A (Second) Introduction to Provable Security

Pooya Farshim

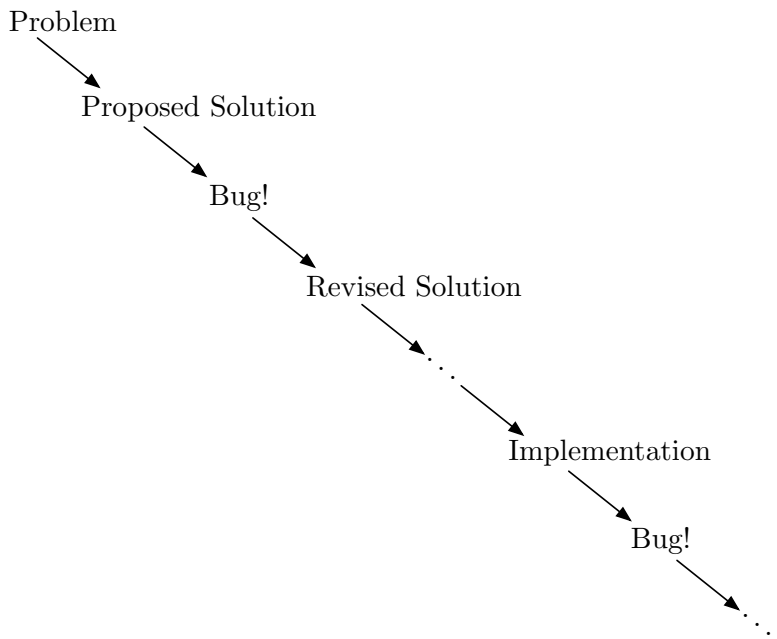
École Normale Supérieure

Summer School on Cryptology Crypto-CO

Defining Cryptography

The Oxford English Dictionary defines:

> **cryptography**, *n.* **1.** The art or practice of writing in code or cipher;



Source: Bellare & Rogaway. Introduction to Modern Cryptography. 2005.

Defining Cryptography

The entry continues

> the science of encryption; the branch of cryptology concerned with this (cf. cryptanalysis *n.*). More generally: the study of codes and ciphers; cryptology.

Defining Cryptography

The entry continues

> the science of encryption; the branch of cryptology concerned with this (cf. cryptanalysis *n.*). More generally: the study of codes and ciphers; cryptology.

A broader definition that's perhaps better suited for us:

> **cryptology**: the scientific/mathematical study of systems in the presence of adversarial behavior.

Mathematical Approach to Cryptography

Goldwasser and Micali (1982) initiated:

The Definition-Theorem-Proof approach in Cryptography.

Mathematical Approach to Cryptography

Goldwasser and Micali (1982) initiated:

The Definition-Theorem-Proof approach in Cryptography.

This consists of:

(1) Specifications of **Syntax** and **Correctness** of a protocol.

Mathematical Approach to Cryptography

Goldwasser and Micali (1982) initiated:

The Definition-Theorem-Proof approach in Cryptography.

This consists of:

(1) Specifications of **Syntax** and **Correctness** of a protocol.

(2) Formulation of a precise **Definition of Security**.

Mathematical Approach to Cryptography

Goldwasser and Micali (1982) initiated:

The Definition-Theorem-Proof approach in Cryptography.

This consists of:

(1) Specifications of **Syntax** and **Correctness** of a protocol.

(2) Formulation of a precise **Definition of Security**.

(3) Statement of **Assumptions** (as instances of (1) and (2)).

Mathematical Approach to Cryptography

Goldwasser and Micali (1982) initiated:

The Definition-Theorem-Proof approach in Cryptography.

This consists of:

(1) Specifications of **Syntax** and **Correctness** of a protocol.

(2) Formulation of a precise **Definition of Security**.

(3) Statement of **Assumptions** (as instances of (1) and (2)).

(4) A **Proof** that an instance/construction of (1) is secure wrt. definition (2) relative to assumptions (3).

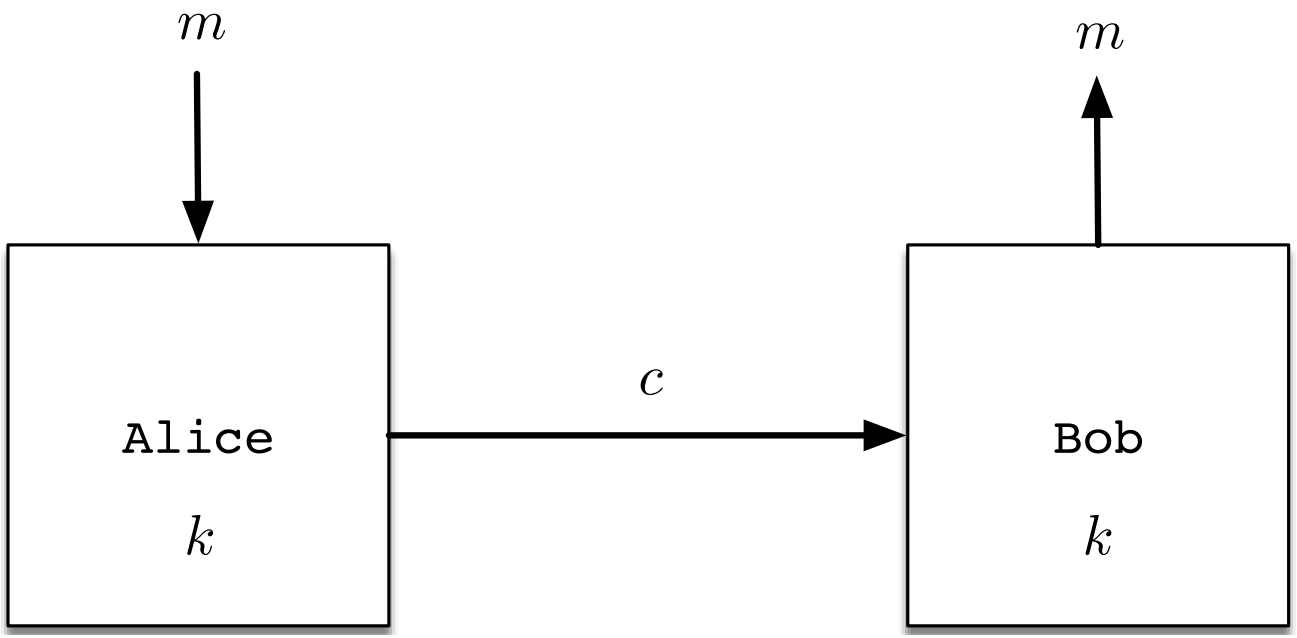
(Roughly reflects the typical structure of a crypto paper.)

Syntax and Correctness

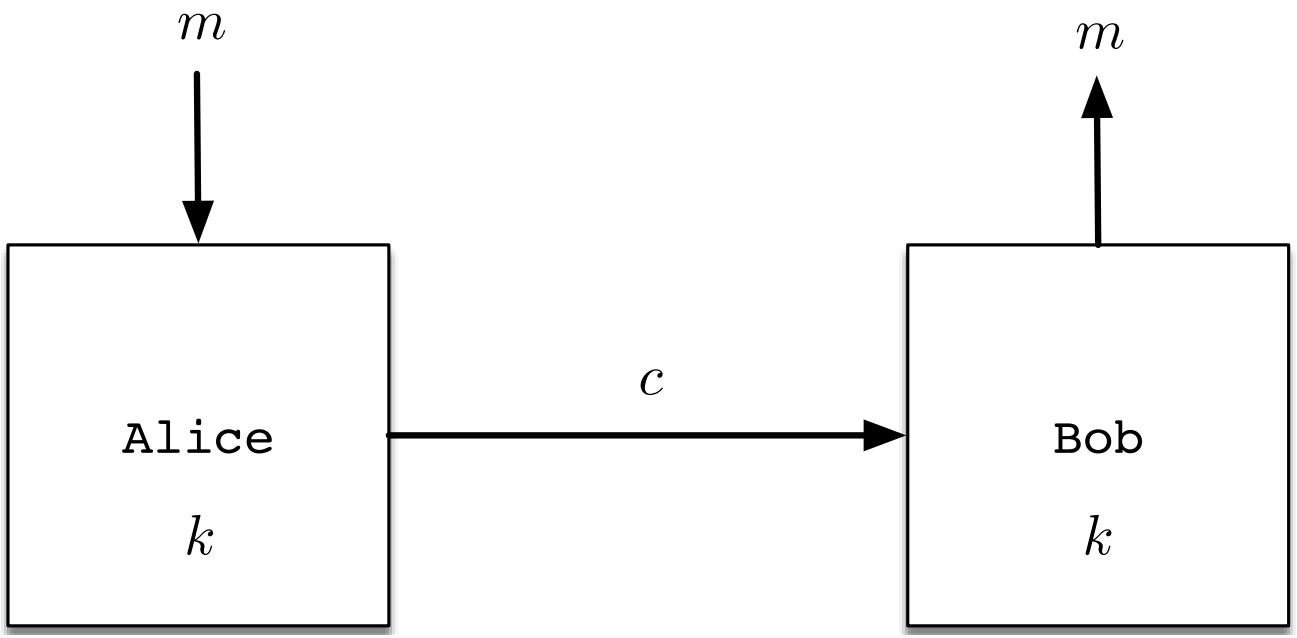
(1) Syntax and Correctness.

Symmetric Encryption

Symmetric Encryption



Symmetric Encryption



Formalize available I/O via three algorithms

$$\mathsf{SE} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$$

and say that Dec undoes Enc.

Syntax of Symmetric Encryption

Syntax of Symmetric Encryption

A symmetric encryption scheme is a triple of efficient algorithms:

1. $\underline{\mathsf{Gen}(\lambda)}$: A randomized algorithm that outputs a key k.

Syntax of Symmetric Encryption

A symmetric encryption scheme is a triple of efficient algorithms:

1. $\underline{\mathsf{Gen}(\lambda)}$: A randomized algorithm that outputs a key k.
2. $\underline{\mathsf{Enc}(k, m)}$: A randomized algorithm that on input a key k and a message m, outputs a ciphertext c.

Syntax of Symmetric Encryption

A symmetric encryption scheme is a triple of efficient algorithms:

1. $\underline{\mathsf{Gen}(\lambda)}$: A randomized algorithm that outputs a key k.
2. $\underline{\mathsf{Enc}(k, m)}$: A randomized algorithm that on input a key k and a message m, outputs a ciphertext c.
3. $\underline{\mathsf{Dec}(k, m)}$: A deterministic algorithm that on input a key k and a ciphertext c, outputs a message m or the special error symbol $\perp$.

Syntax of Symmetric Encryption

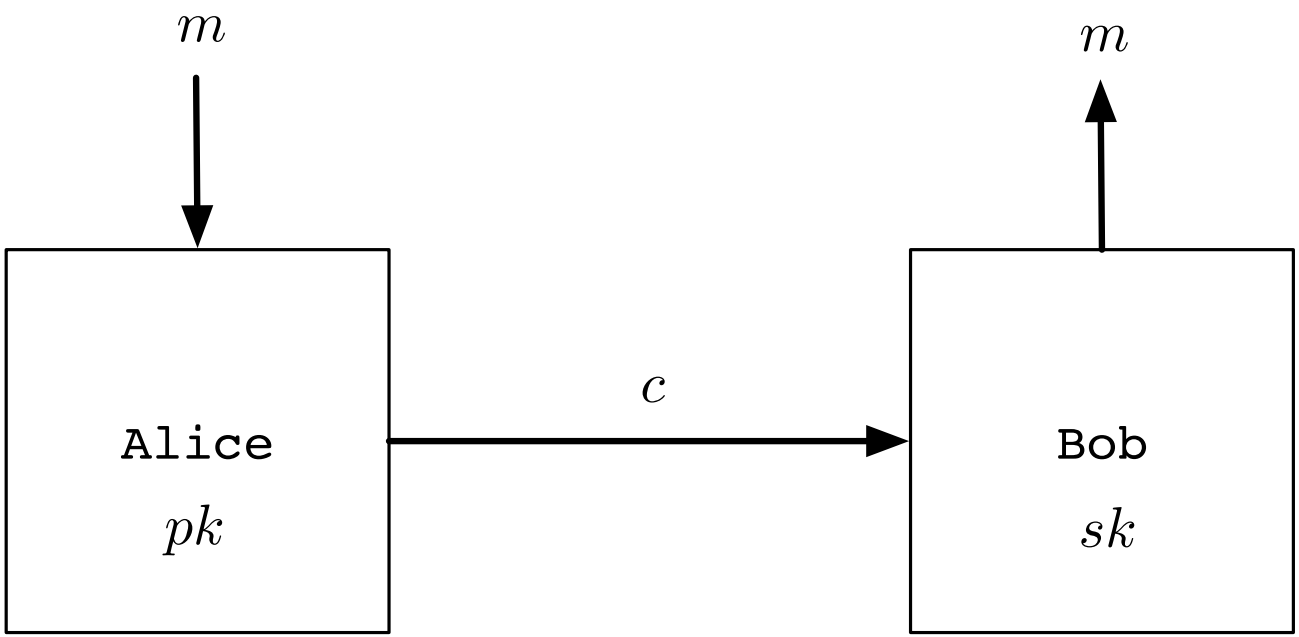
A symmetric encryption scheme is a triple of efficient algorithms:

1. $\underline{\mathsf{Gen}(\lambda)}$: A randomized algorithm that outputs a key k.
2. $\underline{\mathsf{Enc}(k, m)}$: A randomized algorithm that on input a key k and a message m, outputs a ciphertext c.
3. $\underline{\mathsf{Dec}(k, m)}$: A deterministic algorithm that on input a key k and a ciphertext c, outputs a message m or the special error symbol $\perp$.

Correctness: For any $\lambda \in \mathbb{N}$, any $k \leftarrow_{\$} \mathsf{Gen}(\lambda)$, and any $m \in \{0,1\}^*$

$$\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m \ .$$

Public-Key Encryption



SE := (Gen, Enc, Dec)

Syntax of Public-Key Encryption

A public-key encryption scheme is (also) a triple of efficient algorithms:

1. $\underline{\mathsf{Gen}(\lambda)}$: A randomized algorithm that outputs a key pair (sk, pk).

Syntax of Public-Key Encryption

A public-key encryption scheme is (also) a triple of efficient algorithms:

1. $\underline{\mathsf{Gen}(\lambda)}$: A randomized algorithm that outputs a key pair (sk, pk).
2. $\underline{\mathsf{Enc}(pk, m)}$: A randomized algorithm that on input a public key pk and a message m outputs a ciphertext c.

Syntax of Public-Key Encryption

A public-key encryption scheme is (also) a triple of efficient algorithms:

1. $\underline{\mathsf{Gen}(\lambda)}$: A randomized algorithm that outputs a key pair (sk, pk).
2. $\underline{\mathsf{Enc}(pk, m)}$: A randomized algorithm that on input a public key pk and a message m outputs a ciphertext c.
3. $\underline{\mathsf{Dec}(sk, m)}$: A deterministic algorithm that on input a secret key sk and a ciphertext c outputs a message m or error symbol $\perp$.

Correctness: For any $\lambda \in \mathbb{N}$, any $(sk, pk) \leftarrow_{\$} \mathsf{Gen}(\lambda)$, any $m \in \{0,1\}^*$

$$\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m \ .$$

Syntax

- We cannot talk about what an object is supposed to do before defining it.
- Syntax and correctness clarify what the problem at hand is.
  - Available interfaces
  - The input/output behavior
  - The Expected functionality
- It sets the ground for security definitions and proofs.

(2) Security Definitions.

When Is a Symmetric Encryption Secure?

Attempt I: A symmetric encryption scheme is secure if

it's impossible to find the key given a ciphertext.

Consider the following SE scheme:

When Is a Symmetric Encryption Secure?

Attempt I: A symmetric encryption scheme is secure if

it's impossible to find the key given a ciphertext.

Consider the following SE scheme:

- Gen(λ): Return a λ-bit random string k.

When Is a Symmetric Encryption Secure?

Attempt I: A symmetric encryption scheme is secure if

it's impossible to find the key given a ciphertext.

Consider the following SE scheme:

- Gen(λ): Return a λ-bit random string k.
- Enc(k, m): Return $c := m$ (in the clear!)

When Is a Symmetric Encryption Secure?

Attempt I: A symmetric encryption scheme is secure if

it's impossible to find the key given a ciphertext.

Consider the following SE scheme:

- Gen(λ): Return a λ-bit random string k.
- Enc(k, m): Return $c := m$ (in the clear!)
- Dec(k, c): Return $m := c$.

Clearly correct.

When Is a Symmetric Encryption Secure?

Attempt I: A symmetric encryption scheme is secure if

it's impossible to find the key given a ciphertext.

Consider the following SE scheme:

- Gen(λ): Return a λ-bit random string k.
- Enc(k, m): Return $c := m$ (in the clear!)
- Dec(k, c): Return $m := c$.

Clearly correct. Also secure:

It's indeed impossible to find k from c (or even many c's).

Security of Symmetric Encryption

Attempt II: I didn't mean that; I meant:

it's impossible to find the message given a ciphertext.

Security of Symmetric Encryption

Attempt II: I didn't mean that; I meant:

it's impossible to find the message given a ciphertext.

Consider the SE scheme:

Security of Symmetric Encryption

Attempt II: I didn't mean that; I meant:

it's impossible to find the message given a ciphertext.

Consider the SE scheme:

- Gen(λ): Return a λ-bit random string k.

Security of Symmetric Encryption

Attempt II: I didn't mean that; I meant:

it's impossible to find the message given a ciphertext.

Consider the SE scheme:

- Gen(λ): Return a λ-bit random string k.
- Enc($k, m_1|m_2$): Return $c_1 := m_1$ and $c_2 := m_2 \oplus k$.

Security of Symmetric Encryption

Attempt II: I didn't mean that; I meant:

it's impossible to find the message given a ciphertext.

Consider the SE scheme:

- Gen(λ): Return a λ-bit random string k.
- Enc($k, m_1|m_2$): Return $c_1 := m_1$ and $c_2 := m_2 \oplus k$.
- Dec($k, c_1|c_2$): Return $m_1 := c_1$ and $m_2 := c_2 \oplus k$.

Security of Symmetric Encryption

Attempt II: I didn't mean that; I meant:

it's impossible to find the message given a ciphertext.

