CS 6160 Cryptology Lecture 15: Key Management, Public-Key Exchange & Security Definitions for PKC

Maria Francis

November 3, 2020

Need for Public Key Cryptography

- We saw secret/private/symmetric key cryptography so far.

Need for Public Key Cryptography

- We saw secret/private/symmetric key cryptography so far.
  How do the parties share the secret key in the first place?

Need for Public Key Cryptography

- We saw secret/private/symmetric key cryptography so far. How do the parties share the secret key in the first place?
- They can assume a safe communication channel like a trusted courier service or they meet in person to share keys.

Need for Public Key Cryptography

- We saw secret/private/symmetric key cryptography so far. How do the parties share the secret key in the first place?
- They can assume a safe communication channel like a trusted courier service or they meet in person to share keys.
- Not very scalable. What if we need every pair of employees in a company to talk to each other? And to the servers and remote printers?

Need for Public Key Cryptography

- We saw secret/private/symmetric key cryptography so far. How do the parties share the secret key in the first place?
- They can assume a safe communication channel like a trusted courier service or they meet in person to share keys.
- Not very scalable. What if we need every pair of employees in a company to talk to each other? And to the servers and remote printers?
- Smartcards store keys, example of secure hardware, resilient to attacks more than a computer.

Need for Public Key Cryptography

- We saw secret/private/symmetric key cryptography so far. How do the parties share the secret key in the first place?
- They can assume a safe communication channel like a trusted courier service or they meet in person to share keys.
- Not very scalable. What if we need every pair of employees in a company to talk to each other? And to the servers and remote printers?
- Smartcards store keys, example of secure hardware, resilient to attacks more than a computer.
- But limited memory and cannot store many keys.

Need for Public Key Cryptography

- What if we are not talking of closed organizations but open systems with temporary transactions like a credit card transaction with an e-commerce site?

Need for Public Key Cryptography

- What if we are not talking of closed organizations but open systems with temporary transactions like a credit card transaction with an e-commerce site?
- Then no solution from SKC.

Need for Public Key Cryptography

- What if we are not talking of closed organizations but open systems with temporary transactions like a credit card transaction with an e-commerce site?
- Then no solution from SKC.
- Does not make SKC useless since you need to address only key transfer stage and everything else can happen over an insecure channel.

Need for Public Key Cryptography

- What if we are not talking of closed organizations but open systems with temporary transactions like a credit card transaction with an e-commerce site?
- Then no solution from SKC.
- Does not make SKC useless since you need to address only key transfer stage and everything else can happen over an insecure channel.
- The main issues here are :
  - Key distribution

Need for Public Key Cryptography

- What if we are not talking of closed organizations but open systems with temporary transactions like a credit card transaction with an e-commerce site?
- Then no solution from SKC.
- Does not make SKC useless since you need to address only key transfer stage and everything else can happen over an insecure channel.
- The main issues here are :
  - Key distribution
  - Storing and managing large number of secret keys

Need for Public Key Cryptography

- What if we are not talking of closed organizations but open systems with temporary transactions like a credit card transaction with an e-commerce site?
- Then no solution from SKC.
- Does not make SKC useless since you need to address only key transfer stage and everything else can happen over an insecure channel.
- The main issues here are :
  - Key distribution
  - Storing and managing large number of secret keys
  - Inapplicability of SKC to open systems.

Key Distribution Centers

- Key Distribution Center (KDC) – a trusted entity that establishes shared keys.

Key Distribution Centers

- Key Distribution Center (KDC) – a trusted entity that establishes shared keys.
- When ith employee joins an organization:
  - Share a key with that employee and the KDC
  - Generate $i - 1$ keys and give those keys to the employee
  - Send the jth employee k_j key by encrypting the key using the key that jth employee shares with KDC

Key Distribution Centers

- Key Distribution Center (KDC) – a trusted entity that establishes shared keys.
- When ith employee joins an organization:
  - Share a key with that employee and the KDC
  - Generate $i-1$ keys and give those keys to the employee
  - Send the jth employee k_j key by encrypting the key using the key that jth employee shares with KDC
- Or generate keys on demand, online.
  - KDC shares a key with each employee.
  - Alice wants to talk to Bob, she sends that request to KDC.
  - KDC chooses a new random key called session key and sends this key to Alice (encrypted using k_A) and Bob (using k_B).
  - Once Alice and Bob is done with the conversation the session key is erased and for next one KDC has to be contacted again.

Advantages of the second alternative

- Each employee needs to store only one long-term secret key (one with the KDC).

Advantages of the second alternative

- Each employee needs to store only one long-term secret key (one with the KDC).
- KDC needs to store many long-term keys.

Advantages of the second alternative

- Each employee needs to store only one long-term secret key (one with the KDC).
- KDC needs to store many long-term keys.

  The KDC can be kept in a secure location and be given the highest possible protection against network attacks.

Advantages of the second alternative

- Each employee needs to store only one long-term secret key (one with the KDC).
- KDC needs to store many long-term keys.

  The KDC can be kept in a secure location and be given the highest possible protection against network attacks.
- When an employee joins only one key need to generated and no communication with other employees are needed.

Disadvantages of KDC

- KDCs simplify key distribution in large organizations.

Disadvantages of KDC

- KDCs simplify key distribution in large organizations.
- But there are drawbacks:

Disadvantages of KDC

- KDCs simplify key distribution in large organizations.
- But there are drawbacks:
  - Single point of attack. KDCs become a high value target and vulnerable to external and internal attacks.

Disadvantages of KDC

- KDCs simplify key distribution in large organizations.
- But there are drawbacks:
  - Single point of attack. KDCs become a high value target and vulnerable to external and internal attacks.
  - Single point of failure. if the KDC is down, secure communication is temporarily impossible.

Disadvantages of KDC

- KDCs simplify key distribution in large organizations.
- But there are drawbacks:
  - Single point of attack. KDCs become a high value target and vulnerable to external and internal attacks.
  - Single point of failure. if the KDC is down, secure communication is temporarily impossible.
- Solution that is often done: Replicate the KDC.
  - More points of attack possible.
  - More updates needed to add new employees.

Protocols for key distribution

- Needham-Schroeder protocol forms the core of Kerberos - a widely used protocol in universities and companies in Windows and UNIX systems.

Protocols for key distribution

- Needham-Schroeder protocol forms the core of Kerberos - a widely used protocol in universities and companies in Windows and UNIX systems.
- One big difference:
  - Alice contacts the KDC and says I want to talk to Bob, KDC sends encrypted session key + session key encrypted with Bob's key to Alice.

Protocols for key distribution

- Needham-Schroeder protocol forms the core of Kerberos - a widely used protocol in universities and companies in Windows and UNIX systems.
- One big difference:
  - Alice contacts the KDC and says I want to talk to Bob, KDC sends encrypted session key + session key encrypted with Bob's key to Alice.
  - Alice then forwards the second ciphertext (called the ticket) to Bob.

Protocols for key distribution

- Needham-Schroeder protocol forms the core of Kerberos - a widely used protocol in universities and companies in Windows and UNIX systems.
- One big difference:
  - Alice contacts the KDC and says I want to talk to Bob, KDC sends encrypted session key + session key encrypted with Bob's key to Alice.
  - Alice then forwards the second ciphertext (called the ticket) to Bob.
  - It helps authentication as well.

Protocols for key distribution

- Needham-Schroeder protocol forms the core of Kerberos - a widely used protocol in universities and companies in Windows and UNIX systems.
- One big difference:
  - Alice contacts the KDC and says I want to talk to Bob, KDC sends encrypted session key + session key encrypted with Bob's key to Alice.
  - Alice then forwards the second ciphertext (called the ticket) to Bob.
  - It helps authentication as well. Bob can be assured that he is talking to Alice.

Protocols for key distribution

- Needham-Schroeder protocol forms the core of Kerberos - a widely used protocol in universities and companies in Windows and UNIX systems.
- One big difference:
  - Alice contacts the KDC and says I want to talk to Bob, KDC sends encrypted session key + session key encrypted with Bob's key to Alice.
  - Alice then forwards the second ciphertext (called the ticket) to Bob.
  - It helps authentication as well. Bob can be assured that he is talking to Alice.
  - Reduce load on KDC, no communication with Bob and no need to check if Bob is online.

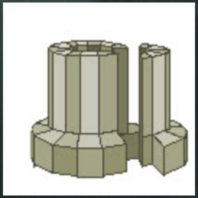
Protocols for key distribution

- Needham-Schroeder protocol forms the core of Kerberos - a widely used protocol in universities and companies in Windows and UNIX systems.
- One big difference:
  - Alice contacts the KDC and says I want to talk to Bob, KDC sends encrypted session key + session key encrypted with Bob's key to Alice.
  - Alice then forwards the second ciphertext (called the ticket) to Bob.
  - It helps authentication as well. Bob can be assured that he is talking to Alice.
  - Reduce load on KDC, no communication with Bob and no need to check if Bob is online.
  - Alice can re-initiate the conversation with Bob without KDC by resending the ticket.

