# Public key cryptography - basic concepts. Encryption and key transport

Applied Cryptography – Spring 2024

Simona Samardjiska

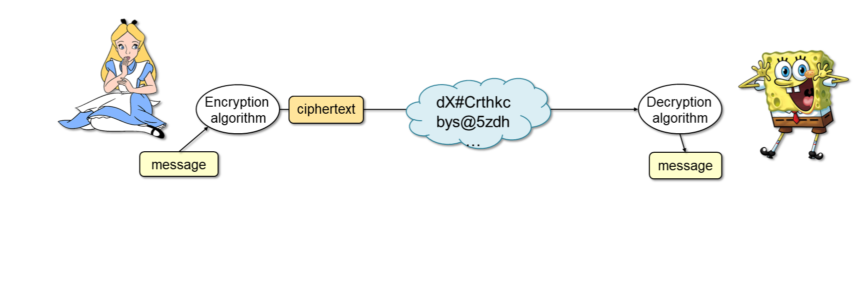
March 11, 2024

Institute for Computing and Information Sciences
Radboud University

# Outline

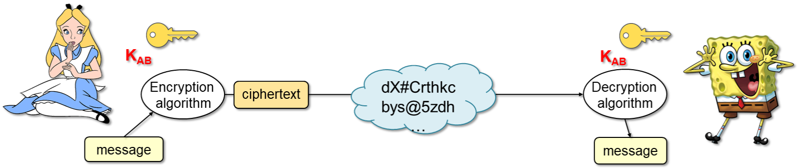
# Public Key Cryptography

# Recall our everyday scenario



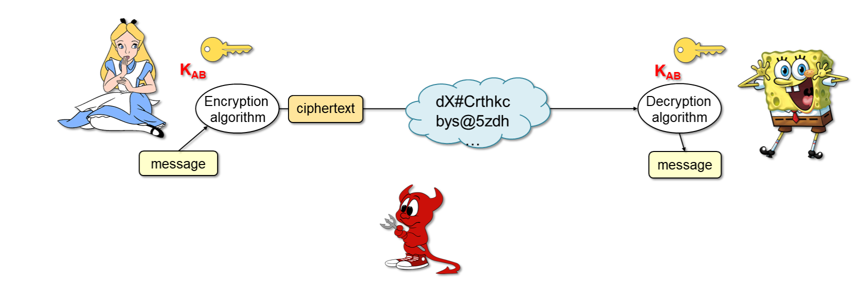

- Recall our favorite characters, Alice and Bob

# Recall our everyday scenario



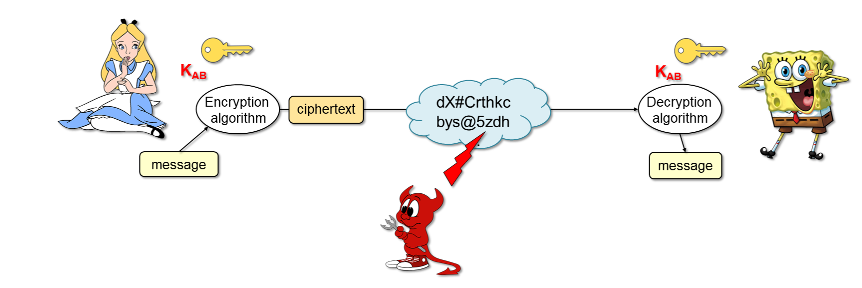

- Recall our favorite characters, Alice and Bob
  - They communicate over a public channel using symmetric cryptography

# Recall our everyday scenario



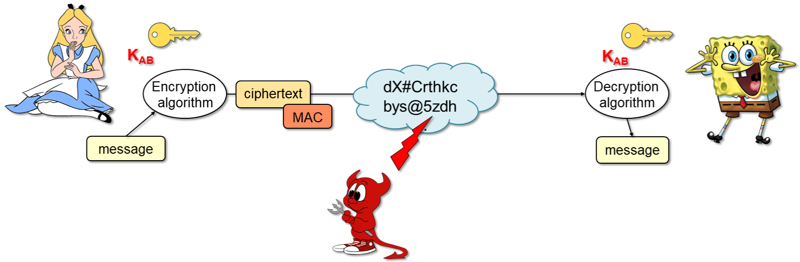

- Recall our favorite characters, Alice and Bob
  - They communicate over a public channel using symmetric cryptography
- while our favorite malicious character, Eve
  - Can listen to the traffic (passive)

# Recall our everyday scenario



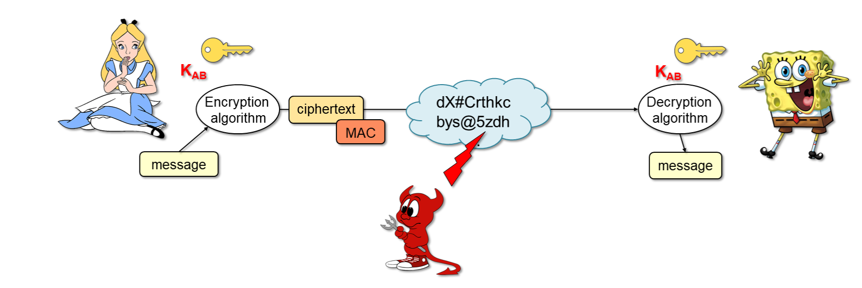

- Recall our favorite characters, Alice and Bob
  - They communicate over a public channel using symmetric cryptography
- while our favorite malicious character, Eve
  - Can listen to the traffic (passive)
  - Can modify the traffic (active)

# Recall our everyday scenario



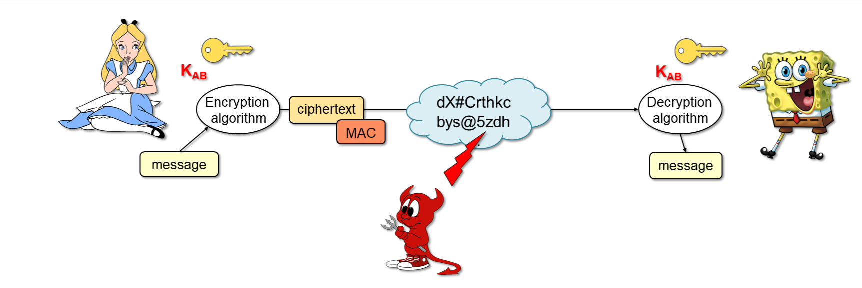

- Recall our favorite characters, Alice and Bob
  - They communicate over a public channel using symmetric cryptography
- while our favorite malicious character, Eve
  - Can listen to the traffic (passive)
  - Can modify the traffic (active)
- Symmetric cryptography provides
  - **Confidentiality:** Eve cannot learn anything about data
  - **Message Authenticity:** Eve cannot manipulate the data

# Recall our everyday scenario



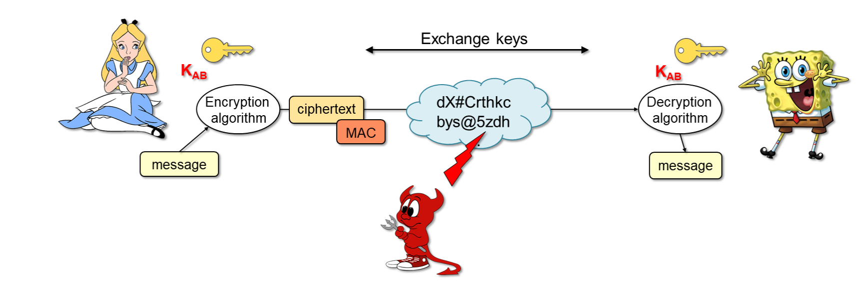

- Recall our favorite characters, Alice and Bob
  - They communicate over a public channel using symmetric cryptography
- while our favorite malicious character, Eve
  - Can listen to the traffic (passive)
  - Can modify the traffic (active)
- Symmetric cryptography provides
  - **Confidentiality:** Eve cannot learn anything about data
  - **Message Authenticity:** Eve cannot manipulate the data
- **What can be a problem in this scenario?**