Consider the SE scheme:

- Gen(λ): Return a λ-bit random string k.
- Enc($k, m_1|m_2$): Return $c_1 := m_1$ and $c_2 := m_2 \oplus k$.
- Dec($k, c_1|c_2$): Return $m_1 := c_1$ and $m_2 := c_2 \oplus k$.

Also secure:

Impossible to find m_2 from (a single!) (c_1, c_2).

But the left half of the message leaks in the clear.

Security of Symmetric Encryption

Attempt III: I actually meant:

> it's impossible to find **any information** about the plaintext.

Security of Symmetric Encryption

Attempt III: I actually meant:

> it's impossible to find **any information** about the plaintext.

Consider the **one-time pad** scheme:

- $\mathsf{Gen}(\lambda)$: Return a λ-bit random string k.
- $\mathsf{Enc}(k, m)$: Return $c := m \oplus k$.
- $\mathsf{Dec}(k, c)$: Return $m := c \oplus k$.

Security of Symmetric Encryption

Attempt III: I actually meant:

it's impossible to find **any information** about the plaintext.

Consider the **one-time pad** scheme:

- Gen(λ): Return a λ-bit random string k.
- Enc(k, m): Return $c := m \oplus k$.
- Dec(k, c): Return $m := c \oplus k$.

This scheme is secure:

Impossible to find m from c as c is random.

But given $m \oplus k$ and $m' \oplus k$, one can learn $m \oplus m'$.

Security of Symmetric Encryption

Attempt IV: Allow many ciphertexts:

it's impossible to find any info. about m given multiple ciphertexts.

Security of Symmetric Encryption

Attempt IV: Allow many ciphertexts:

it's impossible to find any info. about m given multiple ciphertexts.

Can be shown it's **impossible** to meet this notion of security.

Security of Symmetric Encryption

Attempt IV: Allow many ciphertexts:

it's impossible to find any info. about m given multiple ciphertexts.

Can be shown it's **impossible** to meet this notion of security.

Attempt V: **Don't ignore the complexity of attacks**:

it's **infeasible** to find any info. about m given multiple ciphertexts.

Still not possible: can be extremely lucky and guess m...

Security of Symmetric Encryption

Attempt IV: Allow many ciphertexts:

it's impossible to find any info. about m given multiple ciphertexts.

Can be shown it's **impossible** to meet this notion of security.

Attempt V: **Don't ignore the complexity of attacks**:

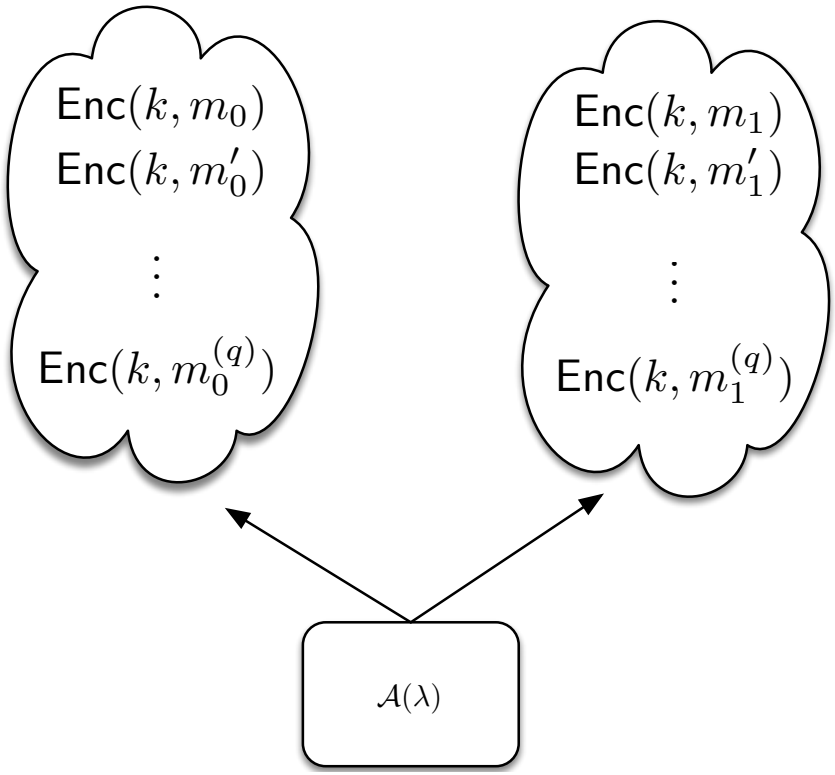
it's **infeasible** to find any info. about m given multiple ciphertexts.

Still not possible: can be extremely lucky and guess m...

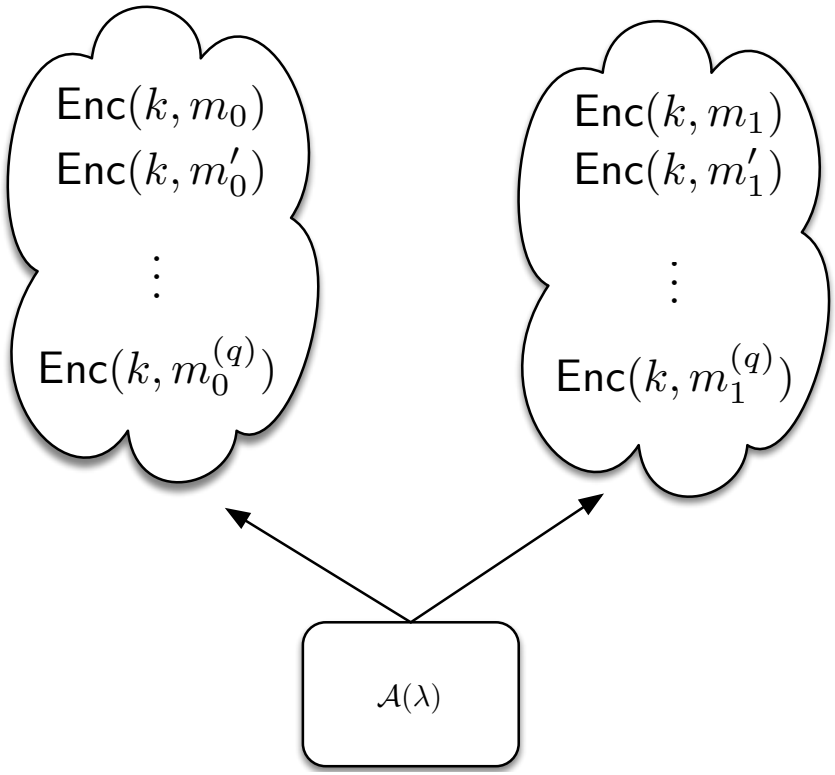
Attempt VI: A final tweak:

it's **infeasible** to find **any info.** about m
given **multiple** ciphertext with good **probability**.

Towards Formalizing Security



Towards Formalizing Security



Ciphertexts look indistinguishable.

Experiments

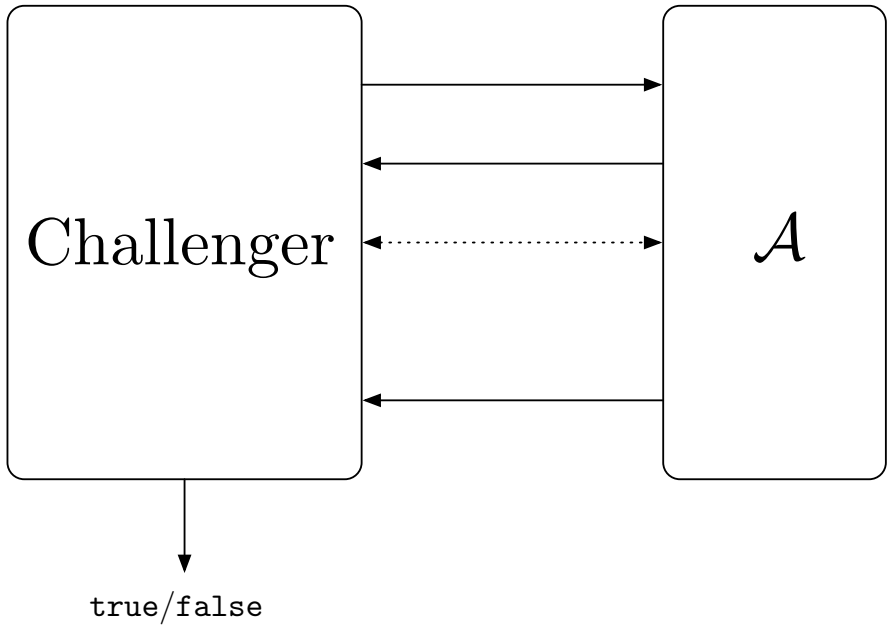
A convenient way to talk about properties of protocols:

- Allow an adversary to interact with a cryptosystem in a restricted way (e.g., hide the key).
- When the adversary terminates, verify if it was successful.

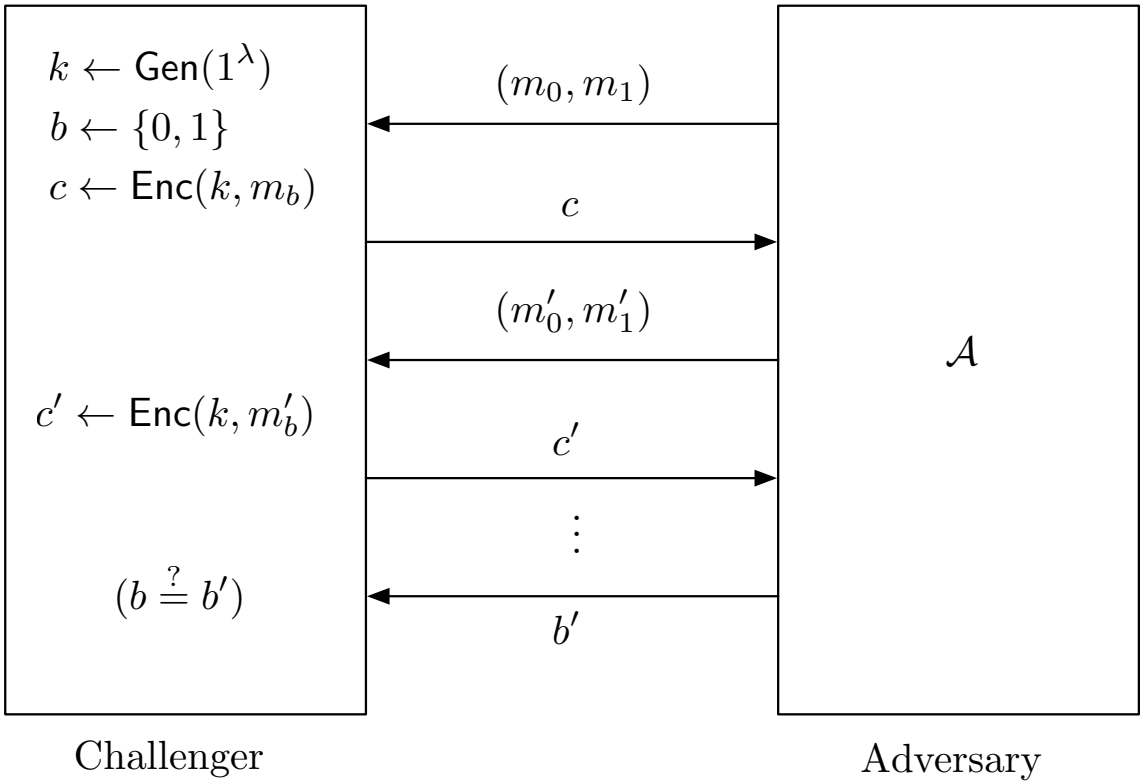
Experiments

A convenient way to talk about properties of protocols:

- Allow an adversary to interact with a cryptosystem in a restricted way (e.g., hide the key).
- When the adversary terminates, verify if it was successful.



Formalizing Security



Formally

$\underline{\text{IND-CPA}^{\mathcal{A}}_{\mathsf{SE}}(\lambda)}$:
$k \leftarrow_{\$} \mathsf{Gen}(\lambda)$
$b \leftarrow_{\$} \{0,1\}$
$b' \leftarrow_{\$} \mathcal{A}^{\text{Enc}(\cdot,\cdot)}(\lambda)$
return $(b = b')$

$\underline{\text{Enc}(m_0, m_1)}$:
$c \leftarrow_{\$} \mathsf{Enc}(k, m_b)$
return c

Formally

$\underline{\text{IND-CPA}^{\mathcal{A}}_{\mathsf{SE}}(\lambda)}$:
$k \leftarrow_{\$} \mathsf{Gen}(\lambda)$
$b \leftarrow_{\$} \{0,1\}$
$b' \leftarrow_{\$} \mathcal{A}^{\text{Enc}(\cdot,\cdot)}(\lambda)$
return $(b = b')$

$\underline{\text{Enc}(m_0, m_1)}$:
$c \leftarrow_{\$} \mathsf{Enc}(k, m_b)$
return c

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{SE},\mathcal{A}}(\lambda) := \Pr[\text{IND-CPA}^{\mathcal{A}}_{\mathsf{SE}}(\lambda) = \text{true}] - 1/2$$

Formally

$\underline{\text{IND-CPA}^{\mathcal{A}}_{\mathsf{SE}}(\lambda)}$:
$k \leftarrow_{\$} \mathsf{Gen}(\lambda)$
$b \leftarrow_{\$} \{0,1\}$
$b' \leftarrow_{\$} \mathcal{A}^{\text{Enc}(\cdot,\cdot)}(\lambda)$
return $(b = b')$

$\underline{\text{Enc}(m_0, m_1)}$:
$c \leftarrow_{\$} \mathsf{Enc}(k, m_b)$
return c

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{SE},\mathcal{A}}(\lambda) := \Pr[\text{IND-CPA}^{\mathcal{A}}_{\mathsf{SE}}(\lambda) = \text{true}] - 1/2$$

Scheme SE is secure if:

$$\forall \text{ efficient } \mathcal{A} \ : \ \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{SE},\mathcal{A}}(\lambda) \text{ is small.}$$

Formally

$\underline{\text{IND-CPA}^{\mathcal{A}}_{\mathsf{SE}}(\lambda)}$:
$k \leftarrow_{\$} \mathsf{Gen}(\lambda)$
$b \leftarrow_{\$} \{0,1\}$
$b' \leftarrow_{\$} \mathcal{A}^{\text{Enc}(\cdot,\cdot)}(\lambda)$
return $(b = b')$

$\underline{\text{Enc}(m_0, m_1)}$:
$c \leftarrow_{\$} \mathsf{Enc}(k, m_b)$
return c

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{SE},\mathcal{A}}(\lambda) := \Pr[\text{IND-CPA}^{\mathcal{A}}_{\mathsf{SE}}(\lambda) = \text{true}] - 1/2$$

Scheme SE is secure if:

$$\forall \text{ efficient } \mathcal{A} \ : \ \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{SE},\mathcal{A}}(\lambda) \text{ is small.}$$

Note: Any encryption scheme meeting this notion must be **randomized**.

Two Quick Notes

Smallness:

A function $\mu : \mathbb{N} \longrightarrow \mathbb{R}$ is negligible/small if it goes to zero faster that $1/\lambda^c$ for any c:

$$\mu(\lambda) \in \lambda^{-\omega(1)} \ .$$

Two Quick Notes

Smallness:

A function $\mu : \mathbb{N} \longrightarrow \mathbb{R}$ is negligible/small if it goes to zero faster that $1/\lambda^c$ for any c:

$$\mu(\lambda) \in \lambda^{-\omega(1)} \ .$$

For example: $2^{-\lambda} = \lambda^{-\log \lambda}$ but not $\frac{1}{\lambda^2}$.

Two Quick Notes

Smallness:

A function $\mu : \mathbb{N} \longrightarrow \mathbb{R}$ is negligible/small if it goes to zero faster that $1/\lambda^c$ for any c:

$$\mu(\lambda) \in \lambda^{-\omega(1)} \ .$$

For example: $2^{-\lambda} = \lambda^{-\log \lambda}$ but not $\frac{1}{\lambda^2}$.

Efficiency:

A (randomized) algorithm $A(x)$ is efficient if there is a $c \in \mathbb{N}$ such that

$$(\forall x) \ : \ \textbf{Time}(A(x)) \leq |x|^c.$$

We call such A **probabilistic polynomial time**.

Two Quick Notes

Smallness:

A function $\mu : \mathbb{N} \longrightarrow \mathbb{R}$ is negligible/small if it goes to zero faster that $1/\lambda^c$ for any c:

$$\mu(\lambda) \in \lambda^{-\omega(1)} .$$

For example: $2^{-\lambda} = \lambda^{-\log \lambda}$ but not $\frac{1}{\lambda^2}$.

Efficiency:

A (randomized) algorithm $A(x)$ is efficient if there is a $c \in \mathbb{N}$ such that

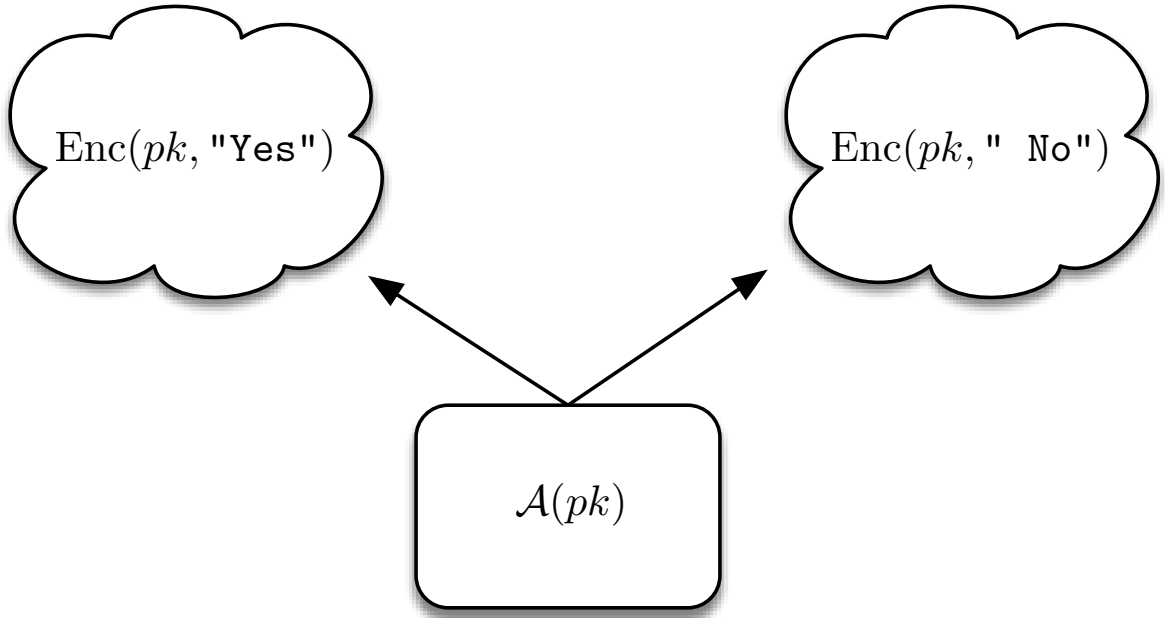
$$(\forall x) \ : \ \textbf{Time}(A(x)) \leq |x|^c.$$

We call such A **probabilistic polynomial time**.