Key Distribution

Alice

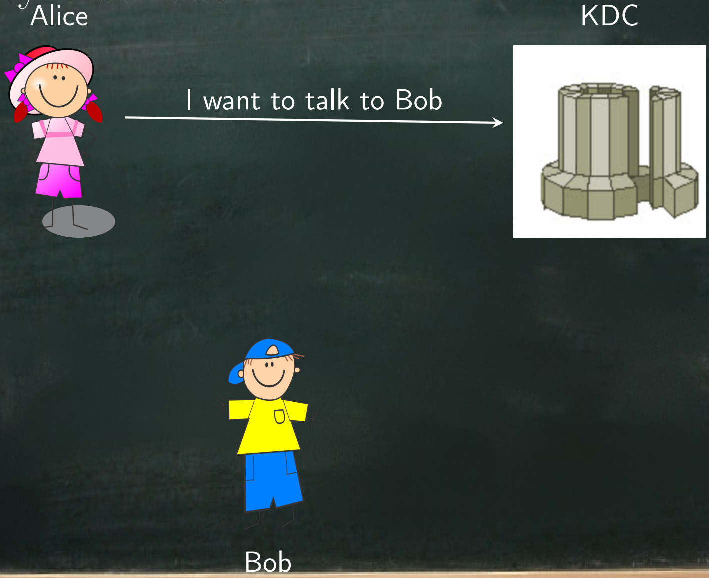
KDC



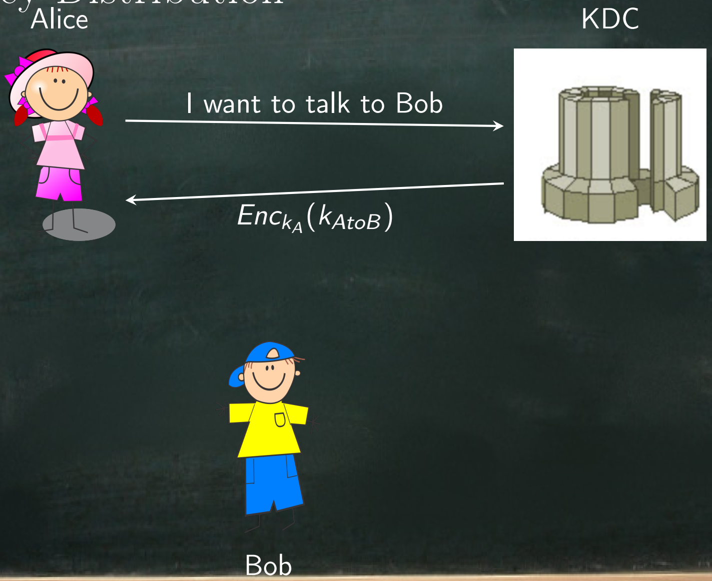
Bob

03/11/2020

Key Distribution

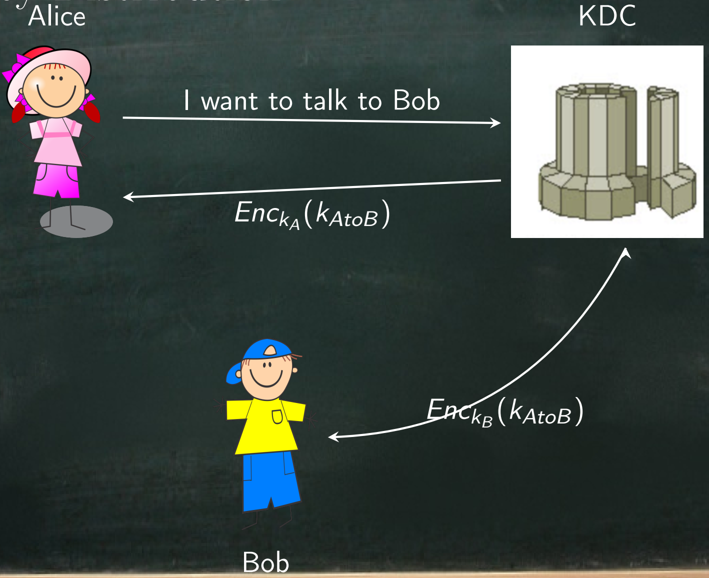


Key Distribution

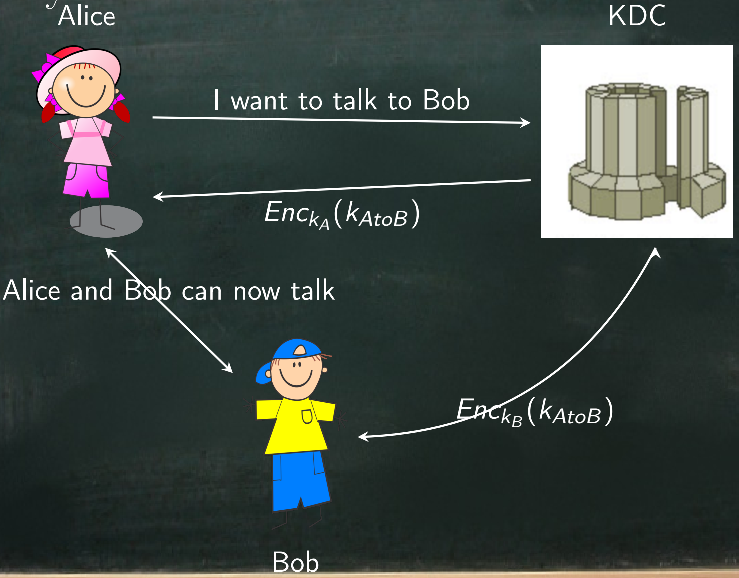


03/11/2020

Key Distribution



Key Distribution

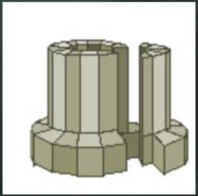


03/11/2020

Kerberos

Alice

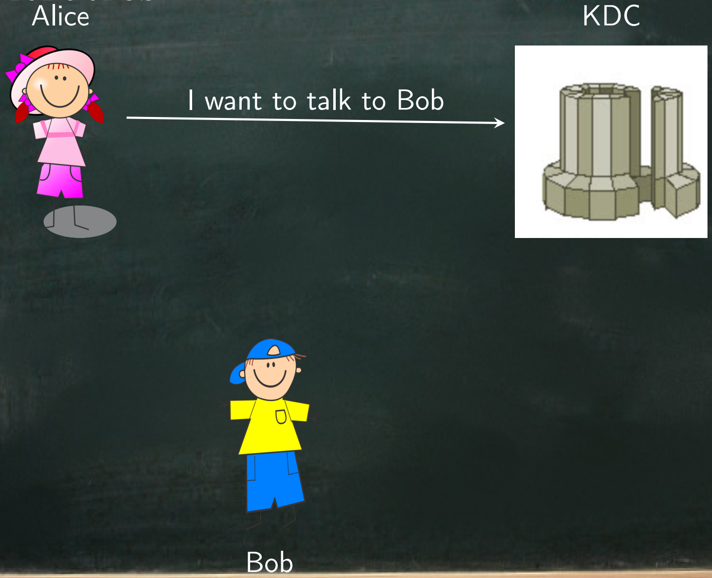
KDC



Bob

03/11/2020

Kerberos

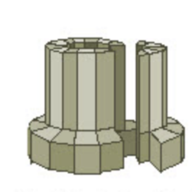


03/11/2020

Kerberos

Alice

KDC

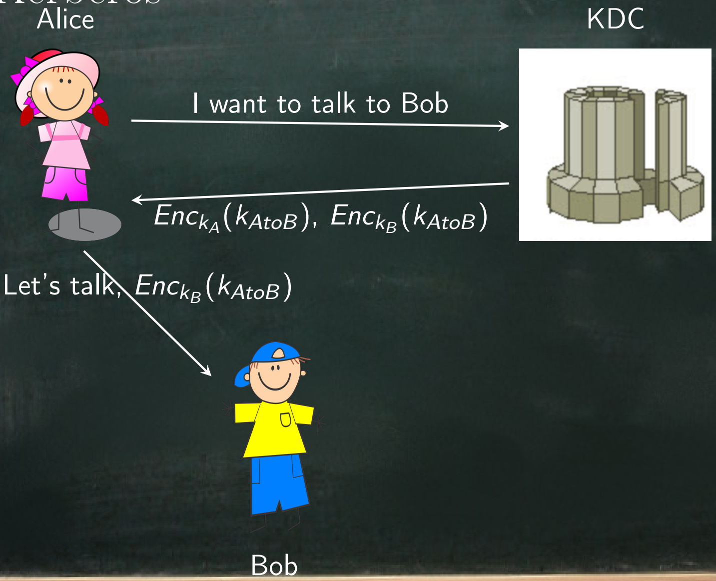


I want to talk to Bob

$Enc_{k_A}(k_{AtoB}),\ Enc_{k_B}(k_{AtoB})$

Bob

Kerberos


Alice
KDC
I want to talk to Bob
$Enc_{k_A}(k_{AtoB}),\ Enc_{k_B}(k_{AtoB})$
Let's talk, $Enc_{k_B}(k_{AtoB})$
Bob

Protocols for key distribution

- KDCs and Kerberos all need a private and authenticated channel to share keys at some point.

Protocols for key distribution

- KDCs and Kerberos all need a private and authenticated channel to share keys at some point.
- This won't work in open systems like Internet.

Protocols for key distribution

- KDCs and Kerberos all need a private and authenticated channel to share keys at some point.
- This won't work in open systems like Internet.
- In 1976, Whitfield Diffie and Martin Hellman published a paper with the innocent-looking title "New Directions in Cryptography."

Protocols for key distribution

- KDCs and Kerberos all need a private and authenticated channel to share keys at some point.
- This won't work in open systems like Internet.
- In 1976, Whitfield Diffie and Martin Hellman published a paper with the innocent-looking title "New Directions in Cryptography."
- They observed asymmetry in the world where there are certain actions that can be easily performed but not easily reversed.

Protocols for key distribution

- KDCs and Kerberos all need a private and authenticated channel to share keys at some point.
- This won't work in open systems like Internet.
- In 1976, Whitfield Diffie and Martin Hellman published a paper with the innocent-looking title "New Directions in Cryptography."
- They observed asymmetry in the world where there are certain actions that can be easily performed but not easily reversed. Like locking with a lock or breaking a vase!