# An update in our everyday scenario



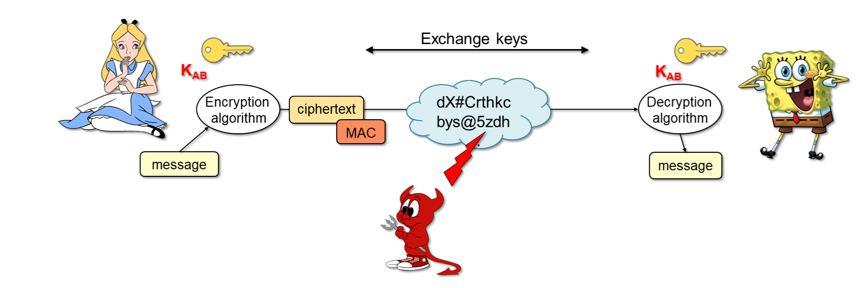

- Alice and Bob have **not** agreed on a joint key yet, but they want to communicate securely

# An update in our everyday scenario



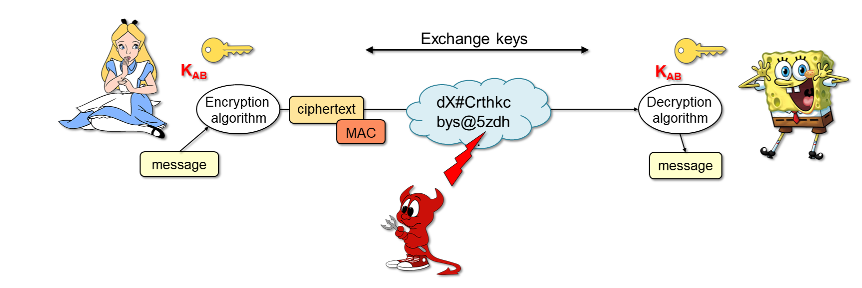

- Alice and Bob have **not** agreed on a joint key yet, but they want to communicate securely
  - They want to exchange symmetric keys over the public channel, first

# An update in our everyday scenario



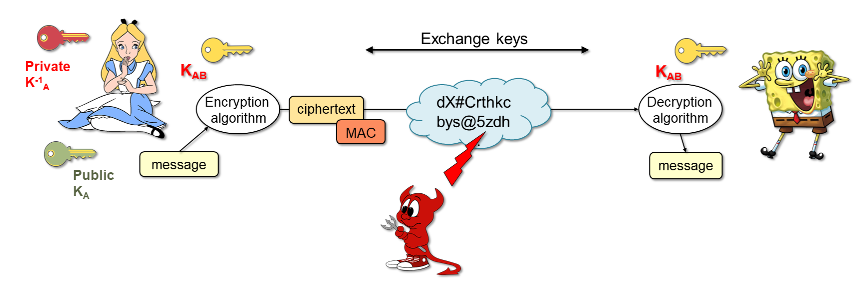

- Alice and Bob have **not** agreed on a joint key yet, but they want to communicate securely
  - They want to exchange symmetric keys over the public channel, first
- Eve can impersonate Alice to Bob or/and Bob to Alice

# An update in our everyday scenario



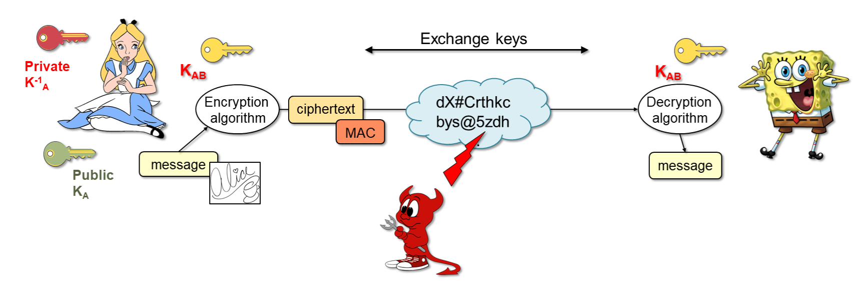

- Alice and Bob have **not** agreed on a joint key yet, but they want to communicate securely
  - They want to exchange symmetric keys over the public channel, first
- Eve can impersonate Alice to Bob or/and Bob to Alice
  - so Alice will never admit she send that angry message to Bob

# An update in our everyday scenario

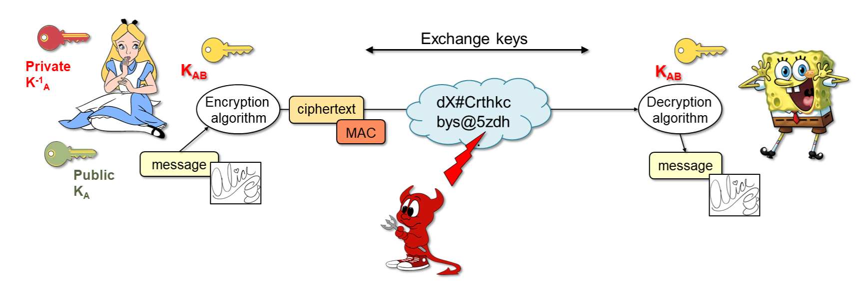



- Alice and Bob have **not** agreed on a joint key yet, but they want to communicate securely
  - They want to exchange symmetric keys over the public channel, first
- Eve can impersonate Alice to Bob or/and Bob to Alice
  - so Alice will never admit she send that angry message to Bob
- Public key cryptography provides
  - **Key Exchange:** Eve can not learn the key

# An update in our everyday scenario




- Alice and Bob have **not** agreed on a joint key yet, but they want to communicate securely
  - They want to exchange symmetric keys over the public channel, first
- Eve can impersonate Alice to Bob or/and Bob to Alice
  - so Alice will never admit she send that angry message to Bob
- Public key cryptography provides
  - **Key Exchange:** Eve can not learn the key




- Alice and Bob have **not** agreed on a joint key yet, but they want to communicate securely
  - They want to exchange symmetric keys over the public channel, first
- Eve can impersonate Alice to Bob or/and Bob to Alice
  - so Alice will never admit she send that angry message to Bob

- Public key cryptography provides
  - **Key Exchange:** Eve can not learn the key
  - **Entity Authentication:** Eve cannot impersonate the parties
  - **Non-repudiation:** The parties can not repudiate the messages

# Core Functionalities

## Public Key Encryption (PKE)

- Uses public key to transform data into ciphertext
- Only with the knowledge of the private key, one can retrieve data back

# Core Functionalities

## Public Key Encryption (PKE)

- Uses public key to transform data into ciphertext
- Only with the knowledge of the private key, one can retrieve data back

## Digital Signatures

- Uses private key of signer to sign the message
- Anyone can verify the signature using the public key of the signer and the message

# Core Functionalities

## Public Key Encryption (PKE)

- Uses public key to transform data into ciphertext
- Only with the knowledge of the private key, one can retrieve data back

## Digital Signatures

- Uses private key of signer to sign the message
- Anyone can verify the signature using the public key of the signer and the message

## Key Encapsulation Mechanism (KEM) and Key Exchange (KEX)

- Goal is to obtain a shared symmetric key
- KEM (simplified)
  - encrypt symmetric key with public key of receiver
  - receiver decrypts symmetric key with his private key
- KEX - a protocol to agree on a shared symmetric key
  - comes in different flavors and constructions (Diffie-Hellman-style, from KEMs, etc.)

