Public-Key Cryptography

Public-Key Cryptography

Lecture 8

Public-Key Encryption

Public-Key Cryptography

Lecture 8

Public-Key Encryption

Diffie-Hellman Key-Exchange

PKE scheme

PKE scheme

- SKE:
  - Syntax
    - KeyGen outputs $K \leftarrow \mathcal{K}$
    - Enc: $\mathcal{M} \times \mathcal{K} \times \mathcal{R} \rightarrow \mathcal{C}$
    - Dec: $\mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$
  - Correctness
    - $\forall K \in$ Range(KeyGen), Dec(Enc(m,K), K) = m
  - Security (SIM/IND-CPA)

Shared/Symmetric-Key Encryption (a.k.a. private-key encryption)

PKE scheme

- SKE:
  - Syntax
    - KeyGen outputs $K \leftarrow \mathcal{K}$
    - Enc: $\mathcal{M} \times \mathcal{K} \times \mathcal{R} \rightarrow \mathcal{C}$
    - Dec: $\mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$
  - Correctness
    - $\forall K \in$ Range(KeyGen), Dec(Enc(m,K), K) = m
  - Security (SIM/IND-CPA)

PKE scheme

Shared/Symmetric-Key Encryption (a.k.a. private-key encryption)

- SKE:
  - Syntax
    - KeyGen outputs $K \leftarrow \mathcal{K}$
    - Enc: $\mathcal{M} \times \mathcal{K} \times \mathcal{R} \rightarrow \mathcal{C}$
    - Dec: $\mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$
  - Correctness
    - $\forall K \in$ Range(KeyGen), Dec(Enc(m,K), K) = m
  - Security (SIM/IND-CPA)
- PKE

PKE scheme

Shared/Symmetric-Key Encryption (a.k.a. private-key encryption)

- SKE:
  - Syntax
    - KeyGen outputs $K \leftarrow \mathcal{K}$
    - Enc: $\mathcal{M} \times \mathcal{K} \times \mathcal{R} \rightarrow \mathcal{C}$
    - Dec: $\mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$
  - Correctness
    - $\forall K \in$ Range(KeyGen), Dec(Enc(m,K), K) = m
  - Security (SIM/IND-CPA)

- PKE

a.k.a. asymmetric-key encryption

PKE scheme

Shared/Symmetric-Key Encryption (a.k.a. private-key encryption)

- SKE:
  - Syntax
    - KeyGen outputs $K \leftarrow \mathcal{K}$
    - Enc: $\mathcal{M} \times \mathcal{K} \times \mathcal{R} \rightarrow \mathcal{C}$
    - Dec: $\mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$
  - Correctness
    - $\forall K \in$ Range(KeyGen), Dec(Enc(m,K), K) = m
  - Security (SIM/IND-CPA)

a.k.a. asymmetric-key encryption

- PKE
  - Syntax

PKE scheme

Shared/Symmetric-Key Encryption (a.k.a. private-key encryption)

- SKE:
  - Syntax
    - KeyGen outputs K ← $\mathcal{K}$
    - Enc: $\mathcal{M} \times \mathcal{K} \times \mathcal{R} \to \mathcal{C}$
    - Dec: $\mathcal{C} \times \mathcal{K} \to \mathcal{M}$
  - Correctness
    - ∀K ∈ Range(KeyGen), Dec(Enc(m,K), K) = m
  - Security (SIM/IND-CPA)

a.k.a. asymmetric-key encryption

- PKE
  - Syntax
    - KeyGen outputs (PK,SK) ← $\mathcal{PK} \times \mathcal{SK}$

PKE scheme

Shared/Symmetric-Key Encryption (a.k.a. private-key encryption)

- SKE:
  - Syntax
    - KeyGen outputs K ← $\mathcal{K}$
    - Enc: $\mathcal{M} \times \mathcal{K} \times \mathcal{R} \to \mathcal{C}$
    - Dec: $\mathcal{C} \times \mathcal{K} \to \mathcal{M}$
  - Correctness
    - $\forall$K $\in$ Range(KeyGen), Dec(Enc(m,K), K) = m
  - Security (SIM/IND-CPA)

- PKE (a.k.a. asymmetric-key encryption)
  - Syntax
    - KeyGen outputs (PK,SK) ← $\mathcal{PK} \times \mathcal{SK}$
    - Enc: $\mathcal{M} \times \mathcal{PK} \times \mathcal{R} \to \mathcal{C}$

PKE scheme

Shared/Symmetric-Key Encryption (a.k.a. private-key encryption)

- SKE:
  - Syntax
    - KeyGen outputs $K \leftarrow \mathcal{K}$
    - Enc: $\mathcal{M} \times \mathcal{K} \times \mathcal{R} \rightarrow \mathcal{C}$
    - Dec: $\mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$
  - Correctness
    - $\forall K \in$ Range(KeyGen), Dec(Enc(m,K), K) = m
  - Security (SIM/IND-CPA)

- PKE (a.k.a. asymmetric-key encryption)
  - Syntax
    - KeyGen outputs $(PK,SK) \leftarrow \mathcal{PK} \times \mathcal{SK}$
    - Enc: $\mathcal{M} \times \mathcal{PK} \times \mathcal{R} \rightarrow \mathcal{C}$
    - Dec: $\mathcal{C} \times \mathcal{SK} \rightarrow \mathcal{M}$

PKE scheme

Shared/Symmetric-Key Encryption (a.k.a. private-key encryption)

- SKE:
  - Syntax
    - KeyGen outputs $K \leftarrow \mathcal{K}$
    - Enc: $\mathcal{M} \times \mathcal{K} \times \mathcal{R} \rightarrow \mathcal{C}$
    - Dec: $\mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$
  - Correctness
    - $\forall K \in$ Range(KeyGen), Dec(Enc(m,K), K) = m
  - Security (SIM/IND-CPA)

a.k.a. asymmetric-key encryption

- PKE
  - Syntax
    - KeyGen outputs $(PK,SK) \leftarrow \mathcal{PK} \times \mathcal{SK}$
    - Enc: $\mathcal{M} \times \mathcal{PK} \times \mathcal{R} \rightarrow \mathcal{C}$
    - Dec: $\mathcal{C} \times \mathcal{SK} \rightarrow \mathcal{M}$
  - Correctness

PKE scheme

Shared/Symmetric-Key Encryption (a.k.a. private-key encryption)

- SKE:
  - Syntax
    - KeyGen outputs $K \leftarrow \mathcal{K}$
    - Enc: $\mathcal{M} \times \mathcal{K} \times \mathcal{R} \rightarrow \mathcal{C}$
    - Dec: $\mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$
  - Correctness
    - $\forall K \in$ Range(KeyGen), Dec(Enc(m,K), K) = m
  - Security (SIM/IND-CPA)

a.k.a. asymmetric-key encryption

- PKE
  - Syntax
    - KeyGen outputs $(PK,SK) \leftarrow \mathcal{PK} \times \mathcal{SK}$
    - Enc: $\mathcal{M} \times \mathcal{PK} \times \mathcal{R} \rightarrow \mathcal{C}$
    - Dec: $\mathcal{C} \times \mathcal{SK} \rightarrow \mathcal{M}$
  - Correctness
    - $\forall (PK,SK) \in$ Range(KeyGen), Dec(Enc(m,PK), SK) = m

PKE scheme

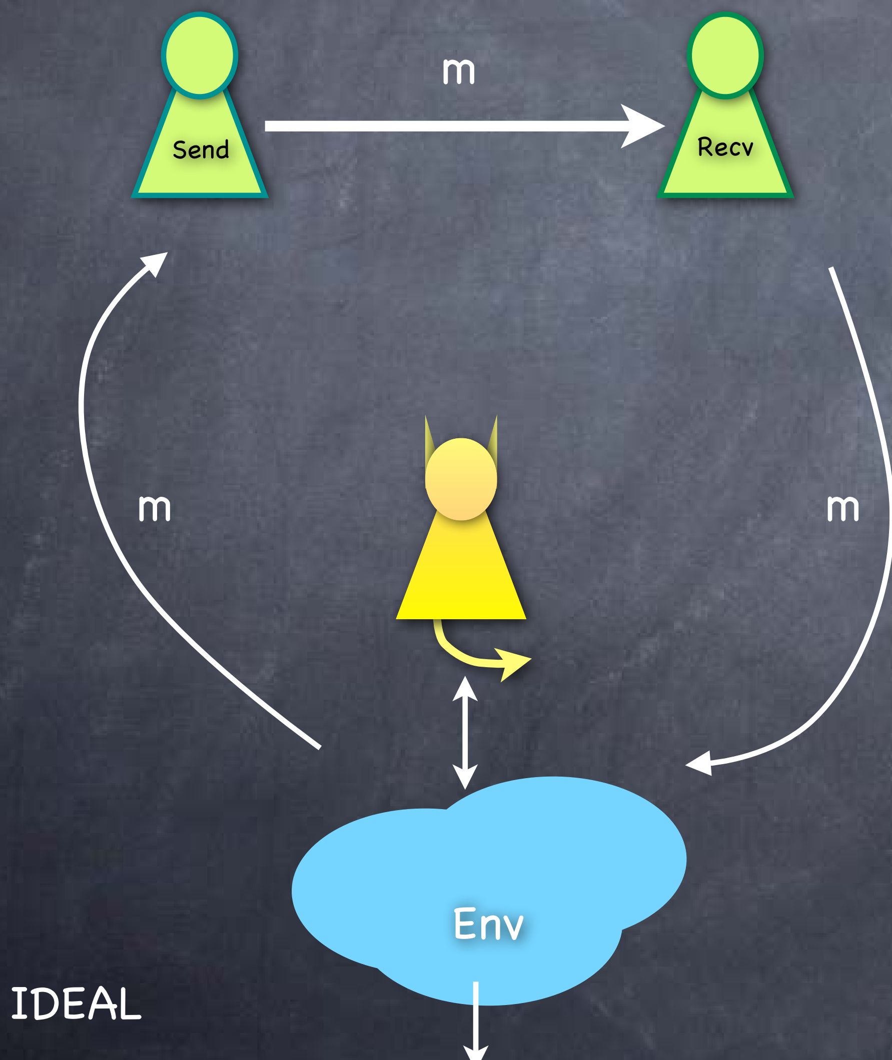
Shared/Symmetric-Key Encryption (a.k.a. private-key encryption)