Protocols for key distribution

- KDCs and Kerberos all need a private and authenticated channel to share keys at some point.
- This won't work in open systems like Internet.
- In 1976, Whitfield Diffie and Martin Hellman published a paper with the innocent-looking title "New Directions in Cryptography."
- They observed asymmetry in the world where there are certain actions that can be easily performed but not easily reversed. Like locking with a lock or breaking a vase!
- Diffie and Hellman used to derive interactive protocols for secure key exchange.

Protocols for key distribution

- KDCs and Kerberos all need a private and authenticated channel to share keys at some point.
- This won't work in open systems like Internet.
- In 1976, Whitfield Diffie and Martin Hellman published a paper with the innocent-looking title "New Directions in Cryptography."
- They observed asymmetry in the world where there are certain actions that can be easily performed but not easily reversed. Like locking with a lock or breaking a vase!
- Diffie and Hellman used to derive interactive protocols for secure key exchange.
- They indeed created a revolution with the first steps into PKC.

Diffie-Hellman key-exchange protocol

- We prove its security against eavesdropping adversaries, i.e. the parties communicate over a public but authenticated channel.

Diffie-Hellman key-exchange protocol

- We prove its security against eavesdropping adversaries, i.e. the parties communicate over a public but authenticated channel.
- There is the key point that no prior information is shared between Alice and Bob so there is nothing prevent $\mathcal{A}$ from impersonating one of them.

Diffie-Hellman key-exchange protocol

- We prove its security against eavesdropping adversaries, i.e. the parties communicate over a public but authenticated channel.
- There is the key point that no prior information is shared between Alice and Bob so there is nothing prevent $\mathcal{A}$ from impersonating one of them.
- We define security :

Diffie-Hellman key-exchange protocol

- We prove its security against eavesdropping adversaries, i.e. the parties communicate over a public but authenticated channel.
- There is the key point that no prior information is shared between Alice and Bob so there is nothing prevent $\mathcal{A}$ from impersonating one of them.
- We define security : A key-exchange protocol is secure if the key output of Alice and Bob is completely unguessable by an an Eve. I.e. distinguish between the key k generated by Alice and Bob from a uniform key of length n.

Diffie-Hellman key-exchange protocol

- We prove its security against eavesdropping adversaries, i.e. the parties communicate over a public but authenticated channel.
- There is the key point that no prior information is shared between Alice and Bob so there is nothing prevent $\mathcal{A}$ from impersonating one of them.
- We define security : A key-exchange protocol is secure if the key output of Alice and Bob is completely unguessable by an an Eve. I.e. distinguish between the key k generated by Alice and Bob from a uniform key of length n.
- Stronger notion that saying unable to compute k exactly, since this is the key used for SKC.

Key-exchange experiment

$KE^{eav}_{\mathcal{A},\Pi}(1^n)$

1. Two parties with 1^n as input execute the probabilistic protocol Π using independent random bits.
   This results in a transcript trans containing all the messages sent by the parties and a key k output by each of the parties.

Key-exchange experiment

$KE^{eav}_{\mathcal{A},\Pi}(1^n)$

1. Two parties with 1^n as input execute the probabilistic protocol Π using independent random bits.
   This results in a transcript trans containing all the messages sent by the parties and a key k output by each of the parties.
2. A uniform bit $b \in \{0, 1\}$ is chosen. If $b = 0$, set $k^{'} = k$ and if $b = 1$ choose $k^{'}$ uniformly at random from $\{0, 1\}^n$.

Key-exchange experiment

$KE^{eav}_{\mathcal{A},\Pi}(1^n)$

1. Two parties with 1^n as input execute the probabilistic protocol Π using independent random bits.
   This results in a transcript trans containing all the messages sent by the parties and a key k output by each of the parties.
2. A uniform bit $b \in \{0, 1\}$ is chosen. If $b = 0$, set $k^{'} = k$ and if $b = 1$ choose $k^{'}$ uniformly at random from $\{0, 1\}^n$.
3. $\mathcal{A}$ is given trans and $k^{'}$ and outputs a bit $b^{'}$.

Key-exchange experiment

$KE^{eav}_{\mathcal{A},\Pi}(1^n)$

1. Two parties with 1^n as input execute the probabilistic protocol Π using independent random bits.
   This results in a transcript trans containing all the messages sent by the parties and a key k output by each of the parties.
2. A uniform bit $b \in \{0, 1\}$ is chosen. If $b = 0$, set $k^{'} = k$ and if $b = 1$ choose $k^{'}$ uniformly at random from $\{0, 1\}^n$.
3. $\mathcal{A}$ is given trans and $k^{'}$ and outputs a bit $b^{'}$.
4. The output is defined to be 1 if $b^{'} = b$ (and $\mathcal{A}$ succeeds) and 0 otherwise.

Key-exchange experiment

$KE^{eav}_{\mathcal{A},\Pi}(1^n)$

1. Two parties with 1^n as input execute the probabilistic protocol Π using independent random bits.
   This results in a transcript trans containing all the messages sent by the parties and a key k output by each of the parties.
2. A uniform bit $b \in \{0,1\}$ is chosen. If $b = 0$, set $k' = k$ and if $b = 1$ choose k' uniformly at random from $\{0,1\}^n$.
3. $\mathcal{A}$ is given trans and k' and outputs a bit b'.
4. The output is defined to be 1 if $b' = b$ (and $\mathcal{A}$ succeeds) and 0 otherwise.

Why give $\mathcal{A}$ the transcript?

Key-exchange experiment

$KE^{eav}_{\mathcal{A},\Pi}(1^n)$

1. Two parties with 1^n as input execute the probabilistic protocol Π using independent random bits.
   This results in a transcript trans containing all the messages sent by the parties and a key k output by each of the parties.
2. A uniform bit $b \in \{0, 1\}$ is chosen. If $b = 0$, set $k^{'} = k$ and if $b = 1$ choose $k^{'}$ uniformly at random from $\{0, 1\}^n$.
3. $\mathcal{A}$ is given trans and $k^{'}$ and outputs a bit $b^{'}$.
4. The output is defined to be 1 if $b^{'} = b$ (and $\mathcal{A}$ succeeds) and 0 otherwise.

Why give $\mathcal{A}$ the transcript? To capture that $\mathcal{A}$ can eavesdrop the entire interaction.

Diffie-Hellman Key-exchange

A key-exchange protocol Π is secure in the presence of an eavesdropper if for all PPT $\mathcal{A}$,

$$Pr[KE^{eav}_{\mathcal{A},\Pi}(1^n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n).$$

Diffie-Hellman Key-exchange

A key-exchange protocol Π is secure in the presence of an eavesdropper if for all PPT $\mathcal{A}$,

$$Pr[KE^{eav}_{\mathcal{A},\Pi}(1^n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n).$$

- Alice runs $\mathcal{G}(1^n)$ to obtain (G, q, g).

Diffie-Hellman Key-exchange

A key-exchange protocol Π is secure in the presence of an eavesdropper if for all PPT $\mathcal{A}$,

$$Pr[KE^{eav}_{\mathcal{A},\Pi}(1^n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n).$$

- Alice runs $\mathcal{G}(1^n)$ to obtain (G, q, g).
- Alice chooses a uniform $x \in \mathbb{Z}_q$ and computes $h_A := g^x$.

Diffie-Hellman Key-exchange

A key-exchange protocol Π is secure in the presence of an eavesdropper if for all PPT $\mathcal{A}$,

$$Pr[KE^{eav}_{\mathcal{A},\Pi}(1^n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n).$$

- Alice runs $\mathcal{G}(1^n)$ to obtain (G, q, g).
- Alice chooses a uniform $x \in \mathbb{Z}_q$ and computes $h_A := g^x$.
- Alice sends (G, q, g, h_A) to Bob.

Diffie-Hellman Key-exchange

A key-exchange protocol Π is secure in the presence of an eavesdropper if for all PPT $\mathcal{A}$,

$$Pr[KE^{eav}_{\mathcal{A},\Pi}(1^n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n).$$

- Alice runs $\mathcal{G}(1^n)$ to obtain (G, q, g).
- Alice chooses a uniform $x \in \mathbb{Z}_q$ and computes $h_A := g^x$.
- Alice sends (G, q, g, h_A) to Bob.
- Bob receives (G, q, g, h_A). He chooses a uniform $y \in \mathbb{Z}_q$ and computes $h_B = g^y$ and sends Alice h_B.

Diffie-Hellman Key-exchange

A key-exchange protocol Π is secure in the presence of an eavesdropper if for all PPT $\mathcal{A}$,

$$Pr[KE^{eav}_{\mathcal{A},\Pi}(1^n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n).$$

- Alice runs $\mathcal{G}(1^n)$ to obtain (G, q, g).
- Alice chooses a uniform $x \in \mathbb{Z}_q$ and computes $h_A := g^x$.
- Alice sends (G, q, g, h_A) to Bob.
- Bob receives (G, q, g, h_A). He chooses a uniform $y \in \mathbb{Z}_q$ and computes $h_B = g^y$ and sends Alice h_B.
- Bob outputs $k_B = h_A^y$.

Diffie-Hellman Key-exchange

A key-exchange protocol Π is secure in the presence of an eavesdropper if for all PPT $\mathcal{A}$,

$$Pr[KE^{eav}_{\mathcal{A},\Pi}(1^n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n).$$

- Alice runs $\mathcal{G}(1^n)$ to obtain (G, q, g).
- Alice chooses a uniform $x \in \mathbb{Z}_q$ and computes $h_A := g^x$.
- Alice sends (G, q, g, h_A) to Bob.
- Bob receives (G, q, g, h_A). He chooses a uniform $y \in \mathbb{Z}_q$ and computes $h_B = g^y$ and sends Alice h_B.
- Bob outputs $k_B = h_A^y$.
- Alice receives h_B and outputs the key $k_A = h_B^x$.

Diffie-Hellman key-exchange

Alice

bob

$$\boxed{\begin{array}{c} x \leftarrow \mathbb{Z}_q \\ h_A = g^x \end{array}}$$

Usually, (G, q, g) are standardized and are fixed and known before the protocol begins.