# Versatility of Public Key Cryptography

## Examples of other, more subtle flavors of Public Key Cryptography

- Group/ring, blind signatures
- Commitments
- Identification schemes
- Secret Sharing schemes
- Threshold encryption
- Homomorphic Encryption
- Identity-based cryptography
- Attribute-based cryptography
- Credential schemes
- Functional Encryption
- Multiparty computation
- Digital cash/cryptocurrency

# Versatility of Public Key Cryptography

## Examples of other, more subtle flavors of Public Key Cryptography

- Group/ring, blind signatures
- Commitments
- Identification schemes
- Secret Sharing schemes
- Threshold encryption
- Homomorphic Encryption
- Identity-based cryptography
- Attribute-based cryptography
- Credential schemes
- Functional Encryption
- Multiparty computation
- Digital cash/cryptocurrency

## Examples of real-world protocols employing Public Key Cryptography

- Secure messaging protocols
- SSL/TLS (https, ftps)
- SSH (sftp, scp)
- IPsec (IKE)
- OpenVPN, Wireguard
- IEEE 802.11
- DNSSEC
- EMV
- Electronic voting
- ...

# What will we learn about Public Key Cryptography in this course?

# What will we learn about Public Key Cryptography in this course?

## Which building blocks are used in pratice?

- PKEs, KEMs, Digital Signatures
- Commitments, Identification schemes
- Protocols for authentication and key-exchange

# What will we learn about Public Key Cryptography in this course?

## Which building blocks are used in pratice?

- PKEs, KEMs, Digital Signatures
- Commitments, Identification schemes
- Protocols for authentication and key-exchange

## How do we formalize their security?

- security models
- security games and reductions

# What will we learn about Public Key Cryptography in this course?

## Which building blocks are used in pratice?

- PKEs, KEMs, Digital Signatures
- Commitments, Identification schemes
- Protocols for authentication and key-exchange

## How do we formalize their security?

- security models
- security games and reductions

## Which instantiations are standardized by standardization bodies?

- are they provably secure or ad-hoc

# What will we learn about Public Key Cryptography in this course?

## Which building blocks are used in pratice?

- PKEs, KEMs, Digital Signatures
- Commitments, Identification schemes
- Protocols for authentication and key-exchange

## How do we formalize their security?

- security models
- security games and reductions

## Which instantiations are standardized by standardization bodies?

- are they provably secure or ad-hoc

## In which real-world protools and products are they used, and how?

- TLS, IPsec, DNSSEC, ...

# What will we learn about Public Key Cryptography in this course?

### Which building blocks are used in pratice?

- PKEs, KEMs, Digital Signatures
- Commitments, Identification schemes
- Protocols for authentication and key-exchange

### How do we formalize their security?

- security models
- security games and reductions

### Which instantiations are standardized by standardization bodies?

- are they provably secure or ad-hoc

### In which real-world protools and products are they used, and how?

- TLS, IPsec, DNSSEC, . . .

### Which practical problems arise in practice?

- complexity of availability, versioning, updates, following standards etc.
- policies, management, distribution of public/private keys, etc.

# What will we learn about Public Key Cryptography in this course?

### Which building blocks are used in pratice?

- PKEs, KEMs, Digital Signatures
- Commitments, Identification schemes
- Protocols for authentication and key-exchange

### How do we formalize their security?

- security models
- security games and reductions

### Which instantiations are standardized by standardization bodies?

- are they provably secure or ad-hoc

### In which real-world protools and products are they used, and how?

- TLS, IPsec, DNSSEC, ...

### Which practical problems arise in practice?

- complexity of availability, versioning, updates, following standards etc.
- policies, management, distribution of public/private keys, etc.

### Prudent practices for future deployment?

- reflections on mistakes made
- how not to repeat them in the future

# Security of Pubic Key Cryptographic Schemes

# Security of Pubic Key Cryptographic Schemes

$\boxed{\text{Provable security}} \quad + \quad \boxed{\text{Cryptanalysis}}$

# Security of Pubic Key Cryptographic Schemes

$\boxed{\text{Provable security}}$ $+$ $\boxed{\text{Cryptanalysis}}$

**Reductionist proof**
from a hard problem

# Security of Pubic Key Cryptographic Schemes

$\boxed{\text{Provable security}}$ + $\boxed{\text{Cryptanalysis}}$

**Reductionist proof** from a hard problem

**Best algorithms** for solving the hard problem

# Security of Pubic Key Cryptographic Schemes

$\boxed{\text{Provable security}} \quad + \quad \boxed{\text{Cryptanalysis}}$

**Reductionist proof**
from a hard problem

**Best algorithms** for solving
the hard problem
(Treated in Cryptology)

# Security of Pubic Key Cryptographic Schemes

$\boxed{\text{Provable security}}$ + $\boxed{\text{Cryptanalysis}}$

**Reductionist proof**
from a hard problem

**Best algorithms** for solving
the hard problem
(Treated in Cryptology)

+ $\boxed{\text{Secure implementation}}$

(Treated in Cryptographic engineering)

# Hardness assumptions

## Easy problems

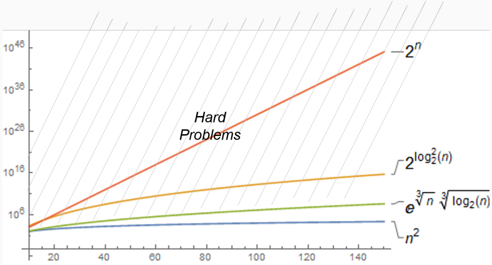
vs

## Hard problems

$\mathcal{O}(n)$

$$\underbrace{\begin{bmatrix}1 & 2 & 3\end{bmatrix}}_{1\times 3} \cdot \underbrace{\begin{bmatrix}2 & 1 & 3\\ 3 & 3 & 2\\ 4 & 1 & 2\end{bmatrix}}_{3\times 3} = \begin{bmatrix}1\cdot 2+2\cdot 3+3\cdot 4\\ 1\cdot 1+2\cdot 3+3\cdot 1\\ 1\cdot 3+2\cdot 2+3\cdot 2\end{bmatrix} = \underbrace{\begin{bmatrix}20\\ 10\\ 13\end{bmatrix}}_{1\times 3}$$

$\mathcal{O}(n^2)$

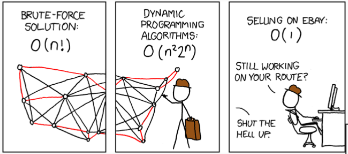



(x OR y OR z) AND (x OR $\bar{y}$ OR z) AND

(x OR y OR $\bar{z}$) AND (x OR $\bar{y}$ OR $\bar{z}$) AND

($\bar{x}$ OR y OR z) AND ($\bar{x}$ OR $\bar{y}$ OR $\bar{z}$)

$\mathcal{O}(2^n)$




$\mathcal{O}(poly(n)2^n)$

Hard problems:
No efficient (polynomial time) algorithm exists

# Hardness assumptions - different flavors

**A computational problem** generated with security parameter $\lambda$ is **hard** if, given as input a problem instance, the probability of finding a correct solution in polynomial time is negligible in $\lambda$ ($negl(\lambda)$).

# Hardness assumptions - different flavors

**A computational problem** generated with security parameter $\lambda$ is **hard** if, given as input a problem instance, the probability of finding a correct solution in polynomial time is negligible in $\lambda$ ($negl(\lambda)$).

**Example**: Computational Diffie-Hellman Problem (CDH)

**Given**: $g, g^a, g^b \in \mathbb{G}$, where $\mathbb{G}$ – general cyclic group

**Find**: $g^{ab}$

# Hardness assumptions - different flavors

**A computational problem** generated with security parameter $\lambda$ is **hard** if, given as input a problem instance, the probability of finding a correct solution in polynomial time is negligible in $\lambda$ ($negl(\lambda)$).

**Example**: Computational Diffie-Hellman Problem (CDH)

**Given**: $g, g^a, g^b \in \mathbb{G}$, where $\mathbb{G}$ – general cyclic group

**Find**: $g^{ab}$

**A decisional problem** generated with s. p. $\lambda$ is **hard** if, given as input a problem instance with a target $Z$, the advantage of correctly guessing in polynomial time whether it is a positive instance is $negl(\lambda)$.

# Hardness assumptions - different flavors

**A computational problem** generated with security parameter $\lambda$ is **hard** if, given as input a problem instance, the probability of finding a correct solution in polynomial time is negligible in $\lambda$ ($negl(\lambda)$).

