Cryptology

Sabyasachi Karati

Assistant Professor
Cryptology and Security Research Unit (C.S.R.U)
R. C. Bose Centre for Cryptology and Security
Indian Statistical Institute (ISI)
Kolkata, India

Lecture 12

Public Key Encryption

Introduction

- Symmetric key Encryption needs a shared secret key prepared ahead of time.

Introduction

- Symmetric key Encryption needs a shared secret key prepared ahead of time.
- In Public key Encryption, that is not required.

Introduction

- Symmetric key Encryption needs a shared secret key prepared ahead of time.
- In Public key Encryption, that is not required.
- Each user has two keys: (sk, pk).

Introduction

- Symmetric key Encryption needs a shared secret key prepared ahead of time.
- In Public key Encryption, that is not required.
- Each user has two keys: (sk, pk).
  - sk is the private key of the user,
  - pk is the public key of the user,
  - It must be hard to compute sk from pk.

Introduction

- Symmetric key Encryption needs a shared secret key prepared ahead of time.
- In Public key Encryption, that is not required.
- Each user has two keys: (sk, pk).
  - sk is the private key of the user,
  - pk is the public key of the user,
  - It must be hard to compute sk from pk.
- Asymmetric: Encryption and decryption keys are different.

Introduction

- Symmetric key Encryption needs a shared secret key prepared ahead of time.
- In Public key Encryption, that is not required.
- Each user has two keys: (sk, pk).
  - sk is the private key of the user,
  - pk is the public key of the user,
  - It must be hard to compute sk from pk.
- Asymmetric: Encryption and decryption keys are different.
  - Any one can encrypt a message using the receiver's public key as $c \xleftarrow{R} \mathcal{E}(pk, m)$.

Introduction

- Symmetric key Encryption needs a shared secret key prepared ahead of time.
- In Public key Encryption, that is not required.
- Each user has two keys: (sk, pk).
  - sk is the private key of the user,
  - pk is the public key of the user,
  - It must be hard to compute sk from pk.
- Asymmetric: Encryption and decryption keys are different.
  - Any one can encrypt a message using the receiver's public key as $c \xleftarrow{R} \mathcal{E}(pk, m)$.
  - Only receiver can decrypt it using its private key as $m \longleftarrow \mathcal{D}(sk, c)$.

Introduction

- Symmetric key Encryption needs a shared secret key prepared ahead of time.
- In Public key Encryption, that is not required.
- Each user has two keys: (sk, pk).
  - sk is the private key of the user,
  - pk is the public key of the user,
  - It must be hard to compute sk from pk.
- Asymmetric: Encryption and decryption keys are different.
  - Any one can encrypt a message using the receiver's public key as $c \xleftarrow{R} \mathcal{E}(pk, m)$.
  - Only receiver can decrypt it using its private key as $m \longleftarrow \mathcal{D}(sk, c)$.
- Public key is known by every one even by the adversary.

Introduction

- Public key cryptosystem is 2-3 times slower than symmetric key cryptography.

Introduction

- Public key cryptosystem is 2-3 times slower than symmetric key cryptography.
- In real world, we use a mixture of both.

Introduction

- Public key cryptosystem is 2-3 times slower than symmetric key cryptography.
- In real world, we use a mixture of both.
- Example:
  - Assume Alice wants to share a movie f with her 10 friends.

Introduction

- Public key cryptosystem is 2-3 times slower than symmetric key cryptography.
- In real world, we use a mixture of both.
- Example:
  - Assume Alice wants to share a movie f with her 10 friends.
  - Symmetric key: Alice will choose a key k and will encrypt the movie by it as $c_f \xleftarrow{R} \mathcal{E}_s(k, f)$.

Introduction

- Public key cryptosystem is 2-3 times slower than symmetric key cryptography.
- In real world, we use a mixture of both.
- Example:
  - Assume Alice wants to share a movie f with her 10 friends.
  - Symmetric key: Alice will choose a key k and will encrypt the movie by it as $c_f \xleftarrow{R} \mathcal{E}_s(k, f)$.
  - Public Key: for each friend,
    - Alice will use the public key pk_i of friend i to encrypt the key k as $c_i \xleftarrow{R} \mathcal{E}_p(pk_i, k)$.

Introduction

- Public key cryptosystem is 2-3 times slower than symmetric key cryptography.
- In real world, we use a mixture of both.
- Example:
  - Assume Alice wants to share a movie f with her 10 friends.
  - Symmetric key: Alice will choose a key k and will encrypt the movie by it as $c_f \xleftarrow{R} \mathcal{E}_s(k, f)$.
  - Public Key: for each friend,
    - Alice will use the public key pk_i of friend i to encrypt the key k as $c_i \xleftarrow{R} \mathcal{E}_p(pk_i, k)$.
    - Sends (c_i, c_f) to friend i.

Introduction

- We have implicitly assumed that the adversary is passive.

Introduction

- We have implicitly assumed that the adversary is passive.
- The adversary only eavesdrops but does not actively interfere.

Introduction

- We have implicitly assumed that the adversary is passive.
- The adversary only eavesdrops but does not actively interfere.
- privacy simply cannot be achieved
  - adversary has the ability to tamper with all communication
  - honest parties hold no shared keys.

Introduction

- We have implicitly assumed that the adversary is passive.
- The adversary only eavesdrops but does not actively interfere.
- privacy simply cannot be achieved
  - adversary has the ability to tamper with all communication
  - honest parties hold no shared keys.
- Assume that senders have a legitimate copy of the receiver's public key.

Introduction

- We have implicitly assumed that the adversary is passive.
- The adversary only eavesdrops but does not actively interfere.
- privacy simply cannot be achieved
  - adversary has the ability to tamper with all communication
  - honest parties hold no shared keys.
- Assume that senders have a legitimate copy of the receiver's public key.
- We assume secure key distribution.

Public Key Encryption

Definition

A public-key encryption scheme $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a triple of efficient algorithms: a key generation algorithm $\mathcal{G}$, an encryption algorithm $\mathcal{E}$, a decryption algorithm $\mathcal{D}$.

Public Key Encryption

Definition

A public-key encryption scheme $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a triple of efficient algorithms: a key generation algorithm $\mathcal{G}$, an encryption algorithm $\mathcal{E}$, a decryption algorithm $\mathcal{D}$.

- $\mathcal{G}$ is a probabilistic polytime algorithm that is invoked as $(sk, pk) \xleftarrow{R} \mathcal{G}()$, where pk is called a public key and sk is called a secret key.

Public Key Encryption

Definition

A public-key encryption scheme $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a triple of efficient algorithms: a key generation algorithm $\mathcal{G}$, an encryption algorithm $\mathcal{E}$, a decryption algorithm $\mathcal{D}$.

- $\mathcal{G}$ is a probabilistic polytime algorithm that is invoked as $(sk, pk) \xleftarrow{R} \mathcal{G}()$, where pk is called a public key and sk is called a secret key.
- $\mathcal{E}$ is a probabilistic polytime algorithm that is invoked as $c \xleftarrow{R} \mathcal{E}(pk, m)$, where pk is a public key (as output by $\mathcal{G}$), m is a message, and c is a ciphertext.

Public Key Encryption

Definition

A public-key encryption scheme $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a triple of efficient algorithms: a key generation algorithm $\mathcal{G}$, an encryption algorithm $\mathcal{E}$, a decryption algorithm $\mathcal{D}$.

- $\mathcal{G}$ is a probabilistic polytime algorithm that is invoked as $(sk, pk) \xleftarrow{R} \mathcal{G}()$, where pk is called a public key and sk is called a secret key.
- $\mathcal{E}$ is a probabilistic polytime algorithm that is invoked as $c \xleftarrow{R} \mathcal{E}(pk, m)$, where pk is a public key (as output by $\mathcal{G}$), m is a message, and c is a ciphertext.
- $\mathcal{D}$ is a deterministic polytime algorithm that is invoked as $m \longleftarrow \mathcal{D}(sk, c)$, where sk is a secret key (as output by $\mathcal{G}$), c is a ciphertext, and m is either a message, or a special reject value (distinct from all messages).

Public Key Encryption

Definition

A public-key encryption scheme $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a triple of efficient algorithms: a key generation algorithm $\mathcal{G}$, an encryption algorithm $\mathcal{E}$, a decryption algorithm $\mathcal{D}$.