Diffie-Hellman key-exchange

Alice

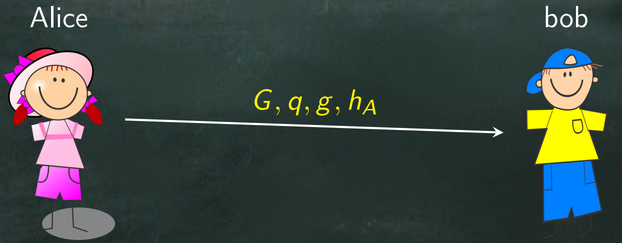
bob

$$G, q, g, h_A \longrightarrow$$

$$\boxed{\begin{array}{c} x \leftarrow \mathbb{Z}_q \\ h_A = g^x \end{array}}$$

Usually, (G, q, g) are standardized and are fixed and known before the protocol begins.

Diffie-Hellman key-exchange

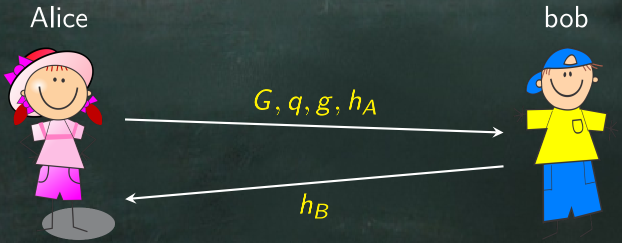


$$x \leftarrow \mathbb{Z}_q \qquad y \leftarrow \mathbb{Z}_q$$
$$h_A = g^x \qquad h_B = g^y$$

Usually, (G, q, g) are standardized and are fixed and known before the protocol begins.

03/11/2020

Diffie-Hellman key-exchange



$x \leftarrow \mathbb{Z}_q$
$h_A = g^x$

$y \leftarrow \mathbb{Z}_q$
$h_B = g^y$

Usually, (G, q, g) are standardized and are fixed and known before the protocol begins.

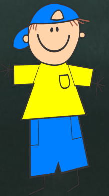
Diffie-Hellman key-exchange

Alice

G, q, g, h_A →

← h_B

bob



$$\boxed{\begin{array}{c} x \leftarrow \mathbb{Z}_q \\ h_A = g^x \end{array}}$$

$$\boxed{\begin{array}{c} y \leftarrow \mathbb{Z}_q \\ h_B = g^y \end{array}}$$

Usually, (G, q, g) are standardized and are fixed and known before the protocol begins.

Security Proof

- Correctness: easy to see. (Verify!)

Security Proof

- Correctness: easy to see. (Verify!) Proof of security wasn't provided by Diffie & Hellman since the framework was not there.

Security Proof

- Correctness: easy to see. (Verify!) Proof of security wasn't provided by Diffie & Hellman since the framework was not there.
- As a minimal requirement we need the discrete-logarithm problem to be hard relative to $\mathcal{G}$.

Security Proof

- Correctness: easy to see. (Verify!) Proof of security wasn't provided by Diffie & Hellman since the framework was not there.
- As a minimal requirement we need the discrete-logarithm problem to be hard relative to $\mathcal{G}$.
- If not,

Security Proof

- Correctness: easy to see. (Verify!) Proof of security wasn't provided by Diffie & Hellman since the framework was not there.
- As a minimal requirement we need the discrete-logarithm problem to be hard relative to $\mathcal{G}$.
- If not, given trans, $\mathcal{A}$ can compute one of the secret values x and compute the shared key.

Security Proof

- Correctness: easy to see. (Verify!) Proof of security wasn't provided by Diffie & Hellman since the framework was not there.
- As a minimal requirement we need the discrete-logarithm problem to be hard relative to $\mathcal{G}$.
- If not, given trans, $\mathcal{A}$ can compute one of the secret values x and compute the shared key.
- It is necessary but is it sufficient?

Security Proof

- Correctness: easy to see. (Verify!) Proof of security wasn't provided by Diffie & Hellman since the framework was not there.
- As a minimal requirement we need the discrete-logarithm problem to be hard relative to $\mathcal{G}$.
- If not, given trans, $\mathcal{A}$ can compute one of the secret values x and compute the shared key.
- It is necessary but is it sufficient?
- No, since there are other ways of computing the key without explicitly computing x or y.

Security Proof

- Correctness: easy to see. (Verify!) Proof of security wasn't provided by Diffie & Hellman since the framework was not there.
- As a minimal requirement we need the discrete-logarithm problem to be hard relative to $\mathcal{G}$.
- If not, given trans, $\mathcal{A}$ can compute one of the secret values x and compute the shared key.
- It is necessary but is it sufficient?
- No, since there are other ways of computing the key without explicitly computing x or y.
- What about CDH?

Security Proof

- Correctness: easy to see. (Verify!) Proof of security wasn't provided by Diffie & Hellman since the framework was not there.
- As a minimal requirement we need the discrete-logarithm problem to be hard relative to $\mathcal{G}$.
- If not, given trans, $\mathcal{A}$ can compute one of the secret values x and compute the shared key.
- It is necessary but is it sufficient?
- No, since there are other ways of computing the key without explicitly computing x or y.
- What about CDH? Not enough either, since it only guarantees that g^{xy} is hard to compute in its entirety from the transcript.

Security Proof

- Correctness: easy to see. (Verify!) Proof of security wasn't provided by Diffie & Hellman since the framework was not there.
- As a minimal requirement we need the discrete-logarithm problem to be hard relative to $\mathcal{G}$.
- If not, given trans, $\mathcal{A}$ can compute one of the secret values x and compute the shared key.
- It is necessary but is it sufficient?
- No, since there are other ways of computing the key without explicitly computing x or y.
- What about CDH? Not enough either, since it only guarantees that g^{xy} is hard to compute in its entirety from the transcript.
- We need g^{xy} *is indistinguishable from uniform for* $\mathcal{A}$.

Security Proof

- Correctness: easy to see. (Verify!) Proof of security wasn't provided by Diffie & Hellman since the framework was not there.
- As a minimal requirement we need the discrete-logarithm problem to be hard relative to $\mathcal{G}$.
- If not, given trans, $\mathcal{A}$ can compute one of the secret values x and compute the shared key.
- It is necessary but is it sufficient?
- No, since there are other ways of computing the key without explicitly computing x or y.
- What about CDH? Not enough either, since it only guarantees that g^{xy} is hard to compute in its entirety from the transcript.
- We need g^{xy} *is indistinguishable from uniform for* $\mathcal{A}$. Which is exactly DDH.

Security Proof

If the decisional Diffie-Hellman problem is hard relative to $\mathcal{G}$ then the DH key-exchange protocol Π is secure in the presence of an eavesdropper.

Security Proof

If the decisional Diffie-Hellman problem is hard relative to $\mathcal{G}$ then the DH key-exchange protocol Π is secure in the presence of an eavesdropper.

- The experiment is slightly different : we ask the shared key to be indistinguishable from a uniform element of G rather than a uniform n-bit string.

Security Proof

If the decisional Diffie-Hellman problem is hard relative to $\mathcal{G}$ then the DH key-exchange protocol Π is secure in the presence of an eavesdropper.

- The experiment is slightly different : we ask the shared key to be indistinguishable from a uniform element of G rather than a uniform n-bit string.
- This is not going to work in practice since group elements are not useful as keys typically and the representation of a uniform group element is not in general uniform bit-string.

Security Proof

If the decisional Diffie-Hellman problem is hard relative to $\mathcal{G}$ then the DH key-exchange protocol Π is secure in the presence of an eavesdropper.

- The experiment is slightly different : we ask the shared key to be indistinguishable from a uniform element of G rather than a uniform n-bit string.
- This is not going to work in practice since group elements are not useful as keys typically and the representation of a uniform group element is not in general uniform bit-string.
- But we assume $\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n)$ denotes a modified experiment where if $b = 1$ $\mathcal{A}$ is given k' chosen uniformly from G instead of uniform n-bit string.

Security Proof Details

$$Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1]$$
$$= \frac{1}{2} \cdot Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1 | b = 0] + \frac{1}{2} \cdot Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1 | b = 1]$$

Security Proof Details

$$Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1]$$
$$= \frac{1}{2} \cdot Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1 | b = 0] + \frac{1}{2} \cdot Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1 | b = 1]$$
$$= \frac{1}{2} \cdot Pr[\mathcal{A}(g^{xy}) = 0] + \frac{1}{2} \cdot Pr[\mathcal{A}(g^{z}) = 1]$$

Security Proof Details

$$Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1]$$
$$= \frac{1}{2} \cdot Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1 | b = 0] + \frac{1}{2} \cdot Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1 | b = 1]$$
$$= \frac{1}{2} \cdot Pr[\mathcal{A}(g^{xy}) = 0] + \frac{1}{2} \cdot Pr[\mathcal{A}(g^{z}) = 1]$$
$$= \frac{1}{2} \cdot (1 - Pr[\mathcal{A}(g^{xy}) = 1]) + \frac{1}{2} \cdot Pr[\mathcal{A}(g^{z}) = 1]$$

Security Proof Details

$$
\begin{gathered}
Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1] \\
= \frac{1}{2} \cdot Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1 | b = 0] + \frac{1}{2} \cdot Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1 | b = 1] \\
= \frac{1}{2} \cdot Pr[\mathcal{A}(g^{xy}) = 0] + \frac{1}{2} \cdot Pr[\mathcal{A}(g^{z}) = 1] \\
= \frac{1}{2} \cdot (1 - Pr[\mathcal{A}(g^{xy}) = 1]) + \frac{1}{2} \cdot Pr[\mathcal{A}(g^{z}) = 1] \\
= \frac{1}{2} + \frac{1}{2}(Pr[\mathcal{A}(g^{z}) = 1] - Pr[\mathcal{A}(g^{xy}) = 1])
\end{gathered}
$$