**Example**: Computational Diffie-Hellman Problem (CDH)

**Given**: $g, g^a, g^b \in \mathbb{G}$, where $\mathbb{G}$ – general cyclic group

**Find**: $g^{ab}$

**A decisional problem** generated with s. p. $\lambda$ is **hard** if, given as input a problem instance with a target $Z$, the advantage of correctly guessing in polynomial time whether it is a positive instance is $negl(\lambda)$.

**Example**: Decisional Diffie-Hellman Problem (DDH)

**Given**: $g, g^a, g^b, Z \in \mathbb{G}$, where $\mathbb{G}$ – general cyclic group

**Decide**: $Z \stackrel{?}{=} g^{ab}$

# Hardness assumptions - different flavors

**A computational problem** generated with security parameter $\lambda$ is **hard** if, given as input a problem instance, the probability of finding a correct solution in polynomial time is negligible in $\lambda$ ($negl(\lambda)$).

**Example**: Computational Diffie-Hellman Problem (CDH)

**Given**: $g, g^a, g^b \in \mathbb{G}$, where $\mathbb{G}$ – general cyclic group

**Find**: $g^{ab}$

**A decisional problem** generated with s. p. $\lambda$ is **hard** if, given as input a problem instance with a target $Z$, the advantage of correctly guessing in polynomial time whether it is a positive instance is $negl(\lambda)$.

**Example**: Decisional Diffie-Hellman Problem (DDH)

**Given**: $g, g^a, g^b, Z \in \mathbb{G}$, where $\mathbb{G}$ – general cyclic group

**Decide**: $Z \stackrel{?}{=} g^{ab}$

**Remark**: For a problem with computational and decisional version, if one can solve the computational version, then they can solve the decisional version as well.(The decisional version is "easier".)
$\Rightarrow$ Assuming the decisional version to be hard is a **stronger assumption**.

# Hardness assumptions - different flavors

**A computational problem** generated with security parameter $\lambda$ is **hard** if, given as input a problem instance, the probability of finding a correct solution in polynomial time is negligible in $\lambda$ ($negl(\lambda)$).

**Example**: Computational Diffie-Hellman Problem (CDH)

**Given**: $g, g^a, g^b \in \mathbb{G}$, where $\mathbb{G}$ – general cyclic group

**Find**: $g^{ab}$

**A decisional problem** generated with s. p. $\lambda$ is **hard** if, given as input a problem instance with a target $Z$, the advantage of correctly guessing in polynomial time whether it is a positive instance is $negl(\lambda)$.

**Example**: Decisional Diffie-Hellman Problem (DDH)

**Given**: $g, g^a, g^b, Z \in \mathbb{G}$, where $\mathbb{G}$ – general cyclic group

**Decide**: $Z \stackrel{?}{=} g^{ab}$

**Remark**: For a problem with computational and decisional version, if one can solve the computational version, then they can solve the decisional version as well.(The decisional version is "easier".)
$\Rightarrow$ Assuming the decisional version to be hard is a **stronger assumption**.

# Security reduction

- **Security reduction** - proof that breaking a scheme implies breaking the hardness assumption (solving a hard mathematical problem)

## Security reduction

- **Security reduction** - proof that breaking a scheme implies breaking the hardness assumption (solving a hard mathematical problem)
- **Security model** - an abstraction that captures a multiple of different real-world attacks, in a form of an interactive game between **adversary** (probabilistic polynomial time) and **challenger**

# Security reduction

- **Security reduction** - proof that breaking a scheme implies breaking the hardness assumption (solving a hard mathematical problem)
- **Security model** - an abstraction that captures a multiple of different real-world attacks, in a form of an interactive game between **adversary** (probabilistic polynomial time) and **challenger**
  - **what** information the adversary can query
  - **when** can that information be queried
  - **how** does the adversary win

# Security reduction

- **Security reduction** - proof that breaking a scheme implies breaking the hardness assumption (solving a hard mathematical problem)
- **Security model** - an abstraction that captures a multiple of different real-world attacks, in a form of an interactive game between **adversary** (probabilistic polynomial time) and **challenger**
  - **what** information the adversary can query
  - **when** can that information be queried
  - **how** does the adversary win
- **Proof by contradiction**

# Security reduction

- **Security reduction** - proof that breaking a scheme implies breaking the hardness assumption (solving a hard mathematical problem)
- **Security model** - an abstraction that captures a multiple of different real-world attacks, in a form of an interactive game between **adversary** (probabilistic polynomial time) and **challenger**
  - **what** information the adversary can query
  - **when** can that information be queried
  - **how** does the adversary win
- **Proof by contradiction**
  - We know (believe) that a mathematical problem is **hard** (hardness assumption)

# Security reduction

- **Security reduction** - proof that breaking a scheme implies breaking the hardness assumption (solving a hard mathematical problem)
- **Security model** - an abstraction that captures a multiple of different real-world attacks, in a form of an interactive game between **adversary** (probabilistic polynomial time) and **challenger**
  - **what** information the adversary can query
  - **when** can that information be queried
  - **how** does the adversary win
- **Proof by contradiction**
  - We know (believe) that a mathematical problem is **hard** (hardness assumption)
  - **Assume** there is an adversary that **breaks the scheme** and show that using this adversary, we can solve the mathematical problem (**break the assumption**)

# Security reduction

- **Security reduction** - proof that breaking a scheme implies breaking the hardness assumption (solving a hard mathematical problem)
- **Security model** - an abstraction that captures a multiple of different real-world attacks, in a form of an interactive game between **adversary** (probabilistic polynomial time) and **challenger**
    - **what** information the adversary can query
    - **when** can that information be queried
    - **how** does the adversary win
- **Proof by contradiction**
    - We know (believe) that a mathematical problem is **hard** (hardness assumption)
    - **Assume** there is an adversary that **breaks the scheme** and show that using this adversary, we can solve the mathematical problem (**break the assumption**)
    - Conclude that our assumption must be **wrong** $\Rightarrow$ There is no such adversary!

# Security reduction

- **Security reduction** - proof that breaking a scheme implies breaking the hardness assumption (solving a hard mathematical problem)
- **Security model** - an abstraction that captures a multiple of different real-world attacks, in a form of an interactive game between **adversary** (probabilistic polynomial time) and **challenger**
  - **what** information the adversary can query
  - **when** can that information be queried
  - **how** does the adversary win
- **Proof by contradiction**
  - We know (believe) that a mathematical problem is **hard** (hardness assumption)
  - **Assume** there is an adversary that **breaks the scheme** and show that using this adversary, we can solve the mathematical problem (**break the assumption**)
  - Conclude that our assumption must be **wrong** $\Rightarrow$ There is no such adversary!

$$\frac{p \Rightarrow q, \quad \neg q}{\neg p}$$

# Security reduction

- **Security reduction** - proof that breaking a scheme implies breaking the hardness assumption (solving a hard mathematical problem)
- **Security model** - an abstraction that captures a multiple of different real-world attacks, in a form of an interactive game between **adversary** (probabilistic polynomial time) and **challenger**
  - **what** information the adversary can query
  - **when** can that information be queried
  - **how** does the adversary win
- **Proof by contradiction**
  - We know (believe) that a mathematical problem is **hard** (hardness assumption)
  - **Assume** there is an adversary that **breaks the scheme** and show that using this adversary, we can solve the mathematical problem (**break the assumption**)
  - Conclude that our assumption must be **wrong** $\Rightarrow$ There is no such adversary!

$$\frac{p \Rightarrow q, \quad \neg q}{\neg p}$$

**Remark**: A security reduction does not show that a scheme is secure, but only **as secure as** the hardness assumption!