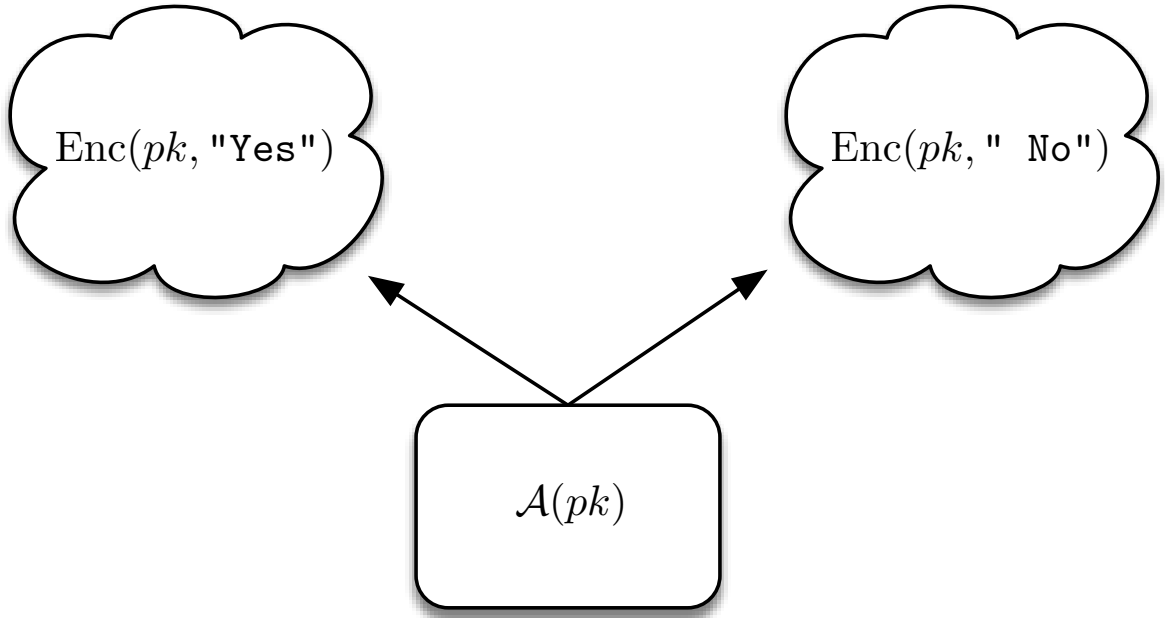
$$\mathrm{Negl} := \{ \text{ all negligible } \mu : \mathbb{N} \longrightarrow \mathbb{R} \ \}.$$

$$\mathrm{PPT} := \{ \text{ all efficient randomized Turing machines } \}.$$

Security of PKEs



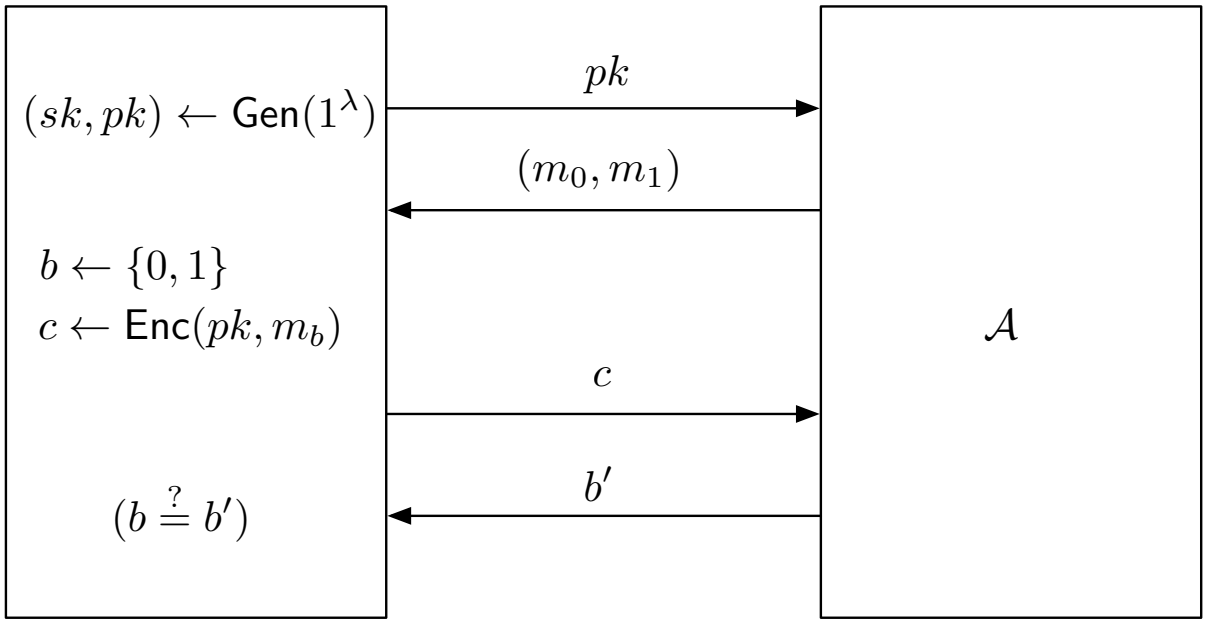
Security of PKEs



Ciphertexts look indistinguishable in the presence of pk.

Turns out: access to multiple ciphertexts doesn't affect the model. (Essentially because pk is public.)

Security of PKEs



Security of Public-Key Encryption

$\underline{\text{IND-CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)}$:
$(sk, pk) \leftarrow_{\$} \mathsf{Gen}(\lambda)$
$(m_0, m_1) \leftarrow_{\$} \mathcal{A}(pk)$
$b \leftarrow_{\$} \{0, 1\}$
$c \leftarrow_{\$} \mathsf{Enc}(pk, m_b)$
$b' \leftarrow_{\$} \mathcal{A}(c)$
return $(b = b')$

Security of Public-Key Encryption

$\underline{\text{IND-CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)}$:
$(sk, pk) \leftarrow_{\$} \mathsf{Gen}(\lambda)$
$(m_0, m_1) \leftarrow_{\$} \mathcal{A}(pk)$
$b \leftarrow_{\$} \{0, 1\}$
$c \leftarrow_{\$} \mathsf{Enc}(pk, m_b)$
$b' \leftarrow_{\$} \mathcal{A}(c)$
return $(b = b')$

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda) := \Pr[\text{IND-CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda) = \mathsf{true}] - 1/2$$

Security of Public-Key Encryption

$\underline{\text{IND-CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)}$:
$(sk, pk) \leftarrow_{\$} \mathsf{Gen}(\lambda)$
$(m_0, m_1) \leftarrow_{\$} \mathcal{A}(pk)$
$b \leftarrow_{\$} \{0,1\}$
$c \leftarrow_{\$} \mathsf{Enc}(pk, m_b)$
$b' \leftarrow_{\$} \mathcal{A}(c)$
return $(b = b')$

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda) := \Pr[\text{IND-CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda) = \mathsf{true}] - 1/2$$

Scheme PKE is IND-CPA secure if

$$(\forall \mathcal{A} \in \text{PPT}) \ : \ \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda) \in \text{Negl} \ .$$

A Familiar Example

Suppose company QWave claims it has built a factoring algorithm $\mathcal{Q}$. How can we check this claim?

A Familiar Example

Suppose company QWave claims it has built a factoring algorithm $\mathcal{Q}$. How can we check this claim?

$\underline{\mathrm{Factor}_{\mathcal{Q}}(\lambda)}$:

1. Generate two random λ-bit primes p, q. Set $N := pq$.
2. Give N to $\mathcal{Q}$ and get back p', q'.
3. Check if $(N = p'q')$.

$$\mathbf{Adv}^{\mathrm{factor}}_{\mathcal{Q}}(\lambda) := \Pr\left[\mathrm{Factor}_{\mathcal{Q}}(\lambda) = \mathsf{true}\right]$$

A Familiar Example

Suppose company QWave claims it has built a factoring algorithm $\mathcal{Q}$. How can we check this claim?

$\underline{\mathrm{Factor}_{\mathcal{Q}}(\lambda)}$:

1. Generate two random λ-bit primes p, q. Set $N := pq$.
2. Give N to $\mathcal{Q}$ and get back p', q'.
3. Check if $(N = p'q')$.

$$\mathbf{Adv}^{\mathrm{factor}}_{\mathcal{Q}}(\lambda) := \Pr\left[\mathrm{Factor}_{\mathcal{Q}}(\lambda) = \mathsf{true}\right]$$

Fun Fact: If we believe factoring is classically hard and $\mathcal{Q}$ can factor, we get evidence that $\mathcal{Q}$ is doing something non-classical/quantum ...

A Familiar Example

Suppose company QWave claims it has built a factoring algorithm $\mathcal{Q}$. How can we check this claim?

$\underline{\mathrm{Factor}_{\mathcal{Q}}(\lambda)}$:

1. Generate two random λ-bit primes p, q. Set $N := pq$.
2. Give N to $\mathcal{Q}$ and get back p', q'.
3. Check if $(N = p'q')$.

$$\mathbf{Adv}^{\mathrm{factor}}_{\mathcal{Q}}(\lambda) := \Pr\left[\mathrm{Factor}_{\mathcal{Q}}(\lambda) = \mathsf{true}\right]$$

Fun Fact: If we believe factoring is classically hard and $\mathcal{Q}$ can factor, we get evidence that $\mathcal{Q}$ is doing something non-classical/quantum ...

Contrapositive:

factoring is hard iff $\mathbf{Adv}^{\mathrm{factor}}_{\mathcal{Q}}(\lambda)$ is small for all efficient $\mathcal{Q}$.

Security Definitions

General template of a security definition:

- What it means to break a system:

Security Definitions

General template of a security definition:

- What it means to break a system:

  Recover keys,

Security Definitions

General template of a security definition:

- What it means to break a system:

  Recover keys, plaintexts,

Security Definitions

General template of a security definition:

- What it means to break a system:

  Recover keys, plaintexts, or some information about plaintexts?

Security Definitions

General template of a security definition:

- What it means to break a system:

  Recover keys, plaintexts, or some information about plaintexts?

- What the adversary's resources are:

Security Definitions

General template of a security definition:

- What it means to break a system:

  Recover keys, plaintexts, or some information about plaintexts?

- What the adversary's resources are:

  Has access to a single or multiple ciphertexts?

Security Definitions

General template of a security definition:

- What it means to break a system:

  Recover keys, plaintexts, or some information about plaintexts?

- What the adversary's resources are:

  Has access to a single or multiple ciphertexts? What's its efficiency?

Security Definitions

General template of a security definition:

- What it means to break a system:

  Recover keys, plaintexts, or some information about plaintexts?

- What the adversary's resources are:

  Has access to a single or multiple ciphertexts? What's its efficiency?

- How adversary's success is measured (advantage metric).

Security Definitions

General template of a security definition:

- What it means to break a system:

  Recover keys, plaintexts, or some information about plaintexts?

- What the adversary's resources are:

  Has access to a single or multiple ciphertexts? What's its efficiency?

- How adversary's success is measured (advantage metric).

A cryptosystem is secure if

Any adversary of specified resources has small advantage in breaking the system.

Assumptions

(3) Assumptions.

Is Factoring Hard?

Recall:

- **P** := All problems that can be solved in polynomial time.
- **NP** := All problems whose solutions can be checked efficiently.

Whether or not

$$\mathbf{P} \stackrel{?}{=} \mathbf{NP}$$

remains **an/the** open problem in CS/Math.

Is Factoring Hard?

Recall:

- **P** := All problems that can be solved in polynomial time.
- **NP** := All problems whose solutions can be checked efficiently.

Whether or not

$$\mathbf{P} \stackrel{?}{=} \mathbf{NP}$$

remains **an/the** open problem in CS/Math. But,

$$\mathbf{P} = \mathbf{NP} \implies \text{factoring easy}$$

Is Factoring Hard?

Recall:

- **P** := All problems that can be solved in polynomial time.
- **NP** := All problems whose solutions can be checked efficiently.

Whether or not

$$\mathbf{P} \stackrel{?}{=} \mathbf{NP}$$

remains **an/the** open problem in CS/Math. But,

$$\mathbf{P} = \mathbf{NP} \implies \text{factoring easy}$$
$$\text{i.e., factoring hard} \implies \mathbf{P} \neq \mathbf{NP}$$

Is Factoring Hard?

Recall:

- **P** := All problems that can be solved in polynomial time.
- **NP** := All problems whose solutions can be checked efficiently.

Whether or not

$$\mathbf{P} \stackrel{?}{=} \mathbf{NP}$$

remains **an/the** open problem in CS/Math. But,

$$\mathbf{P} = \mathbf{NP} \implies \text{factoring easy}$$
$$\text{i.e., factoring hard} \implies \mathbf{P} \neq \mathbf{NP}$$

Indeed:

$$\text{Secure SE, Secure PKE, } \ldots \implies \mathbf{P} \neq \mathbf{NP}.$$

So how can we deal with this?

Our Goal?

Our goal is:

To build highly secure and efficient cryptosystems.

As we just saw, we cannot prove hardness. Instead, we try to

Base the hardness of one cryptosystem/problem
on the hardness of other cryptosystems/problems.

Our Goal?

Our goal is:

To build highly secure and efficient cryptosystems.

As we just saw, we cannot prove hardness. Instead, we try to

Base the hardness of one cryptosystem/problem
on the hardness of other cryptosystems/problems.

i.e., try to relate two hardness notions:

$$\text{Problem } P \text{ hard} \implies \text{Scheme } S \text{ secure}$$

In doing so, we should try to minimize our assumptions and maximize our security goals:

Our Goal?

Our goal is:

To build highly secure and efficient cryptosystems.

As we just saw, we cannot prove hardness. Instead, we try to

Base the hardness of one cryptosystem/problem
on the hardness of other cryptosystems/problems.

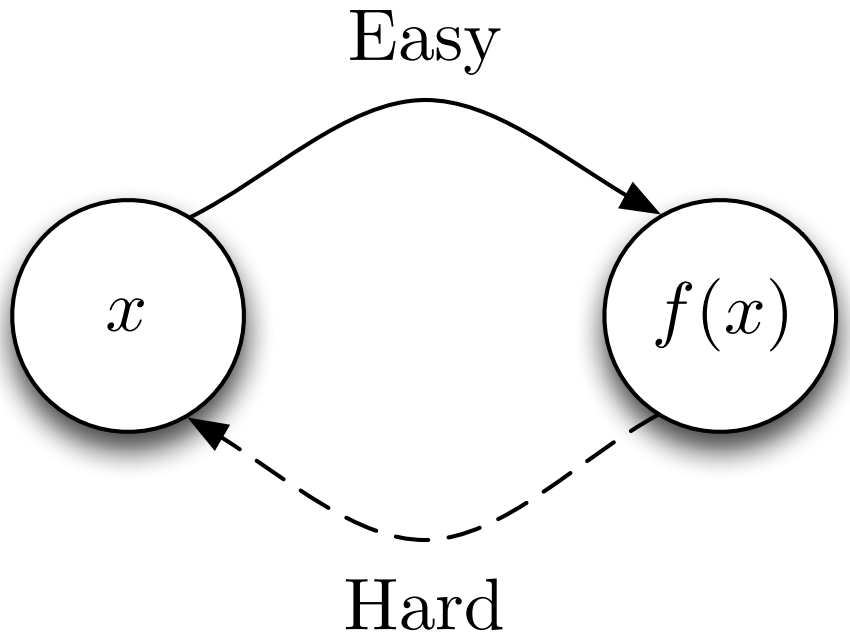
i.e., try to relate two hardness notions:

$$\text{Problem } P \text{ hard} \implies \text{Scheme } S \text{ secure}$$

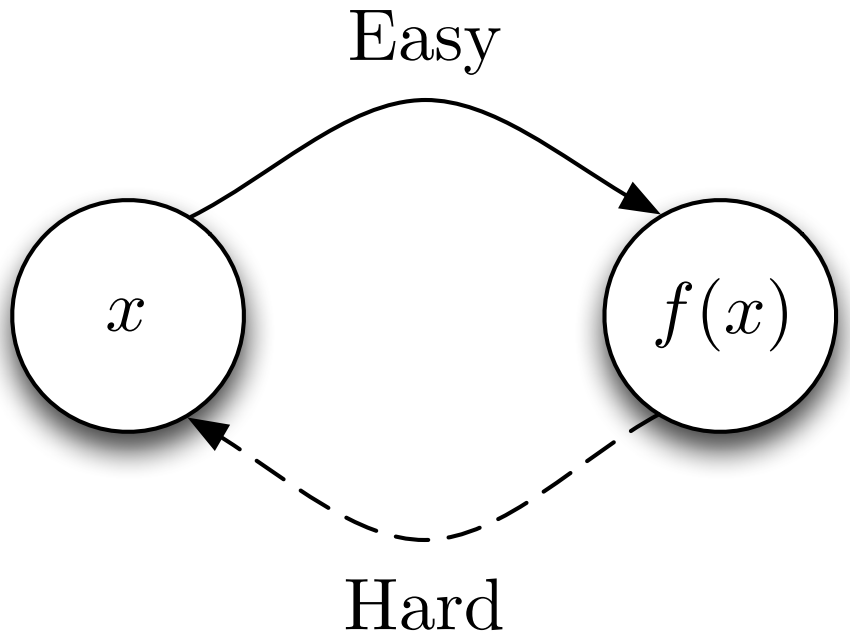
In doing so, we should try to minimize our assumptions and maximize our security goals:

- hardness/security of P to be minimal/simple/well-studied.
- hardness/security of S to be maximal/realistic/novel.