Security Proof Details

$$
\begin{aligned}
&Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1] \\
&= \tfrac{1}{2} \cdot Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1 | b = 0] + \tfrac{1}{2} \cdot Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1 | b = 1] \\
&= \tfrac{1}{2} \cdot Pr[\mathcal{A}(g^{xy}) = 0] + \tfrac{1}{2} \cdot Pr[\mathcal{A}(g^{z}) = 1] \\
&= \tfrac{1}{2} \cdot (1 - Pr[\mathcal{A}(g^{xy}) = 1]) + \tfrac{1}{2} \cdot Pr[\mathcal{A}(g^{z}) = 1] \\
&= \tfrac{1}{2} + \tfrac{1}{2}(Pr[\mathcal{A}(g^{z}) = 1] - Pr[\mathcal{A}(g^{xy}) = 1]) \\
&\le \tfrac{1}{2} + \tfrac{1}{2}|Pr[\mathcal{A}(g^{z}) = 1] - Pr[\mathcal{A}(g^{xy}) = 1]|.
\end{aligned}
$$

Security Proof Details

$$
\begin{aligned}
&Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1] \\
&= \tfrac{1}{2} \cdot Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1 | b = 0] + \tfrac{1}{2} \cdot Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1 | b = 1] \\
&= \tfrac{1}{2} \cdot Pr[\mathcal{A}(g^{xy}) = 0] + \tfrac{1}{2} \cdot Pr[\mathcal{A}(g^{z}) = 1] \\
&= \tfrac{1}{2} \cdot (1 - Pr[\mathcal{A}(g^{xy}) = 1]) + \tfrac{1}{2} \cdot Pr[\mathcal{A}(g^{z}) = 1] \\
&= \tfrac{1}{2} + \tfrac{1}{2}(Pr[\mathcal{A}(g^{z}) = 1] - Pr[\mathcal{A}(g^{xy}) = 1]) \\
&\leq \tfrac{1}{2} + \tfrac{1}{2}|Pr[\mathcal{A}(g^{z}) = 1] - Pr[\mathcal{A}(g^{xy}) = 1]|.
\end{aligned}
$$

If DDH is hard that implies
$|Pr[\mathcal{A}(g^{z}) = 1] - Pr[\mathcal{A}(g^{xy}) = 1]| \leq \mathrm{negl}(n)$.

Security Proof Details

$$Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1]$$
$$= \tfrac{1}{2} \cdot Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1 | b = 0] + \tfrac{1}{2} \cdot Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1 | b = 1]$$
$$= \tfrac{1}{2} \cdot Pr[\mathcal{A}(g^{xy}) = 0] + \tfrac{1}{2} \cdot Pr[\mathcal{A}(g^{z}) = 1]$$
$$= \tfrac{1}{2} \cdot (1 - Pr[\mathcal{A}(g^{xy}) = 1]) + \tfrac{1}{2} \cdot Pr[\mathcal{A}(g^{z}) = 1]$$
$$= \tfrac{1}{2} + \tfrac{1}{2}(Pr[\mathcal{A}(g^{z}) = 1] - Pr[\mathcal{A}(g^{xy}) = 1])$$
$$\leq \tfrac{1}{2} + \tfrac{1}{2}|Pr[\mathcal{A}(g^{z}) = 1] - Pr[\mathcal{A}(g^{xy}) = 1]|.$$

If DDH is hard that implies
$|Pr[\mathcal{A}(g^{z}) = 1] - Pr[\mathcal{A}(g^{xy}) = 1]| \leq \mathrm{negl}(n)$.

$$Pr[\overline{KE}^{eav}_{\mathcal{A},\Pi}(1^n) = 1] \leq \frac{1}{2} + \frac{1}{2}\mathrm{negl}(n).$$

Active Adversaries

- There are two kinds of active attacks:
  - impersonation attacks where the impersonates one party while interacting with the other party,

Active Adversaries

- There are two kinds of active attacks:
  - impersonation attacks where the impersonates one party while interacting with the other party,
  - man-in-the-middle attacks where the adversary just intercepts the messages and modifies them.

Active Adversaries

- There are two kinds of active attacks:
  - impersonation attacks where the impersonates one party while interacting with the other party,
  - man-in-the-middle attacks where the adversary just intercepts the messages and modifies them.
- Security against these attacks need sharing some information between the two parties.

Active Adversaries

- There are two kinds of active attacks:
  - impersonation attacks where the impersonates one party while interacting with the other party,
  - man-in-the-middle attacks where the adversary just intercepts the messages and modifies them.
- Security against these attacks need sharing some information between the two parties.
- Diffie-Hellman is *completely insecure against man-in-the-middle attacks.*

Active Adversaries

- There are two kinds of active attacks:
  - impersonation attacks where the impersonates one party while interacting with the other party,
  - man-in-the-middle attacks where the adversary just intercepts the messages and modifies them.
- Security against these attacks need sharing some information between the two parties.
- Diffie-Hellman is *completely insecure against man-in-the-middle attacks.*
- In practice it is not used in its basic form but is at the core of many standardized key exchange protocols that are resilient to active adversaries.

Active Adversaries

- There are two kinds of active attacks:
  - impersonation attacks where the impersonates one party while interacting with the other party,
  - man-in-the-middle attacks where the adversary just intercepts the messages and modifies them.
- Security against these attacks need sharing some information between the two parties.
- Diffie-Hellman is *completely insecure against man-in-the-middle attacks.*
- In practice it is not used in its basic form but is at the core of many standardized key exchange protocols that are resilient to active adversaries. For e.g: TLS.

Active Adversaries

- There are two kinds of active attacks:
  - impersonation attacks where the impersonates one party while interacting with the other party,
  - man-in-the-middle attacks where the adversary just intercepts the messages and modifies them.
- Security against these attacks need sharing some information between the two parties.
- Diffie-Hellman is *completely insecure against man-in-the-middle attacks.*
- In practice it is not used in its basic form but is at the core of many standardized key exchange protocols that are resilient to active adversaries. For e.g: TLS.
- In any case, it was the first step to asymmetric techniques and therefore is very important.

Public-Key Revolution

- There is a pair of different keys (asymmetric): public key (pk) that is widely disseminated and a private key (sk) that is kept secret.

Public-Key Revolution

- There is a pair of different keys (asymmetric): public key (pk) that is widely disseminated and a private key (sk) that is kept secret.

Public-Key Revolution

- There is a pair of different keys (asymmetric): public key (pk) that is widely disseminated and a private key (sk) that is kept secret.
- Secrecy of messages : public-key encryption. Public key is the encryption key, secret key is the decryption key.

Public-Key Revolution

- There is a pair of different keys (asymmetric): public key (pk) that is widely disseminated and a private key (sk) that is kept secret.
- Secrecy of messages : public-key encryption. Public key is the encryption key, secret key is the decryption key.
- The secrecy of the encrypted messages is preserved even against $\mathcal{A}$ who knows the key used to encrypt!

Public-Key Revolution

- There is a pair of different keys (asymmetric): public key (pk) that is widely disseminated and a private key (sk) that is kept secret.
- Secrecy of messages : public-key encryption. Public key is the encryption key, secret key is the decryption key.
- The secrecy of the encrypted messages is preserved even against $\mathcal{A}$ who knows the key used to encrypt!
- Receiver can simply send her public key over an insecure channel or publicize it and communication can happen privately.

Public-Key Revolution

- There is a pair of different keys (asymmetric): public key (pk) that is widely disseminated and a private key (sk) that is kept secret.
- Secrecy of messages : public-key encryption. Public key is the encryption key, secret key is the decryption key.
- The secrecy of the encrypted messages is preserved even against $\mathcal{A}$ who knows the key used to encrypt!
- Receiver can simply send her public key over an insecure channel or publicize it and communication can happen privately.
- Digital signatures is the PKE analogue of MACs (for data integrity). The private key is the signing/authentication key to generate tags and the public key is the verification key so anyone can verify the signatures.

Public-Key Revolution

- There is a pair of different keys (asymmetric): public key (pk) that is widely disseminated and a private key (sk) that is kept secret.
- Secrecy of messages : public-key encryption. Public key is the encryption key, secret key is the decryption key.
- The secrecy of the encrypted messages is preserved even against $\mathcal{A}$ who knows the key used to encrypt!
- Receiver can simply send her public key over an insecure channel or publicize it and communication can happen privately.
- Digital signatures is the PKE analogue of MACs (for data integrity). The private key is the signing/authentication key to generate tags and the public key is the verification key so anyone can verify the signatures.
- Thus we have key distribution for open environments!

Public-Key Encryption

- Disadvantage: PKE is roughly 2 to 3 orders of magnitude slower than SKE.

Public-Key Encryption

- Disadvantage: PKE is roughly 2 to 3 orders of magnitude slower than SKE.
- Hard to implement on RFID tags, smartcards or a desktop.

Public-Key Encryption

- Disadvantage: PKE is roughly 2 to 3 orders of magnitude slower than SKE.
- Hard to implement on RFID tags, smartcards or a desktop.
- If Mallory tampers with Alice's public key (pk') and the corresponding secret key sk') then nothing that can be done.

Public-Key Encryption

- Disadvantage: PKE is roughly 2 to 3 orders of magnitude slower than SKE.
- Hard to implement on RFID tags, smartcards or a desktop.
- If Mallory tampers with Alice's public key (pk') and the corresponding secret key sk') then nothing that can be done. The tampering can happen in between transmission or in the public directory.

Public-Key Encryption

- Disadvantage: PKE is roughly 2 to 3 orders of magnitude slower than SKE.
- Hard to implement on RFID tags, smartcards or a desktop.
- If Mallory tampers with Alice's public key (pk') and the corresponding secret key sk') then nothing that can be done. The tampering can happen in between transmission or in the public directory.
- This can be prevented by sharing information in advance or a trusted third party.