- SKE:
  - Syntax
    - KeyGen outputs K $\leftarrow \mathcal{K}$
    - Enc: $\mathcal{M} \times \mathcal{K} \times \mathcal{R} \rightarrow \mathcal{C}$
    - Dec: $\mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$
  - Correctness
    - $\forall$K $\in$ Range(KeyGen), Dec(Enc(m,K), K) = m
  - Security (SIM/IND-CPA)

a.k.a. asymmetric-key encryption

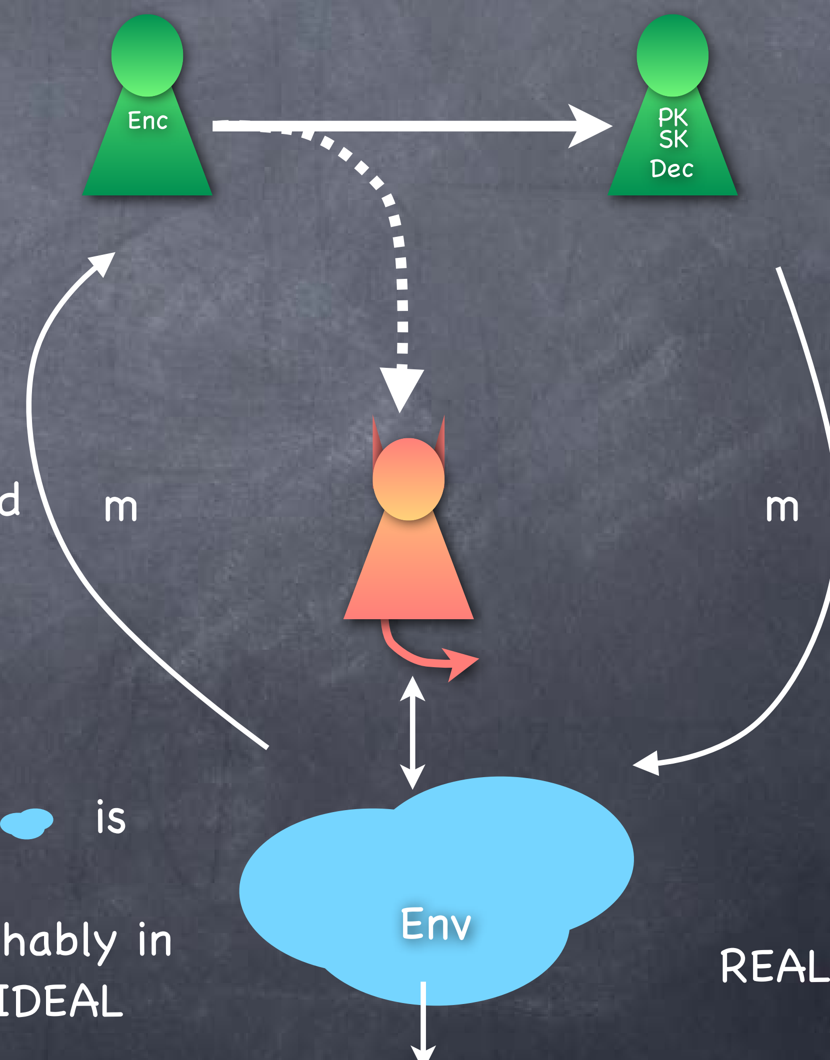
- PKE
  - Syntax
    - KeyGen outputs (PK,SK) $\leftarrow \mathcal{PK} \times \mathcal{SK}$
    - Enc: $\mathcal{M} \times \mathcal{PK} \times \mathcal{R} \rightarrow \mathcal{C}$
    - Dec: $\mathcal{C} \times \mathcal{SK} \rightarrow \mathcal{M}$
  - Correctness
    - $\forall$(PK,SK) $\in$ Range(KeyGen), Dec(Enc(m,PK), SK) = m
  - Security (SIM/IND-CPA, PKE version)

SIM-CPA (PKE Version)

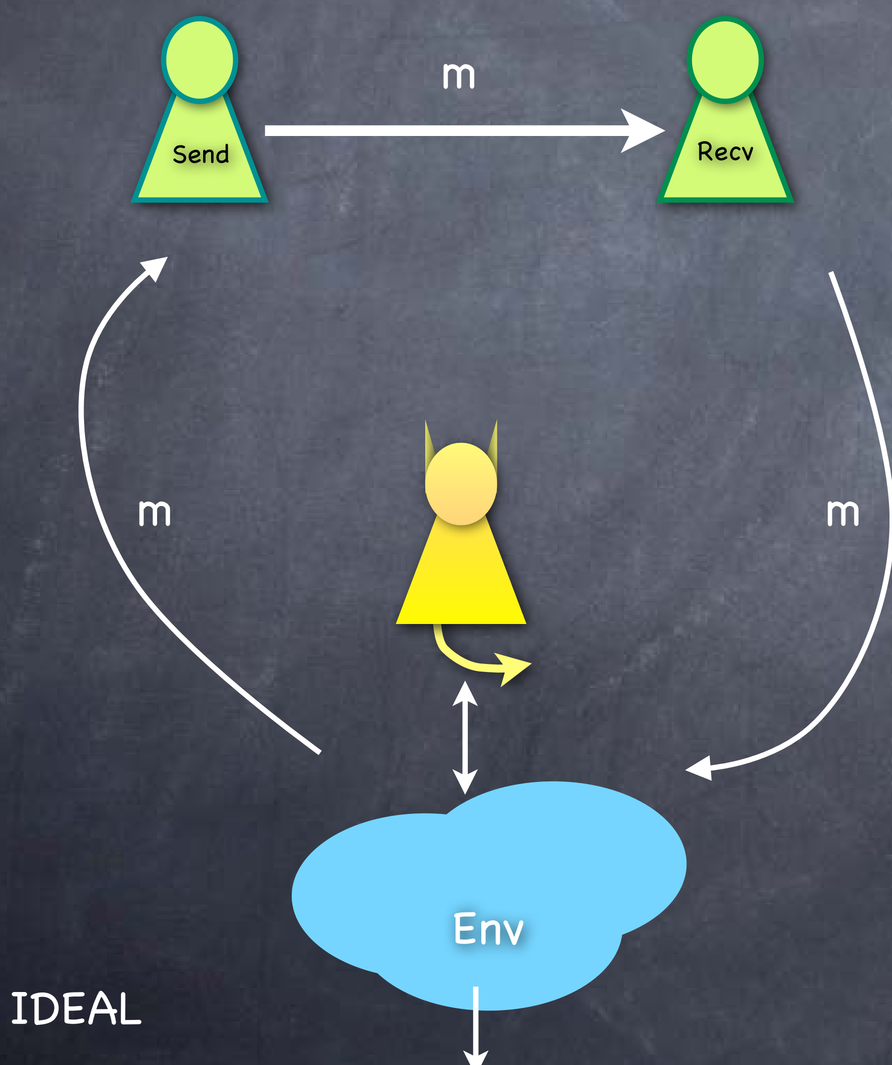


Secure (and correct) if:

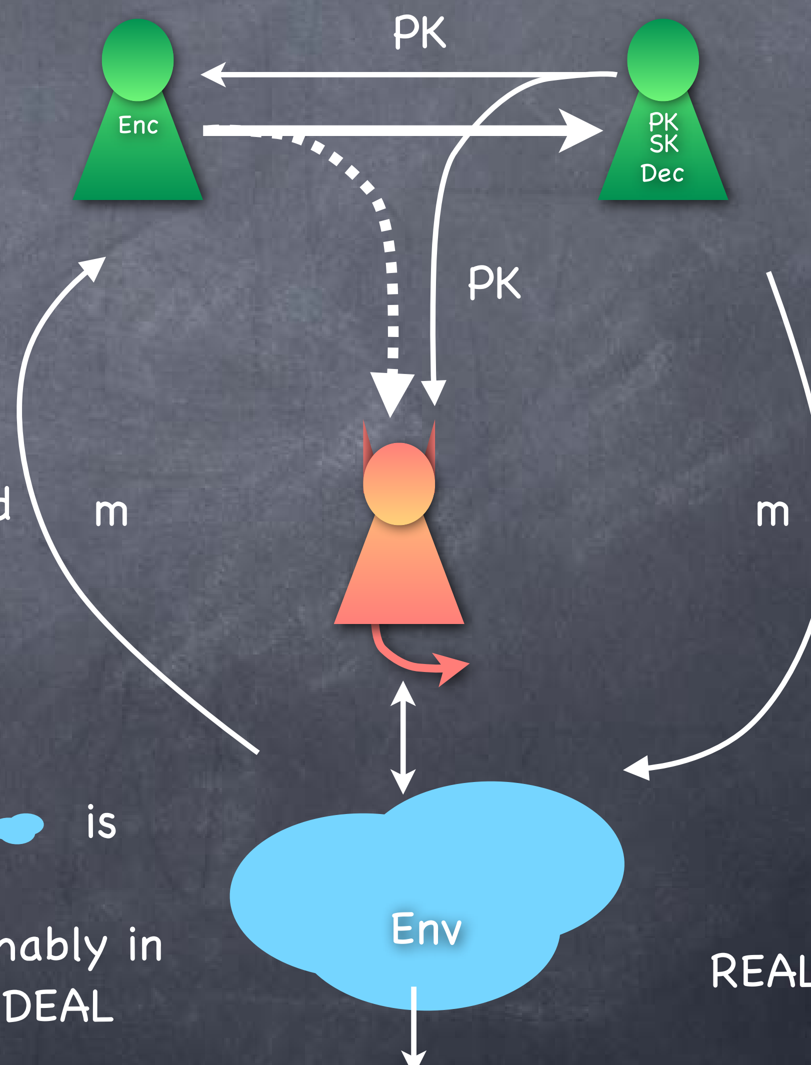
∀ [adversary] ∃ [simulator] s.t. ∀ [environment] output of [environment] is distributed indistinguishably in REAL and IDEAL



SIM-CPA (PKE Version)

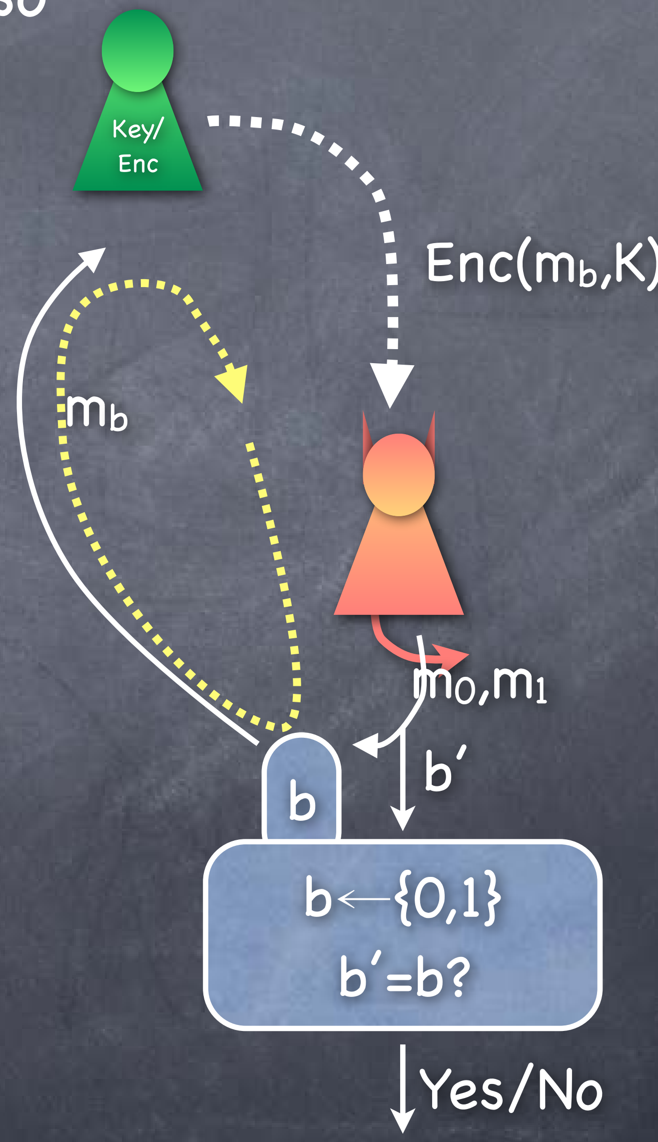


Secure (and correct) if:
∀
∃ s.t.
∀
output of is distributed indistinguishably in REAL and IDEAL