- $\mathcal{G}$ is a probabilistic polytime algorithm that is invoked as $(sk, pk) \xleftarrow{R} \mathcal{G}()$, where pk is called a public key and sk is called a secret key.
- $\mathcal{E}$ is a probabilistic polytime algorithm that is invoked as $c \xleftarrow{R} \mathcal{E}(pk, m)$, where pk is a public key (as output by $\mathcal{G}$), m is a message, and c is a ciphertext.
- $\mathcal{D}$ is a deterministic polytime algorithm that is invoked as $m \longleftarrow \mathcal{D}(sk, c)$, where sk is a secret key (as output by $\mathcal{G}$), c is a ciphertext, and m is either a message, or a special reject value (distinct from all messages).
- For all possible outputs (pk, sk) of $\mathcal{G}$, and all messages m, we have

$$\Pr[\mathcal{D}(sk, \mathcal{E}(pk, m)) = m] = 1.$$

Public Key Encryption

Definition

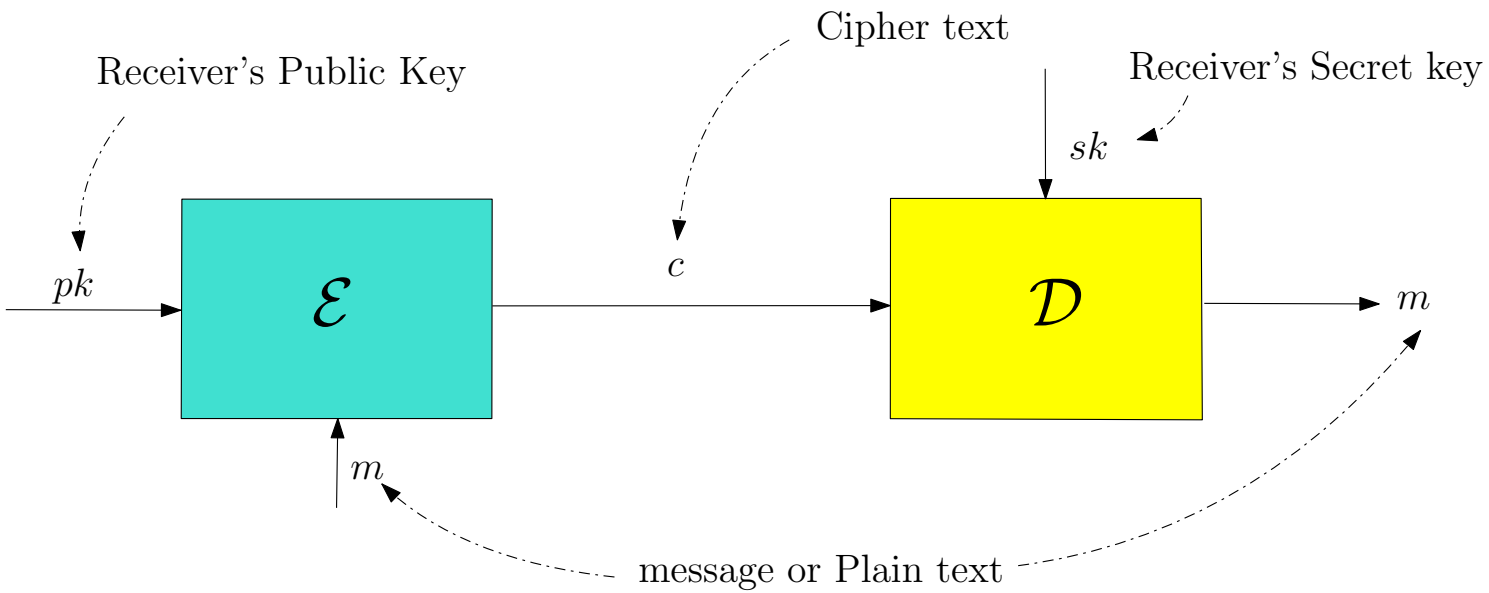
A public-key encryption scheme $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a triple of efficient algorithms: a key generation algorithm $\mathcal{G}$, an encryption algorithm $\mathcal{E}$, a decryption algorithm $\mathcal{D}$.

- $\mathcal{G}$ is a probabilistic polytime algorithm that is invoked as $(sk, pk) \xleftarrow{R} \mathcal{G}()$, where pk is called a public key and sk is called a secret key.
- $\mathcal{E}$ is a probabilistic polytime algorithm that is invoked as $c \xleftarrow{R} \mathcal{E}(pk, m)$, where pk is a public key (as output by $\mathcal{G}$), m is a message, and c is a ciphertext.
- $\mathcal{D}$ is a deterministic polytime algorithm that is invoked as $m \longleftarrow \mathcal{D}(sk, c)$, where sk is a secret key (as output by $\mathcal{G}$), c is a ciphertext, and m is either a message, or a special reject value (distinct from all messages).
- For all possible outputs (pk, sk) of $\mathcal{G}$, and all messages m, we have

$$\Pr[\mathcal{D}(sk, \mathcal{E}(pk, m)) = m] = 1.$$

- Messages are assumed to lie in some finite message space $\mathcal{M}$, and ciphertexts in some finite ciphertext space C. We say that $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ is defined over $(\mathcal{M}, C)$.

Public Key Encryption



Indistinguishability

Indistinguishability Game

For a given public key encryption $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ defined over $(\mathcal{M}, \mathcal{C})$ with security parameter $n \in \mathbb{N}$, and for a given adversary $\mathcal{A}$, we define two experiments: Experiment 0 and Experiment 1. For $b = 0, 1$, we define Experiment b as:

- The challenger computes $(sk, pk) \xleftarrow{R} \mathcal{G}()$, and sends pk to the adversary.

Indistinguishability

Indistinguishability Game

For a given public key encryption $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ defined over $(\mathcal{M}, \mathcal{C})$ with security parameter $n \in \mathbb{N}$, and for a given adversary $\mathcal{A}$, we define two experiments: Experiment 0 and Experiment 1. For $b = 0, 1$, we define Experiment b as:

- The challenger computes $(sk, pk) \xleftarrow{R} \mathcal{G}()$, and sends pk to the adversary.
- The adversary computes $m_0, m_1 \in \mathcal{M}$, of the same length, and sends them to the challenger.

Indistinguishability

Indistinguishability Game

For a given public key encryption $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ defined over $(\mathcal{M}, C)$ with security parameter $n \in \mathbb{N}$, and for a given adversary $\mathcal{A}$, we define two experiments: Experiment 0 and Experiment 1. For $b = 0, 1$, we define Experiment b as:

- The challenger computes $(sk, pk) \xleftarrow{R} \mathcal{G}()$, and sends pk to the adversary.
- The adversary computes $m_0, m_1 \in \mathcal{M}$, of the same length, and sends them to the challenger.
- The challenger computes $c \xleftarrow{R} \mathcal{E}(pk, m_b)$, and sends c to the adversary.