Public-Key Encryption

- Disadvantage: PKE is roughly 2 to 3 orders of magnitude slower than SKE.
- Hard to implement on RFID tags, smartcards or a desktop.
- If Mallory tampers with Alice's public key (pk') and the corresponding secret key sk') then nothing that can be done. The tampering can happen in between transmission or in the public directory.
- This can be prevented by sharing information in advance or a trusted third party.
- In this lecture we assume that senders will have legitimate copy of receiver's public key, i.e. secure key distribution of public keys is assumed.

Public-Key Encryption

- Disadvantage: PKE is roughly 2 to 3 orders of magnitude slower than SKE.
- Hard to implement on RFID tags, smartcards or a desktop.
- If Mallory tampers with Alice's public key (pk') and the corresponding secret key sk') then nothing that can be done. The tampering can happen in between transmission or in the public directory.
- This can be prevented by sharing information in advance or a trusted third party.
- In this lecture we assume that senders will have legitimate copy of receiver's public key, i.e. secure key distribution of public keys is assumed.
- We are not considering active attacks by letting other mechanisms take care of it.

Public-Key Encryption

- A public-key encryption scheme is a triple of PPT algorithms (Gen, Enc, Dec) s.t.:
  1. Key-generation algorithm $Gen(1^n)$ outputs pair of public and secret keys (pk, sk)

Public-Key Encryption

- A public-key encryption scheme is a triple of PPT algorithms (Gen, Enc, Dec) s.t.:
  1. Key-generation algorithm $Gen(1^n)$ outputs pair of public and secret keys (pk, sk)
  2. The encryption algorithm $c \leftarrow Enc(pk, m)$. *It will need to be probabilistic for semantic security.*

Public-Key Encryption

- A public-key encryption scheme is a triple of PPT algorithms (Gen, Enc, Dec) s.t.:
  1. Key-generation algorithm $Gen(1^n)$ outputs pair of public and secret keys (pk, sk)
  2. The encryption algorithm $c \leftarrow Enc(pk, m)$. *It will need to be probabilistic for semantic security.*
  3. The deterministic decryption algorithm $m = Dec_{sk}(c)$.

Public-Key Encryption

- A public-key encryption scheme is a triple of PPT algorithms (Gen, Enc, Dec) s.t.:
  1. Key-generation algorithm $Gen(1^n)$ outputs pair of public and secret keys (pk, sk)
  2. The encryption algorithm $c \leftarrow Enc(pk, m)$. *It will need to be probabilistic for semantic security.*
  3. The deterministic decryption algorithm $m = Dec_{sk}(c)$.
- Except for a negligible probability we have $Dec_{sk}(Enc_{pk}(m)) = m$.

Public-Key Encryption

- A public-key encryption scheme is a triple of PPT algorithms (Gen, Enc, Dec) s.t.:
    1. Key-generation algorithm $Gen(1^n)$ outputs pair of public and secret keys (pk, sk)
    2. The encryption algorithm $c \leftarrow Enc(pk, m)$. *It will need to be probabilistic for semantic security.*
    3. The deterministic decryption algorithm $m = Dec_{sk}(c)$.
- Except for a negligible probability we have $Dec_{sk}(Enc_{pk}(m)) = m$.
- Here *Gen* outputs two keys, we allow for negligible probability of decryption error

Public-Key Encryption

- A public-key encryption scheme is a triple of PPT algorithms (Gen, Enc, Dec) s.t.:
  1. Key-generation algorithm $Gen(1^n)$ outputs pair of public and secret keys (pk, sk)
  2. The encryption algorithm $c \leftarrow Enc(pk, m)$. *It will need to be probabilistic for semantic security.*
  3. The deterministic decryption algorithm $m = Dec_{sk}(c)$.
- Except for a negligible probability we have $Dec_{sk}(Enc_{pk}(m)) = m$.
- Here *Gen* outputs two keys, we allow for negligible probability of decryption error (for e.g: when a composite is generated instead of a prime.)

Security Definitions

- Just like we had security definitions in SKE we have the same for PKE.

Security Definitions

- Just like we had security definitions in SKE we have the same for PKE. We will focus more on the differences.

Security Definitions

- Just like we had security definitions in SKE we have the same for PKE. We will focus more on the differences.
- Eavesdropping indistinguishability experiment $PubK^{eav}_{\mathcal{A},\Pi}(1^n)$ for $\Pi = (Gen, Enc, Dec)$:

Security Definitions

- Just like we had security definitions in SKE we have the same for PKE. We will focus more on the differences.
- Eavesdropping indistinguishability experiment $PubK^{eav}_{\mathcal{A},\Pi}(1^n)$ for $\Pi = (Gen, Enc, Dec)$:
  1. Gen is run to obtain (pk, sk).

Security Definitions

- Just like we had security definitions in SKE we have the same for PKE. We will focus more on the differences.
- Eavesdropping indistinguishability experiment $PubK^{eav}_{\mathcal{A},\Pi}(1^n)$ for $\Pi = (Gen, Enc, Dec)$:
    1. Gen is run to obtain (pk, sk).
    2. $\mathcal{A}$ is given pk, it outputs a pair m_0, m_1 s.t. $|m_0| = |m_1|$.

Security Definitions

- Just like we had security definitions in SKE we have the same for PKE. We will focus more on the differences.
- Eavesdropping indistinguishability experiment $PubK^{eav}_{\mathcal{A},\Pi}(1^n)$ for $\Pi = (Gen, Enc, Dec)$:
  1. Gen is run to obtain (pk, sk).
  2. $\mathcal{A}$ is given pk, it outputs a pair m_0, m_1 s.t. $|m_0| = |m_1|$.
  3. A uniform $b \in \{0, 1\}$ is chosen. Ciphertext $c \leftarrow Enc_{pk}(m_b))$ is given to $\mathcal{A}$. It is called *challenge ciphertext*.

Security Definitions

- Just like we had security definitions in SKE we have the same for PKE. We will focus more on the differences.
- Eavesdropping indistinguishability experiment $PubK^{eav}_{\mathcal{A},\Pi}(1^n)$ for $\Pi = (Gen, Enc, Dec)$:
  1. Gen is run to obtain (pk, sk).
  2. $\mathcal{A}$ is given pk, it outputs a pair m_0, m_1 s.t. $|m_0| = |m_1|$.
  3. A uniform $b \in \{0, 1\}$ is chosen. Ciphertext $c \leftarrow Enc_{pk}(m_b))$ is given to $\mathcal{A}$. It is called *challenge ciphertext*.
  4. $\mathcal{A}$ outputs a bit $b^{'}$.

Security Definitions

- Just like we had security definitions in SKE we have the same for PKE. We will focus more on the differences.
- Eavesdropping indistinguishability experiment $PubK^{eav}_{\mathcal{A},\Pi}(1^n)$ for $\Pi = (Gen, Enc, Dec)$:
  1. Gen is run to obtain (pk, sk).
  2. $\mathcal{A}$ is given pk, it outputs a pair m_0, m_1 s.t. $|m_0| = |m_1|$.
  3. A uniform $b \in \{0, 1\}$ is chosen. Ciphertext $c \leftarrow Enc_{pk}(m_b))$ is given to $\mathcal{A}$. It is called *challenge ciphertext*.
  4. $\mathcal{A}$ outputs a bit $b^{'}$.
  5. If $b = b^{'}$ output 1, else 0. If $PubK^{eav}_{\mathcal{A},\Pi}(1^n) = 1$, then $\mathcal{A}$ succeeds.

Security Definitions

- Just like we had security definitions in SKE we have the same for PKE. We will focus more on the differences.
- Eavesdropping indistinguishability experiment $PubK^{eav}_{\mathcal{A},\Pi}(1^n)$ for $\Pi = (Gen, Enc, Dec)$:
  1. Gen is run to obtain (pk, sk).
  2. $\mathcal{A}$ is given pk, it outputs a pair m_0, m_1 s.t. $|m_0| = |m_1|$.
  3. A uniform $b \in \{0, 1\}$ is chosen. Ciphertext $c \leftarrow Enc_{pk}(m_b))$ is given to $\mathcal{A}$. It is called *challenge ciphertext*.
  4. $\mathcal{A}$ outputs a bit $b^{'}$.
  5. If $b = b^{'}$ output 1, else 0. If $PubK^{eav}_{\mathcal{A},\Pi}(1^n) = 1$, then $\mathcal{A}$ succeeds.
- We say Π has indistinguishable encryptions in the presence of an eavesdropper if for all PPT $\mathcal{A}$

$$Pr[PubK^{eav}_{\mathcal{A},\Pi}(1^n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n).$$

Differences from SKE

- $\mathcal{A}$ is given *pk*.

Differences from SKE

- $\mathcal{A}$ is given pk. Major change:

Differences from SKE

- $\mathcal{A}$ is given pk. Major change: Effectively gives $\mathcal{A}$ an encryption oracle for free.

Differences from SKE

- $\mathcal{A}$ is given pk. Major change: Effectively gives $\mathcal{A}$ an encryption oracle for free.
- Why? Given pk, $\mathcal{A}$ can encrypt any message on its own $Enc_{pk}(m)$.

Differences from SKE

- $\mathcal{A}$ is given pk. Major change: Effectively gives $\mathcal{A}$ an encryption oracle for free.
- Why? Given pk, $\mathcal{A}$ can encrypt any message on its own $Enc_{pk}(m)$.
- If a PKE has indistinguishable encryptions in the presence of an eavesdropper, it is CPA-secure.