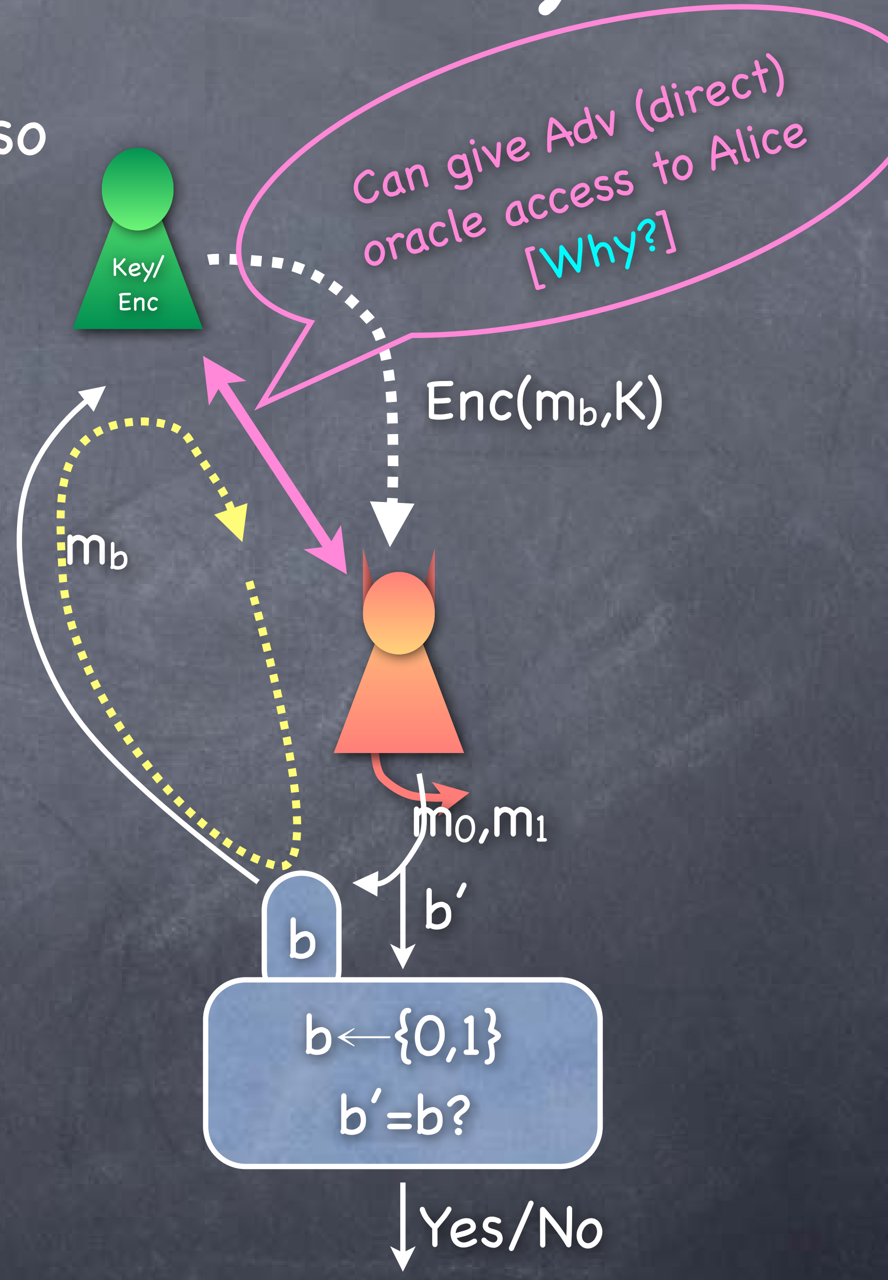
IND-CPA (SKE version)

- Experiment picks a random bit b. It also runs KeyGen to get a key K
  - For as long as Adversary wants
    - Adv sends two messages m_0, m_1 to the experiment
    - Expt returns $Enc(m_b,K)$ to the adversary
  - Adversary returns a guess b′
  - Experiment outputs 1 iff b′=b
- IND-CPA secure if for all PPT adversaries $\Pr[b'=b] - 1/2 \leq \nu(k)$



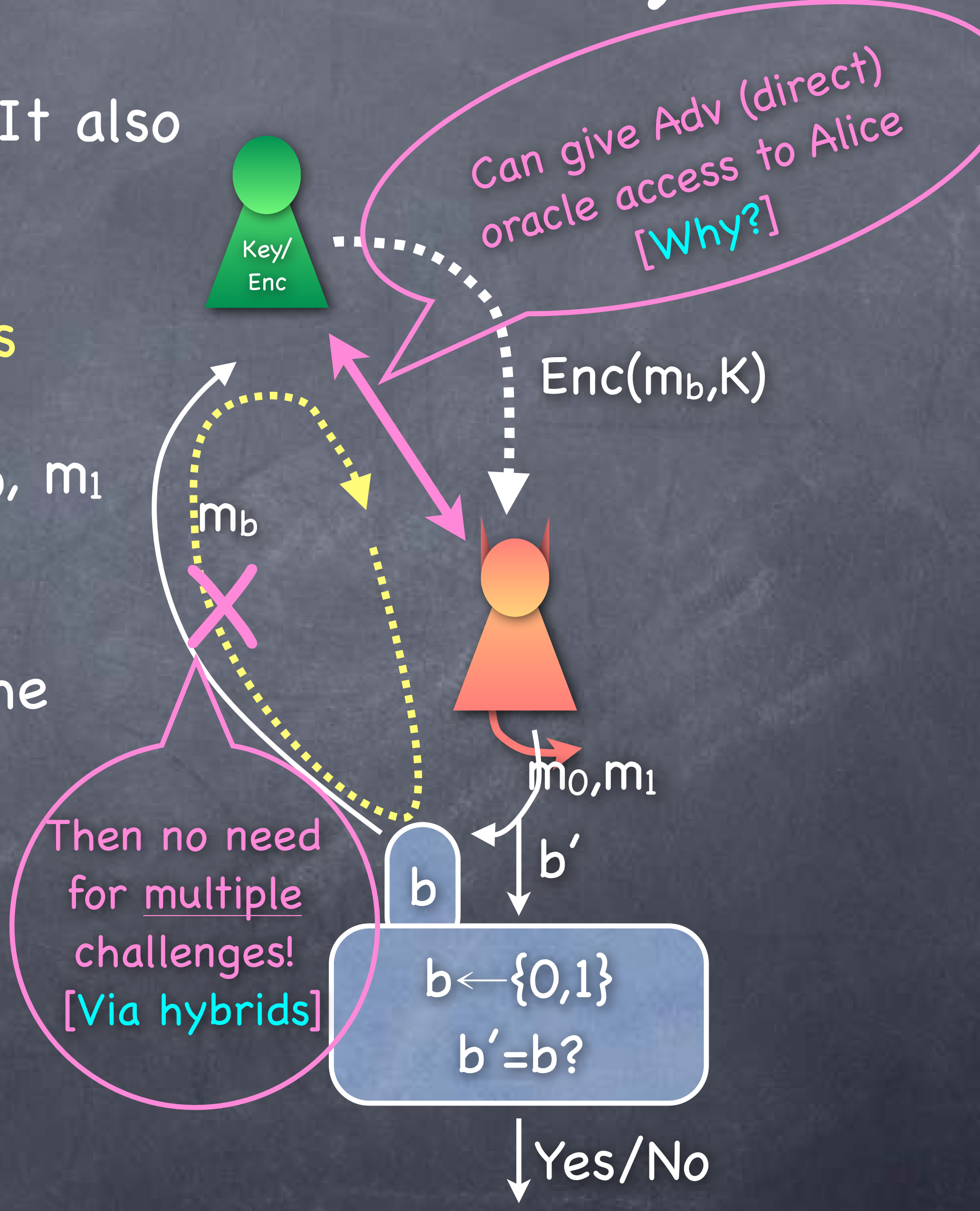
IND-CPA (SKE version)

- Experiment picks a random bit b. It also runs KeyGen to get a key K
  - For as long as Adversary wants
    - Adv sends two messages m_0, m_1 to the experiment
    - Expt returns Enc(m_b,K) to the adversary
  - Adversary returns a guess b′
  - Experiment outputs 1 iff b′=b
- IND-CPA secure if for all PPT adversaries Pr[b′=b] – 1/2 ≤ ν(k)



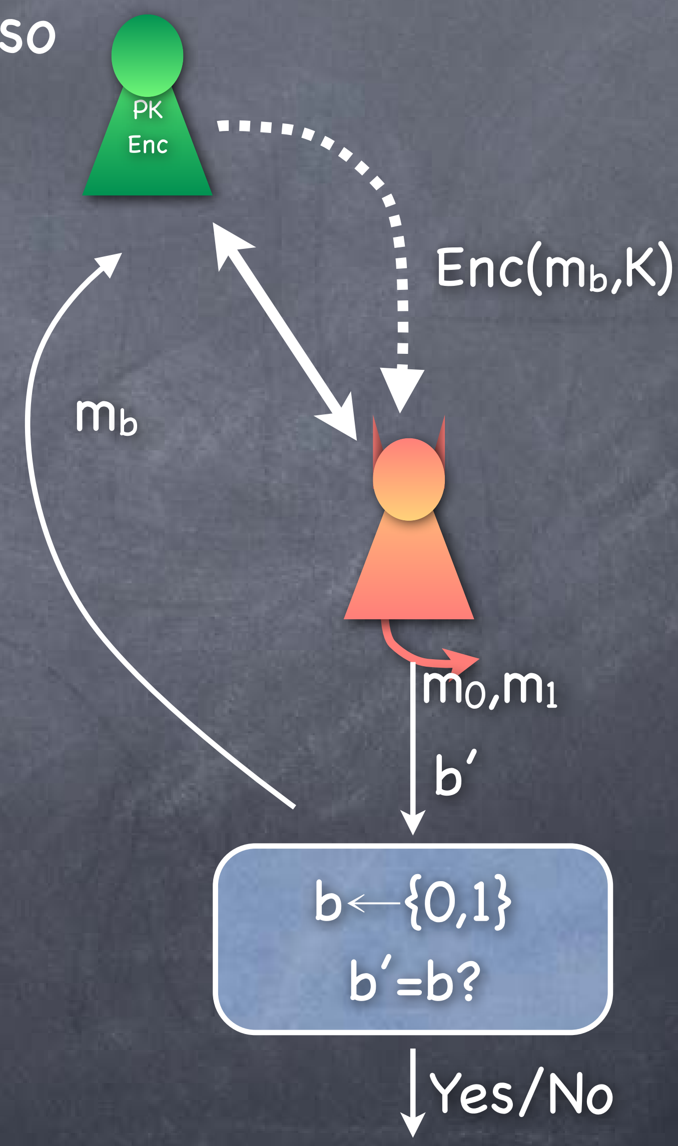
IND-CPA (SKE version)

- Experiment picks a random bit b. It also runs KeyGen to get a key K
  - For as long as Adversary wants
    - Adv sends two messages m_0, m_1 to the experiment
    - Expt returns Enc(m_b,K) to the adversary
  - Adversary returns a guess b′
  - Experiment outputs 1 iff b′=b
- IND-CPA secure if for all PPT adversaries Pr[b′=b] – 1/2 ≤ ν(k)