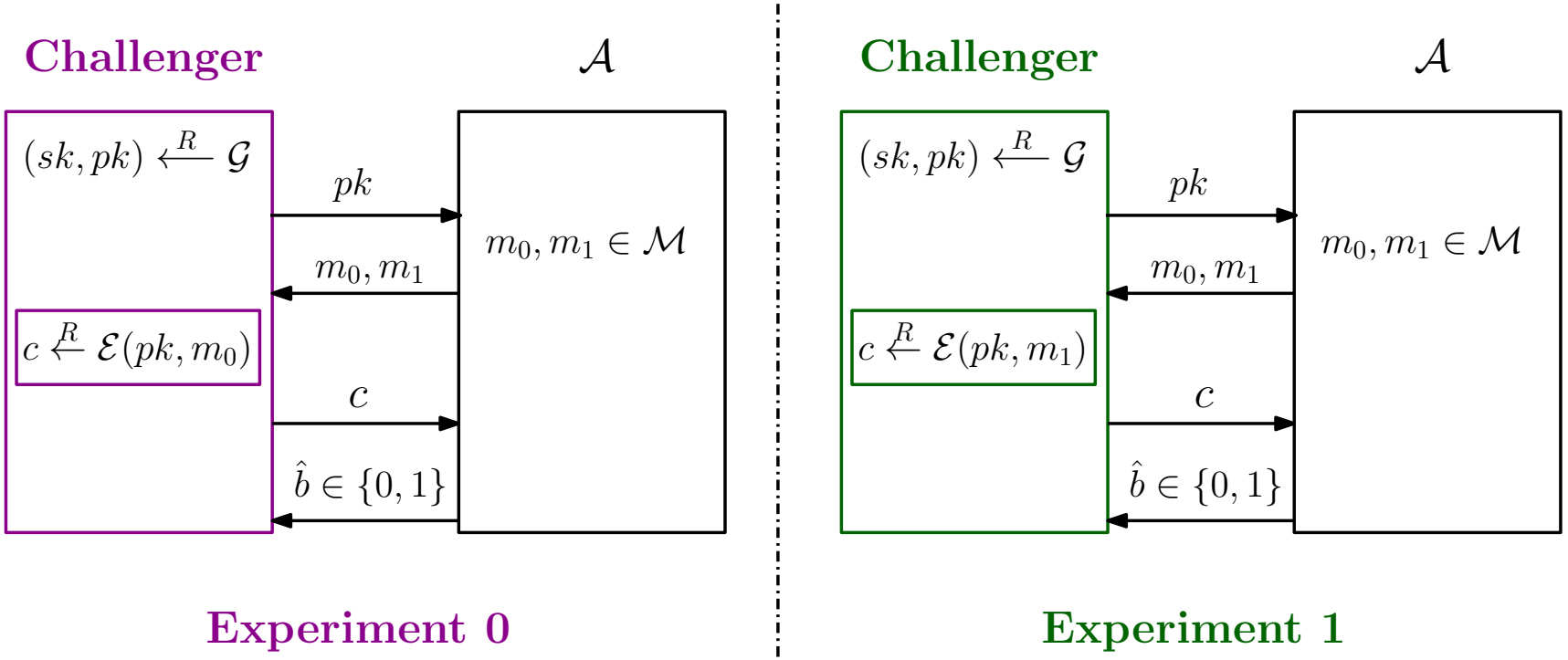
Indistinguishability

Indistinguishability Game

For a given public key encryption $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ defined over $(\mathcal{M}, C)$ with security parameter $n \in \mathbb{N}$, and for a given adversary $\mathcal{A}$, we define two experiments: Experiment 0 and Experiment 1. For $b = 0, 1$, we define Experiment b as:

- The challenger computes $(sk, pk) \xleftarrow{R} \mathcal{G}()$, and sends pk to the adversary.
- The adversary computes $m_0, m_1 \in \mathcal{M}$, of the same length, and sends them to the challenger.
- The challenger computes $c \xleftarrow{R} \mathcal{E}(pk, m_b)$, and sends c to the adversary.
- The adversary outputs a bit $\hat{b} \in \{0, 1\}$.

Indistinguishability



Experiment 0 **Experiment 1**

Indistinguishability

Indistinguishability Advantage

For $b = 0, 1$, let W_b be the event that $\mathcal{A}$ outputs 1 in Experiment b. We define the advantage of $\mathcal{A}$ in the Indistinguishability attack with respect to $\mathfrak{E}$ as

$$\mathsf{INDadv}[\mathcal{A}, \mathfrak{E}] = |\Pr[W_0] - \Pr[W_1]|.$$

Indistinguishability

Indistinguishability Advantage

For $b = 0, 1$, let W_b be the event that $\mathcal{A}$ outputs 1 in Experiment b. We define the advantage of $\mathcal{A}$ in the Indistinguishability attack with respect to $\mathfrak{E}$ as

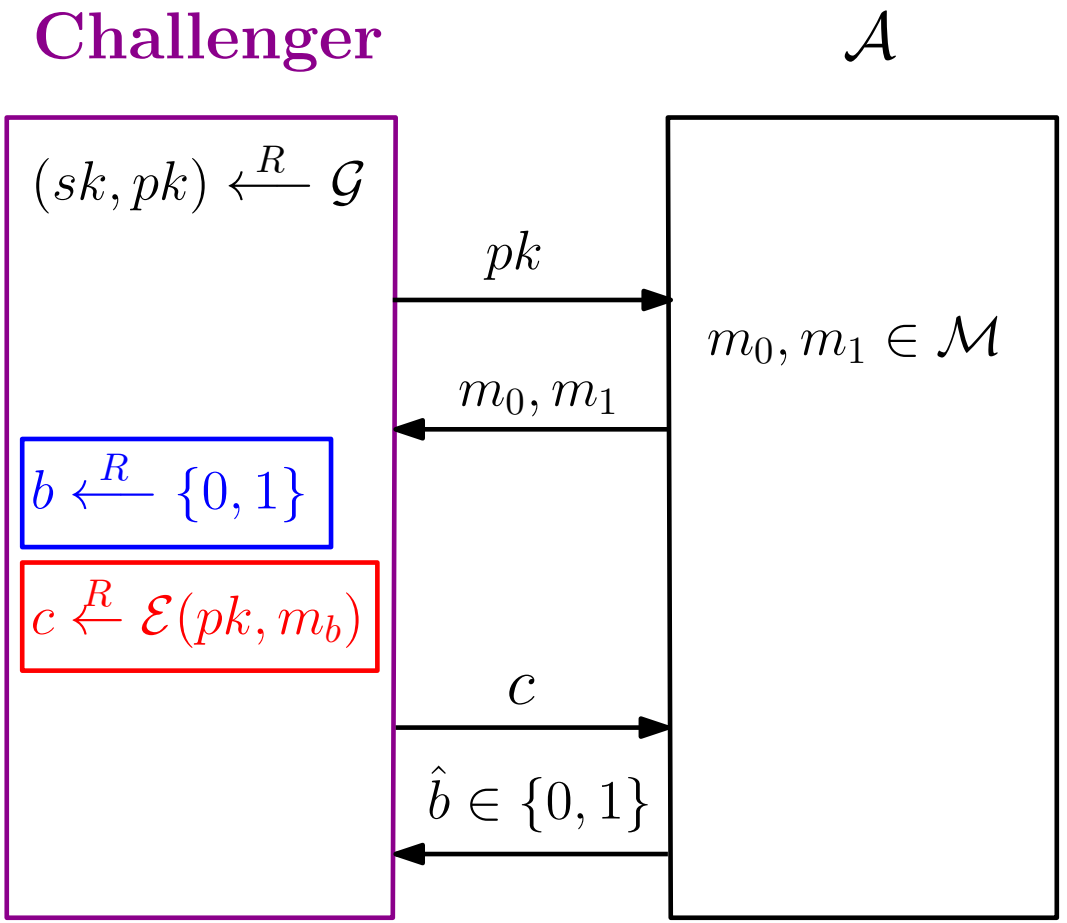
$$\mathsf{INDadv}[\mathcal{A}, \mathfrak{E}] = |\Pr[W_0] - \Pr[W_1]|.$$

Indistinguishable

We say $\mathfrak{E}$ is (computationally) indistinguishable in the presence of an eavesdropper if for all PPT adversaries $\mathcal{A}$ there exists a negligible function ϵ such that

$$\mathsf{INDadv}[\mathcal{A}, \mathfrak{E}] \leqslant \epsilon(n).$$

Indistinguishability Advantage: Bit Guessing version



Experiment

Indistinguishability Advantage: Bit Guessing version

Indistinguishability Advantage

Let W be the event that where $\mathcal{A}$ wins if $\mathcal{A}$ outputs $\hat{b} = b$. We define the advantage of $\mathcal{A}$ in the attack game with respect to $\mathfrak{E}$ as

$$\mathsf{INDadv}^{*}[\mathcal{A}, \mathfrak{E}] = \left| \Pr[\hat{b} = b] - \frac{1}{2} \right|.$$

Indistinguishability Advantage: Bit Guessing version

Indistinguishability Advantage

Let W be the event that where $\mathcal{A}$ wins if $\mathcal{A}$ outputs $\hat{b} = b$. We define the advantage of $\mathcal{A}$ in the attack game with respect to $\mathfrak{E}$ as

$$\mathsf{INDadv}^*[\mathcal{A}, \mathfrak{E}] = \left| \Pr[\hat{b} = b] - \frac{1}{2} \right|.$$

Theorem

For every public-key encryption scheme $\mathfrak{E}$ and every PPT adversary $\mathcal{A}$, we have

$$\mathsf{INDadv}[\mathcal{A}, \mathfrak{E}] = 2 \cdot \mathsf{INDadv}^*[\mathcal{A}, \mathfrak{E}].$$

The need for Randomized Encryption

Randomized Encryption

Suppose $\mathcal{E}$ is deterministic. Then the following adversary $\mathcal{A}$ breaks Indistinguishability of $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$:

The need for Randomized Encryption

Randomized Encryption

Suppose $\mathcal{E}$ is deterministic. Then the following adversary $\mathcal{A}$ breaks Indistinguishability of $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$:

- $\mathcal{A}$ receives a public key pk from its challenger.

The need for Randomized Encryption

Randomized Encryption

Suppose $\mathcal{E}$ is deterministic. Then the following adversary $\mathcal{A}$ breaks Indistinguishability of $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$:

- $\mathcal{A}$ receives a public key pk from its challenger.
- $\mathcal{A}$ chooses two distinct messages m_0 and m_1 in $\mathcal{M}$ and sends them to its challenger.

The need for Randomized Encryption

Randomized Encryption

Suppose $\mathcal{E}$ is deterministic. Then the following adversary $\mathcal{A}$ breaks Indistinguishability of $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$:

- $\mathcal{A}$ receives a public key pk from its challenger.
- $\mathcal{A}$ chooses two distinct messages m_0 and m_1 in $\mathcal{M}$ and sends them to its challenger.
- The challenger responds with $c := \mathcal{E}(pk, m_b)$ for some $b \in \{0, 1\}$.