Differences from SKE

- $\mathcal{A}$ is given pk. Major change: Effectively gives $\mathcal{A}$ an encryption oracle for free.
- Why? Given pk, $\mathcal{A}$ can encrypt any message on its own $Enc_{pk}(m)$.
- If a PKE has indistinguishable encryptions in the presence of an eavesdropper, it is CPA-secure. – another difference w.r.t. SKE.

Differences from SKE

- $\mathcal{A}$ is given pk. Major change: Effectively gives $\mathcal{A}$ an encryption oracle for free.
- Why? Given pk, $\mathcal{A}$ can encrypt any message on its own $Enc_{pk}(m)$.
- If a PKE has indistinguishable encryptions in the presence of an eavesdropper, it is CPA-secure. – another difference w.r.t. SKE.
- Unlike SKE, perfectly secret PKE is impossible.

Differences from SKE

- $\mathcal{A}$ is given pk. Major change: Effectively gives $\mathcal{A}$ an encryption oracle for free.
- Why? Given pk, $\mathcal{A}$ can encrypt any message on its own $Enc_{pk}(m)$.
- If a PKE has indistinguishable encryptions in the presence of an eavesdropper, it is CPA-secure. – another difference w.r.t. SKE.
- Unlike SKE, perfectly secret PKE is impossible. I.e, for all adversaries, even non efficient ones, regardless of how long the keys are or how small the message space is, it is impossible to have $Pr[PubK^{eav}_{\mathcal{A},\Pi}(1^n) = 1] = \frac{1}{2}$.

Differences from SKE

- $\mathcal{A}$ is given pk. Major change: Effectively gives $\mathcal{A}$ an encryption oracle for free.
- Why? Given pk, $\mathcal{A}$ can encrypt any message on its own $Enc_{pk}(m)$.
- If a PKE has indistinguishable encryptions in the presence of an eavesdropper, it is CPA-secure. – another difference w.r.t. SKE.
- Unlike SKE, perfectly secret PKE is impossible. I.e, for all adversaries, even non efficient ones, regardless of how long the keys are or how small the message space is, it is impossible to have $Pr[PubK^{eav}_{\mathcal{A},\Pi}(1^n) = 1] = \frac{1}{2}$.
- In fact, given pk, c we can compute m with probability 1 – Practice q!

Deterministic PKE

- No deterministic public-key encryption scheme is CPA- secure.

Deterministic PKE

- No deterministic public-key encryption scheme is CPA- secure.
- Same as SKE. But more important here.

Deterministic PKE

- No deterministic public-key encryption scheme is CPA- secure.
- Same as SKE. But more important here.
- In SKE deterministic implies $\mathcal{A}$ knows when the same message is sent again.

Deterministic PKE

- No deterministic public-key encryption scheme is CPA- secure.
- Same as SKE. But more important here.
- In SKE deterministic implies $\mathcal{A}$ knows when the same message is sent again.
- In PKE, $\mathcal{A}$ can recover the message with probability 1.

Deterministic PKE

- No deterministic public-key encryption scheme is CPA- secure.
- Same as SKE. But more important here.
- In SKE deterministic implies $\mathcal{A}$ knows when the same message is sent again.
- In PKE, $\mathcal{A}$ can recover the message with probability 1.
  - Consider a prof encrypting students grades.

Deterministic PKE

- No deterministic public-key encryption scheme is CPA- secure.
- Same as SKE. But more important here.
- In SKE deterministic implies $\mathcal{A}$ knows when the same message is sent again.
- In PKE, $\mathcal{A}$ can recover the message with probability 1.
  - Consider a prof encrypting students grades.
  - An eavesdropper knows it has to be one of the grades $\{A, B, C, D, F\}$.

Deterministic PKE

- No deterministic public-key encryption scheme is CPA- secure.
- Same as SKE. But more important here.
- In SKE deterministic implies $\mathcal{A}$ knows when the same message is sent again.
- In PKE, $\mathcal{A}$ can recover the message with probability 1.
  - Consider a prof encrypting students grades.
  - An eavesdropper knows it has to be one of the grades $\{A, B, C, D, F\}$.
  - Just encrypt all the grades and then compare with ciphertext!

Multiple Encryptions

- Same key for encrypting multiple messages:

Multiple Encryptions

- Same key for encrypting multiple messages: we use LR-oracle instead of using lists of plaintexts. The experiment and the security definition are analogous to SKE.

Multiple Encryptions

- Same key for encrypting multiple messages: we use LR-oracle instead of using lists of plaintexts. The experiment and the security definition are analogous to SKE.
- Important result which we do not prove: If public-key encryption scheme Π is CPA-secure, then it also has indistinguishable multiple encryptions.

Multiple Encryptions

- Same key for encrypting multiple messages: we use LR-oracle instead of using lists of plaintexts. The experiment and the security definition are analogous to SKE.
- Important result which we do not prove: If public-key encryption scheme Π is CPA-secure, then it also has indistinguishable multiple encryptions.
- I.e., in the public-key setting, security for encryption of a single message implies security for encryption of multiple messages.

Multiple Encryptions

- Same key for encrypting multiple messages: we use LR-oracle instead of using lists of plaintexts. The experiment and the security definition are analogous to SKE.
- Important result which we do not prove: If public-key encryption scheme Π is CPA-secure, then it also has indistinguishable multiple encryptions.
- I.e., in the public-key setting, security for encryption of a single message implies security for encryption of multiple messages. Much easier to work with single message encryption.

Multiple Encryptions

- Same key for encrypting multiple messages: we use LR-oracle instead of using lists of plaintexts. The experiment and the security definition are analogous to SKE.
- Important result which we do not prove: If public-key encryption scheme Π is CPA-secure, then it also has indistinguishable multiple encryptions.
- I.e., in the public-key setting, security for encryption of a single message implies security for encryption of multiple messages. Much easier to work with single message encryption.
- Indistinguishable encryptions in the presence of an eav, CPA-security, and indistinguishable multiple encryptions are all equivalent.

Multiple Encryptions

- Same key for encrypting multiple messages: we use LR-oracle instead of using lists of plaintexts. The experiment and the security definition are analogous to SKE.
- Important result which we do not prove: If public-key encryption scheme Π is CPA-secure, then it also has indistinguishable multiple encryptions.
- I.e., in the public-key setting, security for encryption of a single message implies security for encryption of multiple messages. Much easier to work with single message encryption.
- Indistinguishable encryptions in the presence of an eav, CPA-security, and indistinguishable multiple encryptions are all equivalent.
- CPA-secure PKE for fixed-length messages implies PKE for arbitrary-length messages satisfying the same notion of security.

Chosen-Ciphertext Attacks

- Relevant in PKE since the receiver expects to receive ciphertexts from unknown receivers and authenticated is more.

Chosen-Ciphertext Attacks

- Relevant in PKE since the receiver expects to receive ciphertexts from unknown receivers and authenticated is more.
- In PKE where Alice is sending a message to Bob, there are two classes of CCA:

Chosen-Ciphertext Attacks

- Relevant in PKE since the receiver expects to receive ciphertexts from unknown receivers and authenticated is more.
- In PKE where Alice is sending a message to Bob, there are two classes of CCA:
  - 1. $\mathcal{A}$ might sent a modified ciphertext c' to Bob on behalf of Alice.

Chosen-Ciphertext Attacks

- Relevant in PKE since the receiver expects to receive ciphertexts from unknown receivers and authenticated is more.
- In PKE where Alice is sending a message to Bob, there are two classes of CCA:
  - 1. $\mathcal{A}$ might sent a modified ciphertext c' to Bob on behalf of Alice.
    - $\mathcal{A}$ may not get the entire decryption m' but he may get some information about m' by the behaviour of Bob and with that he may conclude what m could have been.

Chosen-Ciphertext Attacks

- Relevant in PKE since the receiver expects to receive ciphertexts from unknown receivers and authenticated is more.
- In PKE where Alice is sending a message to Bob, there are two classes of CCA:
  - 1. $\mathcal{A}$ might sent a modified ciphertext c' to Bob on behalf of Alice.
    - $\mathcal{A}$ may not get the entire decryption m' but he may get some information about m' by the behaviour of Bob and with that he may conclude what m could have been.
  - 2. $\mathcal{A}$ sends a modified ciphertext c' to Bob on its own name.

Chosen-Ciphertext Attacks

- Relevant in PKE since the receiver expects to receive ciphertexts from unknown receivers and authenticated is more.
- In PKE where Alice is sending a message to Bob, there are two classes of CCA:
  - 1. $\mathcal{A}$ might sent a modified ciphertext c' to Bob on behalf of Alice.
    - $\mathcal{A}$ may not get the entire decryption m' but he may get some information about m' by the behaviour of Bob and with that he may conclude what m could have been.
  - 2. $\mathcal{A}$ sends a modified ciphertext c' to Bob on its own name.
    - In this case $\mathcal{A}$ obtains entire decryption m' of c' and if there is a known relation between m and m' (malleability) then that will be exploited by $\mathcal{A}$.

Chosen-Ciphertext Attacks

- Relevant in PKE since the receiver expects to receive ciphertexts from unknown receivers and authenticated is more.
- In PKE where Alice is sending a message to Bob, there are two classes of CCA:
  - 1. $\mathcal{A}$ might sent a modified ciphertext c' to Bob on behalf of Alice.
    - $\mathcal{A}$ may not get the entire decryption m' but he may get some information about m' by the behaviour of Bob and with that he may conclude what m could have been.
  - 2. $\mathcal{A}$ sends a modified ciphertext c' to Bob on its own name.
    - In this case $\mathcal{A}$ obtains entire decryption m' of c' and if there is a known relation between m and m' (malleability) then that will be exploited by $\mathcal{A}$.
- Second class of attacks happens only in PKE.

Real life scenarios of CCA

- Scenario 1:
  - A user logs into her bank account by sending her password concantenated with time stamp.