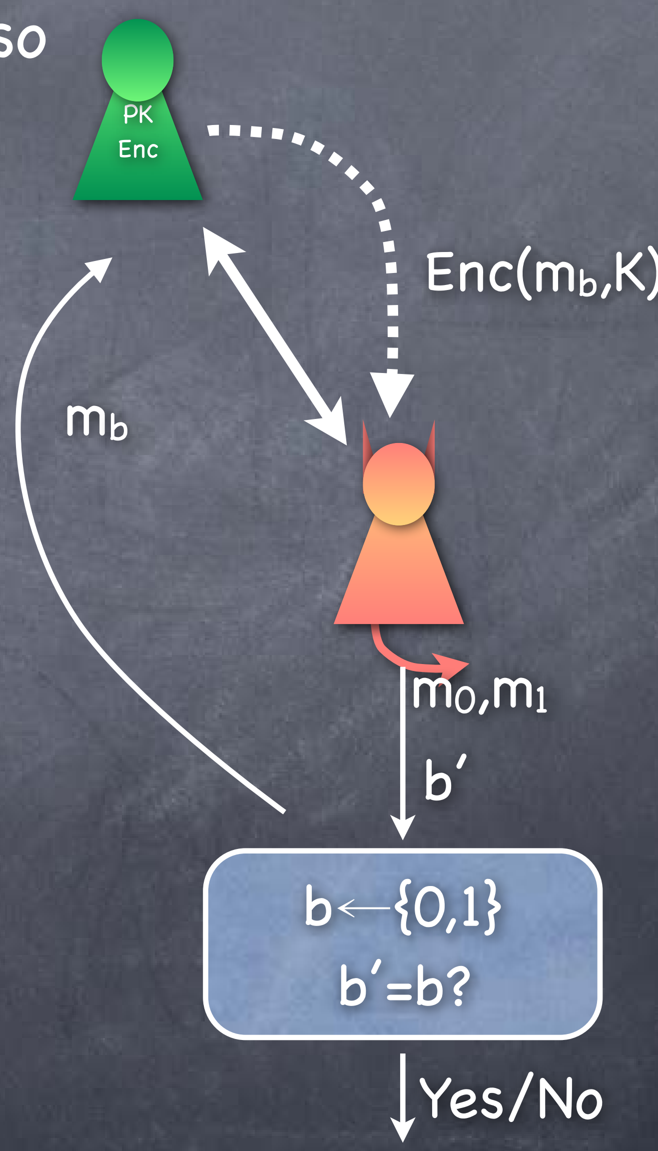
IND-CPA (SKE version)

- Experiment picks a random bit b. It also runs KeyGen to get a key (PK,SK). Adv given PK
  - Adv sends two messages m_0, m_1 to the experiment
  - Expt returns Enc(m_b,K) to the adversary
  - Adversary returns a guess b′
  - Experiment outputs 1 iff b′=b
- IND-CPA secure if for all PPT adversaries $\Pr[b'=b] - 1/2 \leq \nu(k)$



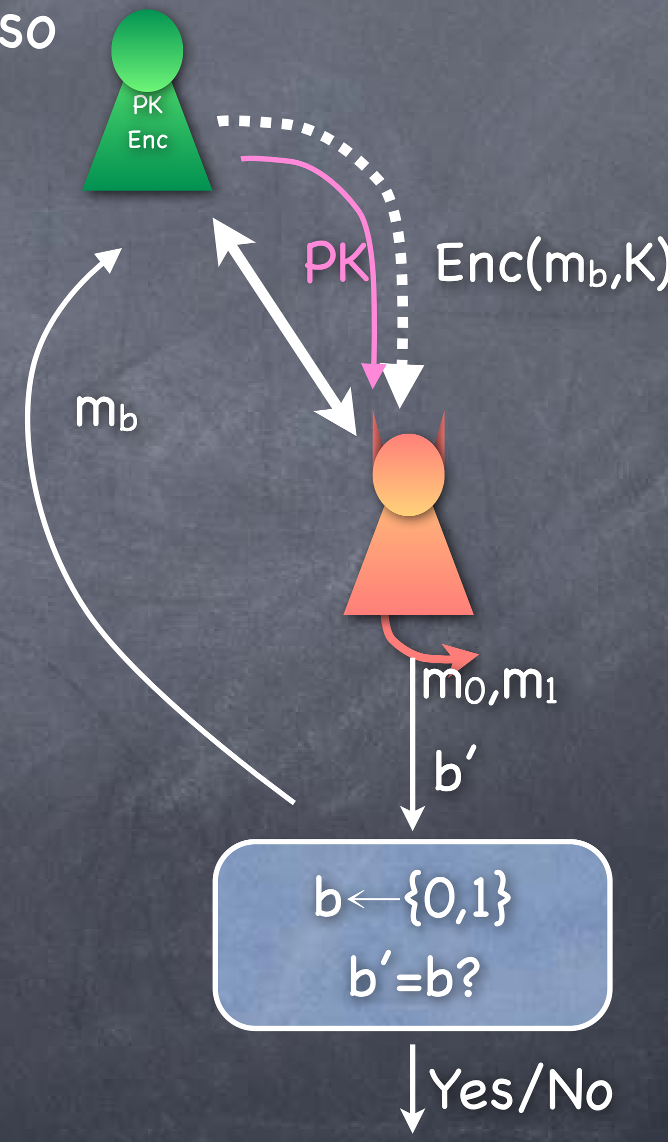
IND-CPA (~~SKE~~ PKE version)

- Experiment picks a random bit b. It also runs KeyGen to get a key (PK,SK). Adv given PK
  - Adv sends two messages m_0, m_1 to the experiment
  - Expt returns Enc(m_b,K) to the adversary
  - Adversary returns a guess b′
  - Experiment outputs 1 iff b′=b
- IND-CPA secure if for all PPT adversaries Pr[b′=b] – 1/2 ≤ ν(k)



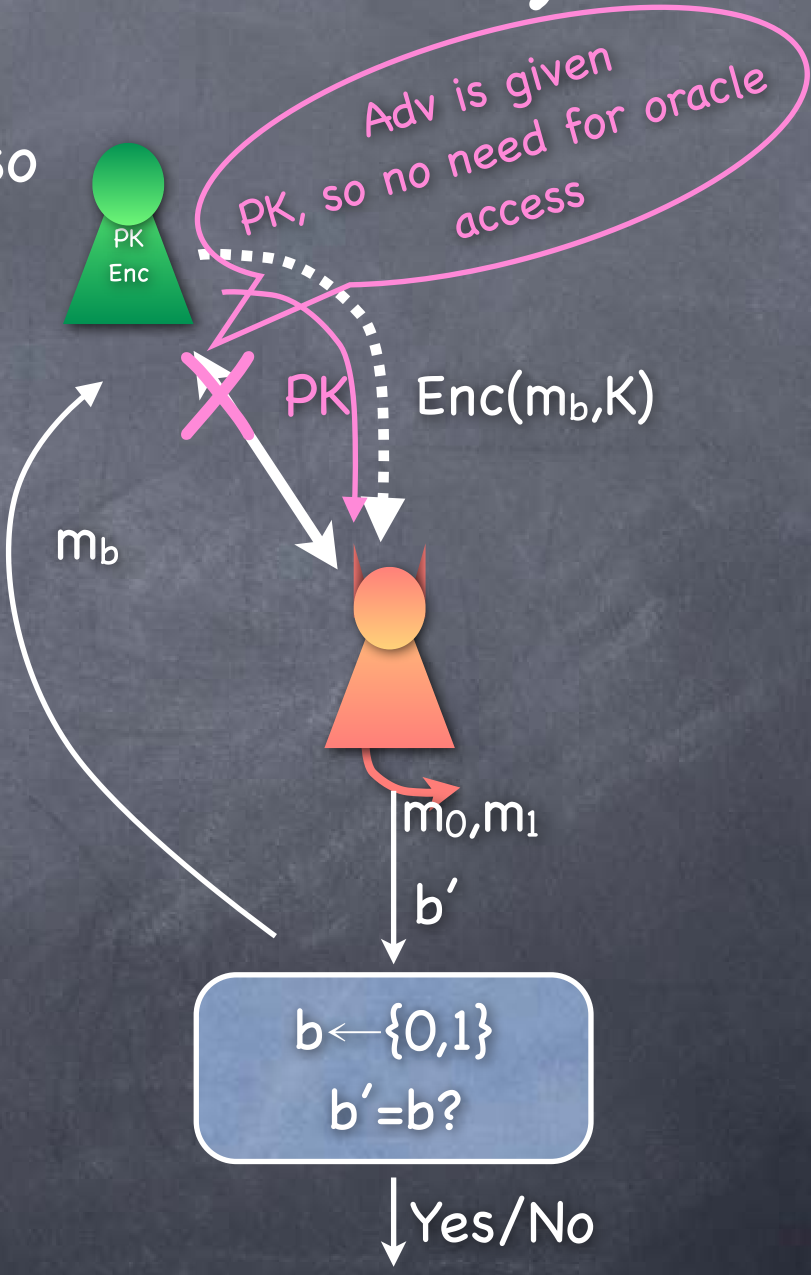
IND-CPA (~~SKE~~ PKE version)

- Experiment picks a random bit b. It also runs KeyGen to get a key (PK,SK). Adv given PK
  - Adv sends two messages m_0, m_1 to the experiment
  - Expt returns Enc(m_b,K) to the adversary
  - Adversary returns a guess b′
  - Experiment outputs 1 iff b′=b
- IND-CPA secure if for all PPT adversaries Pr[b′=b] – 1/2 ≤ ν(k)



IND-CPA (~~SKE~~ PKE version)

- Experiment picks a random bit b. It also runs KeyGen to get a key (PK,SK). Adv given PK
  - Adv sends two messages m_0, m_1 to the experiment
  - Expt returns Enc(m_b,K) to the adversary
  - Adversary returns a guess b′
  - Experiment outputs 1 iff b′=b
- IND-CPA secure if for all PPT adversaries $\Pr[b'=b] - 1/2 \leq \nu(k)$

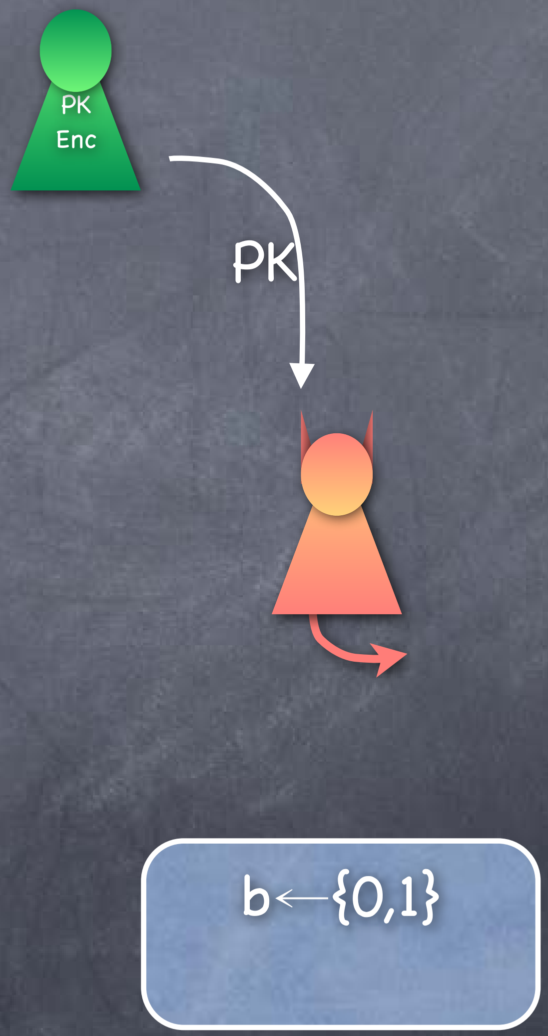


IND-CPA (PKE version)