A Minimal Hardness Notion: One-Way Functions



A Minimal Hardness Notion: One-Way Functions



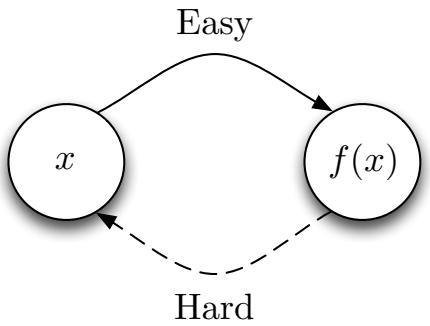
1. F(x): An efficient deterministic algorithm that on input a domain point x outputs an image point y.

One-Way Functions

Security: Hard to compute a preimage for $\mathsf{F}(x)$.

One-Way Functions

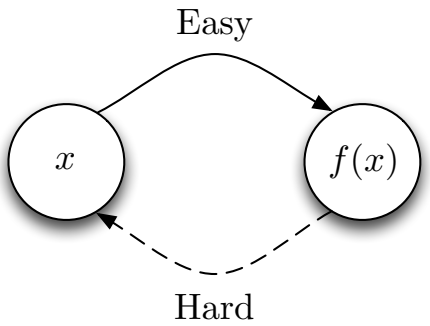
Security: Hard to compute a preimage for $\mathsf{F}(x)$.



$\underline{\mathrm{OW}_{\mathsf{F}}^{\mathcal{A}}(\lambda)}$:

One-Way Functions

Security: Hard to compute a preimage for $\mathsf{F}(x)$.



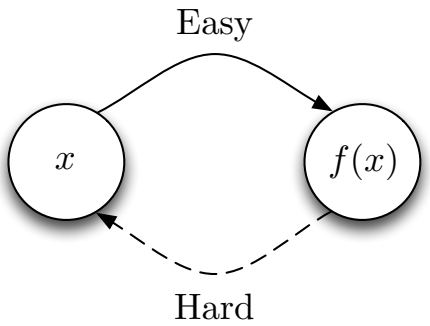
$\underline{\mathrm{OW}_{\mathsf{F}}^{\mathcal{A}}(\lambda)}$:

$x \leftarrow\!\!\$\ \{0,1\}^{\lambda}$

$y \leftarrow \mathsf{F}(x)$

One-Way Functions

Security: Hard to compute a preimage for $\mathsf{F}(x)$.



$\underline{\mathrm{OW}_{\mathsf{F}}^{\mathcal{A}}(\lambda)}$:

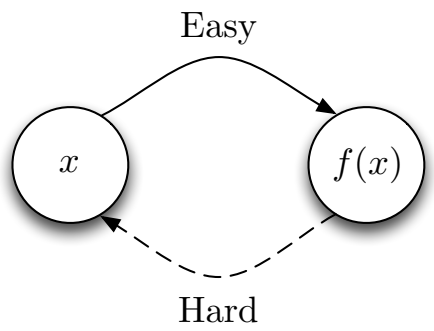
$x \leftarrow\!\!\$\ \{0,1\}^{\lambda}$

$y \leftarrow \mathsf{F}(x)$

$x' \leftarrow\!\!\$\ \mathcal{A}(y)$

One-Way Functions

Security: Hard to compute a preimage for $\mathsf{F}(x)$.



$\underline{\mathrm{OW}_{\mathsf{F}}^{\mathcal{A}}(\lambda)}$:

$x \leftarrow\!\!\$\ \{0,1\}^{\lambda}$

$y \leftarrow \mathsf{F}(x)$

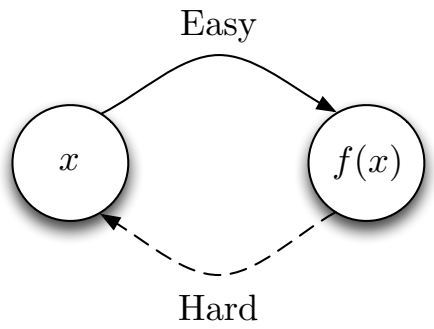
$x' \leftarrow\!\!\$\ \mathcal{A}(y)$

$y' \leftarrow \mathsf{F}(x')$

return $(y = y')$

One-Way Functions

Security: Hard to compute a preimage for $\mathsf{F}(x)$.

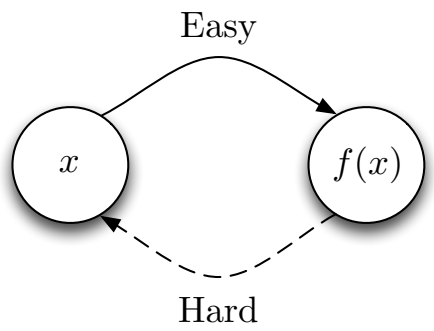


$\underline{\mathrm{OW}^{\mathcal{A}}_{\mathsf{F}}(\lambda)}$:
$x \leftarrow\!\!\$\ \{0,1\}^{\lambda}$
$y \leftarrow \mathsf{F}(x)$
$x' \leftarrow\!\!\$\ \mathcal{A}(y)$
$y' \leftarrow \mathsf{F}(x')$
return $(y = y')$

$$\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F},\mathcal{A}}(\lambda) := \Pr[\mathrm{OW}^{\mathcal{A}}_{\mathsf{F}}(\lambda)]$$

One-Way Functions

Security: Hard to compute a preimage for $\mathsf{F}(x)$.



$\underline{\mathrm{OW}^{\mathcal{A}}_{\mathsf{F}}(\lambda)}$:
$x \leftarrow\!\!\$\ \{0,1\}^{\lambda}$
$y \leftarrow \mathsf{F}(x)$
$x' \leftarrow\!\!\$\ \mathcal{A}(y)$
$y' \leftarrow \mathsf{F}(x')$
return $(y = y')$

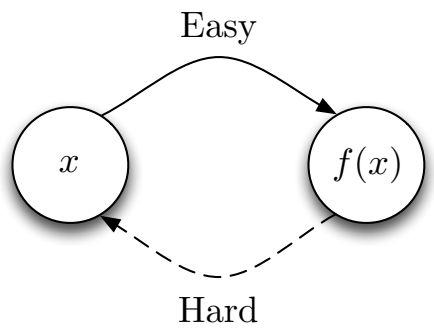
$$\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F},\mathcal{A}}(\lambda) := \Pr[\mathrm{OW}^{\mathcal{A}}_{\mathsf{F}}(\lambda)]$$

We say F is a one-way if:

$$(\forall \mathcal{A} \in \mathrm{PPT}) \ : \ \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F},\mathcal{A}}(\lambda) \in \mathrm{Negl}\ .$$

One-Way Functions

Security: Hard to compute a preimage for $\mathsf{F}(x)$.



$\underline{\mathrm{OW}^{\mathcal{A}}_{\mathsf{F}}(\lambda)}$:
$x \leftarrow\!\!\$\ \{0,1\}^{\lambda}$
$y \leftarrow \mathsf{F}(x)$
$x' \leftarrow\!\!\$\ \mathcal{A}(y)$
$y' \leftarrow \mathsf{F}(x')$
return $(y = y')$

$$\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F},\mathcal{A}}(\lambda) := \Pr[\mathrm{OW}^{\mathcal{A}}_{\mathsf{F}}(\lambda)]$$

We say F is a one-way if:

$$(\forall \mathcal{A} \in \mathrm{PPT}) \ : \ \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F},\mathcal{A}}(\lambda) \in \mathrm{Negl}\ .$$

Question 1: Is $f(x) := x \pmod 2$ one way?
Question 2: If factoring is hard, do OWFs exist?

Minimal Notion II: Collision-Resistance Hash Functions

Security: F for which it's hard to find $x_1 \neq x_2$ with $\mathsf{F}(x_1) = \mathsf{F}(x_2)$.
F is "essentially injective."

Minimal Notion II: Collision-Resistance Hash Functions

Security: F for which it's hard to find $x_1 \neq x_2$ with $\mathsf{F}(x_1) = \mathsf{F}(x_2)$.
F is "essentially injective."

$\underline{\mathrm{CR}^{\mathcal{A}}_{\mathsf{F}}(\lambda)}$:
$(x_1, x_2) \leftarrow\!\!\$\ \mathcal{A}(\lambda)$
$y_1 \leftarrow \mathsf{F}(x_1)$; $y_2 \leftarrow \mathsf{F}(x_2)$
return $(x_1 \neq x_2 \wedge y_1 = y_2)$

Minimal Notion II: Collision-Resistance Hash Functions

Security: F for which it's hard to find $x_1 \neq x_2$ with $\mathsf{F}(x_1) = \mathsf{F}(x_2)$.
F is "essentially injective."

$\underline{\mathrm{CR}^{\mathcal{A}}_{\mathsf{F}}(\lambda)}$:
$(x_1, x_2) \leftarrow\!\!\$\ \mathcal{A}(\lambda)$
$y_1 \leftarrow \mathsf{F}(x_1)$; $y_2 \leftarrow \mathsf{F}(x_2)$
return $(x_1 \neq x_2 \wedge y_1 = y_2)$

F is collision resistant if:

$$(\forall \mathcal{A} \in \mathrm{PPT}) \ : \ \mathbf{Adv}^{\mathrm{cr}}_{\mathsf{F},\mathcal{A}}(\lambda) := \mathsf{Pr}[\mathrm{CR}^{\mathcal{A}}_{\mathsf{F}}(\lambda)] \in \mathrm{Negl} \ .$$

Goal

Recall our goal was to:

Base the hardness of one cryptosystem/problem
on the hardness of other cryptosystems/problems.

i.e., show:

$$\text{Problem } P \text{ hard} \implies \text{Cryosystem } S \text{ secure}$$

How can we prove this?

(4) Security Proofs via Reductions.

Toy Reduction

Suppose a calculator can only square numbers. Can we use it to multiply?

$$\text{Build} \qquad M : (x, y) \mapsto x \cdot y \qquad \text{from} \qquad S : x \mapsto x^2$$

Toy Reduction

Suppose a calculator can only square numbers. Can we use it to multiply?

$$\text{Build} \qquad M : (x, y) \mapsto x \cdot y \qquad \text{from} \qquad S : x \mapsto x^2$$

Yes:

$$S(x + y) - S(x) - S(y) = (x + y)^2 - x^2 - y^2$$

Toy Reduction

Suppose a calculator can only square numbers. Can we use it to multiply?

$$\text{Build} \qquad M : (x, y) \mapsto x \cdot y \qquad \text{from} \qquad S : x \mapsto x^2$$

Yes:

$$S(x + y) - S(x) - S(y) = (x + y)^2 - x^2 - y^2 = 2xy$$

Toy Reduction

Suppose a calculator can only square numbers. Can we use it to multiply?

$$\text{Build} \qquad M : (x, y) \mapsto x \cdot y \qquad \text{from} \qquad S : x \mapsto x^2$$

Yes:

$$S(x + y) - S(x) - S(y) = (x + y)^2 - x^2 - y^2 = 2xy$$

and now drop the LSB. We have **reduced** M to S.

Toy Reduction

Suppose a calculator can only square numbers. Can we use it to multiply?

$$\text{Build} \qquad M : (x, y) \mapsto x \cdot y \qquad \text{from} \qquad S : x \mapsto x^2$$

Yes:

$$S(x + y) - S(x) - S(y) = (x + y)^2 - x^2 - y^2 = 2xy$$

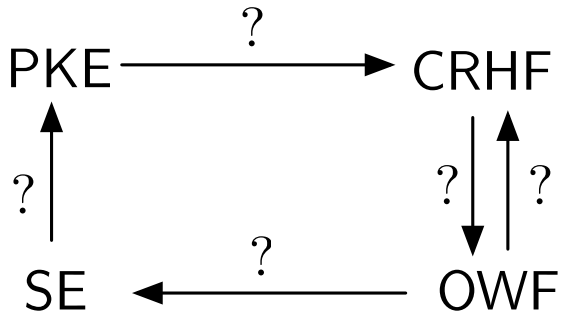
and now drop the LSB. We have **reduced** M to S.

A **reduction** from M to S is an algorithm R^S that solves M using a subroutine for S.

Question: If I believe M is hard to compute and M reduces to S, does it follow that S is hard to compute?

Crypto

Can we reduce the security/correctness of one cryptosystem to the security/correctness of another?



Crypto Reductions

A reduction from a primitive $\mathcal{P}$ (think SE) to a primitive $\mathcal{Q}$ (think OWF) is a **pair** of oracle algorithms

$$(C^?, R^?)$$

Crypto Reductions

A reduction from a primitive $\mathcal{P}$ (think SE) to a primitive $\mathcal{Q}$ (think OWF) is a **pair** of oracle algorithms

$$(C^?, R^?)$$

called the **construction** and the **security reduction** respectively, where:

Crypto Reductions

A reduction from a primitive $\mathcal{P}$ (think SE) to a primitive $\mathcal{Q}$ (think OWF) is a **pair** of oracle algorithms

$$(C^?, R^?)$$

called the **construction** and the **security reduction** respectively, where:

- If Q is an instance of $\mathcal{Q}$, then $P := C^Q$ is an instance of $\mathcal{P}$.

Crypto Reductions

A reduction from a primitive $\mathcal{P}$ (think SE) to a primitive $\mathcal{Q}$ (think OWF) is a **pair** of oracle algorithms

$$(C^{?}, R^{?})$$

called the **construction** and the **security reduction** respectively, where:

- If Q is an instance of $\mathcal{Q}$, then $P := C^{Q}$ is an instance of $\mathcal{P}$.
- If $\mathcal{A}$ breaks Q, then $\mathcal{B} := R^{\mathcal{A}}$ breaks $P = C^{Q}$.

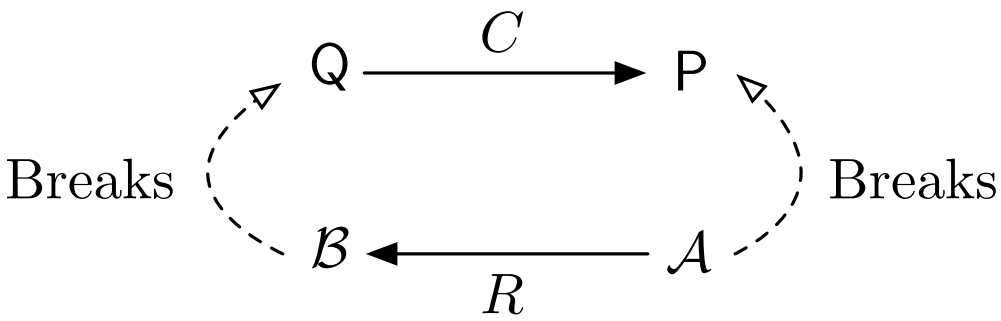
Crypto Reductions

A reduction from a primitive $\mathcal{P}$ (think SE) to a primitive $\mathcal{Q}$ (think OWF) is a **pair** of oracle algorithms

$$(C^?, R^?)$$

called the **construction** and the **security reduction** respectively, where:

- If Q is an instance of $\mathcal{Q}$, then $P := C^Q$ is an instance of $\mathcal{P}$.
- If $\mathcal{A}$ breaks Q, then $\mathcal{B} := R^{\mathcal{A}}$ breaks $P = C^Q$.



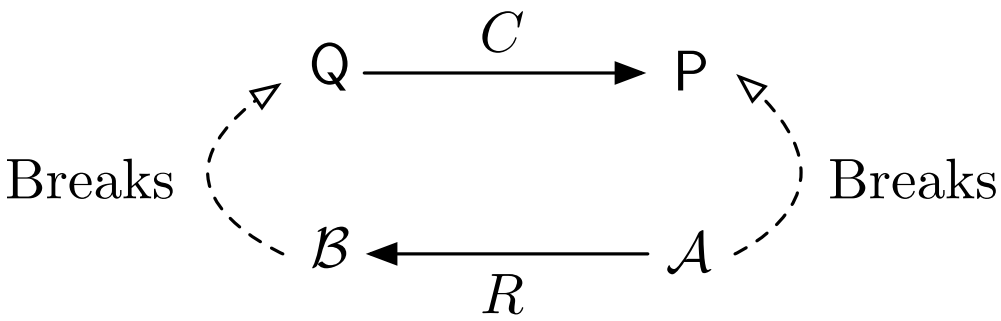
Crypto Reductions

A reduction from a primitive $\mathcal{P}$ (think SE) to a primitive $\mathcal{Q}$ (think OWF) is a **pair** of oracle algorithms

$$(C^?, R^?)$$

called the **construction** and the **security reduction** respectively, where:

- If Q is an instance of $\mathcal{Q}$, then $P := C^Q$ is an instance of $\mathcal{P}$.
- If $\mathcal{A}$ breaks Q, then $\mathcal{B} := R^{\mathcal{A}}$ breaks $P = C^Q$.



So if Q is secure, then so is P.

Temple for Statement of Security

Results in provable security typically look like this:

Theorem

If primitive Q (think OWF) is secure, then construction C^Q is a secure instance of P (think SE).

Temple for Statement of Security

Results in provable security typically look like this:

Theorem

If primitive Q (think OWF) is secure, then construction C^Q is a secure instance of P (think SE). More precisely, for any $t(\lambda)$-time $\mathcal{A}$ against C^Q (wrt. its security definition), there is a $t'(\lambda)$-time adversary $\mathcal{B} := R^{\mathcal{A}}$ against Q (wrt. its security definition) such that:

$$\mathbf{Adv}^{\mathrm{sec1}}_{P,\mathcal{A}}(\lambda) \leq \ell(\lambda) \cdot \mathbf{Adv}^{\mathrm{sec2}}_{Q,\mathcal{B}}(\lambda) + \varepsilon(\lambda)$$

for a (polynomial) function $\ell(\lambda)$ and a negligible function $\varepsilon(\lambda)$.