The need for Randomized Encryption

Randomized Encryption

Suppose $\mathcal{E}$ is deterministic. Then the following adversary $\mathcal{A}$ breaks Indistinguishability of $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$:

- $\mathcal{A}$ receives a public key pk from its challenger.
- $\mathcal{A}$ chooses two distinct messages m_0 and m_1 in $\mathcal{M}$ and sends them to its challenger.
- The challenger responds with $c := \mathcal{E}(pk, m_b)$ for some $b \in \{0, 1\}$.
- $\mathcal{A}$ computes $c_0 := \mathcal{E}(pk, m_0)$ and outputs 0 if $c = c_0$. Otherwise, it outputs 1.

The need for Randomized Encryption

Randomized Encryption

Suppose $\mathcal{E}$ is deterministic. Then the following adversary $\mathcal{A}$ breaks Indistinguishability of $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$:

- $\mathcal{A}$ receives a public key pk from its challenger.
- $\mathcal{A}$ chooses two distinct messages m_0 and m_1 in $\mathcal{M}$ and sends them to its challenger.
- The challenger responds with $c := \mathcal{E}(pk, m_b)$ for some $b \in \{0, 1\}$.
- $\mathcal{A}$ computes $c_0 := \mathcal{E}(pk, m_0)$ and outputs 0 if $c = c_0$. Otherwise, it outputs 1.

Success

- $\mathcal{E}$ is deterministic $\Rightarrow c = c_0$ whenever $b = 0$.

The need for Randomized Encryption

Randomized Encryption

Suppose $\mathcal{E}$ is deterministic. Then the following adversary $\mathcal{A}$ breaks Indistinguishability of $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$:

- $\mathcal{A}$ receives a public key pk from its challenger.
- $\mathcal{A}$ chooses two distinct messages m_0 and m_1 in $\mathcal{M}$ and sends them to its challenger.
- The challenger responds with $c := \mathcal{E}(pk, m_b)$ for some $b \in \{0, 1\}$.
- $\mathcal{A}$ computes $c_0 := \mathcal{E}(pk, m_0)$ and outputs 0 if $c = c_0$. Otherwise, it outputs 1.

Success

- $\mathcal{E}$ is deterministic $\Rightarrow c = c_0$ whenever $b = 0$.
- When $b = 0$ the adversary always outputs 0.

The need for Randomized Encryption

Randomized Encryption

Suppose $\mathcal{E}$ is deterministic. Then the following adversary $\mathcal{A}$ breaks Indistinguishability of $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$:

- $\mathcal{A}$ receives a public key pk from its challenger.
- $\mathcal{A}$ chooses two distinct messages m_0 and m_1 in $\mathcal{M}$ and sends them to its challenger.
- The challenger responds with $c := \mathcal{E}(pk, m_b)$ for some $b \in \{0, 1\}$.
- $\mathcal{A}$ computes $c_0 := \mathcal{E}(pk, m_0)$ and outputs 0 if $c = c_0$. Otherwise, it outputs 1.

Success

- $\mathcal{E}$ is deterministic $\Rightarrow c = c_0$ whenever $b = 0$.
- When $b = 0$ the adversary always outputs 0.
- Similarly, when $b = 1$ it always outputs 1.

The need for Randomized Encryption

Randomized Encryption

Suppose $\mathcal{E}$ is deterministic. Then the following adversary $\mathcal{A}$ breaks Indistinguishability of $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$:

- $\mathcal{A}$ receives a public key pk from its challenger.
- $\mathcal{A}$ chooses two distinct messages m_0 and m_1 in $\mathcal{M}$ and sends them to its challenger.
- The challenger responds with $c := \mathcal{E}(pk, m_b)$ for some $b \in \{0, 1\}$.
- $\mathcal{A}$ computes $c_0 := \mathcal{E}(pk, m_0)$ and outputs 0 if $c = c_0$. Otherwise, it outputs 1.

Success

- $\mathcal{E}$ is deterministic $\Rightarrow c = c_0$ whenever $b = 0$.
- When $b = 0$ the adversary always outputs 0.
- Similarly, when $b = 1$ it always outputs 1.

$$\mathsf{INDadv}[\mathcal{A}, \mathfrak{E}] = 1.$$

CPA Security

CPA Security Game

For a given public-key encryption $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ defined over $(\mathcal{M}, \mathcal{C})$ with security parameter $n \in \mathbb{N}$, and for a given adversary $\mathcal{A}$, we define two experiments: Experiment 0 and Experiment 1. For $b = 0, 1$, we define Experiment b as:

CPA Security

CPA Security Game

For a given public-key encryption $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ defined over $(\mathcal{M}, \mathcal{C})$ with security parameter $n \in \mathbb{N}$, and for a given adversary $\mathcal{A}$, we define two experiments: Experiment 0 and Experiment 1. For $b = 0, 1$, we define Experiment b as:

- The challenger computes $(sk, pk) \xleftarrow{R} \mathcal{G}()$, and sends pk to the adversary.

CPA Security

CPA Security Game

For a given public-key encryption $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ defined over $(\mathcal{M}, \mathcal{C})$ with security parameter $n \in \mathbb{N}$, and for a given adversary $\mathcal{A}$, we define two experiments: Experiment 0 and Experiment 1. For $b = 0, 1$, we define Experiment b as:

- The challenger computes $(sk, pk) \xleftarrow{R} \mathcal{G}()$, and sends pk to the adversary.
- The adversary submits a sequence of queries to the challenger.

CPA Security

CPA Security Game

For a given public-key encryption $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ defined over $(\mathcal{M}, C)$ with security parameter $n \in \mathbb{N}$, and for a given adversary $\mathcal{A}$, we define two experiments: Experiment 0 and Experiment 1. For $b = 0, 1$, we define Experiment b as:

- The challenger computes $(sk, pk) \xleftarrow{R} \mathcal{G}()$, and sends pk to the adversary.
- The adversary submits a sequence of queries to the challenger.
  - For $i = 1, 2, \ldots$, the ith query is a pair of messages, $m_{i0}, m_{i1} \in \mathcal{M}$ of the same length, and sends them to the challenger.

CPA Security

CPA Security Game

For a given public-key encryption $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ defined over $(\mathcal{M}, C)$ with security parameter $n \in \mathbb{N}$, and for a given adversary $\mathcal{A}$, we define two experiments: Experiment 0 and Experiment 1. For $b = 0, 1$, we define Experiment b as:

- The challenger computes $(sk, pk) \xleftarrow{R} \mathcal{G}()$, and sends pk to the adversary.
- The adversary submits a sequence of queries to the challenger.
  - For $i = 1, 2, \ldots$, the ith query is a pair of messages, $m_{i0}, m_{i1} \in \mathcal{M}$ of the same length, and sends them to the challenger.
  - The challenger computes $c_i \xleftarrow{R} \mathcal{E}(pk, m_{ib})$, and sends c_i to the adversary.

CPA Security

CPA Security Game

For a given public-key encryption $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ defined over $(\mathcal{M}, C)$ with security parameter $n \in \mathbb{N}$, and for a given adversary $\mathcal{A}$, we define two experiments: Experiment 0 and Experiment 1. For $b = 0, 1$, we define Experiment b as:

- The challenger computes $(sk, pk) \xleftarrow{R} \mathcal{G}()$, and sends pk to the adversary.
- The adversary submits a sequence of queries to the challenger.
  - For $i = 1, 2, \ldots$, the ith query is a pair of messages, $m_{i0}, m_{i1} \in \mathcal{M}$ of the same length, and sends them to the challenger.
  - The challenger computes $c_i \xleftarrow{R} \mathcal{E}(pk, m_{ib})$, and sends c_i to the adversary.
- The adversary outputs a bit $\hat{b} \in \{0, 1\}$.

CPA Security

CPA Security Advantage

For $b = 0, 1$, let W_b be the event that $\mathcal{A}$ outputs 1 in Experiment b. We define the advantage of $\mathcal{A}$ in the CPA Security attack with respect to $\mathfrak{E}$ as

$$\mathsf{CPAadv}[\mathcal{A}, \mathfrak{E}] = |\Pr[W_0] - \Pr[W_1]|.$$

CPA Security

CPA Security Advantage

For $b = 0, 1$, let W_b be the event that $\mathcal{A}$ outputs 1 in Experiment b. We define the advantage of $\mathcal{A}$ in the CPA Security attack with respect to $\mathfrak{E}$ as

$$\mathsf{CPAadv}[\mathcal{A}, \mathfrak{E}] = |\Pr[W_0] - \Pr[W_1]|.$$

CPA Secure

We say $\mathfrak{E}$ is CPA Secure in the presence of an eavesdropper if for all PPT adversaries $\mathcal{A}$ there exists a negligible function ϵ such that

$$\mathsf{CPAadv}[\mathcal{A}, \mathfrak{E}] \leqslant \epsilon(n).$$

CPA Security

Theorem

If a public-key encryption scheme $\mathfrak{E}$ is Indistinguishabile, then it is also CPA secure.