# Security reduction - the setup

- **The virtual "players"** that interact with the adversary
  - **Challenger** - creates an instance of the **real** cryptographic scheme, following its algorithms, and interacts with the adversary by answering queries about the scheme
  - **Simulator** - creates an instance of a **simulated** scheme, produced from the hard problem. The Simulator wants the adversary to break this scheme with the same advantage as the real scheme

# Security reduction - the setup

- **The virtual "players"** that interact with the adversary
  - **Challenger** - creates an instance of the **real** cryptographic scheme, following its algorithms, and interacts with the adversary by answering queries about the scheme
  - **Simulator** - creates an instance of a **simulated** scheme, produced from the hard problem. The Simulator wants the adversary to break this scheme with the same advantage as the real scheme
- **Can the adversary figure out it is a simulation?**
  - For the adversary, the simulated scheme should be **indistinguishable** from the real one
  - $\Rightarrow$ the **attack** on the simulated scheme should be indistinguishable from the real one

## Security reduction - the setup

- **The virtual "players"** that interact with the adversary
  - **Challenger** - creates an instance of the **real** cryptographic scheme, following its algorithms, and interacts with the adversary by answering queries about the scheme
  - **Simulator** - creates an instance of a **simulated** scheme, produced from the hard problem. The Simulator wants the adversary to break this scheme with the same advantage as the real scheme
- **Can the adversary figure out it is a simulation?**
  - For the adversary, the simulated scheme should be **indistinguishable** from the real one
  - $\Rightarrow$ the **attack** on the simulated scheme should be indistinguishable from the real one
- **The attack**

# Security reduction - the setup

- **The virtual "players"** that interact with the adversary
  - **Challenger** - creates an instance of the **real** cryptographic scheme, following its algorithms, and interacts with the adversary by answering queries about the scheme
  - **Simulator** - creates an instance of a **simulated** scheme, produced from the hard problem. The Simulator wants the adversary to break this scheme with the same advantage as the real scheme
- **Can the adversary figure out it is a simulation?**
  - For the adversary, the simulated scheme should be **indistinguishable** from the real one
  - $\Rightarrow$ the **attack** on the simulated scheme should be indistinguishable from the real one
- **The attack**
  - **Computational attack** - the adversary will spit out an answer that in a reduction can be used to solve a hard computational problem

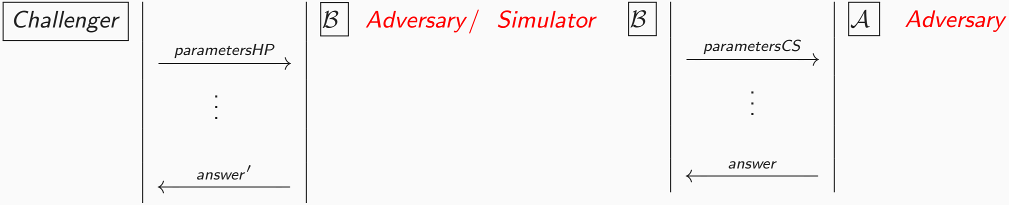
# Security reduction - the setup

- **The virtual "players"** that interact with the adversary
  - **Challenger** - creates an instance of the **real** cryptographic scheme, following its algorithms, and interacts with the adversary by answering queries about the scheme
  - **Simulator** - creates an instance of a **simulated** scheme, produced from the hard problem. The Simulator wants the adversary to break this scheme with the same advantage as the real scheme
- **Can the adversary figure out it is a simulation?**
  - For the adversary, the simulated scheme should be **indistinguishable** from the real one
  - $\Rightarrow$ the **attack** on the simulated scheme should be indistinguishable from the real one
- **The attack**
  - **Computational attack** - the adversary will spit out an answer that in a reduction can be used to solve a hard computational problem
  - **Decisional attack** - the adversary will spit out a decision that in a reduction can be used to solve a hard decisional problem

# A high level view of security reduction

Hard-problem (HP) security game

Cryptographic scheme (CS) security game




real game

simulated game

# Security reduction - reduction cost and reduction lost

- **Adversary** breaks scheme in $(t, \epsilon)$ (read: "in time $t$ and non-negligible advantage $\epsilon$")
- $\Rightarrow$ **Simulator** needs $(t', \epsilon')$ to solve the hard problem

$$t' = t + T, \qquad \epsilon' = \frac{\epsilon}{L}$$

$T$ - reduction **cost** (time cost), $L$- reduction (security) **loss**

# Security reduction - reduction cost and reduction lost

- **Adversary** breaks scheme in $(t, \epsilon)$ (read: "in time $t$ and non-negligible advantage $\epsilon$")
- $\Rightarrow$ **Simulator** needs $(t', \epsilon')$ to solve the hard problem

$$t' = t + T, \qquad \epsilon' = \frac{\epsilon}{L}$$

  $T$ - reduction **cost** (time cost), $L$- reduction (security) **loss**

- **Tight security reduction** - $L$ constant (or sub-linear) in the number of queries
- **Loose security reduction** - $L$ at least linear in the number of queries
    - ***k*-bit security loss** - if $L = 2^k$

# Security reduction - reduction cost and reduction lost

- **Adversary** breaks scheme in $(t, \epsilon)$ (read: "in time $t$ and non-negligible advantage $\epsilon$")
- $\Rightarrow$ **Simulator** needs $(t', \epsilon')$ to solve the hard problem

$$t' = t + T, \qquad \epsilon' = \frac{\epsilon}{L}$$

  $T$ - reduction **cost** (time cost), $L$- reduction (security) **loss**

- **Tight security reduction** - $L$ constant (or sub-linear) in the number of queries
- **Loose security reduction** - $L$ at least linear in the number of queries
  - ***k*-bit security loss** - if $L = 2^k$
  - the parameters need to be increased to add additional $k$ bits of security
  - Example:
    - The underlying problem has 128 bits of security, the reduction has loss of 12 bits,
    - $\Rightarrow$ the scheme can be claimed to have only 116 bits of security
  - A vastly overlooked/ignored issue in public-key cryptography

# Public Key Encryption (PKE)

# Public Key Encryption (PKE) – definition

Given security parameter $\lambda \in \mathbb{N}$ and two finite sets $\mathcal{M}, \mathcal{R} \subseteq \{0,1\}^*$, a Public Key Encryption $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three algorithms:

- **Key-generation algorithm** (probabilistic): $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
- **Encryption algorithm** (probabilistic): Takes message $M \in \mathcal{M}$ and random $r \in \mathcal{R}$ and outputs $C \leftarrow \mathsf{Enc}(\mathsf{pk}, M, r)$
- **Decryption algorithm** (deterministic): Takes as input a secret key sk and ciphertext $C$, and outputs either a message $M' = \mathsf{Dec}(\mathsf{sk}, C) \in \mathcal{M}$ or $\perp \notin \mathcal{M}$ to indicate an invalid ciphertext.

# Public Key Encryption (PKE) – definition

Given security parameter $\lambda \in \mathbb{N}$ and two finite sets $\mathcal{M}, \mathcal{R} \subseteq \{0,1\}^*$, a Public Key Encryption $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three algorithms:

- **Key-generation algorithm** (probabilistic): $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
- **Encryption algorithm** (probabilistic): Takes message $M \in \mathcal{M}$ and random $r \in \mathcal{R}$ and outputs $C \leftarrow \mathsf{Enc}(\mathsf{pk}, M, r)$
- **Decryption algorithm** (deterministic): Takes as input a secret key sk and ciphertext $C$, and outputs either a message $M' = \mathsf{Dec}(\mathsf{sk}, C) \in \mathcal{M}$ or $\perp \notin \mathcal{M}$ to indicate an invalid ciphertext.