Temple for Statement of Security

Results in provable security typically look like this:

Theorem

If primitive Q (think OWF) is secure, then construction C^Q is a secure instance of P (think SE). More precisely, for any $t(\lambda)$-time $\mathcal{A}$ against C^Q (wrt. its security definition), there is a $t'(\lambda)$-time adversary $\mathcal{B} := R^{\mathcal{A}}$ against Q (wrt. its security definition) such that:

$$\mathbf{Adv}^{\mathrm{sec1}}_{P,\mathcal{A}}(\lambda) \leq \ell(\lambda) \cdot \mathbf{Adv}^{\mathrm{sec2}}_{Q,\mathcal{B}}(\lambda) + \varepsilon(\lambda)$$

for a (polynomial) function $\ell(\lambda)$ and a negligible function $\varepsilon(\lambda)$.

The reduction R is **tight** if

$$t(\lambda) \approx t'(\lambda) \quad \text{and} \quad \ell(\lambda) \approx 1 \quad \text{and} \quad \varepsilon(\lambda) \approx 0 \ .$$

This means most of the security inherent in P is **preserved**.

Temple for Statement of Security

Results in provable security typically look like this:

Theorem

If primitive Q (think OWF) is secure, then construction C^Q is a secure instance of P (think SE). More precisely, for any $t(\lambda)$-time $\mathcal{A}$ against C^Q (wrt. its security definition), there is a $t'(\lambda)$-time adversary $\mathcal{B} := R^{\mathcal{A}}$ against Q (wrt. its security definition) such that:

$$\mathbf{Adv}^{\mathrm{sec1}}_{P,\mathcal{A}}(\lambda) \leq \ell(\lambda) \cdot \mathbf{Adv}^{\mathrm{sec2}}_{Q,\mathcal{B}}(\lambda) + \varepsilon(\lambda)$$

for a (polynomial) function $\ell(\lambda)$ and a negligible function $\varepsilon(\lambda)$.

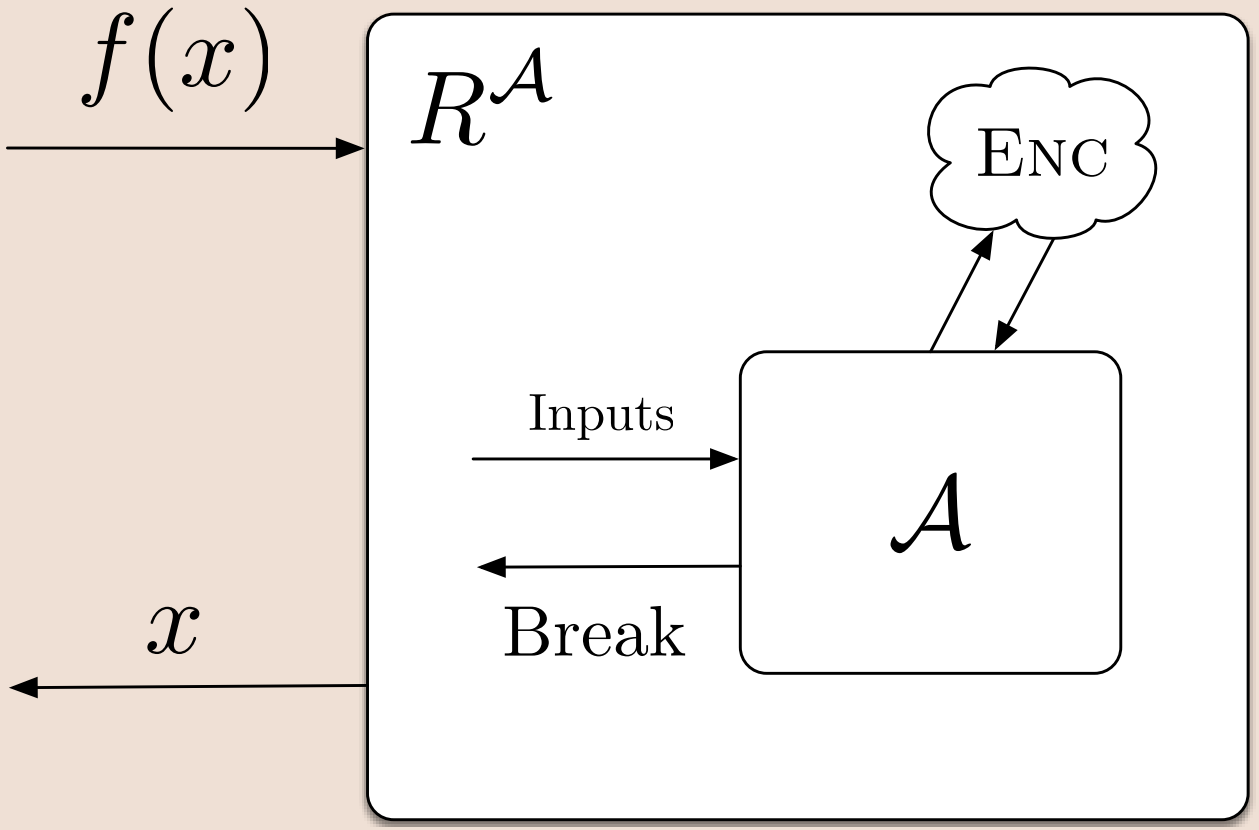
The reduction R is **tight** if

$$t(\lambda) \approx t'(\lambda) \quad \text{and} \quad \ell(\lambda) \approx 1 \quad \text{and} \quad \varepsilon(\lambda) \approx 0 \ .$$

This means most of the security inherent in P is **preserved**.
Guides setting key sizes.

Proof Template via Reductions

Proof.



Reductions

Example I: One-Way and Collision-Resistant Functions.

Can We Reduce OWFs to CRHFs?

Suppose we are given an f that is collision resistant.
Can we build a one-way function f' using f?

Can We Reduce OWFs to CRHFs?

Suppose we are given an f that is collision resistant.
Can we build a one-way function f' using f?

$f(x) := x$ is collision resistant. So no reduction to such f. (Why?)

Can We Reduce OWFs to CRHFs?

Suppose we are given an f that is collision resistant.
Can we build a one-way function f' using f?

$f(x) := x$ is collision resistant. So no reduction to such f. (Why?)

Let's look at r-regular functions:

$$(\forall x) \ : \ |f^{-1}(f(x))| = r \qquad \text{for an } r > 1.$$

Note: r-regular functions are **compressing**.

Can We Reduce OWFs to CRHFs?

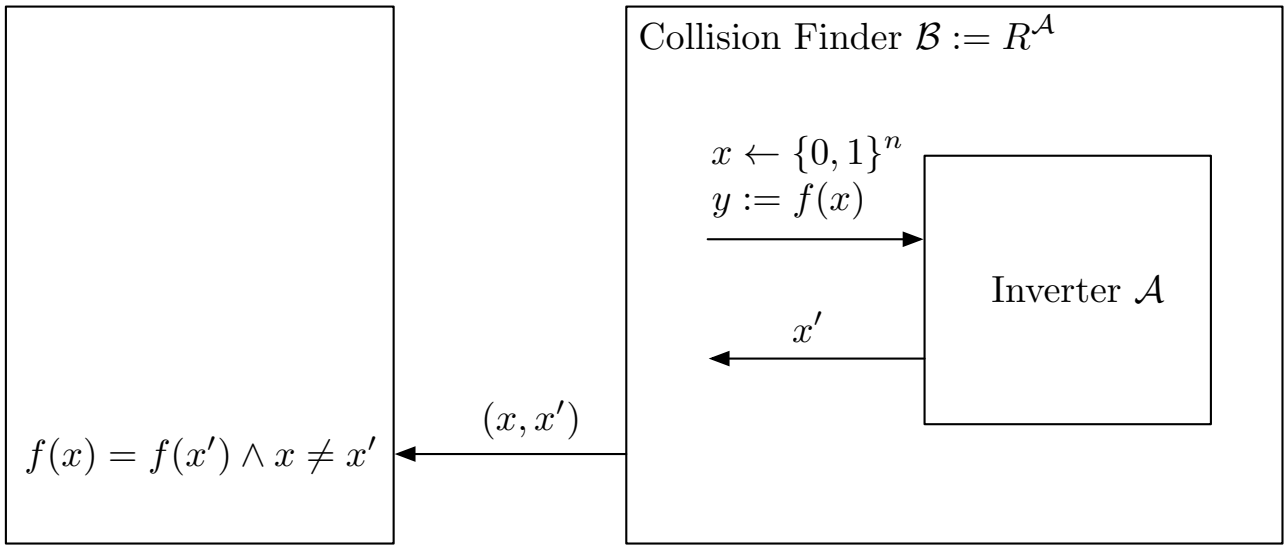
Theorem

Any r-regular collision-resistant hash function is one way.

Can We Reduce OWFs to CRHFs?

Theorem

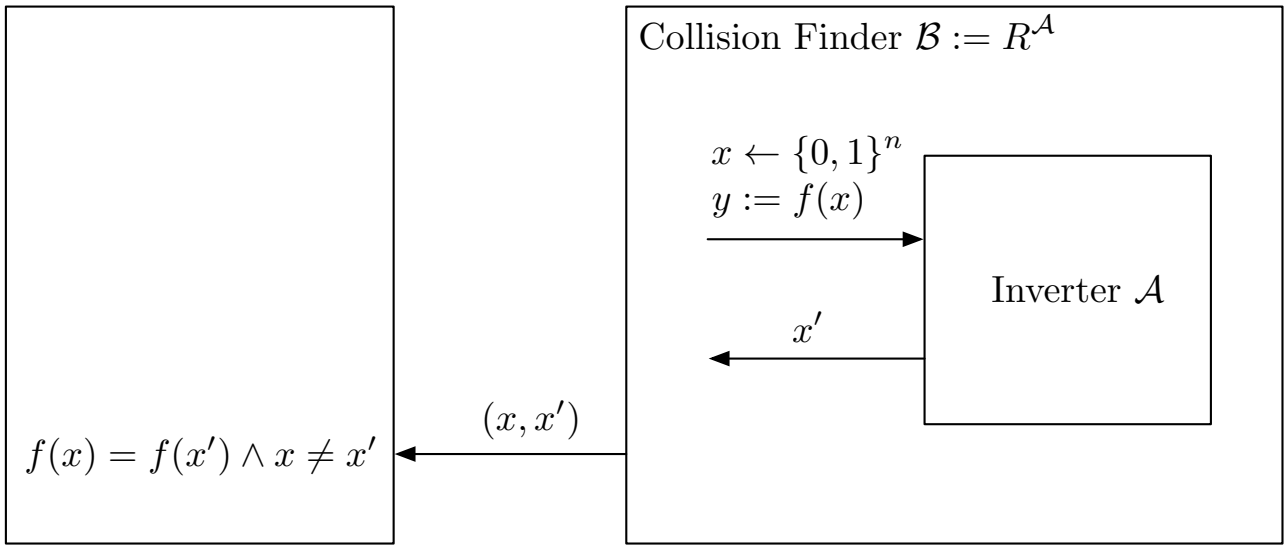
Any r-regular collision-resistant hash function is one way.



Can We Reduce OWFs to CRHFs?

Theorem

Any r-regular collision-resistant hash function is one way.

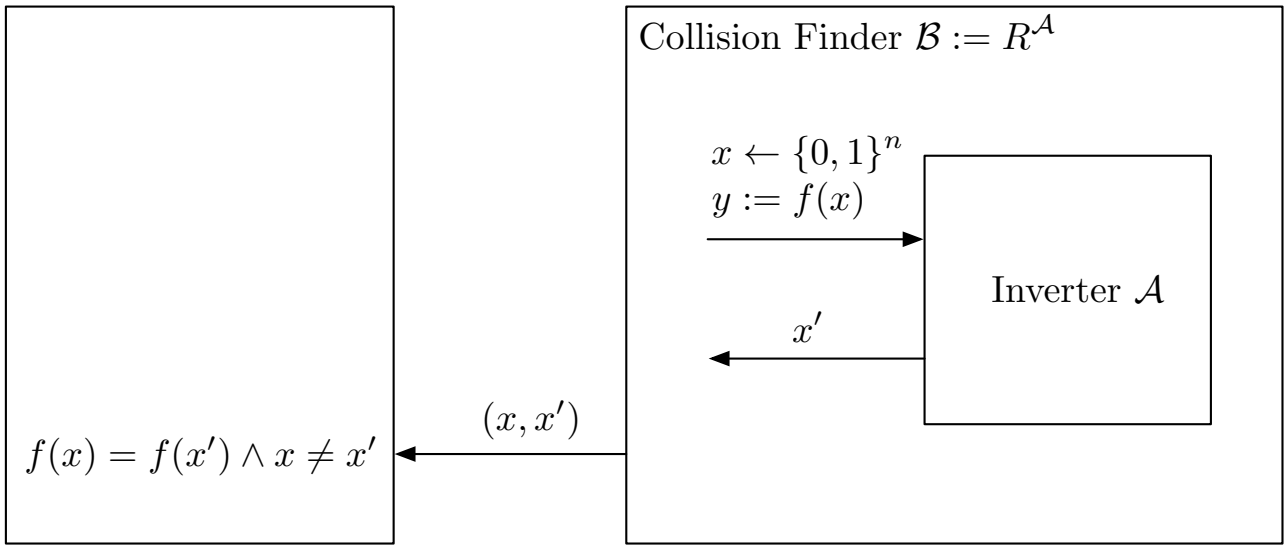


- Will $f(x') = f(x)$?

Can We Reduce OWFs to CRHFs?

Theorem

Any r-regular collision-resistant hash function is one way.

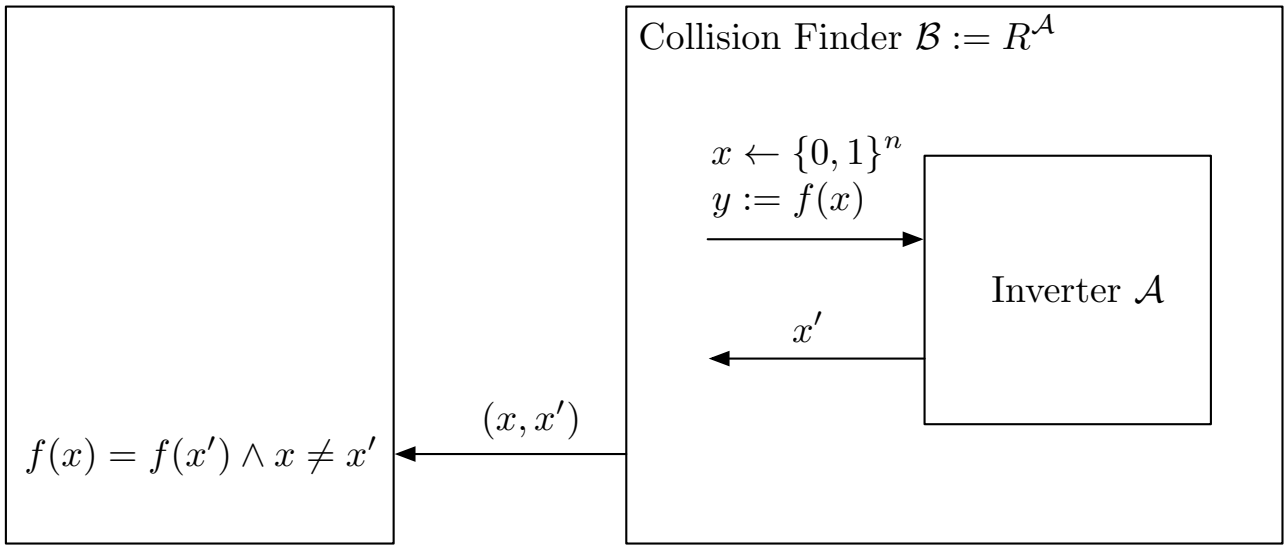


- Will $f(x') = f(x)$? With good prob. yes since $\mathcal{A}$ is a good inverter.

Can We Reduce OWFs to CRHFs?

Theorem

Any r-regular collision-resistant hash function is one way.

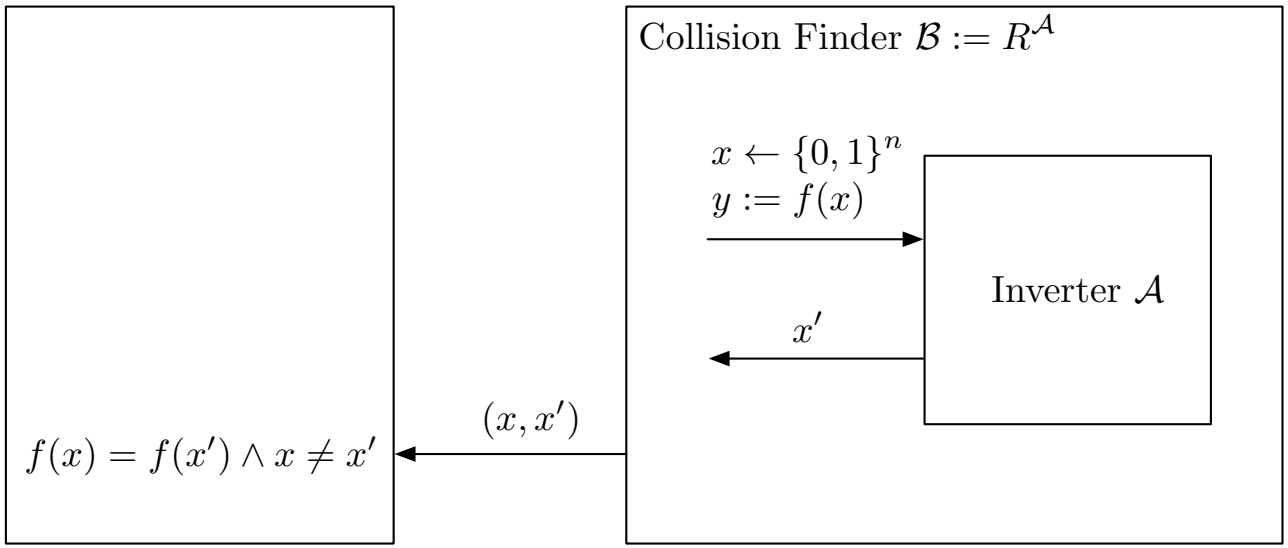


- Will $f(x') = f(x)$? With good prob. yes since $\mathcal{A}$ is a good inverter.
- Will $x \neq x'$?

Can We Reduce OWFs to CRHFs?

Theorem

Any r-regular collision-resistant hash function is one way.