CPA Security

Theorem

If a public-key encryption scheme $\mathfrak{E}$ is Indistinguishabile, then it is also CPA secure.

In particular, for every CPA adversary $\mathcal{A}$ that plays CPA Attack Game with respect to $\mathfrak{E}$, and which makes at most Q queries to its challenger, there exists an Indistinguishability adversary $\mathcal{B}$, where $\mathcal{B}$ is an elementary wrapper around $\mathcal{A}$, such that

$$\mathsf{CPAadv}[\mathcal{A}, \mathfrak{E}] = Q \cdot \mathsf{INDadv}[\mathcal{B}, \mathfrak{E}].$$

Text-Book RSA

Key Generation

1. Let the security parameter be n.

Text-Book RSA

Key Generation

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.

Text-Book RSA

Key Generation

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.

Text-Book RSA

Key Generation

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
4. Choose e such that $\gcd(e, \phi(N)) = 1$.

Text-Book RSA

Key Generation

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
4. Choose e such that $\gcd(e, \phi(N)) = 1$.
5. Compute $d \equiv e^{-1} \pmod{\phi(N)}$.

Text-Book RSA

Key Generation

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
4. Choose e such that $\gcd(e, \phi(N)) = 1$.
5. Compute $d \equiv e^{-1} \pmod{\phi(N)}$.
6. $sk = (N, d)$ and $pk = (N, e)$.

Text-Book RSA

Key Generation

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
4. Choose e such that $\gcd(e, \phi(N)) = 1$.
5. Compute $d \equiv e^{-1} \pmod{\phi(N)}$.
6. $sk = (N, d)$ and $pk = (N, e)$.

$\mathcal{E}(pk, m \in \mathbb{Z}_N)$

1. Compute $c \longleftarrow m^e \pmod{N}$.

Text-Book RSA

Key Generation

1. Let the security parameter be n.
2. Choose two primes p and q of bit-length almost n.
3. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
4. Choose e such that $\gcd(e, \phi(N)) = 1$.
5. Compute $d \equiv e^{-1} \pmod{\phi(N)}$.
6. $sk = (N, d)$ and $pk = (N, e)$.

$\mathcal{E}(pk, m \in \mathbb{Z}_N)$

1. Compute $c \longleftarrow m^e \pmod{N}$.

$\mathcal{D}(sk, c \in \mathbb{Z}_N)$

1. Compute $m \longleftarrow c^d \pmod{N}$.

Text-Book RSA

Correctness

1. $ed \equiv 1 \pmod{\phi(N)} \Leftrightarrow ed = t\phi(N) + 1$ for some $t \in \mathbb{Z}$.

Text-Book RSA

Correctness

1. $ed \equiv 1 \pmod{\phi(N)} \Leftrightarrow ed = t\phi(N) + 1$ for some $t \in \mathbb{Z}$.
2. If $m \in \mathbb{Z}_N^*$, then by Euler's theorem,

$$m^{ed} = m^{t\phi(N)+1} = \left(m^{\phi(N)}\right)^t \cdot m = m \pmod{N}.$$

Text-Book RSA

Correctness

1. $ed \equiv 1 \pmod{\phi(N)} \Leftrightarrow ed = t\phi(N) + 1$ for some $t \in \mathbb{Z}$.
2. If $m \in \mathbb{Z}_N^*$, then by Euler's theorem,

$$m^{ed} = m^{t\phi(N)+1} = \left(m^{\phi(N)}\right)^t \cdot m = m \pmod{N}.$$

3. If $m \in \mathbb{Z}_N$, then for some $h, k \in \mathbb{Z}$
   - $ed - 1 = t\phi(N) = t(p-1)(q-1) \Rightarrow ed - 1 = h(p-1) = k(q-1)$.

Text-Book RSA

Correctness

1. $ed \equiv 1 \pmod{\phi(N)} \Leftrightarrow ed = t\phi(N) + 1$ for some $t \in \mathbb{Z}$.
2. If $m \in \mathbb{Z}_N^*$, then by Euler's theorem,

$$m^{ed} = m^{t\phi(N)+1} = \left(m^{\phi(N)}\right)^t \cdot m = m \pmod{N}.$$

3. If $m \in \mathbb{Z}_N$, then for some $h, k \in \mathbb{Z}$
   - $ed - 1 = t\phi(N) = t(p-1)(q-1) \Rightarrow ed - 1 = h(p-1) = k(q-1)$.
     - if $m \equiv 0 \pmod{p}$, $m^{ed} = 0^{ed} = m \pmod{p}$.

Text-Book RSA

Correctness

1. $ed \equiv 1 \pmod{\phi(N)} \Leftrightarrow ed = t\phi(N) + 1$ for some $t \in \mathbb{Z}$.
2. If $m \in \mathbb{Z}_N^*$, then by Euler's theorem,

$$m^{ed} = m^{t\phi(N)+1} = \left(m^{\phi(N)}\right)^t \cdot m = m \pmod{N}.$$

3. If $m \in \mathbb{Z}_N$, then for some $h, k \in \mathbb{Z}$
   - $ed - 1 = t\phi(N) = t(p-1)(q-1) \Rightarrow ed - 1 = h(p-1) = k(q-1)$.
     - if $m \equiv 0 \pmod{p}$, $m^{ed} = 0^{ed} = m \pmod{p}$.
     - if $m \not\equiv 0 \pmod{p}$, by Fermat's Little theorem,

$$m^{ed} = m^{ed-1}m = \left(m^{p-1}\right)^h \cdot m = m \pmod{p}.$$

Text-Book RSA

Correctness

1. $ed \equiv 1 \pmod{\phi(N)} \Leftrightarrow ed = t\phi(N) + 1$ for some $t \in \mathbb{Z}$.
2. If $m \in \mathbb{Z}_N^*$, then by Euler's theorem,

$$m^{ed} = m^{t\phi(N)+1} = \left(m^{\phi(N)}\right)^t \cdot m = m \pmod{N}.$$

3. If $m \in \mathbb{Z}_N$, then for some $h, k \in \mathbb{Z}$
   - $ed - 1 = t\phi(N) = t(p-1)(q-1) \Rightarrow ed - 1 = h(p-1) = k(q-1)$.
     - if $m \equiv 0 \pmod{p}$, $m^{ed} = 0^{ed} = m \pmod{p}$.
     - if $m \not\equiv 0 \pmod{p}$, by Fermat's Little theorem,

$$m^{ed} = m^{ed-1}m = \left(m^{p-1}\right)^h \cdot m = m \pmod{p}.$$

   - Similarly, $m^{ed} = m \pmod{q}$.

Text-Book RSA

Correctness

1. $ed \equiv 1 \pmod{\phi(N)} \Leftrightarrow ed = t\phi(N) + 1$ for some $t \in \mathbb{Z}$.
2. If $m \in \mathbb{Z}_N^*$, then by Euler's theorem,

$$m^{ed} = m^{t\phi(N)+1} = \left(m^{\phi(N)}\right)^t \cdot m = m \pmod{N}.$$

3. If $m \in \mathbb{Z}_N$, then for some $h, k \in \mathbb{Z}$
   - $ed - 1 = t\phi(N) = t(p-1)(q-1) \Rightarrow ed - 1 = h(p-1) = k(q-1)$.
     - if $m \equiv 0 \pmod{p}$, $m^{ed} = 0^{ed} = m \pmod{p}$.
     - if $m \not\equiv 0 \pmod{p}$, by Fermat's Little theorem,

$$m^{ed} = m^{ed-1}m = \left(m^{p-1}\right)^h \cdot m = m \pmod{p}.$$

   - Similarly, $m^{ed} = m \pmod{q}$.
   - $m^{ed} = m \pmod{p}$ and $m^{ed} = m \pmod{q}$
     $\Rightarrow m^{ed} = m \pmod{pq} = m \pmod{N}$ by CRT.

Factoring Assumption

Factorization Problem

Let $N = p \cdot q$, where p and q be two n-bit integers. Given N, compute p and q.

Factoring Assumption

Factorization Problem

Let $N = p \cdot q$, where p and q be two n-bit integers. Given N, compute p and q.

Attack Game

Let `GenModulus` be a polynomial-time algorithm that, on input 1^n , outputs (N, p, q) where $N = pq$, and p and q are n-bit primes except with probability negligible in n.

Factoring Assumption

Factorization Problem

Let $N = p \cdot q$, where p and q be two n-bit integers. Given N, compute p and q.