**Correctness:** For all $M \in \mathcal{M}$, $Pr[\mathsf{Dec}(\mathsf{sk}, C) = M : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda), C \leftarrow \mathsf{Enc}(\mathsf{pk}, M, r)] \geqslant 1 - \delta$

# Public Key Encryption (PKE) – definition

Given security parameter $\lambda \in \mathbb{N}$ and two finite sets $\mathcal{M}, \mathcal{R} \subseteq \{0,1\}^*$, a Public Key Encryption $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three algorithms:

- **Key-generation algorithm** (probabilistic): $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
- **Encryption algorithm** (probabilistic): Takes message $M \in \mathcal{M}$ and random $r \in \mathcal{R}$ and outputs $C \leftarrow \mathsf{Enc}(\mathsf{pk}, M, r)$
- **Decryption algorithm** (deterministic): Takes as input a secret key sk and ciphertext $C$, and outputs either a message $M' = \mathsf{Dec}(\mathsf{sk}, C) \in \mathcal{M}$ or $\perp \notin \mathcal{M}$ to indicate an invalid ciphertext.

**Correctness:** For all $M \in \mathcal{M}$, $Pr[\mathsf{Dec}(\mathsf{sk}, C) = M : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda), C \leftarrow \mathsf{Enc}(\mathsf{pk}, M, r)] \geqslant 1 - \delta$

- (Negligible) **Decryption error** ($\delta$) is also allowed (not all schemes have it)

# Public Key Encryption (PKE) – definition

Given security parameter $\lambda \in \mathbb{N}$ and two finite sets $\mathcal{M}, \mathcal{R} \subseteq \{0,1\}^*$, a Public Key Encryption $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three algorithms:

- **Key-generation algorithm** (probabilistic): $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
- **Encryption algorithm** (probabilistic): Takes message $M \in \mathcal{M}$ and random $r \in \mathcal{R}$ and outputs $C \leftarrow \mathsf{Enc}(\mathsf{pk}, M, r)$
- **Decryption algorithm** (deterministic): Takes as input a secret key sk and ciphertext $C$, and outputs either a message $M' = \mathsf{Dec}(\mathsf{sk}, C) \in \mathcal{M}$ or $\perp \notin \mathcal{M}$ to indicate an invalid ciphertext.

**Correctness:** For all $M \in \mathcal{M}$, $Pr[\mathsf{Dec}(\mathsf{sk}, C) = M : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda), C \leftarrow \mathsf{Enc}(\mathsf{pk}, M, r)] \geqslant 1 - \delta$

- (Negligible) **Decryption error** ($\delta$) is also allowed (not all schemes have it)
- **Passive attacker** (eavesdropper) - too weak security for PKE

# Public Key Encryption (PKE) – definition

Given security parameter $\lambda \in \mathbb{N}$ and two finite sets $\mathcal{M}, \mathcal{R} \subseteq \{0,1\}^*$, a Public Key Encryption $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three algorithms:

- **Key-generation algorithm** (probabilistic): $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
- **Encryption algorithm** (probabilistic): Takes message $M \in \mathcal{M}$ and random $r \in \mathcal{R}$ and outputs $C \leftarrow \mathsf{Enc}(\mathsf{pk}, M, r)$
- **Decryption algorithm** (deterministic): Takes as input a secret key sk and ciphertext $C$, and outputs either a message $M' = \mathsf{Dec}(\mathsf{sk}, C) \in \mathcal{M}$ or $\perp \notin \mathcal{M}$ to indicate an invalid ciphertext.

**Correctness:** For all $M \in \mathcal{M}$, $Pr[\mathsf{Dec}(\mathsf{sk}, C) = M : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda), C \leftarrow \mathsf{Enc}(\mathsf{pk}, M, r)] \geqslant 1 - \delta$

- (Negligible) **Decryption error** ($\delta$) is also allowed (not all schemes have it)
- **Passive attacker** (eavesdropper) - too weak security for PKE **Why?**

# Public Key Encryption (PKE) – definition

Given security parameter $\lambda \in \mathbb{N}$ and two finite sets $\mathcal{M}, \mathcal{R} \subseteq \{0,1\}^*$, a Public Key Encryption $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three algorithms:

- **Key-generation algorithm** (probabilistic): $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
- **Encryption algorithm** (probabilistic): Takes message $M \in \mathcal{M}$ and random $r \in \mathcal{R}$ and outputs $C \leftarrow \mathsf{Enc}(\mathsf{pk}, M, r)$
- **Decryption algorithm** (deterministic): Takes as input a secret key sk and ciphertext $C$, and outputs either a message $M' = \mathsf{Dec}(\mathsf{sk}, C) \in \mathcal{M}$ or $\perp \notin \mathcal{M}$ to indicate an invalid ciphertext.

**Correctness:** For all $M \in \mathcal{M}$, $Pr[\mathsf{Dec}(\mathsf{sk}, C) = M : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda), C \leftarrow \mathsf{Enc}(\mathsf{pk}, M, r)] \geqslant 1 - \delta$

- (Negligible) **Decryption error** ($\delta$) is also allowed (not all schemes have it)

- **Passive attacker** (eavesdropper) - too weak security for PKE **Why?**
- **Active attacker** - **can craft messages** to encrypt as much as they want - **Always possible!**

# Public Key Encryption (PKE) – definition

Given security parameter $\lambda \in \mathbb{N}$ and two finite sets $\mathcal{M}, \mathcal{R} \subseteq \{0,1\}^*$, a Public Key Encryption $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three algorithms:

- **Key-generation algorithm** (probabilistic): $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
- **Encryption algorithm** (probabilistic): Takes message $M \in \mathcal{M}$ and random $r \in \mathcal{R}$ and outputs $C \leftarrow \mathsf{Enc}(\mathsf{pk}, M, r)$
- **Decryption algorithm** (deterministic): Takes as input a secret key sk and ciphertext $C$, and outputs either a message $M' = \mathsf{Dec}(\mathsf{sk}, C) \in \mathcal{M}$ or $\perp \notin \mathcal{M}$ to indicate an invalid ciphertext.

**Correctness:** For all $M \in \mathcal{M}$, $Pr[\mathsf{Dec}(\mathsf{sk}, C) = M : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda), C \leftarrow \mathsf{Enc}(\mathsf{pk}, M, r)] \geqslant 1 - \delta$

- (Negligible) **Decryption error** ($\delta$) is also allowed (not all schemes have it)
- **Passive attacker** (eavesdropper) - too weak security for PKE **Why?**
- **Active attacker** - **can craft messages** to encrypt as much as they want - **Always possible!**
  - encryption only requires the public key!

# Public Key Encryption (PKE) – definition

Given security parameter $\lambda \in \mathbb{N}$ and two finite sets $\mathcal{M}, \mathcal{R} \subseteq \{0,1\}^*$, a Public Key Encryption $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three algorithms:

- **Key-generation algorithm** (probabilistic): $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
- **Encryption algorithm** (probabilistic): Takes message $M \in \mathcal{M}$ and random $r \in \mathcal{R}$ and outputs $C \leftarrow \mathsf{Enc}(\mathsf{pk}, M, r)$
- **Decryption algorithm** (deterministic): Takes as input a secret key sk and ciphertext $C$, and outputs either a message $M' = \mathsf{Dec}(\mathsf{sk}, C) \in \mathcal{M}$ or $\perp \notin \mathcal{M}$ to indicate an invalid ciphertext.

**Correctness:** For all $M \in \mathcal{M}$, $Pr[\mathsf{Dec}(\mathsf{sk}, C) = M : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda), C \leftarrow \mathsf{Enc}(\mathsf{pk}, M, r)] \geqslant 1 - \delta$

- (Negligible) **Decryption error** ($\delta$) is also allowed (not all schemes have it)
- **Passive attacker** (eavesdropper) - too weak security for PKE **Why?**
- **Active attacker** - **can craft messages** to encrypt as much as they want - **Always possible!**
    - encryption only requires the public key!
    - What more could the attacker do?

