \text{Sig}, \text{Ver}) \rightarrow$	Adversary
$(pk_*, sk_*) \leftarrow \text{KeyGen}(\ell)$		
$m_* \leftarrow \mathcal{M}$		
send(pk_*, m_*)		**receive**(pk_*, m_*)
		???
receive(s_*)		**send**(s_*)
accept if $\text{Ver}(pk_*, m_*, s_*) = 1$		

Too simple: a real adversary can learn some valid pairs message/signature for the target public key.

Universal Unforgeability (UF)

The basic security notion for signatures: forge a valid signature for any given message only from the public key:

Challenger	$\leftarrow (\ell, \text{KeyGen}, \text{Sig}, \text{Ver}) \rightarrow$	Adversary
$(pk_*, sk_*) \leftarrow \text{KeyGen}(\ell)$		
$m_* \leftarrow \mathcal{M}$		
send(pk_*, m_*)		**receive**(pk_*, m_*)
		???
receive(s_*)		**send**(s_*)
accept if $\text{Ver}(pk_*, m_*, s_*) = 1$		

Too simple: a real adversary can learn some valid pairs message/signature for the target public key.

Actually, the random selection of m_* is not well defined!.

UF-RMA Security

In UF-RMA security, the adversary can ask for valid signatures on random messages.

Experiment Exp-UF-RMA$(\Sigma, \mathcal{A}, \ell)$:
  $(pk_*, sk_*) \leftarrow \text{KeyGen}(\ell)$
  $m_* \leftarrow \mathcal{M}$
  $s_* \leftarrow \mathcal{A}^{\mathcal{O}()}(pk_*, m_*)$
  if $\text{Ver}(pk_*, m_*, s_*) = 1$ **output** 1; // $\mathcal{A}$ wins
  else output 0;
Oracle $\mathcal{O}()$ // Signing a random message oracle
  $m \leftarrow \mathcal{M}$
  return $(m, \text{Sig}(sk_*, m))$

UF-RMA Security

In UF-RMA security, the adversary can ask for valid signatures on random messages.

Experiment Exp-UF-RMA$(\Sigma, \mathcal{A}, \ell)$:
  $(pk_*, sk_*) \leftarrow \mathsf{KeyGen}(\ell)$
  $m_* \leftarrow \mathcal{M}$
  $s_* \leftarrow \mathcal{A}^{\mathcal{O}()}(pk_*, m_*)$
  if $\mathsf{Ver}(pk_*, m_*, s_*) = 1$ **output** 1; // $\mathcal{A}$ wins
  else output 0;
Oracle $\mathcal{O}()$ // Signing a random message oracle
  $m \leftarrow \mathcal{M}$
  return $(m, \mathsf{Sig}(sk_*, m))$

Definition (Sig-UF-RMA)

The signature scheme Σ is UF-RMA secure if for all PPTM $\mathcal{A}$

$$\Pr[\text{Exp-UF-RMA}(\Sigma, \mathcal{A}, \ell) = 1] \in \mathbf{negl}(\ell)$$

UF-RMA Security

The given examples of signature schemes (RSA-FDH, Pointcheval-Stern, DSA and ECDSA) are UF-RMA secure, in the Random Oracle Model.

UF-RMA Security

The given examples of signature schemes (RSA-FDH, Pointcheval-Stern, DSA and ECDSA) are UF-RMA secure, in the Random Oracle Model.

Still too simple (even plain RSA signature is UF-RMA secure!): an adversary being able to produce a signature for some specific messages, and not all, can be considered successful.

Again, the random selection of m_* and m are not well defined!.

UF-RMA Security

The given examples of signature schemes (RSA-FDH, Pointcheval-Stern, DSA and ECDSA) are UF-RMA secure, in the Random Oracle Model.

Still too simple (even plain RSA signature is UF-RMA secure!): an adversary being able to produce a signature for some specific messages, and not all, can be considered successful.

Again, the random selection of m_* and m are not well defined!.

Improvement: Make the adversary choose the target message to be signed.