- Experiment picks a random bit b. It also runs KeyGen to get a key (PK,SK). Adv given PK
  - Adv sends two messages m_0, m_1 to the experiment
  - Expt returns Enc(m_b,K) to the adversary
  - Adversary returns a guess b'
  - Experiment outputs 1 iff b'=b
- IND-CPA secure if for all PPT adversaries $\Pr[b'=b] - 1/2 \leq \nu(k)$

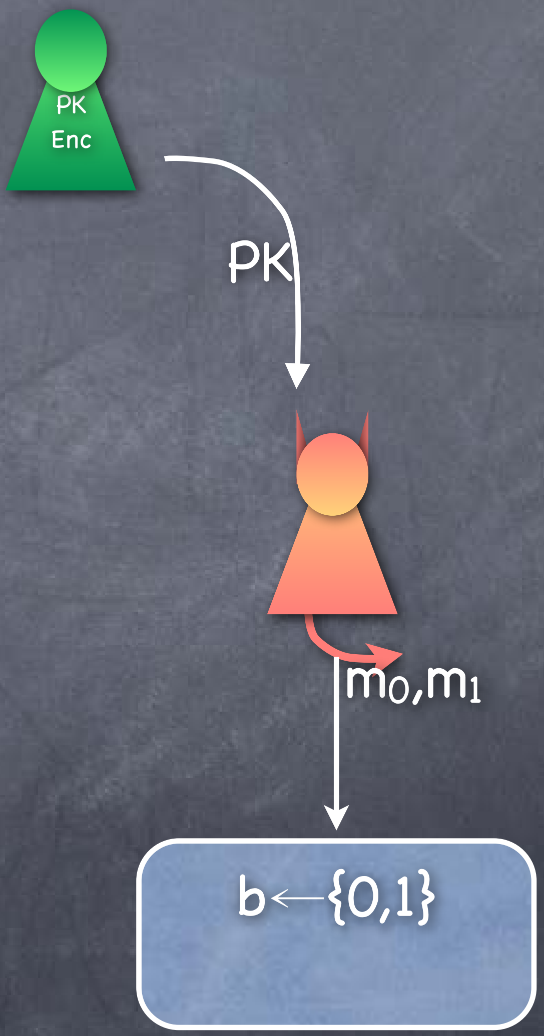
IND-CPA (PKE version)

- Experiment picks a random bit b. It also runs KeyGen to get a key (PK,SK). Adv given PK
  - Adv sends two messages m_0, m_1 to the experiment
  - Expt returns $Enc(m_b,K)$ to the adversary
  - Adversary returns a guess b'
  - Experiment outputs 1 iff b'=b
- IND-CPA secure if for all PPT adversaries $\Pr[b'=b] - 1/2 \leq \nu(k)$



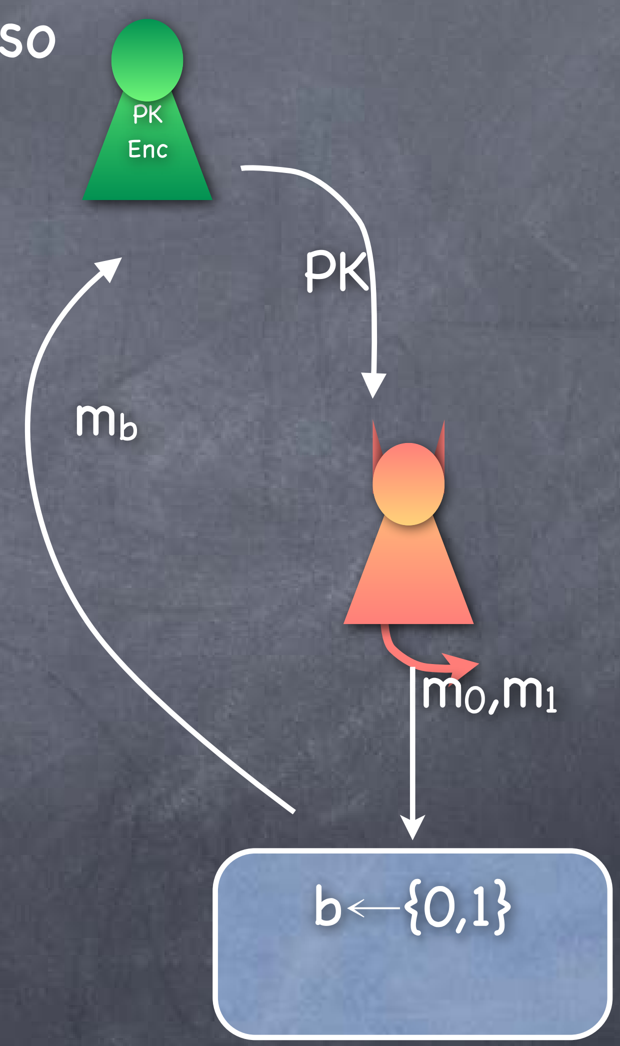
IND-CPA (PKE version)

- Experiment picks a random bit b. It also runs KeyGen to get a key (PK,SK). Adv given PK
  - Adv sends two messages m_0, m_1 to the experiment
  - Expt returns Enc(m_b,K) to the adversary
  - Adversary returns a guess b′
  - Experiment outputs 1 iff b′=b
- IND-CPA secure if for all PPT adversaries $\Pr[b'=b] - 1/2 \leq \nu(k)$



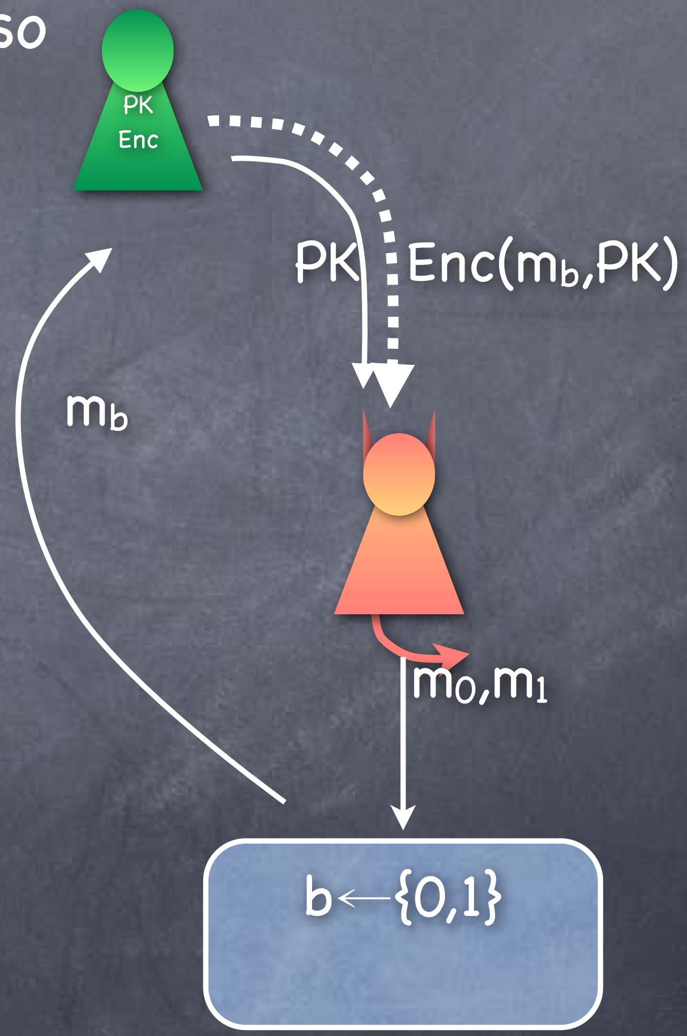
IND-CPA (PKE version)

- Experiment picks a random bit b. It also runs KeyGen to get a key (PK,SK). Adv given PK
    - Adv sends two messages m_0, m_1 to the experiment
    - Expt returns Enc(m_b,K) to the adversary
    - Adversary returns a guess b′
    - Experiment outputs 1 iff b′=b
- IND-CPA secure if for all PPT adversaries Pr[b′=b] - 1/2 ≤ ν(k)



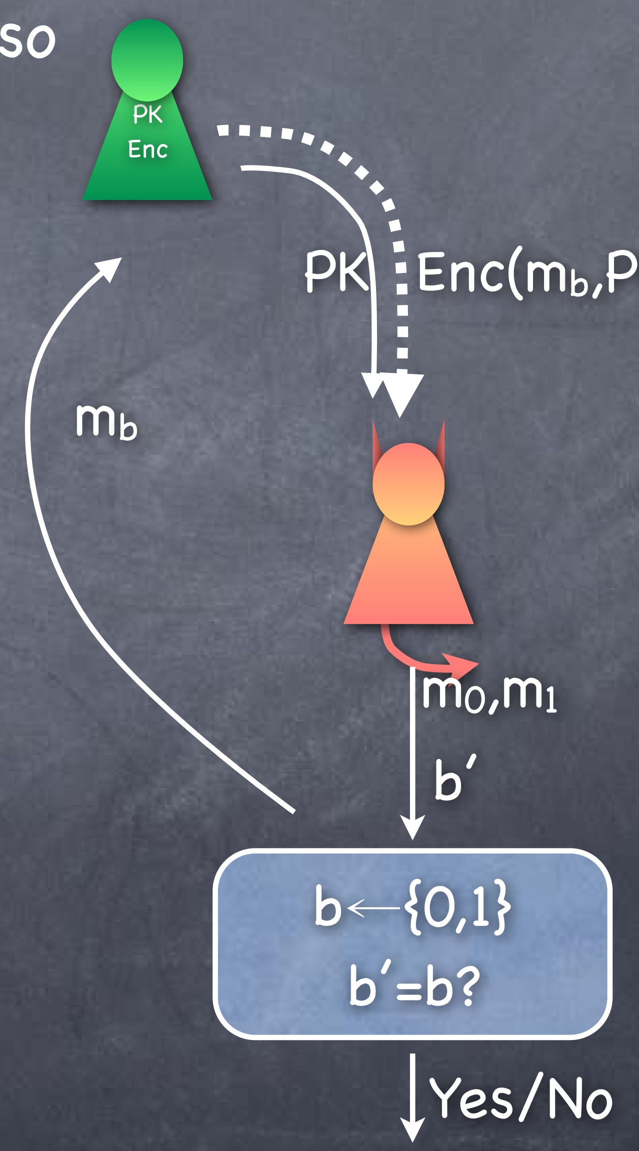
IND-CPA (PKE version)

- Experiment picks a random bit b. It also runs KeyGen to get a key (PK,SK). Adv given PK
  - Adv sends two messages m_0, m_1 to the experiment
  - Expt returns Enc(m_b,K) to the adversary
  - Adversary returns a guess b'
  - Experiment outputs 1 iff b'=b
- IND-CPA secure if for all PPT adversaries $\Pr[b'=b] - 1/2 \le \nu(k)$



IND-CPA (PKE version)