Attack Game

Let `GenModulus` be a polynomial-time algorithm that, on input 1^n , outputs (N, p, q) where $N = pq$, and p and q are n-bit primes except with probability negligible in n. Then consider the following attack game for a given adversary $\mathcal{A}$ and parameter n:

- Run $\texttt{GenModulus}(1^n)$ to obtain (N, p, q).

Factoring Assumption

Factorization Problem

Let $N = p \cdot q$, where p and q be two n-bit integers. Given N, compute p and q.

Attack Game

Let `GenModulus` be a polynomial-time algorithm that, on input 1^n , outputs (N, p, q) where $N = pq$, and p and q are n-bit primes except with probability negligible in n. Then consider the following attack game for a given adversary $\mathcal{A}$ and parameter n:

- Run $\texttt{GenModulus}(1^n)$ to obtain (N, p, q).
- $\mathcal{A}$ is given N, and outputs $p', q' > 1$.

Factoring Assumption

Factorization Problem

Let $N = p \cdot q$, where p and q be two n-bit integers. Given N, compute p and q.

Attack Game

Let `GenModulus` be a polynomial-time algorithm that, on input 1^n , outputs (N, p, q) where $N = pq$, and p and q are n-bit primes except with probability negligible in n. Then consider the following attack game for a given adversary $\mathcal{A}$ and parameter n:

- Run $\texttt{GenModulus}(1^n)$ to obtain (N, p, q).
- $\mathcal{A}$ is given N, and outputs $p', q' > 1$.

We define $\mathcal{A}$'s advantage in solving the factoring problem for `GenModulus`, denoted $\mathsf{FACTORadv}[\mathcal{A}, \texttt{GenModulus}]$, as the probability that $p' \cdot q' = N$.

Factoring Assumption

Factorization Problem

Let $N = p \cdot q$, where p and q be two n-bit integers. Given N, compute p and q.

Attack Game

Let `GenModulus` be a polynomial-time algorithm that, on input 1^n , outputs (N, p, q) where $N = pq$, and p and q are n-bit primes except with probability negligible in n. Then consider the following attack game for a given adversary $\mathcal{A}$ and parameter n:

- Run $\texttt{GenModulus}(1^n)$ to obtain (N, p, q).
- $\mathcal{A}$ is given N, and outputs $p', q' > 1$.

We define $\mathcal{A}$'s advantage in solving the factoring problem for `GenModulus`, denoted $\mathsf{FACTORadv}[\mathcal{A}, \texttt{GenModulus}]$, as the probability that $p' \cdot q' = N$.

Factoring Assumption

We say that the Factoring assumption holds for `GenModulus` if for all efficient adversaries $\mathcal{A}$ the quantity $\mathsf{FACTORadv}[\mathcal{A}, \texttt{GenModulus}]$ is negligible.

RSA Assumption

RSA Assumption

Given N, an integer $e > 0$ that is relatively prime to $\phi(N)$, and an element $y \in \mathbb{Z}_N$, compute $y^{1/e} \pmod{N}$;

RSA Assumption

RSA Assumption

Given N, an integer $e > 0$ that is relatively prime to $\phi(N)$, and an element $y \in \mathbb{Z}_N$, compute $y^{1/e} \pmod{N}$; that is, given (N, e, y) find x such $x^e = y \pmod{N}$.

RSA Assumption

RSA Assumption

Given N, an integer $e > 0$ that is relatively prime to $\phi(N)$, and an element $y \in \mathbb{Z}_N$, compute $y^{1/e} \pmod{N}$; that is, given (N, e, y) find x such $x^e = y \pmod{N}$.

GenRSA(1^n)

1. $(N, p, q) \longleftarrow \texttt{GenModulus}(1^n)$.

RSA Assumption

RSA Assumption

Given N, an integer $e > 0$ that is relatively prime to $\phi(N)$, and an element $y \in \mathbb{Z}_N$, compute $y^{1/e} \pmod{N}$; that is, given (N, e, y) find x such $x^e = y \pmod{N}$.

GenRSA(1^n)

1. $(N, p, q) \longleftarrow \texttt{GenModulus}(1^n)$.
2. $\phi(N) = (p-1)(q-1)$.

RSA Assumption

RSA Assumption

Given N, an integer $e > 0$ that is relatively prime to $\phi(N)$, and an element $y \in \mathbb{Z}_N$, compute $y^{1/e} \pmod{N}$; that is, given (N, e, y) find x such $x^e = y \pmod{N}$.

GenRSA(1^n)

1. $(N, p, q) \longleftarrow \texttt{GenModulus}(1^n)$.
2. $\phi(N) = (p-1)(q-1)$.
3. Choose e such that $\gcd(e, \phi(N)) = 1$.

RSA Assumption

RSA Assumption

Given N, an integer $e > 0$ that is relatively prime to $\phi(N)$, and an element $y \in \mathbb{Z}_N$, compute $y^{1/e} \pmod{N}$; that is, given (N, e, y) find x such $x^e = y \pmod{N}$.

GenRSA(1^n)

1. $(N, p, q) \longleftarrow \texttt{GenModulus}(1^n)$.
2. $\phi(N) = (p-1)(q-1)$.
3. Choose e such that $\gcd(e, \phi(N)) = 1$.
4. Compute $d \equiv e^{-1} \pmod{\phi(N)}$.

RSA Assumption

RSA Assumption

Given N, an integer $e > 0$ that is relatively prime to $\phi(N)$, and an element $y \in \mathbb{Z}_N$, compute $y^{1/e} \pmod{N}$; that is, given (N, e, y) find x such $x^e = y \pmod{N}$.

GenRSA(1^n)

1. $(N, p, q) \longleftarrow \texttt{GenModulus}(1^n)$.
2. $\phi(N) = (p-1)(q-1)$.
3. Choose e such that $\gcd(e, \phi(N)) = 1$.
4. Compute $d \equiv e^{-1} \pmod{\phi(N)}$.
5. Return (N, e, d)

RSA Assumption

Attack Game: RSA – inv

For given integers $n > 2$, and a given adversary $\mathcal{A}$, the attack game runs as follows:

- Challenger runs $\mathsf{GenRSA}(1^n)$ to obtain (N, e, d).

RSA Assumption

Attack Game: RSA – inv

For given integers $n > 2$, and a given adversary $\mathcal{A}$, the attack game runs as follows:

- Challenger runs $\mathsf{GenRSA}(1^n)$ to obtain (N, e, d).
- Challenger choose $x \xleftarrow{R} \mathbb{Z}_N$ and computes $y \longleftarrow x^e \pmod N$. Challenger send y to $\mathcal{A}$.

RSA Assumption

Attack Game: RSA – inv

For given integers $n > 2$, and a given adversary $\mathcal{A}$, the attack game runs as follows:

- Challenger runs $\mathsf{GenRSA}(1^n)$ to obtain (N, e, d).
- Challenger choose $x \xleftarrow{R} \mathbb{Z}_N$ and computes $y \longleftarrow x^e \pmod N$. Challenger send y to $\mathcal{A}$.
- The adversary outputs $\hat{x} \in \mathbb{Z}_N$.

RSA Assumption

Attack Game: RSA – inv

For given integers $n > 2$, and a given adversary $\mathcal{A}$, the attack game runs as follows:

- Challenger runs $\mathsf{GenRSA}(1^n)$ to obtain (N, e, d).
- Challenger choose $x \xleftarrow{R} \mathbb{Z}_N$ and computes $y \longleftarrow x^e \pmod N$. Challenger send y to $\mathcal{A}$.
- The adversary outputs $\hat{x} \in \mathbb{Z}_N$.

We define $\mathcal{A}$'s advantage, denoted $\mathsf{RSAadv}[\mathcal{A}, \mathsf{RSA-inv}]$, as the probability that $\hat{x} = x$.

RSA Assumption

Attack Game: RSA – inv

For given integers $n > 2$, and a given adversary $\mathcal{A}$, the attack game runs as follows:

- Challenger runs $\mathsf{GenRSA}(1^n)$ to obtain (N, e, d).
- Challenger choose $x \xleftarrow{R} \mathbb{Z}_N$ and computes $y \longleftarrow x^e \pmod N$. Challenger send y to $\mathcal{A}$.
- The adversary outputs $\hat{x} \in \mathbb{Z}_N$.

We define $\mathcal{A}$'s advantage, denoted $\mathsf{RSAadv}[\mathcal{A}, \mathsf{RSA-inv}]$, as the probability that $\hat{x} = x$.

RSA Assumption

We say that the RSA assumption holds for GenRSA if for all efficient adversaries $\mathcal{A}$ the quantity $\mathsf{RSAadv}[\mathcal{A}, \mathsf{RSA-inv}]$ is negligible.