Real life scenarios of CCA

- Scenario 1:
  - A user logs into her bank account by sending her password concantenated with time stamp.
  - Let us say bank returns two types of error messages : password incorrect or timestamp incorrect.

Real life scenarios of CCA

- Scenario 1:
  - A user logs into her bank account by sending her password concantenated with time stamp.
  - Let us say bank returns two types of error messages : password incorrect or timestamp incorrect.
  - $\mathcal{A}$ sends c' and observes which error message it gets and conclude some information about password.

Real life scenarios of CCA

- Scenario 1:
  - A user logs into her bank account by sending her password concantenated with time stamp.
  - Let us say bank returns two types of error messages : password incorrect or timestamp incorrect.
  - $\mathcal{A}$ sends c' and observes which error message it gets and conclude some information about password.
- Scenario 2:
  - Alice sends an encrypted email c to Bob.
  - $\mathcal{A}$ sends in its own name an encrypted email c' to Bob.

Real life scenarios of CCA

- Scenario 1:
  - A user logs into her bank account by sending her password concantenated with time stamp.
  - Let us say bank returns two types of error messages : password incorrect or timestamp incorrect.
  - $\mathcal{A}$ sends c' and observes which error message it gets and conclude some information about password.
- Scenario 2:
  - Alice sends an encrypted email c to Bob.
  - $\mathcal{A}$ sends in its own name an encrypted email c' to Bob.
  - Bob replies to $\mathcal{A}$ with m'

Real life scenarios of CCA

- Scenario 1:
  - A user logs into her bank account by sending her password concantenated with time stamp.
  - Let us say bank returns two types of error messages : password incorrect or timestamp incorrect.
  - $\mathcal{A}$ sends c' and observes which error message it gets and conclude some information about password.
- Scenario 2:
  - Alice sends an encrypted email c to Bob.
  - $\mathcal{A}$ sends in its own name an encrypted email c' to Bob.
  - Bob replies to $\mathcal{A}$ with m' –Bob is now the decryption oracle.

Real life scenarios of CCA

- Scenario 1:
  - A user logs into her bank account by sending her password concantenated with time stamp.
  - Let us say bank returns two types of error messages : password incorrect or timestamp incorrect.
  - $\mathcal{A}$ sends c' and observes which error message it gets and conclude some information about password.
- Scenario 2:
  - Alice sends an encrypted email c to Bob.
  - $\mathcal{A}$ sends in its own name an encrypted email c' to Bob.
  - Bob replies to $\mathcal{A}$ with m' –Bob is now the decryption oracle.
- Scenario 3: Malleability, i.e. m' has a known relation to m
  - Consider an auction with encrypted bids from Alice and $\mathcal{A}$ using pk of Bob.
  - Say you can always modify c s.t. $m' = 2m$.

Real life scenarios of CCA

- Scenario 1:
  - A user logs into her bank account by sending her password concantenated with time stamp.
  - Let us say bank returns two types of error messages : password incorrect or timestamp incorrect.
  - $\mathcal{A}$ sends c' and observes which error message it gets and conclude some information about password.
- Scenario 2:
  - Alice sends an encrypted email c to Bob.
  - $\mathcal{A}$ sends in its own name an encrypted email c' to Bob.
  - Bob replies to $\mathcal{A}$ with m' –Bob is now the decryption oracle.
- Scenario 3: Malleability, i.e. m' has a known relation to m
  - Consider an auction with encrypted bids from Alice and $\mathcal{A}$ using pk of Bob.
  - Say you can always modify c s.t. $m' = 2m$.
  - Then $\mathcal{A}$ submits highest bid always and wins.

CCA indistinguishability experiment

$PubK^{cca}_{\mathcal{A},\Pi}(1^n)$

1. (pk, sk) is generated by running $Gen(1^n)$.
2. $\mathcal{A}$ is given pk and $Dec_k(\cdot)$.

CCA indistinguishability experiment

$PubK^{cca}_{\mathcal{A},\Pi}(1^n)$

1. (pk, sk) is generated by running $Gen(1^n)$.
2. $\mathcal{A}$ is given pk and $Dec_k(\cdot)$.
3. It outputs a pair of messages m_0, m_1 of the same length.

CCA indistinguishability experiment

$PubK^{cca}_{\mathcal{A},\Pi}(1^n)$

1. (pk, sk) is generated by running $Gen(1^n)$.
2. $\mathcal{A}$ is given pk and $Dec_k(\cdot)$.
3. It outputs a pair of messages m_0, m_1 of the same length.
4. A uniform bit $b \in \{0, 1\}$ is chosen and then a challenge ciphertext, $c \leftarrow Enc_{pk}(m_b)$ is computed and given to $\mathcal{A}$.

CCA indistinguishability experiment

$PubK^{cca}_{\mathcal{A},\Pi}(1^n)$

1. (pk, sk) is generated by running $Gen(1^n)$.
2. $\mathcal{A}$ is given pk and $Dec_k(\cdot)$.
3. It outputs a pair of messages m_0, m_1 of the same length.
4. A uniform bit $b \in \{0, 1\}$ is chosen and then a challenge ciphertext, $c \leftarrow Enc_{pk}(m_b)$ is computed and given to $\mathcal{A}$.
5. $\mathcal{A}$ continues to have oracle access to $Dec_k(\cdot)$.

CCA indistinguishability experiment

$PubK^{cca}_{\mathcal{A},\Pi}(1^n)$

1. (pk, sk) is generated by running $Gen(1^n)$.
2. $\mathcal{A}$ is given pk and $Dec_k(\cdot)$.
3. It outputs a pair of messages m_0, m_1 of the same length.
4. A uniform bit $b \in \{0, 1\}$ is chosen and then a challenge ciphertext, $c \leftarrow Enc_{pk}(m_b)$ is computed and given to $\mathcal{A}$.
5. $\mathcal{A}$ continues to have oracle access to $Dec_k(\cdot)$. But it is not allowed to query on the challenge ciphertext.

CCA indistinguishability experiment

$PubK^{cca}_{\mathcal{A},\Pi}(1^n)$

1. (pk, sk) is generated by running $Gen(1^n)$.
2. $\mathcal{A}$ is given pk and $Dec_k(\cdot)$.
3. It outputs a pair of messages m_0, m_1 of the same length.
4. A uniform bit $b \in \{0,1\}$ is chosen and then a challenge ciphertext, $c \leftarrow Enc_{pk}(m_b)$ is computed and given to $\mathcal{A}$.
5. $\mathcal{A}$ continues to have oracle access to $Dec_k(\cdot)$. But it is not allowed to query on the challenge ciphertext.
6. Eventually, $\mathcal{A}$ outputs a bit b'.

CCA indistinguishability experiment

$PubK^{cca}_{\mathcal{A},\Pi}(1^n)$

1. (pk, sk) is generated by running $Gen(1^n)$.
2. $\mathcal{A}$ is given pk and $Dec_k(\cdot)$.
3. It outputs a pair of messages m_0, m_1 of the same length.
4. A uniform bit $b \in \{0, 1\}$ is chosen and then a challenge ciphertext, $c \leftarrow Enc_{pk}(m_b)$ is computed and given to $\mathcal{A}$.
5. $\mathcal{A}$ continues to have oracle access to $Dec_k(\cdot)$. But it is not allowed to query on the challenge ciphertext.
6. Eventually, $\mathcal{A}$ outputs a bit b'.
7. Output is 1 if $b' = b$ and 0 otherwise. If output is 1 we say that $\mathcal{A}$ succeeds.

CCA indistinguishability experiment

$PubK^{cca}_{\mathcal{A},\Pi}(1^n)$

1. (pk, sk) is generated by running $Gen(1^n)$.
2. $\mathcal{A}$ is given pk and $Dec_k(\cdot)$.
3. It outputs a pair of messages m_0, m_1 of the same length.
4. A uniform bit $b \in \{0, 1\}$ is chosen and then a challenge ciphertext, $c \leftarrow Enc_{pk}(m_b)$ is computed and given to $\mathcal{A}$.
5. $\mathcal{A}$ continues to have oracle access to $Dec_k(\cdot)$. But it is not allowed to query on the challenge ciphertext.
6. Eventually, $\mathcal{A}$ outputs a bit b'.
7. Output is 1 if $b' = b$ and 0 otherwise. If output is 1 we say that $\mathcal{A}$ succeeds.

Difference is $\mathcal{A}$ is given pk which means no need separate access to Enc oracle.

CCA-secure

Now that we have an indistinguishability experiment, we can have the security definition.

Definition

A PKE Π has indistinguishable encryptions under a chosen-ciphertext attack or is CCA-secure if for PPT adversaries $\mathcal{A}$ there is a negligible function negl such that:

$$Pr[PubK^{cca}_{\mathcal{A},\Pi}(1^n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n).$$

CCA-secure

Now that we have an indistinguishability experiment, we can have the security definition.

Definition

A PKE Π has indistinguishable encryptions under a chosen-ciphertext attack or is CCA-secure if for PPT adversaries $\mathcal{A}$ there is a negligible function negl such that:

$$Pr[PubK^{cca}_{\mathcal{A},\Pi}(1^n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n).$$

Also, if a scheme has indistinguishable encryptions under a chosen-ciphertext attack then it has indistinguishable multiple encryptions under a chosen-ciphertext attack.

CCA-secure

Now that we have an indistinguishability experiment, we can have the security definition.

Definition

A PKE Π has indistinguishable encryptions under a chosen-ciphertext attack or is CCA-secure if for PPT adversaries $\mathcal{A}$ there is a negligible function negl such that:

$$Pr[PubK^{cca}_{\mathcal{A},\Pi}(1^n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n).$$

Also, if a scheme has indistinguishable encryptions under a chosen-ciphertext attack then it has indistinguishable multiple encryptions under a chosen-ciphertext attack.

CCA-secure for fixed-length messages do NOT hold for arbitrary-length messages.