- Will $f(x') = f(x)$? With good prob. yes since $\mathcal{A}$ is a good inverter.
- Will $x \neq x'$? Adversary $\mathcal{A}$ only sees $f(x)$. This gives no information about which of r possible x's was used to calculate $f(x)$. So $\mathcal{A}$ will output an $x' \neq x$ with probability $1 - 1/r \geq 1/2$.

Can We Reduce CRHFs to OWFs?

Suppose now we are given an f that is one way.
Can we build a collision-resistant function f' using f?

Observation: Setting $f' := f$ (i.e., $C := id$) does not work.

Can We Reduce CRHFs to OWFs?

Suppose now we are given an f that is one way.
Can we build a collision-resistant function f' using f?

Observation: Setting $f' := f$ (i.e., $C := id$) does not work. Consider

$$f^*(x) := \begin{cases} 0 & \text{if } x = 0, 1 \\ f(x) & \text{otherwise.} \end{cases}$$

Can We Reduce CRHFs to OWFs?

Suppose now we are given an f that is one way.
Can we build a collision-resistant function f' using f?

Observation: Setting $f' := f$ (i.e., $C := id$) does not work. Consider

$$f^*(x) := \begin{cases} 0 & \text{if } x = 0, 1 \\ f(x) & \text{otherwise.} \end{cases}$$

Note:

- f^* is not collision resistant: $f(0) = f(1) = 0$ and $0 \neq 1$.
- If f is one way, then so is f^* as for a random x with overwhelming probability $f(x) = f^*(x)$.

Can We Reduce CRHFs to OWFs?

Suppose now we are given an f that is one way.
Can we build a collision-resistant function f' using f?

Observation: Setting $f' := f$ (i.e., $C := id$) does not work. Consider

$$f^*(x) := \begin{cases} 0 & \text{if } x = 0, 1 \\ f(x) & \text{otherwise.} \end{cases}$$

Note:

- f^* is not collision resistant: $f(0) = f(1) = 0$ and $0 \neq 1$.
- If f is one way, then so is f^* as for a random x with overwhelming probability $f(x) = f^*(x)$.

This is called a **separating example**.

Can We Reduce CRHFs to OWFs?

Suppose now we are given an f that is one way.
Can we build a collision-resistant function f' using f?

Observation: Setting $f' := f$ (i.e., $C := id$) does not work. Consider

$$f^*(x) := \begin{cases} 0 & \text{if } x = 0, 1 \\ f(x) & \text{otherwise.} \end{cases}$$

Note:

- f^* is not collision resistant: $f(0) = f(1) = 0$ and $0 \neq 1$.
- If f is one way, then so is f^* as for a random x with overwhelming probability $f(x) = f^*(x)$.

This is called a **separating example**.
Much more is known:

Allowing arbitrary C's does not help either [Sim98].

This is called an **impossibility** result.

Reducing Big CRHFs to Small CRHFs

Suppose we have build a CRHF with finite domain:

$$h : \{0,1\}^{2n} \longrightarrow \{0,1\}^{n}$$

How can we build a CRHF with arbitrary domain?

$$H : \{0,1\}^{*} \longrightarrow \{0,1\}^{n}$$

Reducing Big CRHFs to Small CRHFs

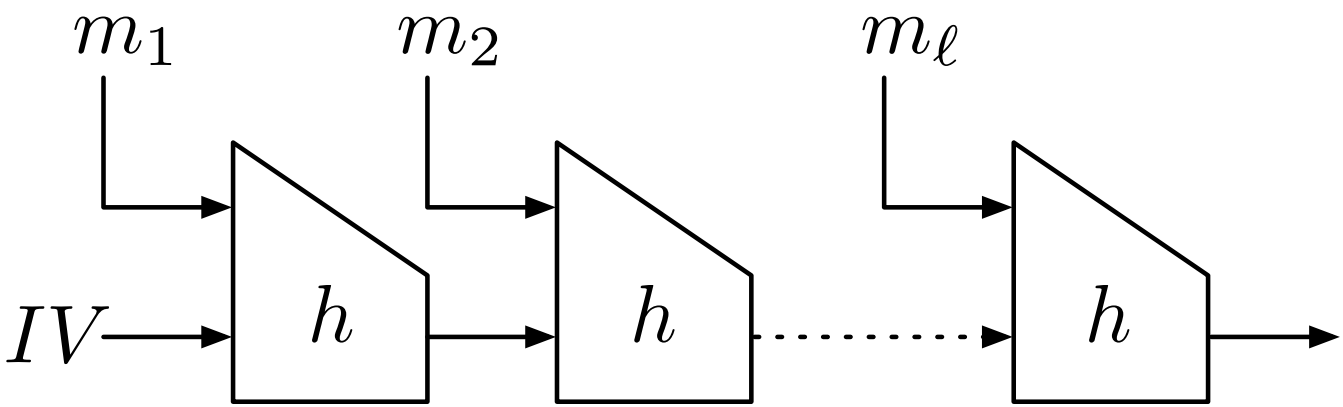
Suppose we have build a CRHF with finite domain:

$$h : \{0,1\}^{2n} \longrightarrow \{0,1\}^n$$

How can we build a CRHF with arbitrary domain?

$$H : \{0,1\}^* \longrightarrow \{0,1\}^n$$

Merkle–Damgård chaining: Given $m = (m_1, m_2, \ldots, m_\ell)$ do:



Call this construction MD^h.

Reducing Big CRHFs to Small CRHFs

Theorem

If h is collision resistant then MD^h is also collision resistant.

Reducing Big CRHFs to Small CRHFs

Theorem

If h is collision resistant then MD^h is also collision resistant.

For simplicity, let's look at $\ell = 2$.

Reducing Big CRHFs to Small CRHFs

Theorem

If h is collision resistant then MD^h is also collision resistant.

For simplicity, let's look at $\ell = 2$.
Suppose an adversary $\mathcal{A}$ outputs $(m_1, m_2) \neq (m_1', m_2')$ such that:

$$h(h(IV, m_1), m_2) = h(h(IV, m_1'), m_2')$$

Reducing Big CRHFs to Small CRHFs

Theorem

If h is collision resistant then MD^h is also collision resistant.

For simplicity, let's look at $\ell = 2$.
Suppose an adversary $\mathcal{A}$ outputs $(m_1, m_2) \neq (m_1', m_2')$ such that:

$$h(h(IV, m_1), m_2) = h(h(IV, m_1'), m_2')$$

Let's consider some cases:

1. If $m_1 = m_1'$ then $m_2 \neq m_2'$.

Reducing Big CRHFs to Small CRHFs

Theorem

If h is collision resistant then MD^h is also collision resistant.

For simplicity, let's look at $\ell = 2$.
Suppose an adversary $\mathcal{A}$ outputs $(m_1, m_2) \neq (m_1', m_2')$ such that:

$$h(h(IV, m_1), m_2) = h(h(IV, m_1'), m_2')$$

Let's consider some cases:

1. If $m_1 = m_1'$ then $m_2 \neq m_2'$. Hence $(h(IV, m_1), m_2) \neq (h(IV, m_1'), m_2')$,

Reducing Big CRHFs to Small CRHFs

Theorem

If h is collision resistant then MD^h is also collision resistant.

For simplicity, let's look at $\ell = 2$.
Suppose an adversary $\mathcal{A}$ outputs $(m_1, m_2) \neq (m_1', m_2')$ such that:

$$h(h(IV, m_1), m_2) = h(h(IV, m_1'), m_2')$$

Let's consider some cases:

1. If $m_1 = m_1'$ then $m_2 \neq m_2'$. Hence $(h(IV, m_1), m_2) \neq (h(IV, m_1'), m_2')$, and this is a collision for h.

Reducing Big CRHFs to Small CRHFs

Theorem

If h is collision resistant then MD^h is also collision resistant.

For simplicity, let's look at $\ell = 2$.
Suppose an adversary $\mathcal{A}$ outputs $(m_1, m_2) \neq (m_1', m_2')$ such that:

$$h(h(IV, m_1), m_2) = h(h(IV, m_1'), m_2')$$

Let's consider some cases:

1. If $m_1 = m_1'$ then $m_2 \neq m_2'$. Hence $(h(IV, m_1), m_2) \neq (h(IV, m_1'), m_2')$, and this is a collision for h.
2. If $m_1 \neq m_1'$:

Reducing Big CRHFs to Small CRHFs

Theorem

If h is collision resistant then MD^h is also collision resistant.

For simplicity, let's look at $\ell = 2$.
Suppose an adversary $\mathcal{A}$ outputs $(m_1, m_2) \neq (m_1', m_2')$ such that:

$$h(h(IV, m_1), m_2) = h(h(IV, m_1'), m_2')$$

Let's consider some cases:

1. If $m_1 = m_1'$ then $m_2 \neq m_2'$. Hence $(h(IV, m_1), m_2) \neq (h(IV, m_1'), m_2')$, and this is a collision for h.
2. If $m_1 \neq m_1'$:
   - If $h(IV, m_1) = h(IV, m_1')$:

Reducing Big CRHFs to Small CRHFs

Theorem

If h is collision resistant then MD^h is also collision resistant.

For simplicity, let's look at $\ell = 2$.
Suppose an adversary $\mathcal{A}$ outputs $(m_1, m_2) \neq (m_1', m_2')$ such that:

$$h(h(IV, m_1), m_2) = h(h(IV, m_1'), m_2')$$

Let's consider some cases:

1. If $m_1 = m_1'$ then $m_2 \neq m_2'$. Hence $(h(IV, m_1), m_2) \neq (h(IV, m_1'), m_2')$, and this is a collision for h.
2. If $m_1 \neq m_1'$:
   - If $h(IV, m_1) = h(IV, m_1')$: then we have a collision for h.

Reducing Big CRHFs to Small CRHFs

Theorem

If h is collision resistant then MD^h is also collision resistant.

For simplicity, let's look at $\ell = 2$.
Suppose an adversary $\mathcal{A}$ outputs $(m_1, m_2) \neq (m_1', m_2')$ such that:

$$h(h(IV, m_1), m_2) = h(h(IV, m_1'), m_2')$$

Let's consider some cases:

1. If $m_1 = m_1'$ then $m_2 \neq m_2'$. Hence $(h(IV, m_1), m_2) \neq (h(IV, m_1'), m_2')$, and this is a collision for h.
2. If $m_1 \neq m_1'$:
   - If $h(IV, m_1) = h(IV, m_1')$: then we have a collision for h.
   - If $\underbrace{h(IV, m_1)}_{x_1} \neq \underbrace{h(IV, m_1')}_{x_1'}$:

Reducing Big CRHFs to Small CRHFs

Theorem

If h is collision resistant then MD^h is also collision resistant.

For simplicity, let's look at $\ell = 2$.
Suppose an adversary $\mathcal{A}$ outputs $(m_1, m_2) \neq (m_1', m_2')$ such that:

$$h(h(IV, m_1), m_2) = h(h(IV, m_1'), m_2')$$

Let's consider some cases:

1. If $m_1 = m_1'$ then $m_2 \neq m_2'$. Hence $(h(IV, m_1), m_2) \neq (h(IV, m_1'), m_2')$, and this is a collision for h.
2. If $m_1 \neq m_1'$:
   - If $h(IV, m_1) = h(IV, m_1')$: then we have a collision for h.
   - If $\underbrace{h(IV, m_1)}_{x_1} \neq \underbrace{h(IV, m_1')}_{x_1'}$: then $(x_1, m_2) \neq (x_1', m_2')$,

Reducing Big CRHFs to Small CRHFs

Theorem

If h is collision resistant then MD^h is also collision resistant.

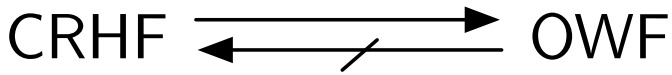
For simplicity, let's look at $\ell = 2$.
Suppose an adversary $\mathcal{A}$ outputs $(m_1, m_2) \neq (m_1', m_2')$ such that:

$$h(h(IV, m_1), m_2) = h(h(IV, m_1'), m_2')$$

Let's consider some cases:

1. If $m_1 = m_1'$ then $m_2 \neq m_2'$. Hence $(h(IV, m_1), m_2) \neq (h(IV, m_1'), m_2')$, and this is a collision for h.
2. If $m_1 \neq m_1'$:
   - If $h(IV, m_1) = h(IV, m_1')$: then we have a collision for h.
   - If $\underbrace{h(IV, m_1)}_{x_1} \neq \underbrace{h(IV, m_1')}_{x_1'}$: then $(x_1, m_2) \neq (x_1', m_2')$, a collision for h.

Summary



Example II: Encryption.

Groups

Recall cyclic groups $\mathbb{G} = \langle g \rangle$ of order p:

$$\mathbb{G} = \{g^a \ : \ a \in \mathbb{Z}_p\}, \quad g^a \cdot g^b = g^{a+b}.$$

One question for such groups is:

Given (g, g^x) can one efficiently compute x?

This is the **Discrete Logarithm** (DL) Problem.

DLP is a well-studied problem and is believed to be hard in certain groups.

Some Other Problems

Given (g, g^x, g^y) compute g^{xy}.

This is the **Computational Diffie–Hellman** (CDH) Problem.

Some Other Problems

Given (g, g^x, g^y) compute g^{xy}.

This is the **Computational Diffie–Hellman** (CDH) Problem.

Given (g, g^x, g^y, g^r) is $r = xy$?

This is the **Decisional Diffie–Hellman** (DDH) Problem.

Some Other Problems

Given (g, g^x, g^y) compute g^{xy}.

This is the **Computational Diffie–Hellman** (CDH) Problem.

Given (g, g^x, g^y, g^r) is $r = xy$?

This is the **Decisional Diffie–Hellman** (DDH) Problem. **Check**:

DDH Hard $\implies$ CDH Hard $\implies$ DL Hard

Can we use these problems to build a secure encryption scheme?

ElGamal Encryption

ElGamal Encryption

- $\mathsf{Gen}(\lambda; x) := (\underbrace{x}_{sk}, \underbrace{g, g^x}_{pk})$

ElGamal Encryption

- $\mathsf{Gen}(\lambda; x) := (\underbrace{x}_{sk}, \underbrace{g, g^x}_{pk})$
- $\mathsf{Enc}(pk, m; r) := (\underbrace{g^r}_{c_1}, \underbrace{m \cdot pk^r}_{c_2}) = (g^r, m \cdot (g^x)^r)$

ElGamal Encryption

- $\mathsf{Gen}(\lambda; x) := (\underbrace{x}_{sk}, \underbrace{g, g^x}_{pk})$
- $\mathsf{Enc}(pk, m; r) := (\underbrace{g^r}_{c_1}, \underbrace{m \cdot pk^r}_{c_2}) = (g^r, m \cdot (g^x)^r)$
- $\mathsf{Dec}(sk, (c_1, c_2)) := c_2 / c_1^{sk}$.

ElGamal Encryption

- $\mathsf{Gen}(\lambda; x) := (\underbrace{x}_{sk}, \underbrace{g, g^x}_{pk})$
- $\mathsf{Enc}(pk, m; r) := (\underbrace{g^r}_{c_1}, \underbrace{m \cdot pk^r}_{c_2}) = (g^r, m \cdot (g^x)^r)$
- $\mathsf{Dec}(sk, (c_1, c_2)) := c_2 / c_1^{sk}$.

Is this scheme correct?

ElGamal Encryption

- $\mathsf{Gen}(\lambda; x) := (\underbrace{x}_{sk}, \underbrace{g, g^x}_{pk})$
- $\mathsf{Enc}(pk, m; r) := (\underbrace{g^r}_{c_1}, \underbrace{m \cdot pk^r}_{c_2}) = (g^r, m \cdot (g^x)^r)$
- $\mathsf{Dec}(sk, (c_1, c_2)) := c_2 / c_1^{sk}$.

Is this scheme correct?

$$\mathsf{Dec}(sk, (c_1, c_2)) = \frac{c_2}{c_1^{sk}} = \frac{m \cdot (g^x)^r}{(g^r)^x} = m$$

ElGamal Security

Is ElGamal secure?

ElGamal Security

Is ElGamal secure? I.e., is it IND-CPA secure?

ElGamal Security

Is ElGamal secure? I.e., is it IND-CPA secure?

Theorem

If the DDH problem is hard, then ElGamal is IND-CPA secure.

As a result:

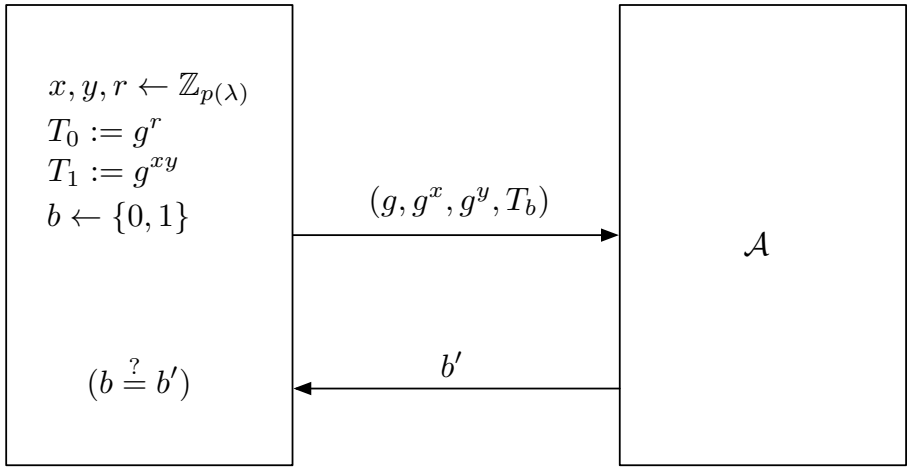
We have reduce the security of a PKE scheme with more complex syntax, correctness, and security to that of the DDH problem, which is simpler and better studied.

The DDH Assumption

Given (g, g^x, g^y, g^r) is $r = xy$?

The DDH Assumption

Given (g, g^x, g^y, g^r) is $r = xy$?