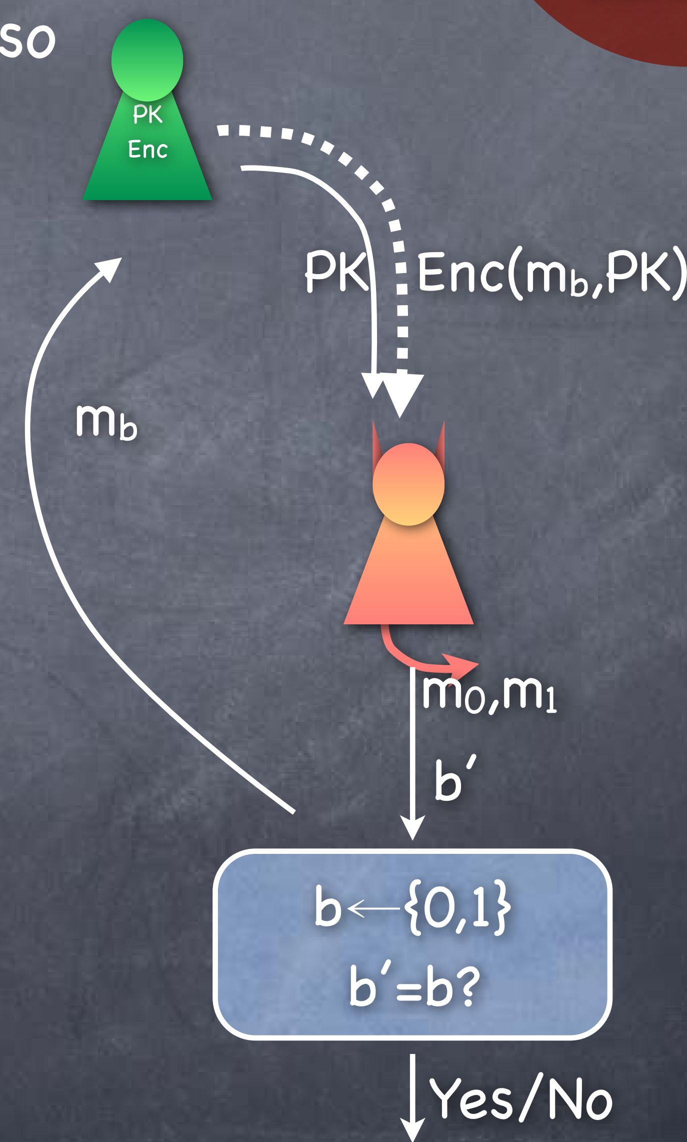
- Experiment picks a random bit b. It also runs KeyGen to get a key (PK,SK). Adv given PK
  - Adv sends two messages m_0, m_1 to the experiment
  - Expt returns Enc(m_b,K) to the adversary
  - Adversary returns a guess b′
  - Experiment outputs 1 iff b′=b
- IND-CPA secure if for all PPT adversaries Pr[b′=b] – 1/2 ≤ ν(k)



IND-CPA (PKE version)

IND-CPA + ~correctness equivalent to SIM-CPA

- Experiment picks a random bit b. It also runs KeyGen to get a key (PK,SK). Adv given PK
  - Adv sends two messages m_0, m_1 to the experiment
  - Expt returns Enc(m_b,K) to the adversary
  - Adversary returns a guess b′
  - Experiment outputs 1 iff b′=b
- IND-CPA secure if for all PPT adversaries Pr[b′=b] – 1/2 ≤ ν(k)



Perfect Secrecy?

Perfect Secrecy?

- No perfectly secret and correct PKE (even for one-time encryption)

Perfect Secrecy?

- No perfectly secret and correct PKE (even for one-time encryption)
  - Public-key and ciphertext (the total shared information between Alice and Bob at the end) should together have entire information about the message

Perfect Secrecy?

- No perfectly secret and correct PKE (even for one-time encryption)
  - Public-key and ciphertext (the total shared information between Alice and Bob at the end) should together have entire information about the message
    - Intuition: If Eve thinks Bob could decrypt it as two messages based on different SKs, Alice should be concerned too

Perfect Secrecy?

- No perfectly secret and correct PKE (even for one-time encryption)
  - Public-key and ciphertext (the total shared information between Alice and Bob at the end) should together have entire information about the message
    - Intuition: If Eve thinks Bob could decrypt it as two messages based on different SKs, Alice should be concerned too
    - i.e., Alice conveys same information to Bob and Eve

Perfect Secrecy?

- No perfectly secret and correct PKE (even for one-time encryption)
  - Public-key and ciphertext (the total shared information between Alice and Bob at the end) should together have entire information about the message
    - Intuition: If Eve thinks Bob could decrypt it as two messages based on different SKs, Alice should be concerned too
    - i.e., Alice conveys same information to Bob and Eve
    - [Exercise]

Perfect Secrecy?

- No perfectly secret and correct PKE (even for one-time encryption)
  - Public-key and ciphertext (the total shared information between Alice and Bob at the end) should together have entire information about the message
    - Intuition: If Eve thinks Bob could decrypt it as two messages based on different SKs, Alice should be concerned too
    - i.e., Alice conveys same information to Bob and Eve
    - [Exercise]
- PKE only with computational security

Perfect Secrecy?

- No perfectly secret and correct PKE (even for one-time encryption)
  - Public-key and ciphertext (the total shared information between Alice and Bob at the end) should together have entire information about the message
    - Intuition: If Eve thinks Bob could decrypt it as two messages based on different SKs, Alice should be concerned too
    - i.e., Alice conveys same information to Bob and Eve
    - [Exercise]
- PKE only with computational security

Unless assumptions of imperfect eavesdropping

Diffie-Hellman Key-exchange

Diffie-Hellman Key-exchange

- A candidate for how Alice and Bob could generate a shared key, which is “hidden” from Eve

Diffie-Hellman Key-exchange

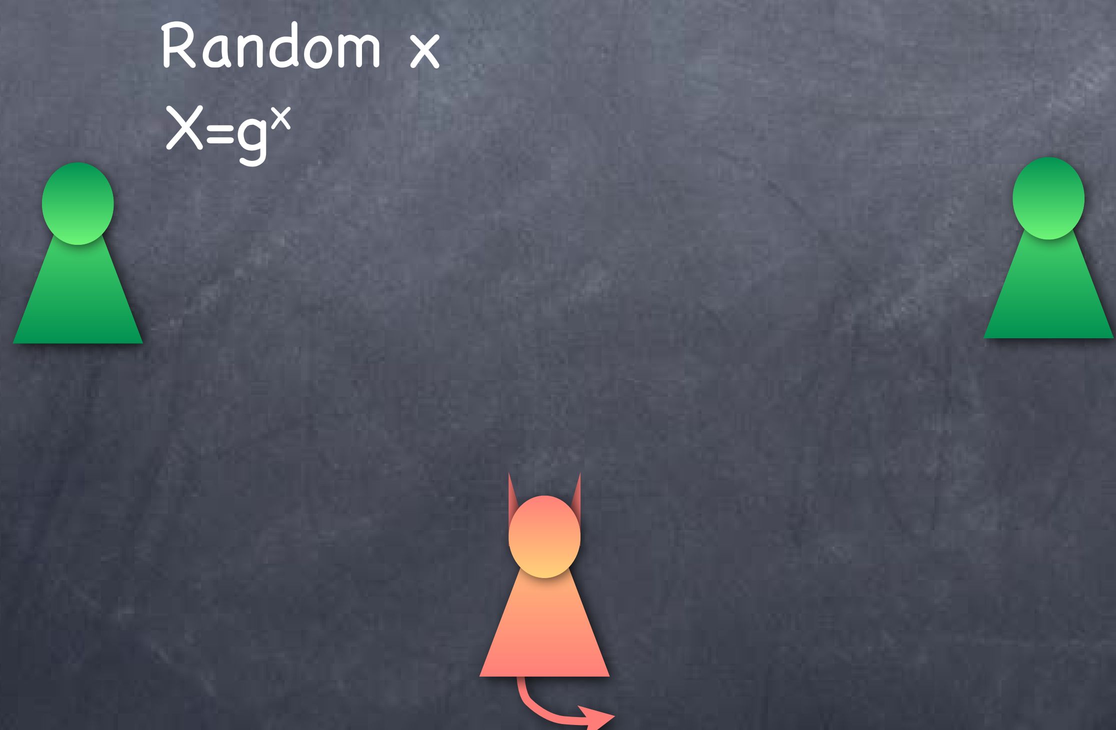
- A candidate for how Alice and Bob could generate a shared key, which is "hidden" from Eve

Diffie-Hellman Key-exchange

- A candidate for how Alice and Bob could generate a shared key, which is "hidden" from Eve

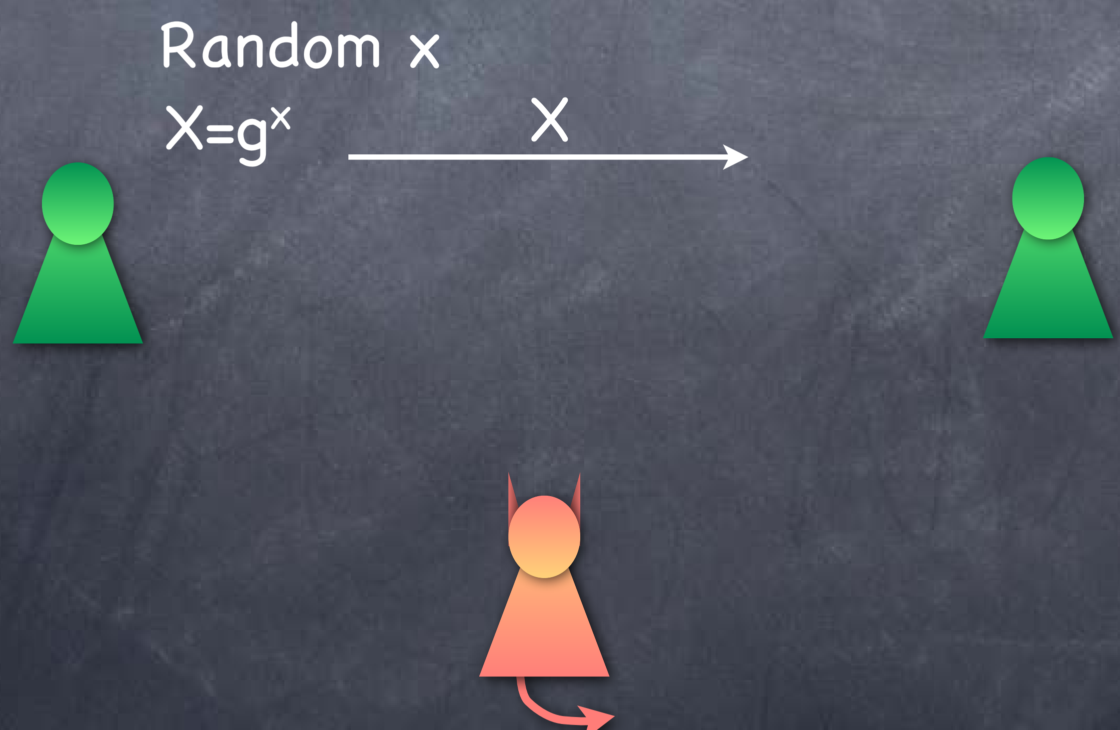
Diffie-Hellman Key-exchange

- A candidate for how Alice and Bob could generate a shared key, which is “hidden” from Eve