# Security of Public Key Encryption (PKE)

**Baseline security**: **indistinguishability under chosen-plaintext attacks (IND-CPA)**

A PKE scheme $\Pi$ is called IND-CPA-secure if any PPT adversary $\mathcal{A}$ has only negligible advantage

$$Adv = \mathbf{Pr}\left(\mathrm{Exp}^{\text{ind-cpa}}_{\Pi(1^k)}(\mathcal{A}) = 1\right) - 1/2 = negl(k)\,.$$

in the following $\mathrm{Exp}^{\textbf{ind-cpa}}_{\Pi(1^k)}(\mathcal{A})$ **game (experiment)**:

| **Challenger** | | **Adversary** |
|---|---|---|
| $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{KGen}()$ | $\xrightarrow{\mathrm{pk}}$ | |
| | $\xleftarrow{M_i}$ | $M_i$ for number of $i$-s |
| | $\xrightarrow{\mathrm{Enc}(\mathrm{pk}, M_i)}$ | |
| $b \xleftarrow{\$} \{0,1\}$ | $\xleftarrow{(M_0^*, M_1^*)}$ | $M_0^*, M_1^*$ |
| $C \leftarrow \mathrm{Enc}(\mathrm{pk}, M_b^*)$ | $\xrightarrow{C}$ | |
| | $\xleftarrow{b'}$ | $b'$ |

Return 1 iff $b = b'$ otherwise 0.

- **A deterministic PKE can never be IND-CPA! Why?**

# Security of Public Key Encryption (PKE)

- **A deterministic PKE can never be IND-CPA! Why?**
  - (A deterministic PKE always gives the same ciphertext for the same message.)

# Security of Public Key Encryption (PKE)

- **A deterministic PKE can never be IND-CPA! Why?**
  - (A deterministic PKE always gives the same ciphertext for the same message.)
  - **Answer**: $\mathcal{A}$ can always query the encryption oracle for the messages $M_0^*, M_1^*$ and compare to the challenge ciphertext
  - $\Rightarrow$ For IND-CPA we need probabilistic encryption!

# Security of Public Key Encryption (PKE)

- **A deterministic PKE can never be IND-CPA! Why?**
  - (A deterministic PKE always gives the same ciphertext for the same message.)
  - **Answer**: $\mathcal{A}$ can always query the encryption oracle for the messages $M_0^*, M_1^*$ and compare to the challenge ciphertext
  - $\Rightarrow$ For IND-CPA we need probabilistic encryption!

**Why CPA is not sufficient ...**

# Security of Public Key Encryption (PKE)

- **A deterministic PKE can never be IND-CPA! Why?**
  - (A deterministic PKE always gives the same ciphertext for the same message.)
  - **Answer**: $\mathcal{A}$ can always query the encryption oracle for the messages $M_0^*, M_1^*$ and compare to the challenge ciphertext
  - $\Rightarrow$ For IND-CPA we need probabilistic encryption!

**Why CPA is not sufficient ...**

- **Active attacker**
  - **can craft messages** to encrypt as much as they want - **Always possible!**

# Security of Public Key Encryption (PKE)

- **A deterministic PKE can never be IND-CPA! Why?**
  - (A deterministic PKE always gives the same ciphertext for the same message.)
  - **Answer**: $\mathcal{A}$ can always query the encryption oracle for the messages $M_0^*, M_1^*$ and compare to the challenge ciphertext
  - $\Rightarrow$ For IND-CPA we need probabilistic encryption!

**Why CPA is not sufficient ...**

- **Active attacker**
  - **can craft messages** to encrypt as much as they want - **Always possible!**
  - **can craft ciphertexts** and use the decryption algorithm as an **oracle** to obtain the plaintexts

# Security of Public Key Encryption (PKE)

- **A deterministic PKE can never be IND-CPA! Why?**
  - (A deterministic PKE always gives the same ciphertext for the same message.)
  - **Answer**: $\mathcal{A}$ can always query the encryption oracle for the messages $M_0^*, M_1^*$ and compare to the challenge ciphertext
  - $\Rightarrow$ For IND-CPA we need probabilistic encryption!

**Why CPA is not sufficient ...**

- **Active attacker**
  - **can craft messages** to encrypt as much as they want - **Always possible!**
  - **can craft ciphertexts** and use the decryption algorithm as an **oracle** to obtain the plaintexts
    - access switched off **before** target ciphertext is given to the attacker
    - unlimited access, **before and after** the target ciphertext is made available (of course the target ciphertext can not be queried)

# Security of Public Key Encryption (PKE)

A PKE scheme $\Pi$ is called IND-CCA-secure (IND-CCA2-secure) if any PPT adversary $\mathcal{A}$ has only negligible advantage

$$Adv = \mathbf{Pr}\left(\mathsf{Exp}^{\text{ind-cca}}_{\Pi(1^k)}(\mathcal{A}) = 1\right) - 1/2 = negl(k)\,.$$

in the following $\mathsf{Exp}^{\text{ind-cca}}_{\Pi(1^k)}(\mathcal{A})$ **game (experiment)**:

| **Challenger** | | **Adversary** |
|---|---|---|
| $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}()$ | $\xrightarrow{\mathsf{pk}}$ | |
| | $\xleftarrow{M_i \text{ or } C_i}$ | $M_i$ or $C_i$ for number of $i$-s |
| | $\xrightarrow{\mathsf{Enc}(\mathsf{pk}, M_i) \text{ or } \mathsf{Dec}(\mathsf{pk}, C_i)}$ | |
| $b \xleftarrow{\$} \{0,1\}$ | $\xleftarrow{(M_0^*, M_1^*)}$ | $M_0^*, M_1^*$ |
| $C \leftarrow \mathsf{Enc}(\mathsf{pk}, M_b^*)$ | $\xrightarrow{C}$ | |
| | $\xleftarrow{M_i \text{ or } C_i}$ | (only in IND-CCA2 game) |
| | $\xrightarrow{\mathsf{Enc}(\mathsf{pk}, M_i) \text{ or } \mathsf{Dec}(\mathsf{pk}, C_i)}$ | |
| | $\xleftarrow{b'}$ | $b'$ |
| Return 1 iff $b = b'$ otherwise 0. | | |

Recall textbook RSA (for more info see I2C slides):

**Textbook RSA:**

**KeyGen**:

1. Choose two primes $p, q$ s.t. $|p| \approx |q|$
2. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$
3. Choose a random $e < \phi(N)$, s.t. $\gcd(e, \phi(N)) = 1$
4. Compute $d$ such that $ed = 1 \pmod{\phi(N)}$
5. Output public key $\mathrm{pk} = (N, e)$ and private key $\mathrm{sk} = d$

**Encrypt**:

Compute ciphertext as $C \leftarrow M^e \pmod{N}$

**Decrypt**:

Decrypt ciphertext as $M \leftarrow C^d \pmod{N}$

**Passive (Meet-in-the-middle) attack:**

- Let $C = M^e \pmod{N}$, and Eve knows that $M < 2^\ell$ (for example PIN, short password)

# An example walkthrough - Insecurity of textbook RSA

**Passive (Meet-in-the-middle) attack:**

- Let $C = M^{e} \pmod{N}$, and Eve knows that $M < 2^{\ell}$ (for example PIN, short password)
- With non-negligible probability $M = M_1 \cdot M_2$ with $M_1, M_2 < 2^{\ell/2}$

# An example walkthrough - Insecurity of textbook RSA

**Passive (Meet-in-the-middle) attack:**

- Let $C = M^{e} \pmod{N}$, and Eve knows that $M < 2^{\ell}$ (for example PIN, short password)
- With non-negligible probability $M = M_1 \cdot M_2$ with $M_1, M_2 < 2^{\ell/2}$
  - For $\ell$ of length 40 to 64 bits, probability that the plaintext can be factored in factors of approx. equal size is $18\% - 50\%$

**Passive (Meet-in-the-middle) attack:**

- Let $C = M^e \pmod N$, and Eve knows that $M < 2^\ell$ (for example PIN, short password)
- With non-negligible probability $M = M_1 \cdot M_2$ with $M_1, M_2 < 2^{\ell/2}$
  - For $\ell$ of length 40 to 64 bits, probability that the plaintext can be factored in factors of approx. equal size is $18\% - 50\%$
- RSA is multiplicative: $C = M_1^e \cdot M_2^e \pmod N$