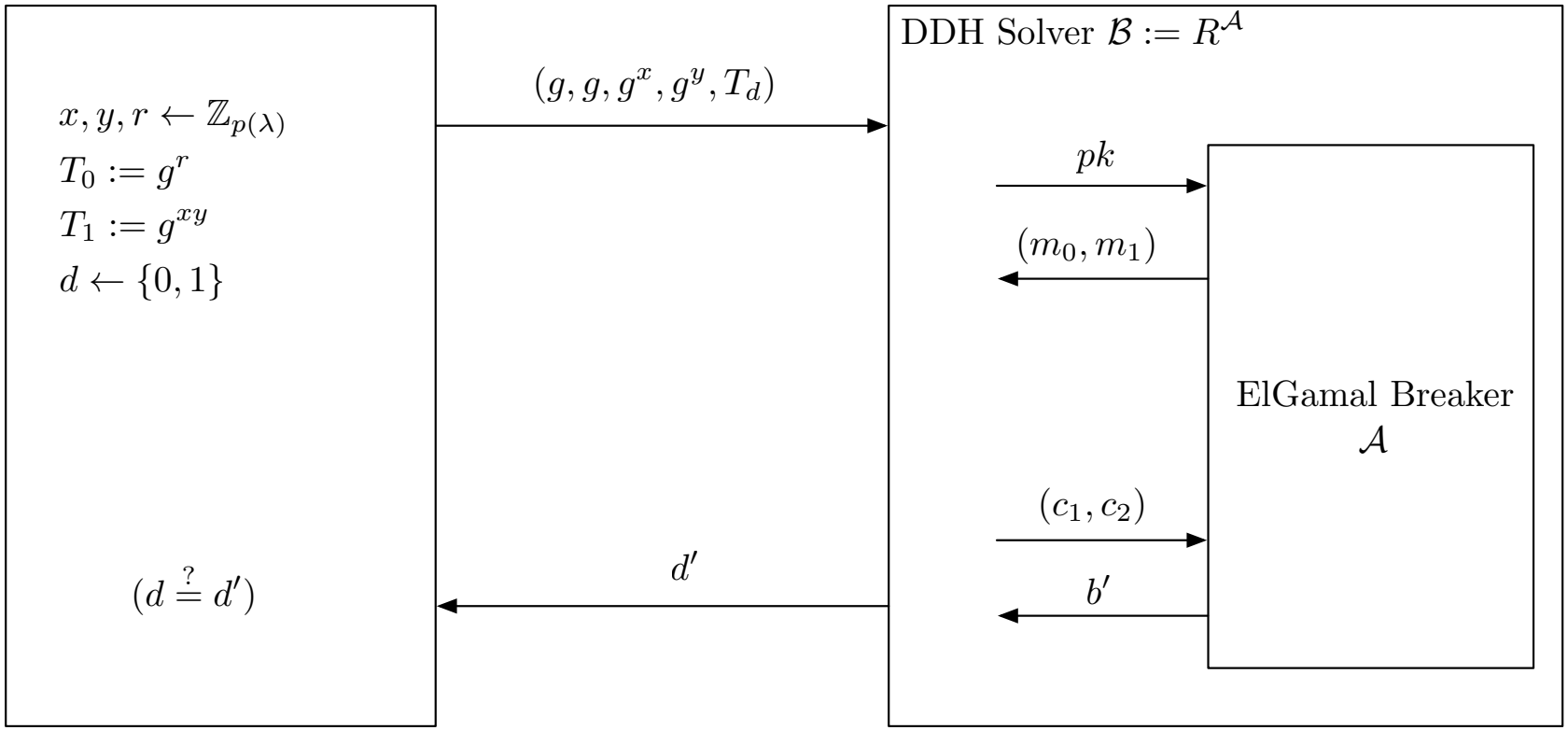


$$\mathbf{Adv}^{\mathrm{ddh}}_{G,\mathcal{A}}(\lambda) := \Pr[\mathrm{DDH}^{\mathcal{A}}_{G}(\lambda)] - 1/2$$

The IND-CPA Security of ElGamal

Theorem

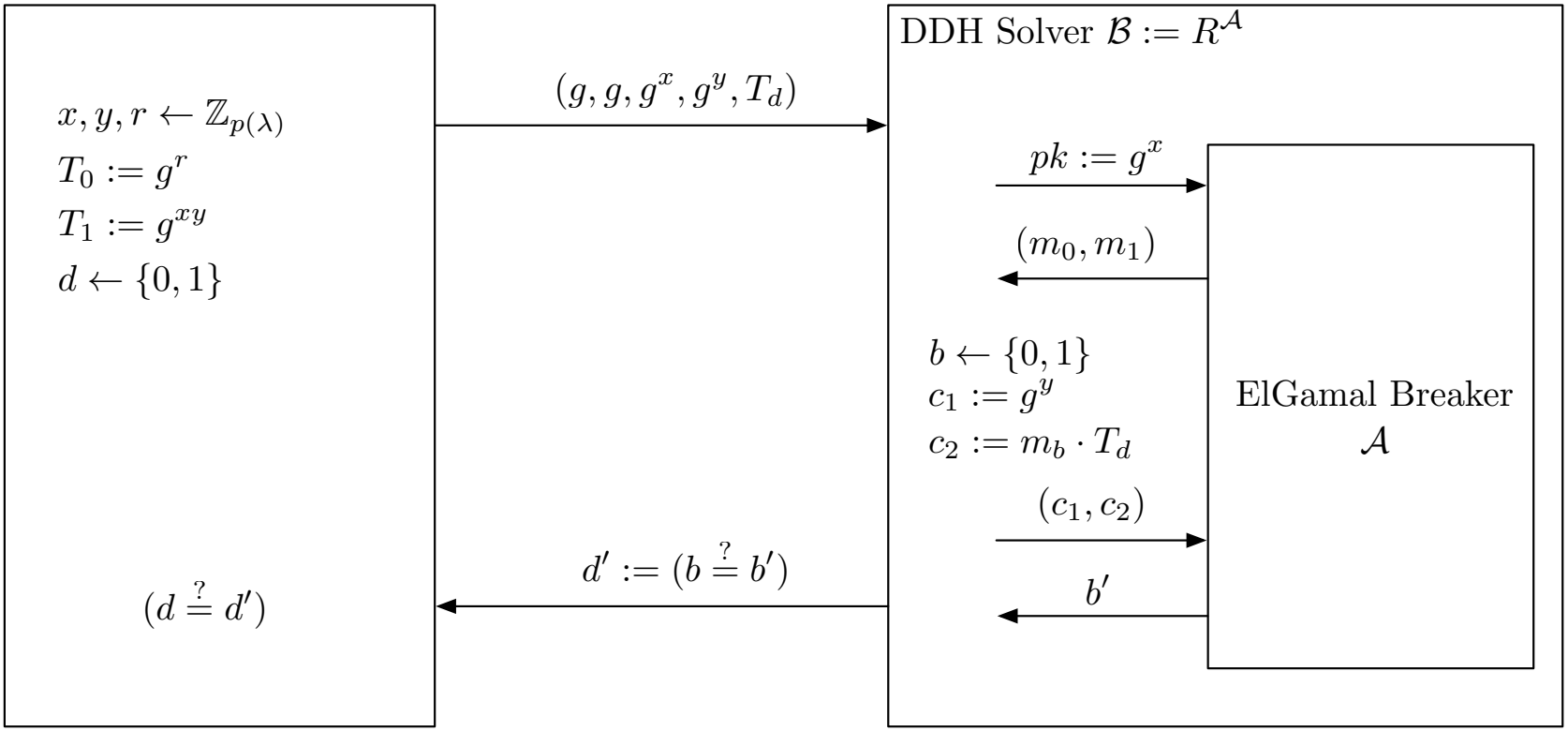
If the DDH problem is hard, then ElGamal is IND-CPA secure.



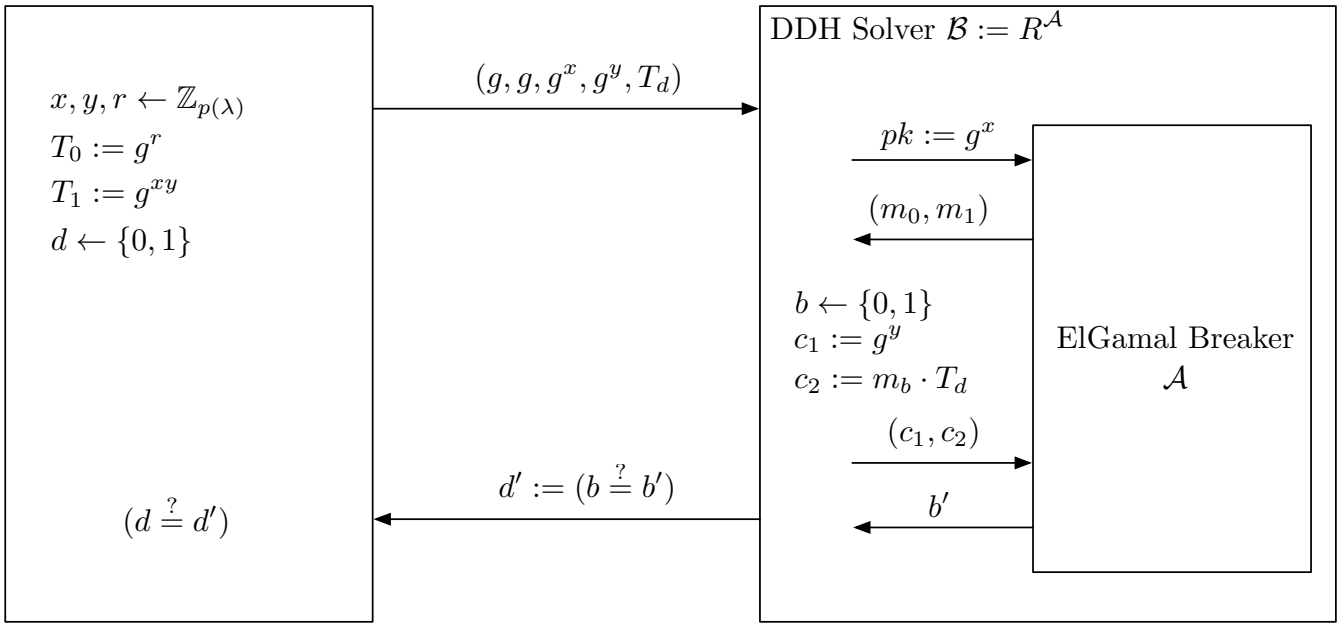
The IND-CPA Security of ElGamal

Theorem

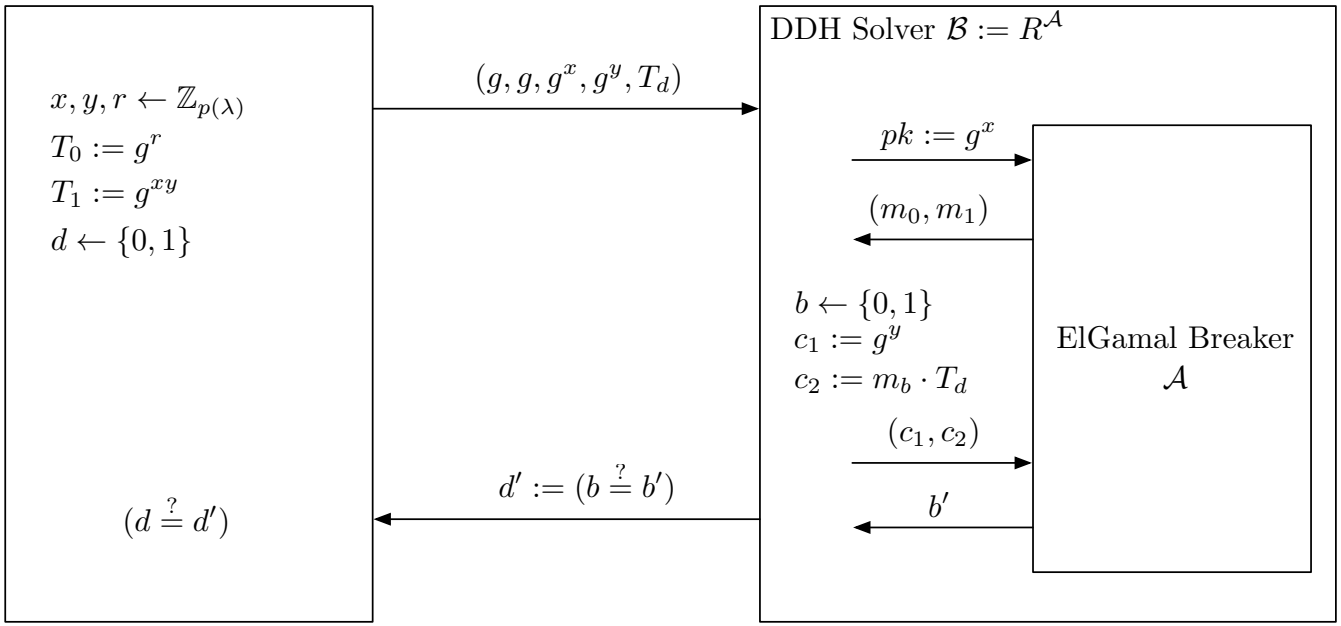
If the DDH problem is hard, then ElGamal is IND-CPA secure.



Proof

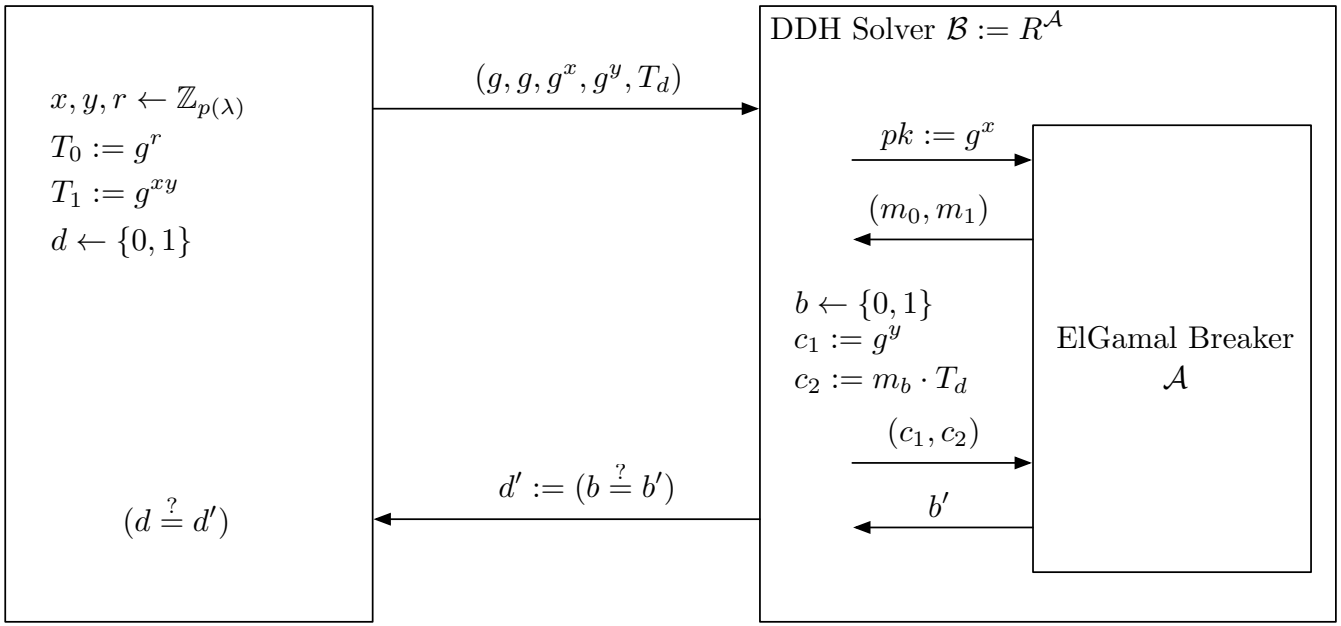

$x, y, r \leftarrow \mathbb{Z}_{p(\lambda)}$
$T_0 := g^r$
$T_1 := g^{xy}$
$d \leftarrow \{0,1\}$
$(d \stackrel{?}{=} d')$
(g, g, g^x, g^y, T_d)
$d' := (b \stackrel{?}{=} b')$
DDH Solver $\mathcal{B} := R^{\mathcal{A}}$
$pk := g^x$
(m_0, m_1)
$b \leftarrow \{0,1\}$
$c_1 := g^y$
$c_2 := m_b \cdot T_d$
(c_1, c_2)
b'
ElGamal Breaker
$\mathcal{A}$

Proof



- $d = 1$: Then $T_d = g^{xy}$.

Proof

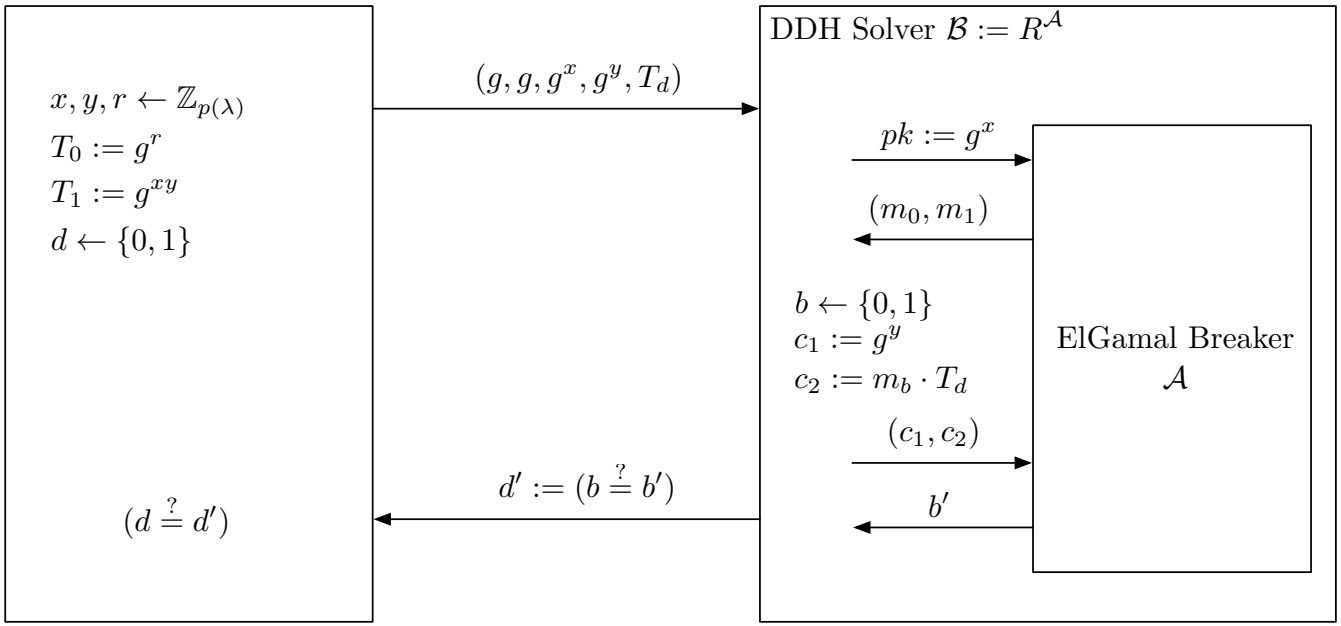


- $d = 1$: Then $T_d = g^{xy}$. So $\mathcal{A}$ is run according to the IND-CPA experiment.

$$\Pr[d = d' | d = 1] = \Pr[d' = 1] = \Pr[b = b'] = \mathbf{Adv}^{\text{ind-cpa}}_{\text{PKE},\mathcal{A}}(\lambda) + 1/2$$

- $d = 0$: Then $T_d = g^r$.