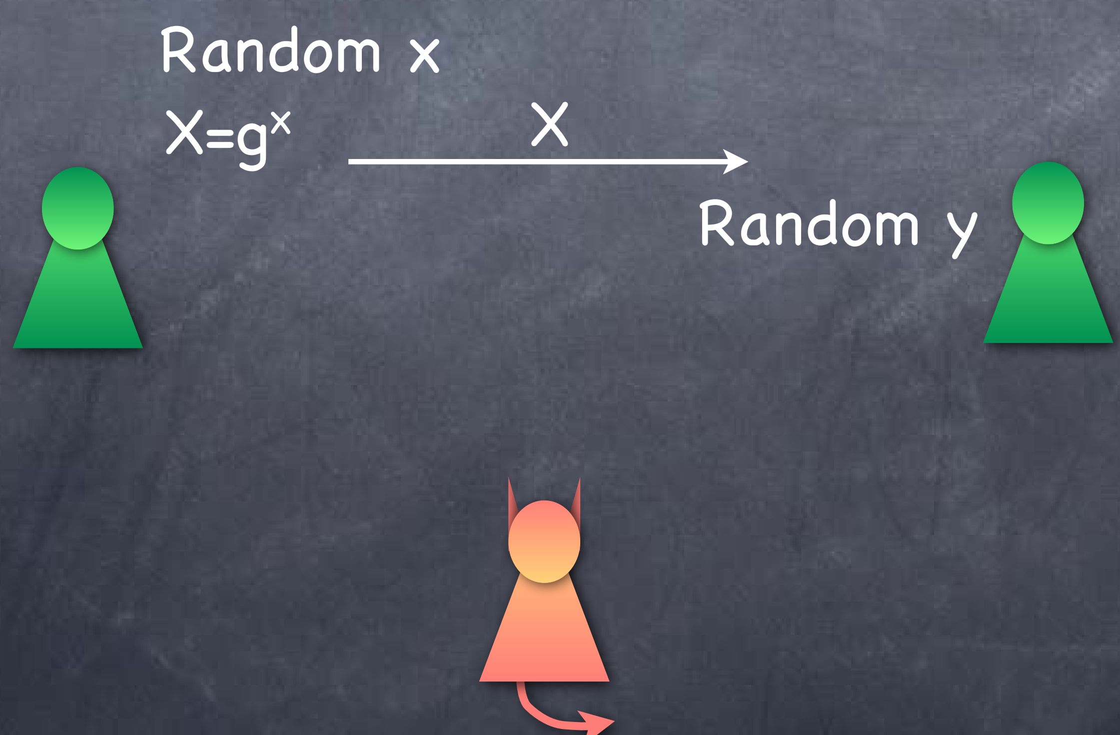
Diffie-Hellman Key-exchange

- A candidate for how Alice and Bob could generate a shared key, which is "hidden" from Eve



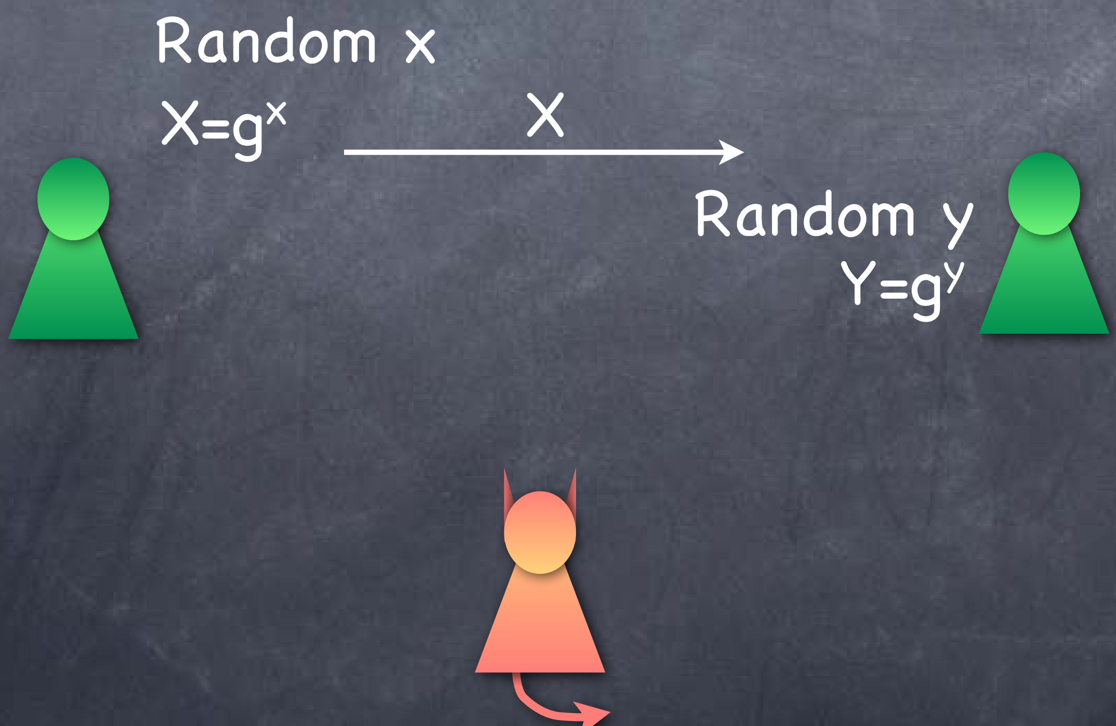
Diffie-Hellman Key-exchange

- A candidate for how Alice and Bob could generate a shared key, which is "hidden" from Eve



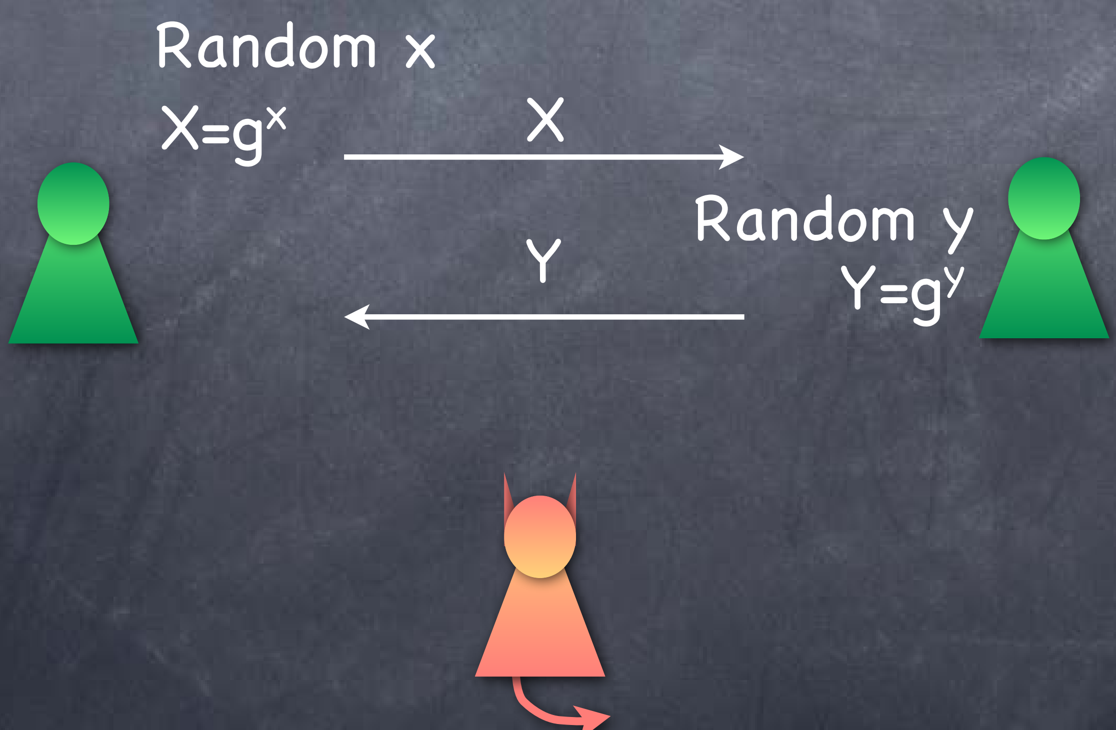
Diffie-Hellman Key-exchange

- A candidate for how Alice and Bob could generate a shared key, which is "hidden" from Eve



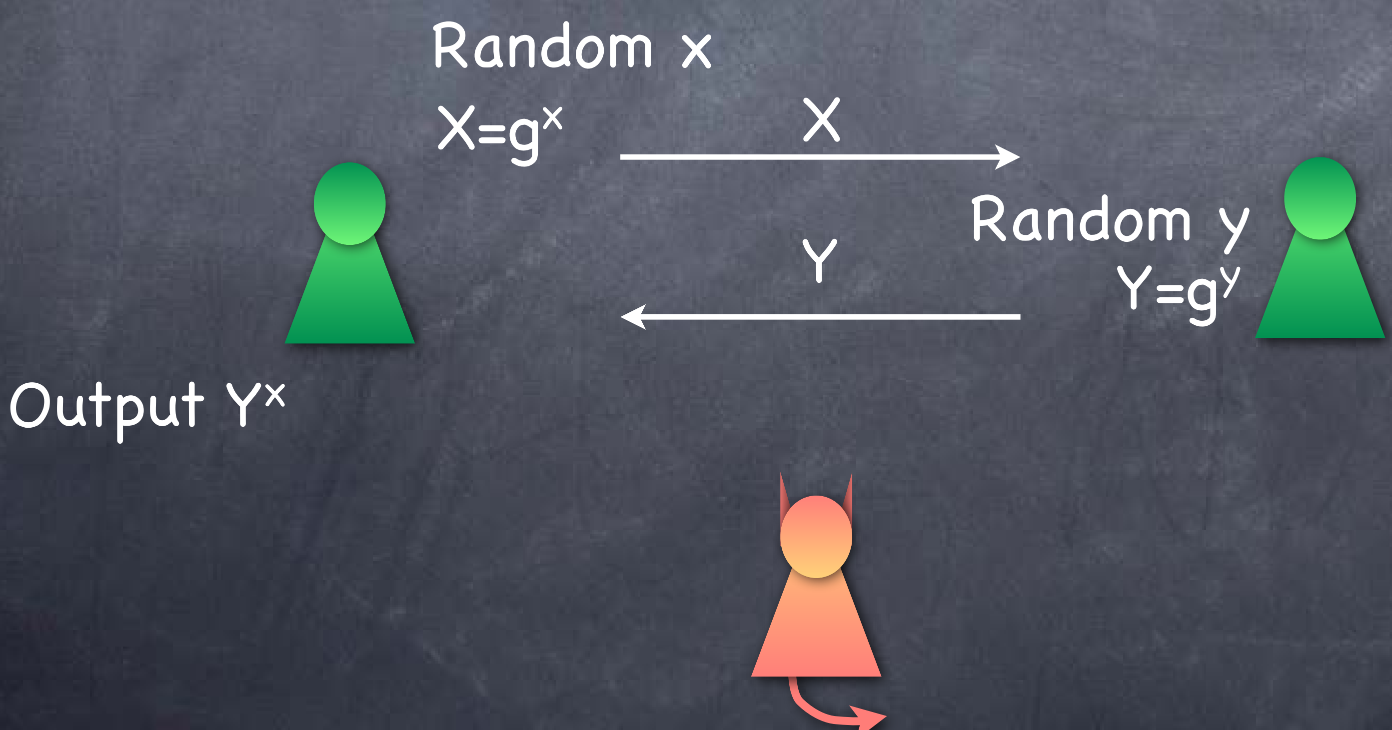
Diffie-Hellman Key-exchange

- A candidate for how Alice and Bob could generate a shared key, which is "hidden" from Eve