Existential Unforgeability (EF)

Experiment Exp-EF-RMA$(\Sigma, \mathcal{A}, \ell)$:
  $(pk_*, sk_*) \leftarrow \mathsf{KeyGen}(\ell)$
  $L = \emptyset$
  $(m_*, s_*) \leftarrow \mathcal{A}^{\mathcal{O}()}(pk_*)$
  if $m_* \notin L$ **and** $\mathsf{Ver}(pk_*, m_*, s_*) = 1$ **output** 1; // $\mathcal{A}$ wins
  else output 0;
Oracle $\mathcal{O}()$ // Signing a random message oracle
  $m \leftarrow \mathcal{M}$
  $L = L \cup \{m\}$
  return $(m, \mathsf{Sig}(sk_*, m))$

Definition (Sig-EF-RMA)

The signature scheme Σ is EF-RMA secure if for all PPTM $\mathcal{A}$

$$\Pr[\text{Exp-EF-RMA}(\Sigma, \mathcal{A}, \ell) = 1] \in \mathbf{negl}(\ell)$$

We need to maintain the list of messages signed by the oracle to exclude trivial attacks.

Beyond RMA Security

Plain RSA and ElGamal signatures are EF-RMA insecure, while RSA-FDH, Pointcheval-Stern, DSA and ECDSA are EF-RMA secure.

Beyond RMA Security

Plain RSA and ElGamal signatures are EF-RMA insecure, while RSA-FDH, Pointcheval-Stern, DSA and ECDSA are EF-RMA secure.

Still, the random selection of m in the RMA oracle is not well defined.

In a practical setting, there exist ways to inject some messages to be signed by a honest signer, and this is a type of attack that is outside the RMA model.

Beyond RMA Security

Plain RSA and ElGamal signatures are EF-RMA insecure, while RSA-FDH, Pointcheval-Stern, DSA and ECDSA are EF-RMA secure.

Still, the random selection of m in the RMA oracle is not well defined.

In a practical setting, there exist ways to inject some messages to be signed by a honest signer, and this is a type of attack that is outside the RMA model.

Improvement: Allow the adversary choose the messages to be signed by the oracle.

EF-CMA Security

Experiment Exp-EF-CMA$(\Sigma, \mathcal{A}, \ell)$:
  $(pk_*, sk_*) \leftarrow \mathsf{KeyGen}(\ell)$
  $L = \emptyset$
  $(m_*, s_*) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(pk_*)$
  if $m_* \notin L$ **and** $\mathsf{Ver}(pk_*, m_*, s_*) = 1$ **output** 1; // $\mathcal{A}$ wins
  else output 0;
Oracle $\mathcal{O}(m)$ // Signing oracle
  $L = L \cup \{m\}$
  return $(m, \mathsf{Sig}(sk_*, m))$

Definition (Sig-EF-CMA)

The signature scheme Σ is EF-CMA secure if for all PPTM $\mathcal{A}$

$$\Pr[\text{Exp-EF-CMA}(\Sigma, \mathcal{A}, \ell) = 1] \in \mathbf{negl}(\ell)$$

EF-CMA Security

EF-CMA is nowadays considered the standard security level for a digital signature scheme.

RSA-FDH, Pointcheval-Stern, DSA and ECDSA achieve EF-CMA security in the Random Oracle Model.

There are other signature schemes (e.g., Cramer-Shoup) that also achieve EF-CMA security without random oracles.

Outline

Assumptions Related to Integer Factorization

- **IF:** For random ℓ-bit long primes p, q, **given** $n = pq$, **compute** p or q.

Assumptions Related to Integer Factorization

- **IF:** For random ℓ-bit long primes p, q, **given** $n = pq$, **compute** p or q.
- **RSA:** For random ℓ-bit long primes p, q, $n = pq$, a random e coprime with $\phi(n)$ and a random $m \in \mathbb{Z}_n$, **given** n, e and $m^e \bmod n$, **compute** m.

Assumptions Related to Integer Factorization

- **IF:** For random ℓ-bit long primes p, q, **given** $n = pq$, **compute** p or q.
- **RSA:** For random ℓ-bit long primes p, q, $n = pq$, a random e coprime with $\phi(n)$ and a random $m \in \mathbb{Z}_n$, **given** n, e and $m^e \bmod n$, **compute** m.
- e-**RSA:** For random ℓ-bit long primes p, q such that e does not divide $p-1$ or $q-1$, $n = pq$ and a random $m \in \mathbb{Z}_n$, **given** n and $m^e \bmod n$, **compute** m.