Proof



- $d = 1$: Then $T_d = g^{xy}$. So $\mathcal{A}$ is run according to the IND-CPA experiment.

$$\Pr[d = d' | d = 1] = \Pr[d' = 1] = \Pr[b = b'] = \mathbf{Adv}^{\text{ind-cpa}}_{\text{PKE},\mathcal{A}}(\lambda) + 1/2$$

- $d = 0$: Then $T_d = g^r$. Now $\mathcal{A}$ gets **no information** about b as a random group element g^r completely blinds m_b:

$$\Pr[d = d' | d = 0] = \Pr[b \neq b'] = 1/2 \ .$$

Proof

Continuing:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{ddh}}_{G,\mathcal{B}}(\lambda) &= \Pr[d = d'] - 1/2 \\
&= \Pr[d = d' | d = 1] \Pr[d = 1] + \Pr[d = d' | d = 0] \Pr[d = 0] - 1/2 \\
&= 1/2 \cdot (\Pr[d = d' | d = 1] + \Pr[d = d' | d = 0] - 1) \\
&= 1/2 \cdot (\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda) + 1/2 + 1/2 - 1) \\
&= 1/2 \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda)
\end{aligned}$$

What's going here?

- We are running the IND-CPA game with respect to T_d.
- If $d = 1$, and we have a good breaker, then $\Pr[b = b']$ will be noticeably above $1/2$.
- On the other hand, if $d = 0$, then $\Pr[b = b']$ will be exactly $1/2$ for **any** breaker.
- We use this difference in the two probabilities to decide d.

Proof Summary

DDH says:

$$(g, g^x, g^y, \underbrace{g^{xy}}_{d=1}) \quad \overset{c}{\approx} \quad (g, g^a, g^b, \underbrace{g^r}_{d=0})$$

The proof can be seen a transition

$$(\underbrace{g, g^x}_{pk}, \quad \underbrace{g^r, m_b \cdot g^{xy}}_{c}) \quad \rightarrow \quad (\underbrace{g, g^x}_{pk}, \quad \underbrace{g^r, m_b \cdot \boxed{g^r}}_{c})$$

based on DDH and according to the value of d.

Proof Summary

DDH says:

$$(g, g^x, g^y, \underbrace{g^{xy}}_{d=1}) \quad \overset{c}{\approx} \quad (g, g^a, g^b, \underbrace{g^r}_{d=0})$$

The proof can be seen a transition

$$(\underbrace{g, g^x}_{pk}, \quad \underbrace{g^r, m_b \cdot g^{xy}}_{c}) \quad \rightarrow \quad (\underbrace{g, g^x}_{pk}, \quad \underbrace{g^r, m_b \cdot \boxed{g^r}}_{c})$$

based on DDH and according to the value of d.

Proof.

Exp.	pk	c_1	c_2	**Remark**
0	g, g^x	g^y	$m_b \cdot g^{xy}$	

Proof Summary

DDH says:

$$(g, g^x, g^y, \underbrace{g^{xy}}_{d=1}) \overset{c}{\approx} (g, g^a, g^b, \underbrace{g^r}_{d=0})$$

The proof can be seen a transition

$$(\underbrace{g, g^x}_{pk}, \underbrace{g^r, m_b \cdot g^{xy}}_{c}) \rightarrow (\underbrace{g, g^x}_{pk}, \underbrace{g^r, m_b \cdot \boxed{g^r}}_{c})$$

based on DDH and according to the value of d.

Proof.

Exp.	pk	c_1	c_2	**Remark**
0	g, g^x	g^y	$m_b \cdot g^{xy}$	
1	g, g^x	g^y	$\boxed{m_b \cdot g^r}$	DDH Assumption

Proof Summary

DDH says:

$$(g, g^x, g^y, \underbrace{g^{xy}}_{d=1}) \overset{c}{\approx} (g, g^a, g^b, \underbrace{g^r}_{d=0})$$

The proof can be seen a transition

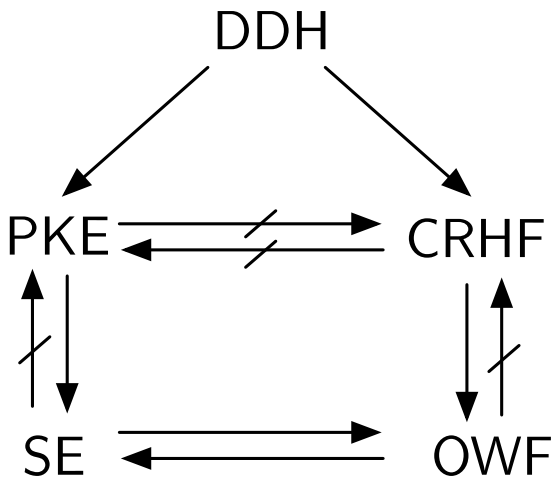
$$(\underbrace{g, g^x}_{pk}, \underbrace{g^r, m_b \cdot g^{xy}}_{c}) \rightarrow (\underbrace{g, g^x}_{pk}, \underbrace{g^r, m_b \cdot \boxed{g^r}}_{c})$$

based on DDH and according to the value of d.

Proof.

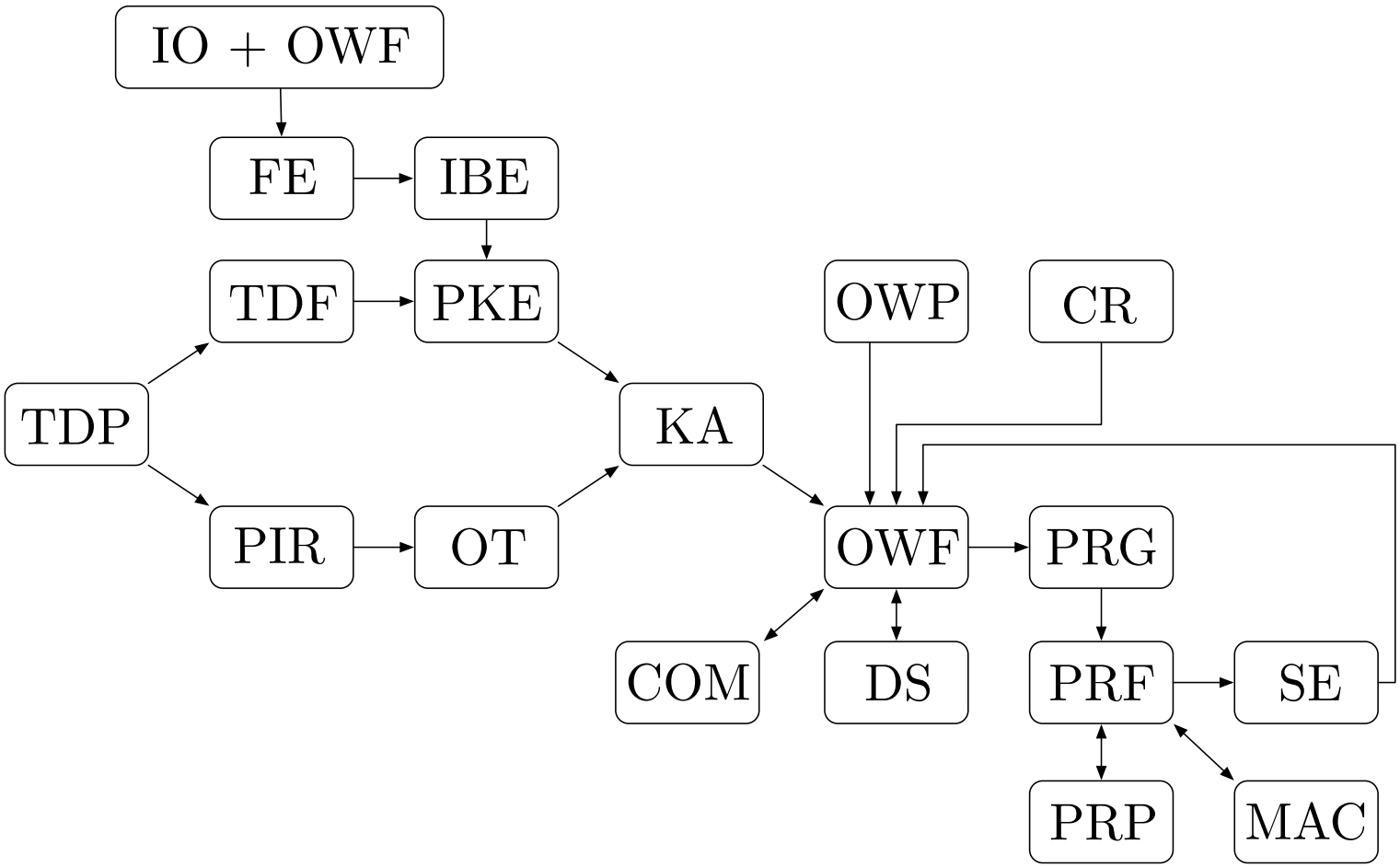
Exp.	pk	c_1	c_2	**Remark**
0	g, g^x	g^y	$m_b \cdot g^{xy}$	
1	g, g^x	g^y	$\boxed{m_b \cdot g^r}$	DDH Assumption
				Adv $= 0$

Landscape I



OWF is the weakest primitive, and hence the safest assumption.

Landscape II



Conclusion: Definitions

- Provide abstraction boundaries and enable a mathematical treatment

Conclusion: Definitions

- Provide abstraction boundaries and enable a mathematical treatment
- Direct our design goals

Conclusion: Definitions

- Provide abstraction boundaries and enable a mathematical treatment
- Direct our design goals
- Make it easier to come up with attacks

Conclusion: Definitions

- Provide abstraction boundaries and enable a mathematical treatment
- Direct our design goals
- Make it easier to come up with attacks
- Allow comparison of protocols beyond efficiency

Conclusion: Definitions

- Provide abstraction boundaries and enable a mathematical treatment
- Direct our design goals
- Make it easier to come up with attacks
- Allow comparison of protocols beyond efficiency

Definitions shape and direct the way we think.

Conclusion: Definitions

- Provide abstraction boundaries and enable a mathematical treatment
- Direct our design goals
- Make it easier to come up with attacks
- Allow comparison of protocols beyond efficiency

Definitions shape and direct the way we think. 

Conclusion: Significance of Proofs

Conclusion: Significance of Proofs

- Lots of protocols are broken . . .
- Yet AES, ECDLP, etc. have not been broken.

Conclusion: Significance of Proofs

- Lots of protocols are broken . . .
- Yet AES, ECDLP, etc. have not been broken.
- Proofs allow protocols to **inherit** this strength.

Conclusion: Significance of Proofs

- Lots of protocols are broken ...
- Yet AES, ECDLP, etc. have not been broken.
- Proofs allow protocols to **inherit** this strength.
- And tell us **how much** strength is inherited.

Conclusion: Significance of Proofs

- Lots of protocols are broken ...
- Yet AES, ECDLP, etc. have not been broken.
- Proofs allow protocols to **inherit** this strength.
- And tell us **how much** strength is inherited.

Nowadays provable security is required to support standardization.
(e.g., by ISO, IETF, CAESAR, etc.)

Conclusion: Significance of Proofs

- Lots of protocols are broken ...
- Yet AES, ECDLP, etc. have not been broken.
- Proofs allow protocols to **inherit** this strength.
- And tell us **how much** strength is inherited.

Nowadays provable security is required to support standardization.
(e.g., by ISO, IETF, CAESAR, etc.)

Caveats:

- Provable security never proves security in an absolute sense.
- Security is always relative to the assumptions and the model.
- Definitions, assumptions, constructions and proofs evolve.

What to Look for?

1. Are syntax, correctness, and security precisely defined?

What to Look for?

1. Are syntax, correctness, and security precisely defined?
2. Is the security model strong/realistic enough?
3. Are the assumptions safe? Are they weak, simple, compact, and well-studied?

What to Look for?

1. Are syntax, correctness, and security precisely defined?
2. Is the security model strong/realistic enough?
3. Are the assumptions safe? Are they weak, simple, compact, and well-studied?
4. How is the efficiency of the construction?
5. How is the tightness of the security reduction? Does it give good key sizes?

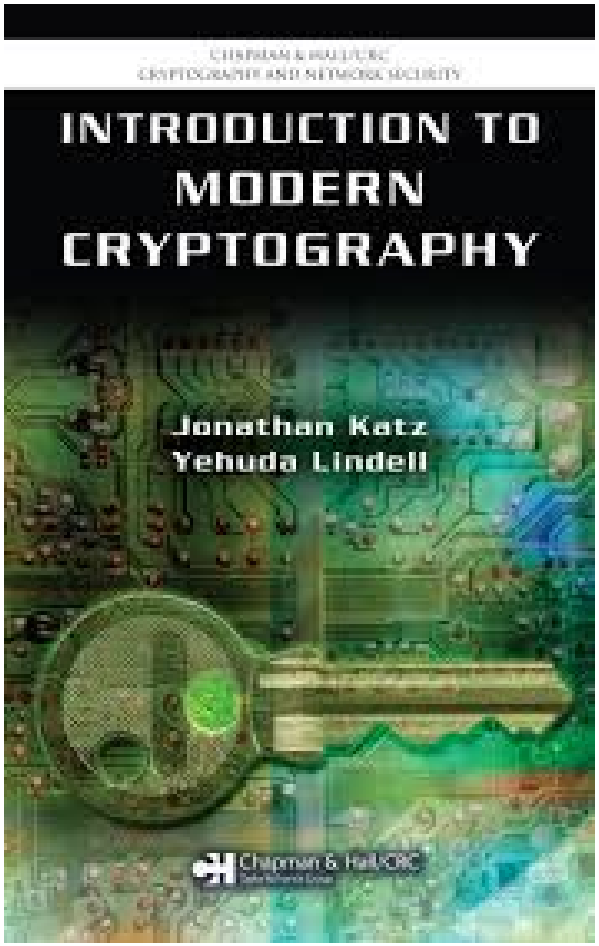
What to Look for?

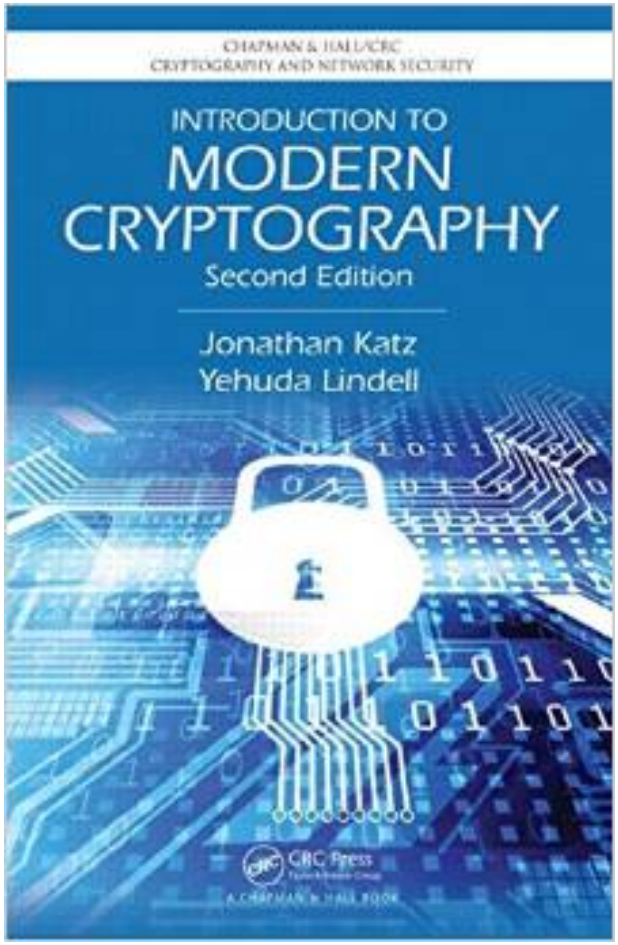
1. Are syntax, correctness, and security precisely defined?
2. Is the security model strong/realistic enough?
3. Are the assumptions safe? Are they weak, simple, compact, and well-studied?
4. How is the efficiency of the construction?
5. How is the tightness of the security reduction? Does it give good key sizes?
6. Are the relations (that is, implications and separations) among security notions studied?

What to Look for?

1. Are syntax, correctness, and security precisely defined?
2. Is the security model strong/realistic enough?
3. Are the assumptions safe? Are they weak, simple, compact, and well-studied?
4. How is the efficiency of the construction?
5. How is the tightness of the security reduction? Does it give good key sizes?
6. Are the relations (that is, implications and separations) among security notions studied?
7. And of course: is the crypto task well-motivated?

Further Reading





Further Viewing

Further Reading

1998 Practice-Oriented Provable-Security. (Bellare)

2004 On the Role of Definitions in and Beyond Cryptography. (Rogaway)

2004 Another look at "Provable Security". (Kobliz & Menezes)

2006 On Post-Modern Cryptography. (Goldreich)

2009 Practice-Oriented Provable Security and the Social Construction of Cryptography. (Rogaway)

2015 The Moral Character of Cryptographic Work. (Rogaway)

Further Reading

1998 Practice-Oriented Provable-Security. (Bellare)

2004 On the Role of Definitions in and Beyond Cryptography. (Rogaway)

2004 Another look at "Provable Security". (Kobliz & Menezes)

2006 On Post-Modern Cryptography. (Goldreich)

2009 Practice-Oriented Provable Security and the Social Construction of Cryptography. (Rogaway)

2015 The Moral Character of Cryptographic Work. (Rogaway)

Thanks.