**Passive (Meet-in-the-middle) attack:**

- Let $C = M^e \pmod{N}$, and Eve knows that $M < 2^\ell$ (for example PIN, short password)
- With non-negligible probability $M = M_1 \cdot M_2$ with $M_1, M_2 < 2^{\ell/2}$
  - For $\ell$ of length 40 to 64 bits, probability that the plaintext can be factored in factors of approx. equal size is $18\% - 50\%$
- RSA is multiplicative: $C = M_1^e \cdot M_2^e \pmod{N}$

**So:**

- Eve builds a sorted database $\{1^e, 2^e, \ldots (2^{\ell/2})^e\} \pmod{N}$

**Passive (Meet-in-the-middle) attack:**

- Let $C = M^e \pmod{N}$, and Eve knows that $M < 2^\ell$ (for example PIN, short password)
- With non-negligible probability $M = M_1 \cdot M_2$ with $M_1, M_2 < 2^{\ell/2}$
  - For $\ell$ of length 40 to 64 bits, probability that the plaintext can be factored in factors of approx. equal size is $18\% - 50\%$
- RSA is multiplicative: $C = M_1^e \cdot M_2^e \pmod{N}$

**So:**

- Eve builds a sorted database $\{1^e, 2^e, \ldots (2^{\ell/2})^e\} \pmod{N}$
- And searches for $\mathbf{c} = C/i^e$, $i \in \{1, 2, \ldots 2^{\ell/2}\}$ in the database

# An example walkthrough - Insecurity of textbook RSA

**Passive (Meet-in-the-middle) attack:**

- Let $C = M^e \pmod{N}$, and Eve knows that $M < 2^\ell$ (for example PIN, short password)
- With non-negligible probability $M = M_1 \cdot M_2$ with $M_1, M_2 < 2^{\ell/2}$
  - For $\ell$ of length 40 to 64 bits, probability that the plaintext can be factored in factors of approx. equal size is $18\% - 50\%$
- RSA is multiplicative: $C = M_1^e \cdot M_2^e \pmod{N}$

**So:**

- Eve builds a sorted database $\{1^e, 2^e, \ldots (2^{\ell/2})^e\} \pmod{N}$
- And searches for $\mathbf{c} = C/i^e$, $i \in \{1, 2, ... 2^{\ell/2}\}$ in the database
- $\mathbf{c}$ is by design of the shape $j^e$, and will show up in at most $2^{\ell/2}$ trials!

**Passive (Meet-in-the-middle) attack:**

- Let $C = M^e \pmod N$, and Eve knows that $M < 2^\ell$ (for example PIN, short password)
- With non-negligible probability $M = M_1 \cdot M_2$ with $M_1, M_2 < 2^{\ell/2}$
    - For $\ell$ of length 40 to 64 bits, probability that the plaintext can be factored in factors of approx. equal size is $18\% - 50\%$
- RSA is multiplicative: $C = M_1^e \cdot M_2^e \pmod N$

**So:**

- Eve builds a sorted database $\{1^e, 2^e, \ldots (2^{\ell/2})^e\} \pmod N$
- And searches for $\mathbf{c} = C/i^e$, $i \in \{1, 2, ...2^{\ell/2}\}$ in the database
- $\mathbf{c}$ is by design of the shape $j^e$, and will show up in at most $2^{\ell/2}$ trials!
- $\Rightarrow$ Message $M = i \cdot j$ recovered!
- $\Rightarrow$ **Message recovery in time and space cost of $\tilde{\mathcal{O}}(2^{\ell/2})$ (factors polynomial in $\ell$ neglected)**

# An example walkthrough - Insecurity of textbook RSA

**Active (Oracle) attack:**

- Suppose Eve wants to find out the message from the ciphertext $C = M^e \pmod{N}$, that was previously sent to Alice

## An example walkthrough - Insecurity of textbook RSA

**Active (Oracle) attack:**

- Suppose Eve wants to find out the message from the ciphertext $C = M^e \pmod{N}$, that was previously sent to Alice

**So:**

- Eve picks a random $R \xleftarrow{\$} \mathbb{Z}_N^*$ and computes $C' = C \cdot R^e \pmod{N}$

# An example walkthrough - Insecurity of textbook RSA

**Active (Oracle) attack:**

- Suppose Eve wants to find out the message from the ciphertext $C = M^e \pmod N$, that was previously sent to Alice

**So:**

- Eve picks a random $R \xleftarrow{\$} \mathbb{Z}_N^*$ and computes $C' = C \cdot R^e \pmod N$
- Eve sends $C'$ to Alice

# An example walkthrough - Insecurity of textbook RSA

**Active (Oracle) attack:**

- Suppose Eve wants to find out the message from the ciphertext $C = M^e \pmod{N}$, that was previously sent to Alice

**So:**

- Eve picks a random $R \xleftarrow{\$} \mathbb{Z}_N^*$ and computes $C' = C \cdot R^e \pmod{N}$
- Eve sends $C'$ to Alice
- Alice decrypts and sends to Eve: $M' \leftarrow C'^d \pmod{N} = (M^e R^e)^d \pmod{N} = MR \pmod{N}$

# An example walkthrough - Insecurity of textbook RSA

**Active (Oracle) attack:**

- Suppose Eve wants to find out the message from the ciphertext $C = M^e \pmod{N}$, that was previously sent to Alice

  **So:**
- Eve picks a random $R \xleftarrow{\$} \mathbb{Z}_N^*$ and computes $C' = C \cdot R^e \pmod{N}$
- Eve sends $C'$ to Alice
- Alice decrypts and sends to Eve: $M' \leftarrow C'^d \pmod{N} = (M^e R^e)^d \pmod{N} = MR \pmod{N}$
  - Alice does not notice anything, because for her $MR$ is just a random element of $\mathbb{Z}_N^*$, in no way connected to $M$

# An example walkthrough - Insecurity of textbook RSA

**Active (Oracle) attack:**

- Suppose Eve wants to find out the message from the ciphertext $C = M^e \pmod{N}$, that was previously sent to Alice

  **So:**

- Eve picks a random $R \xleftarrow{\$} \mathbb{Z}_N^*$ and computes $C' = C \cdot R^e \pmod{N}$
- Eve sends $C'$ to Alice
- Alice decrypts and sends to Eve: $M' \leftarrow C'^d \pmod{N} = (M^e R^e)^d \pmod{N} = MR \pmod{N}$
  - Alice does not notice anything, because for her $MR$ is just a random element of $\mathbb{Z}_N^*$, in no way connected to $M$
- $\Rightarrow$ **Message recovery in 1 oracle query!**

**Active (Oracle) attack:**

- Suppose Eve wants to find out the message from the ciphertext $C = M^e \pmod N$, that was previously sent to Alice

**So:**

- Eve picks a random $R \xleftarrow{\$} \mathbb{Z}_N^*$ and computes $C' = C \cdot R^e \pmod N$
- Eve sends $C'$ to Alice
- Alice decrypts and sends to Eve: $M' \leftarrow C'^d \pmod N = (M^e R^e)^d \pmod N = MR \pmod N$
    - Alice does not notice anything, because for her $MR$ is just a random element of $\mathbb{Z}_N^*$, in no way connected to $M$
- $\Rightarrow$ **Message recovery in 1 oracle query!**

**Conclussion:**

- We need some sort of randomization of the message! (we need IND-CPA)
- The adversary should not be able to construct valid ciphertexts! (we need IND-CCA)

# Summary

**Today:**

- Public Key Cryptography - a Recap
- Security of PKC
- Security of Public Key Encryption and Key Encapsulation

# Summary

**Today:**

- Public Key Cryptography - a Recap
- Security of PKC
- Security of Public Key Encryption and Key Encapsulation

**Next time:**

- Security of Public Key Encryption and Key Encapsulation (contd.)
- Security of Digital Signatures