Assumptions Related to Integer Factorization

- **IF:** For random ℓ-bit long primes p, q, **given** $n = pq$, **compute** p or q.
- **RSA:** For random ℓ-bit long primes p, q, $n = pq$, a random e coprime with $\phi(n)$ and a random $m \in \mathbb{Z}_n$, **given** n, e and $m^e \bmod n$, **compute** m.
- e-**RSA:** For random ℓ-bit long primes p, q such that e does not divide $p-1$ or $q-1$, $n = pq$ and a random $m \in \mathbb{Z}_n$, **given** n and $m^e \bmod n$, **compute** m.
- **DCR:** For random ℓ-bit long primes p, q, $n = pq$ and a random $r \in \mathbb{Z}_{n^2}$, **tell apart** the two probability distributions (n, r) and $(n, r^n \bmod n^2)$.

The last problem is called a **decision** problem, while the other three are **search** problems.

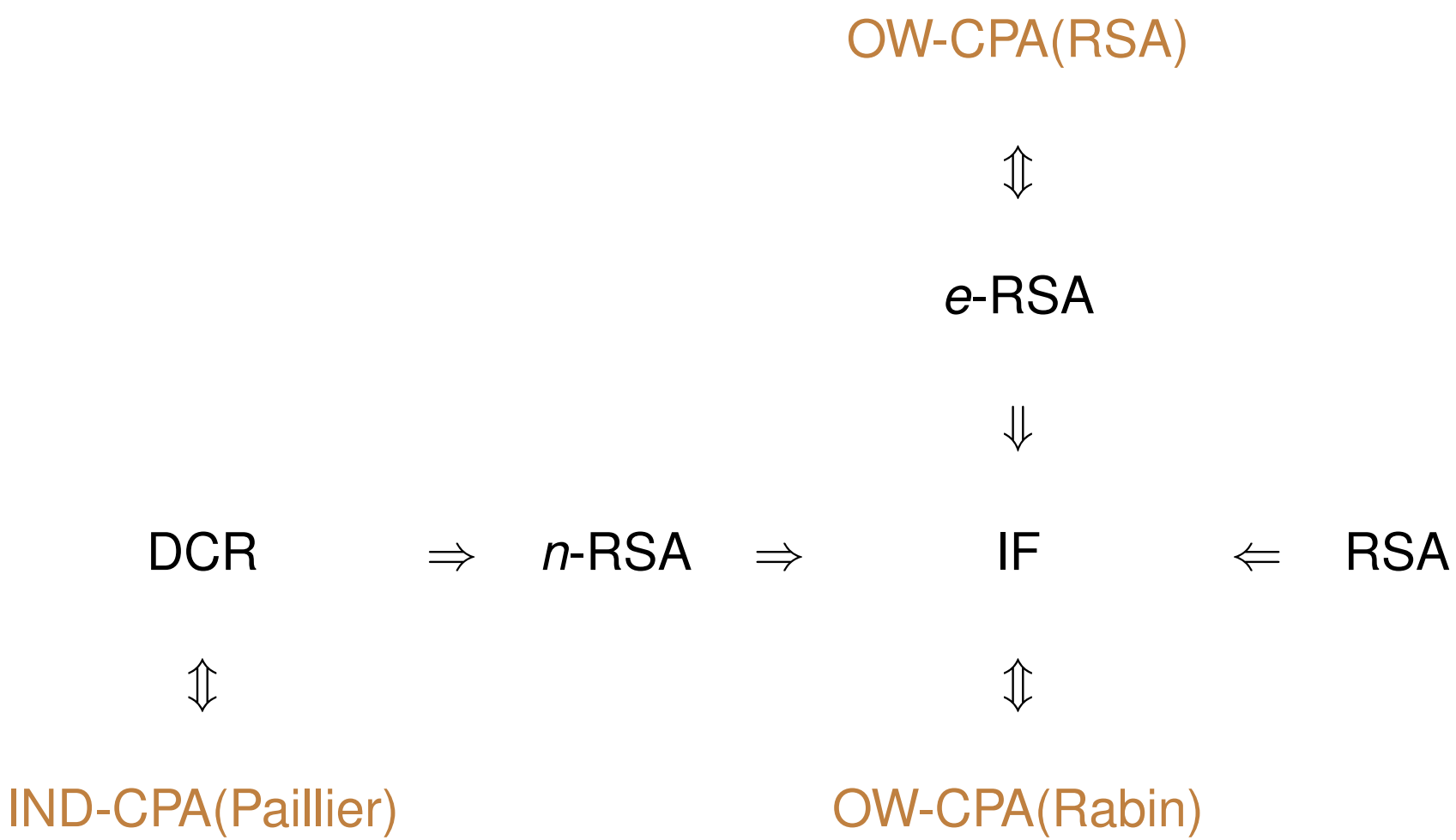
Assumptions Related to Integer Factorization