Text-Book RSA

Insecurities of Text-Book RSA

- Deterministic, thus not secure in Indistinguishabililty game.

Text-Book RSA

Insecurities of Text-Book RSA

- Deterministic, thus not secure in Indistinguishabililty game.
- RSA Key Inversion Problem (RSAKIP): Given $N = pq$, where p and q are two large primes of almost same bit-length, and e with $\gcd(e, \phi(N)) = 1$, find d such that $d \equiv e^{-1} \pmod{\phi(N)}$.
- Claim: Solving RSAKIP is as hard as factoring moduli output by GenRSA.

Text-Book RSA

Insecurities of Text-Book RSA

- Deterministic, thus not secure in Indistinguishabililty game.
- RSA Key Inversion Problem (RSAKIP): Given $N = pq$, where p and q are two large primes of almost same bit-length, and e with $\gcd(e, \phi(N)) = 1$, find d such that $d \equiv e^{-1} \pmod{\phi(N)}$.
- Claim: Solving RSAKIP is as hard as factoring moduli output by GenRSA.
  - Given $N = pq$ and e, d with $ed \equiv 1 \pmod{\phi(N)}$, it is possible to compute the factors of N in polynomial time.

Text-Book RSA

Insecurities of Text-Book RSA

- Deterministic, thus not secure in Indistinguishabililty game.
- RSA Key Inversion Problem (RSAKIP): Given $N = pq$, where p and q are two large primes of almost same bit-length, and e with $\gcd(e, \phi(N)) = 1$, find d such that $d \equiv e^{-1} \pmod{\phi(N)}$.
- Claim: Solving RSAKIP is as hard as factoring moduli output by GenRSA.
  - Given $N = pq$ and e, d with $ed \equiv 1 \pmod{\phi(N)}$, it is possible to compute the factors of N in polynomial time.
- RSA assumption or RSAKIP are polytime reducible to factoring problem.
- But there is no proof of the converse.

Text-Book RSA

Insecurities of Text-Book RSA

- Deterministic, thus not secure in Indistinguishabililty game.
- RSA Key Inversion Problem (RSAKIP): Given $N = pq$, where p and q are two large primes of almost same bit-length, and e with $\gcd(e, \phi(N)) = 1$, find d such that $d \equiv e^{-1} \pmod{\phi(N)}$.
- Claim: Solving RSAKIP is as hard as factoring moduli output by GenRSA.
  - Given $N = pq$ and e, d with $ed \equiv 1 \pmod{\phi(N)}$, it is possible to compute the factors of N in polynomial time.
- RSA assumption or RSAKIP are polytime reducible to factoring problem.
- But there is no proof of the converse.
- Very weak security: If the RSA assumption holds for GenRSA, then no PPT adversary given the public key (N, e) and the resulting ciphertext c can recover the entire message m.

Text-Book RSA

Insecurities of Text-Book RSA

- Deterministic, thus not secure in Indistinguishabililty game.
- RSA Key Inversion Problem (RSAKIP): Given $N = pq$, where p and q are two large primes of almost same bit-length, and e with $\gcd(e, \phi(N)) = 1$, find d such that $d \equiv e^{-1} \pmod{\phi(N)}$.
- Claim: Solving RSAKIP is as hard as factoring moduli output by GenRSA.
  - Given $N = pq$ and e, d with $ed \equiv 1 \pmod{\phi(N)}$, it is possible to compute the factors of N in polynomial time.
- RSA assumption or RSAKIP are polytime reducible to factoring problem.
- But there is no proof of the converse.
- Very weak security: If the RSA assumption holds for GenRSA, then no PPT adversary given the public key (N, e) and the resulting ciphertext c can recover the entire message m.
  - m must be chosen at random.

Text-Book RSA

Decryption Using CRT

- $[c^d \pmod N] \leftrightarrow \Big([c^d \pmod p], [c^d \pmod q]\Big)$.

Text-Book RSA

Decryption Using CRT

- $[c^d \pmod N] \leftrightarrow \Big([c^d \pmod p], [c^d \pmod q]\Big)$.
- Compute $m_p := c^d \pmod p = c^{d \,(\mathrm{mod}\, p-1)} \pmod p$.

Text-Book RSA

Decryption Using CRT

- $[c^d \pmod N] \leftrightarrow \Big([c^d \pmod p], [c^d \pmod q]\Big)$.
- Compute $m_p := c^d \pmod p = c^{d \pmod{p-1}} \pmod p$.
- Compute $m_q := c^d \pmod q = c^{d \pmod{q-1}} \pmod q$.

Text-Book RSA

Decryption Using CRT

- $[c^d \pmod N] \leftrightarrow \Big([c^d \pmod p], [c^d \pmod q]\Big)$.
- Compute $m_p := c^d \pmod p = c^{d \pmod{p-1}} \pmod p$.
- Compute $m_q := c^d \pmod q = c^{d \pmod{q-1}} \pmod q$.
- Compute $m \pmod N$ from equations $m = m_p \pmod p$ and $m = m_q \pmod q$ using CRT.

Attacks on Textbook RSA

Encrypting short messages using small e

- Let $e = 3$ and $m < N^{1/3}$.

Attacks on Textbook RSA

Encrypting short messages using small e

- Let $e = 3$ and $m < N^{1/3}$.
- encryption does not involve modular reduction.

Attacks on Textbook RSA

Encrypting short messages using small e

- Let $e = 3$ and $m < N^{1/3}$.
- encryption does not involve modular reduction.
- m can be retrieved from c as $c := m^{1/3}$ over integers.

Attacks on Textbook RSA

Encrypting short messages using small e

- Let $e = 3$ and $m < N^{1/3}$.
- encryption does not involve modular reduction.
- m can be retrieved from c as $c := m^{1/3}$ over integers.

A general attack when small e is used

- $pk_i = (N_i, 3)$ for $i = 1, 2, 3$ and $\gcd(N_i, N_j) = 1$ where $i \neq j$

Attacks on Textbook RSA

Encrypting short messages using small e

- Let $e = 3$ and $m < N^{1/3}$.
- encryption does not involve modular reduction.
- m can be retrieved from c as $c := m^{1/3}$ over integers.

A general attack when small e is used

- $pk_i = (N_i, 3)$ for $i = 1, 2, 3$ and $\gcd(N_i, N_j) = 1$ where $i \neq j$
- Let the same message m has been encrypted: $c_i := m^3 \pmod{N_i}$ for $i = 1, 2, 3$.

Attacks on Textbook RSA

Encrypting short messages using small e

- Let $e = 3$ and $m < N^{1/3}$.
- encryption does not involve modular reduction.
- m can be retrieved from c as $c := m^{1/3}$ over integers.

A general attack when small e is used

- $pk_i = (N_i, 3)$ for $i = 1, 2, 3$ and $\gcd(N_i, N_j) = 1$ where $i \neq j$
- Let the same message m has been encrypted: $c_i := m^3 \pmod{N_i}$ for $i = 1, 2, 3$.
- By CRT, solve

$$c = c_1 \pmod{N_1}; \quad c = c_2 \pmod{N_2}; \quad c = c_3 \pmod{N_3}.$$

Attacks on Textbook RSA

Encrypting short messages using small e

- Let $e = 3$ and $m < N^{1/3}$.
- encryption does not involve modular reduction.
- m can be retrieved from c as $c := m^{1/3}$ over integers.

A general attack when small e is used

- $pk_i = (N_i, 3)$ for $i = 1, 2, 3$ and $\gcd(N_i, N_j) = 1$ where $i \neq j$
- Let the same message m has been encrypted: $c_i := m^3 \pmod{N_i}$ for $i = 1, 2, 3$.
- By CRT, solve

$$c = c_1 \pmod{N_1}; \quad c = c_2 \pmod{N_2}; \quad c = c_3 \pmod{N_3}.$$

- Now $c \equiv m^3 \pmod{N}$ where $N = N_1 N_2 N_3$, $m \leqslant \min\{N_1, N_2, N_3\}$ and $m < N^{1/3}$.

Attacks on Textbook RSA

Encrypting short messages using small e

- Let $e = 3$ and $m < N^{1/3}$.
- encryption does not involve modular reduction.
- m can be retrieved from c as $c := m^{1/3}$ over integers.

A general attack when small e is used

- $pk_i = (N_i, 3)$ for $i = 1, 2, 3$ and $\gcd(N_i, N_j) = 1$ where $i \neq j$
- Let the same message m has been encrypted: $c_i := m^3 \pmod{N_i}$ for $i = 1, 2, 3$.
- By CRT, solve

$$c = c_1 \pmod{N_1}; \quad c = c_2 \pmod{N_2}; \quad c = c_3 \pmod{N_3}.$$

- Now $c \equiv m^3 \pmod{N}$ where $N = N_1 N_2 N_3$, $m \leqslant \min\{N_1, N_2, N_3\}$ and $m < N^{1/3}$.
- Now apply previous attack.