- **IF:** For random ℓ-bit long primes p, q, **given** $n = pq$, **compute** p or q.
- **RSA:** For random ℓ-bit long primes p, q, $n = pq$, a random e coprime with $\phi(n)$ and a random $m \in \mathbb{Z}_n$, **given** n, e and $m^e \bmod n$, **compute** m.
- e-**RSA:** For random ℓ-bit long primes p, q such that e does not divide $p-1$ or $q-1$, $n = pq$ and a random $m \in \mathbb{Z}_n$, **given** n and $m^e \bmod n$, **compute** m.
- **DCR:** For random ℓ-bit long primes p, q, $n = pq$ and a random $r \in \mathbb{Z}_{n^2}$, **tell apart** the two probability distributions (n, r) and $(n, r^n \bmod n^2)$.

The last problem is called a **decision** problem, while the other three are **search** problems.

All these problems are conjectured hard.

Known Hardness Implications



Assumptions Related to Discrete Logarithm

For a ℓ-bit long prime q, a cyclic group G of order q and a generator g:

- **DLOG:** For random $x \in \mathbb{Z}_q$, **given** g^x, **compute** x.

Assumptions Related to Discrete Logarithm

For a ℓ-bit long prime q, a cyclic group G of order q and a generator g:

- **DLOG:** For random $x \in \mathbb{Z}_q$, **given** g^x, **compute** x.
- **CDH:** For random $x, y \in \mathbb{Z}_q$, **given** g^x and g^y, **compute** g^{xy}.

Assumptions Related to Discrete Logarithm

For a ℓ-bit long prime q, a cyclic group G of order q and a generator g:

- **DLOG:** For random $x \in \mathbb{Z}_q$, **given** g^x, **compute** x.
- **CDH:** For random $x, y \in \mathbb{Z}_q$, **given** g^x and g^y, **compute** g^{xy}.
- **DDH:** For random $x, y, z \in \mathbb{Z}_q$, **tell apart** the two probability distributions (g^x, g^y, g^z) and (g^x, g^y, g^{xy}).

Assumptions Related to Discrete Logarithm

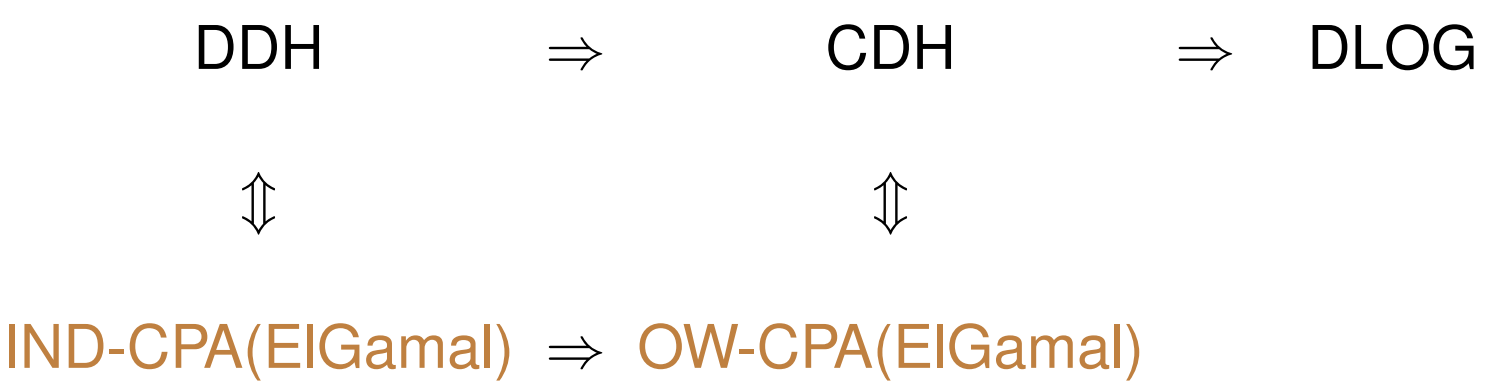
For a ℓ-bit long prime q, a cyclic group G of order q and a generator g:

- **DLOG:** For random $x \in \mathbb{Z}_q$, **given** g^x, **compute** x.
- **CDH:** For random $x, y \in \mathbb{Z}_q$, **given** g^x and g^y, **compute** g^{xy}.
- **DDH:** For random $x, y, z \in \mathbb{Z}_q$, **tell apart** the two probability distributions (g^x, g^y, g^z) and (g^x, g^y, g^{xy}).

All these problems are conjectured hard on suitable groups like:

- Subgroups of the multiplicative group of a large enough finite field.
- Subgroups of an elliptic curve over a large enough finite field.

Known Hardness Implications

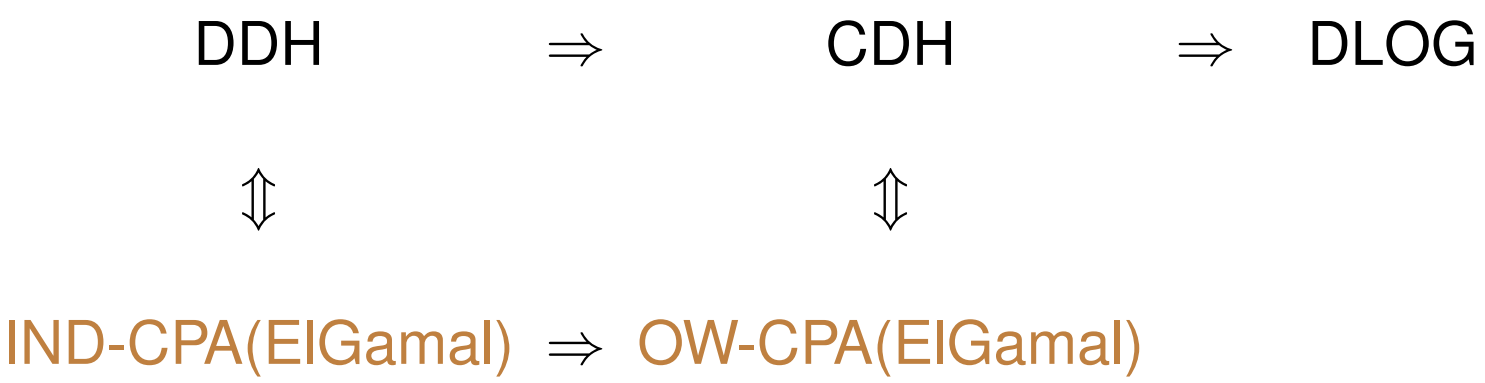


Known Hardness Implications

$$\begin{array}{ccccc} \text{DDH} & \Rightarrow & \text{CDH} & \Rightarrow & \text{DLOG} \\ \Updownarrow & & \Updownarrow & & \\ \text{IND-CPA(ElGamal)} & \Rightarrow & \text{OW-CPA(ElGamal)} & & \end{array}$$

In some special groups (e.g., elliptic curves with efficient pairing maps) DDH is easy while CDH is still conjectured to be hard.

Known Hardness Implications



In some special groups (e.g., elliptic curves with efficient pairing maps) DDH is easy while CDH is still conjectured to be hard.

Secure variants of ElGamal encryption and new encryption and signature schemes with new features have been proposed in these special groups (pairing based cryptography).

Further Assumptions

There exist other settings that offer well-known problems also conjectured hard, that can be used to build cryptographic protocols:

- Lattices
- Coding Theory
- Multivariate Polynomials
- Isogenies
- Non-abelian Groups

Further Assumptions

There exist other settings that offer well-known problems also conjectured hard, that can be used to build cryptographic protocols:

- Lattices
- Coding Theory
- Multivariate Polynomials
- Isogenies
- Non-abelian Groups

Some of the problems could remain hard even in the presence of quantum computers.

CRYE 6138 Security Models

Jorge L. Villar

UBa Cyber Crypto Center, Fall 2025

Computational Security Notions

—END—