Attacks on Textbook RSA

Common Modulus N: Case 1

- (sk, pk) for i-th user $((N, d_i), (N, e_i))$.

Attacks on Textbook RSA

Common Modulus N: Case 1

- (sk, pk) for i-th user $((N, d_i), (N, e_i))$.
- r-th user can compute factor of N from (N, e_r, d_r) in polynomial time.

Attacks on Textbook RSA

Common Modulus N: Case 1

- (sk, pk) for i-th user $((N, d_i), (N, e_i))$.
- r-th user can compute factor of N from (N, e_r, d_r) in polynomial time.
- Then r-th user can compute d_i of i-th user from (N, e_i), and decrypt any message intended for i-th user.

Attacks on Textbook RSA

Common Modulus N: Case 1

- (sk, pk) for i-th user $((N, d_i), (N, e_i))$.
- r-th user can compute factor of N from (N, e_r, d_r) in polynomial time.
- Then r-th user can compute d_i of i-th user from (N, e_i), and decrypt any message intended for i-th user.

Common Modulus N: Case 2

- $pk_i = (N, e_i)$ for $i = 1, 2$ and $\gcd(e_1, e_2) = 1$.

Attacks on Textbook RSA

Common Modulus N: Case 1

- (sk, pk) for i-th user $((N, d_i), (N, e_i))$.
- r-th user can compute factor of N from (N, e_r, d_r) in polynomial time.
- Then r-th user can compute d_i of i-th user from (N, e_i), and decrypt any message intended for i-th user.

Common Modulus N: Case 2

- $pk_i = (N, e_i)$ for $i = 1, 2$ and $\gcd(e_1, e_2) = 1$.
- Let the same message m has been encrypted: $c_i := m^{e_i} \pmod{N}$ for $i = 1, 2$.

Attacks on Textbook RSA

Common Modulus N: Case 1

- (sk, pk) for i-th user $((N, d_i), (N, e_i))$.
- r-th user can compute factor of N from (N, e_r, d_r) in polynomial time.
- Then r-th user can compute d_i of i-th user from (N, e_i), and decrypt any message intended for i-th user.

Common Modulus N: Case 2

- $pk_i = (N, e_i)$ for $i = 1, 2$ and $\gcd(e_1, e_2) = 1$.
- Let the same message m has been encrypted: $c_i := m^{e_i} \pmod{N}$ for $i = 1, 2$.
- $\gcd(e_1, e_2) = 1 \Leftrightarrow Xe_1 + Ye_2 = 1$, and computing X and Y are easy.

Attacks on Textbook RSA

Common Modulus N: Case 1

- (sk, pk) for i-th user $((N, d_i), (N, e_i))$.
- r-th user can compute factor of N from (N, e_r, d_r) in polynomial time.
- Then r-th user can compute d_i of i-th user from (N, e_i), and decrypt any message intended for i-th user.

Common Modulus N: Case 2

- $pk_i = (N, e_i)$ for $i = 1, 2$ and $\gcd(e_1, e_2) = 1$.
- Let the same message m has been encrypted: $c_i := m^{e_i} \pmod N$ for $i = 1, 2$.
- $\gcd(e_1, e_2) = 1 \Leftrightarrow Xe_1 + Ye_2 = 1$, and computing X and Y are easy.
- By CRT, solve

$$c_1^X \cdot c_2^Y = m^{Xe_1} \cdot m^{Ye_2} = m^{Xe_1 + Ye_2} = m^1 = m \pmod N.$$

Padded RSA

- Let ℓ be a function s.t. $\ell(n) \leqslant 2n-2, \forall n$.

Key Generation

1. $(N, e, d) \longleftarrow \mathsf{GenRSA}(1^n)$

Padded RSA

- Let ℓ be a function s.t. $\ell(n) \leqslant 2n-2, \forall n$.

Key Generation

1. $(N, e, d) \longleftarrow \mathsf{GenRSA}(1^n)$
2. $sk = (N, d)$ and $pk = (N, e)$.

Padded RSA

- Let ℓ be a function s.t. $\ell(n) \leqslant 2n-2, \forall n$.

Key Generation

1. $(N, e, d) \longleftarrow \mathsf{GenRSA}(1^n)$
2. $sk = (N, d)$ and $pk = (N, e)$.

$\mathcal{E}(pk, m \in \{0,1\}^{\ell(n)})$

1. Choose a random string $r \xleftarrow{R} \{0,1\}^{|N|-\ell(n)-1}$
2. Compute $c \longleftarrow (r \| m)^e \pmod N$.

Padded RSA

- Let ℓ be a function s.t. $\ell(n) \leqslant 2n-2, \forall n$.

Key Generation

1. $(N, e, d) \longleftarrow \mathsf{GenRSA}(1^n)$
2. $sk = (N, d)$ and $pk = (N, e)$.

$\mathcal{E}(pk, m \in \{0,1\}^{\ell(n)})$

1. Choose a random string $r \xleftarrow{R} \{0,1\}^{|N|-\ell(n)-1}$
2. Compute $c \longleftarrow (r \| m)^e \pmod N$.

$\mathcal{D}(sk, c \in \mathbb{Z}_N)$

1. Compute $(r \| m) \longleftarrow c^d \pmod N$.
2. Output least significant $\ell(n)$ bits.

Padded RSA

Theorem

If the RSA problem is hard relative to GenRSA then padded RSA with $\ell(n) = O(\log n)$ has indistinguishable encryptions under a chosen-plaintext attack.

ElGamal Public-Key Encryption Scheme

Domain parameter

- Let G be a cyclic multiplicative group.

ElGamal Public-Key Encryption Scheme

Domain parameter

- Let G be a cyclic multiplicative group.
- $O(G) = n$.

ElGamal Public-Key Encryption Scheme

Domain parameter

- Let G be a cyclic multiplicative group.
- $O(G) = n$.
- $\exists g \in G$ such that $O(g) = n$.

ElGamal Public-Key Encryption Scheme

Domain parameter

- Let G be a cyclic multiplicative group.
- $O(G) = n$.
- $\exists g \in G$ such that $O(g) = n$.

ElGamal Public-Key Encryption Scheme

Domain parameter

- Let G be a cyclic multiplicative group.
- $O(G) = n$.
- $\exists g \in G$ such that $O(g) = n$.

Key Generation

- Choose $d \xleftarrow{R} \mathbb{Z}_n \setminus \{0, 1\}$.

ElGamal Public-Key Encryption Scheme

Domain parameter

- Let G be a cyclic multiplicative group.
- $O(G) = n$.
- $\exists g \in G$ such that $O(g) = n$.

Key Generation

- Choose $d \xleftarrow{R} \mathbb{Z}_n \setminus \{0, 1\}$.
- Compute $Q \equiv g^d$.

ElGamal Public-Key Encryption Scheme

Domain parameter

- Let G be a cyclic multiplicative group.
- $O(G) = n$.
- $\exists g \in G$ such that $O(g) = n$.

Key Generation

- Choose $d \xleftarrow{R} \mathbb{Z}_n \setminus \{0, 1\}$.
- Compute $Q \equiv g^d$.
- $sk = (G, g, n, d)$ and $pk = (G, g, n, Q)$.

ElGamal Public-Key Encryption Scheme

$\mathcal{E}(pk, m)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_n \setminus \{0, 1\}$.
2. Compute $r \longleftarrow g^k$.
3. Compute $s \longleftarrow mQ^k$.
4. $c = (r, s)$.

ElGamal Public-Key Encryption Scheme

$\mathcal{E}(pk, m)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_n \setminus \{0, 1\}$.
2. Compute $r \longleftarrow g^k$.
3. Compute $s \longleftarrow mQ^k$.
4. $c = (r, s)$.

$\mathcal{D}(sk, c))$

1. Compute $m = sr^{-d}$.

ElGamal Public-Key Encryption Scheme

$\mathcal{E}(pk, m)$

1. Choose $k \xleftarrow{R} \mathbb{Z}_n \setminus \{0, 1\}$.
2. Compute $r \longleftarrow g^k$.
3. Compute $s \longleftarrow mQ^k$.
4. $c = (r, s)$.

$\mathcal{D}(sk, c))$

1. Compute $m = sr^{-d}$.

Correctness

$$m' = sr^{-d} = mQ^k(g^k)^{-d} = m(g^d)^k(g^k)^{-d} = mg^{kd-kd} = m.$$

ElGamal Public-Key Encryption Scheme

Theorem

If the DDH problem is hard relative to G, then the ElGamal encryption scheme has indistinguishable encryptions under a chosen-plaintext attack.

End