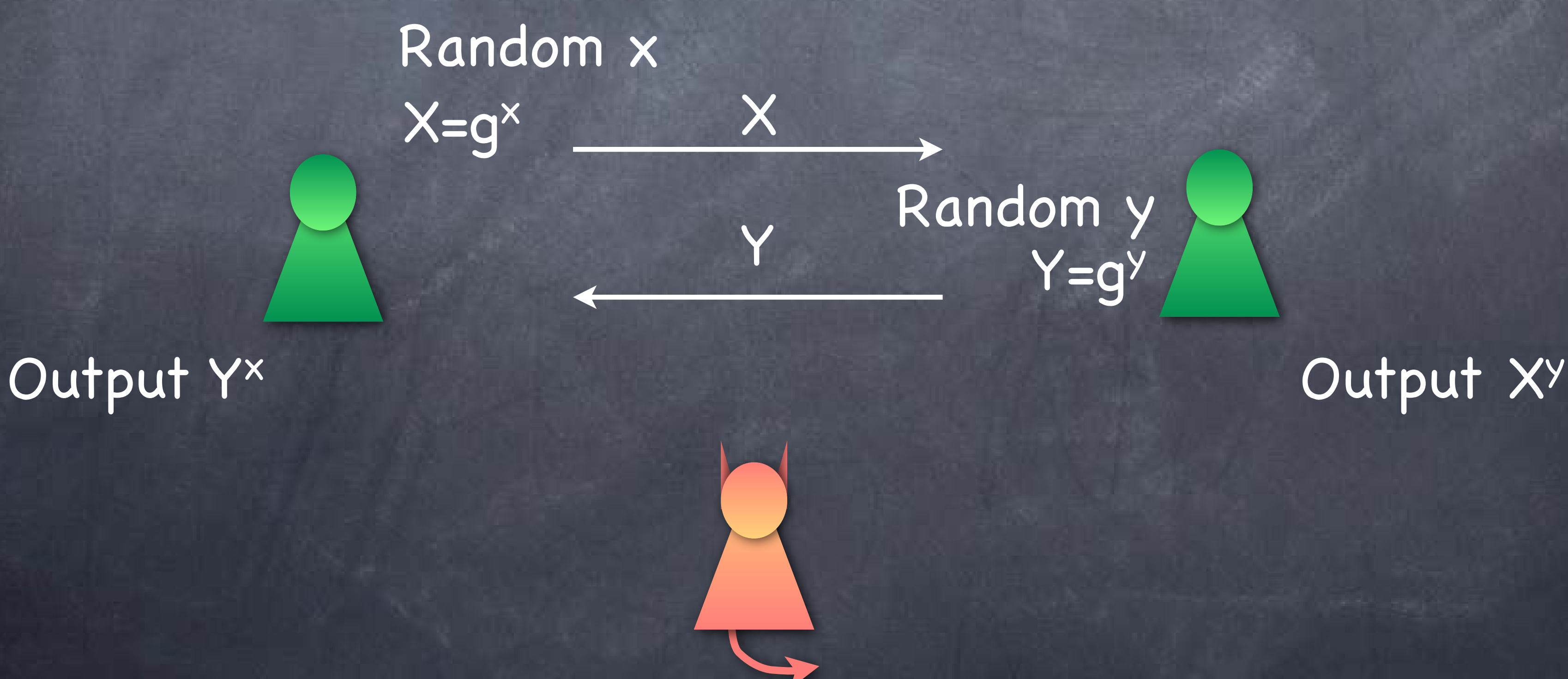
Diffie-Hellman Key-exchange

- A candidate for how Alice and Bob could generate a shared key, which is “hidden” from Eve



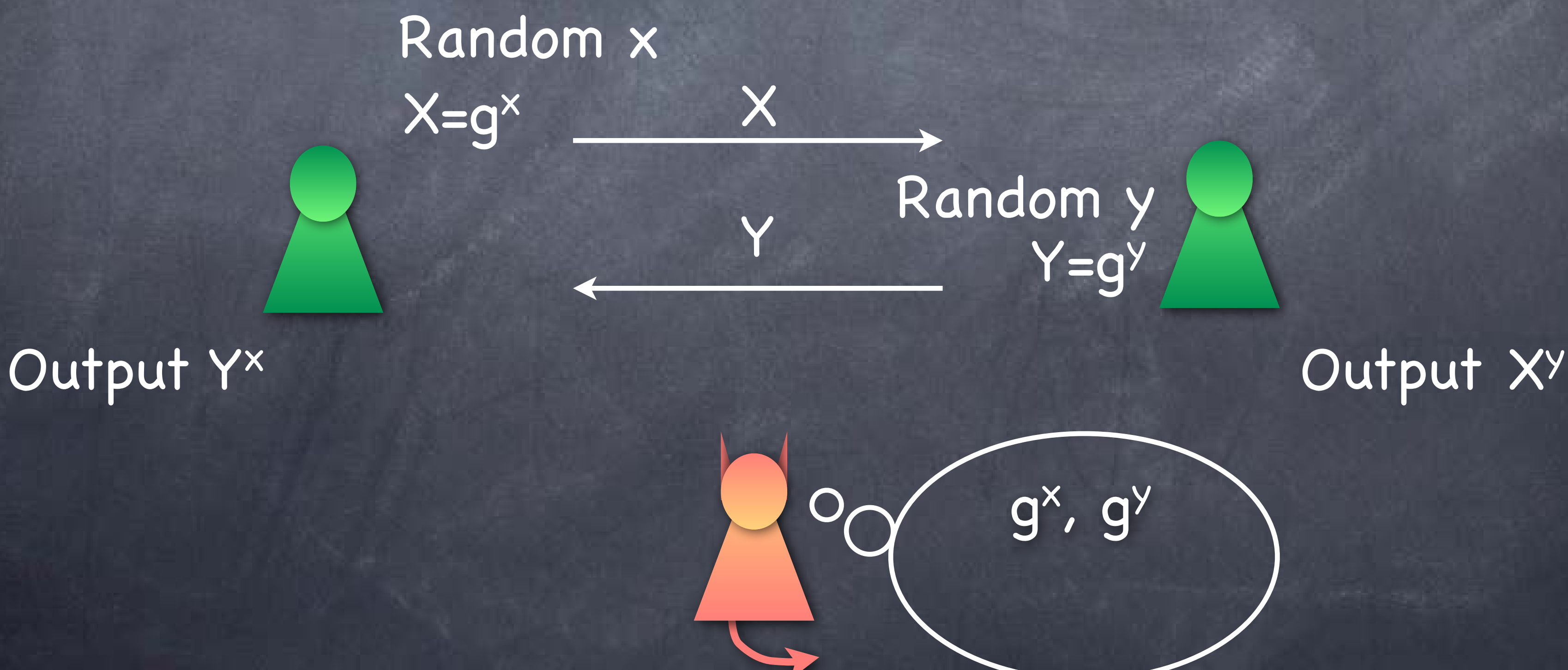
Diffie-Hellman Key-exchange

- A candidate for how Alice and Bob could generate a shared key, which is "hidden" from Eve



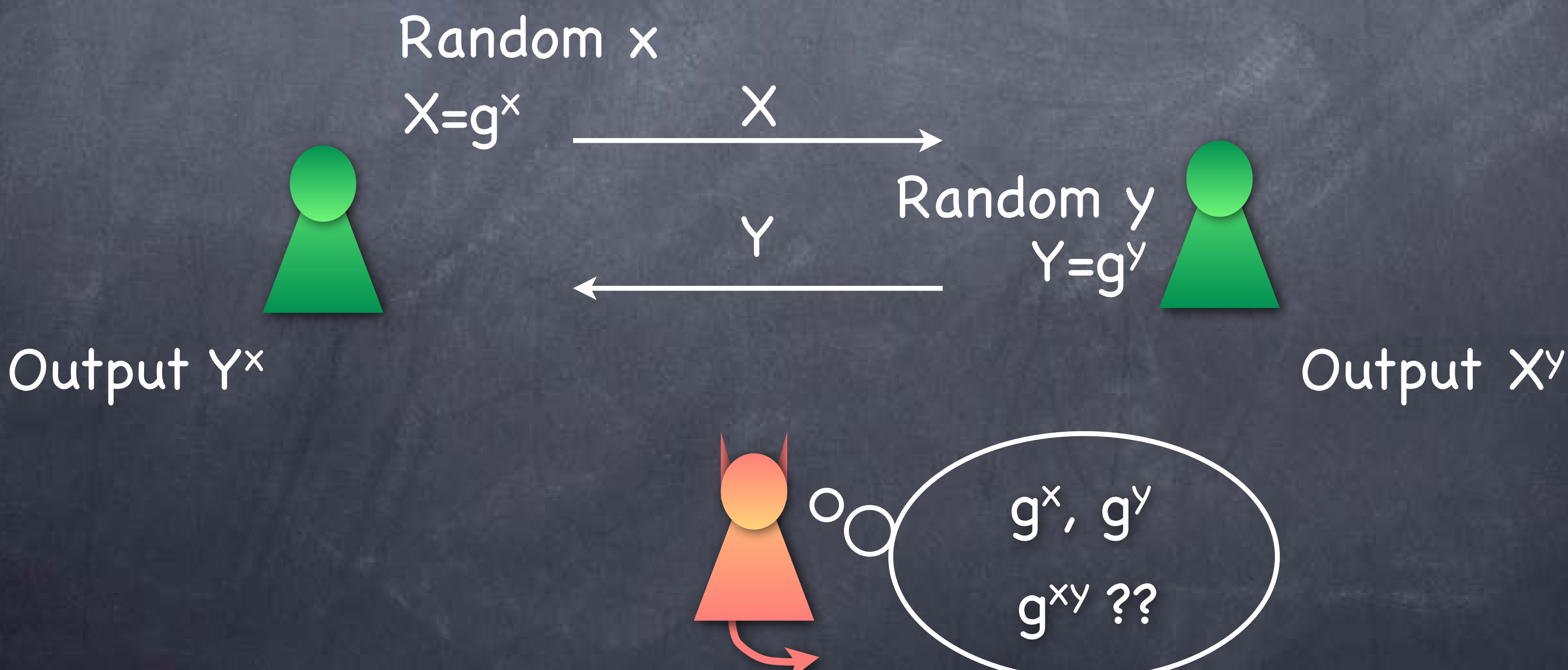
Diffie-Hellman Key-exchange

- A candidate for how Alice and Bob could generate a shared key, which is "hidden" from Eve



Diffie-Hellman Key-exchange

- A candidate for how Alice and Bob could generate a shared key, which is "hidden" from Eve



Why DH-Key-exchange could be secure

Why DH-Key-exchange could be secure

- Given g^x, g^y for random x, y, g^{xy} should be "hidden"

Why DH-Key-exchange could be secure

- Given g^x, g^y for random x, y, g^{xy} should be "hidden"
  - i.e., could still be used as a pseudorandom element

Why DH-Key-exchange could be secure

- Given g^x, g^y for random x, y, g^{xy} should be "hidden"
  - i.e., could still be used as a pseudorandom element
  - i.e., $(g^x, g^y, g^{xy}) \approx (g^x, g^y, R)$

Why DH-Key-exchange could be secure

- Given g^x, g^y for random x, y, g^{xy} should be "hidden"
  - i.e., could still be used as a pseudorandom element
  - i.e., $(g^x, g^y, g^{xy}) \approx (g^x, g^y, R)$
- Is that reasonable to expect?

Why DH-Key-exchange could be secure

- Given g^x, g^y for random x, y, g^{xy} should be "hidden"
  - i.e., could still be used as a pseudorandom element
  - i.e., $(g^x, g^y, g^{xy}) \approx (g^x, g^y, R)$
- Is that reasonable to expect?
  - Depends on the "group"

Groups, by examples

Groups, by examples

- A set G (for us finite, unless otherwise specified) and a "group operation" * that is associative, has an identity, is invertible, and (for us) commutative

Groups, by examples

- A set G (for us finite, unless otherwise specified) and a "group operation" * that is associative, has an identity, is invertible, and (for us) commutative
- Examples: $\mathbb{Z}$ = (integers, +) (this is an infinite group),

  $\mathbb{Z}_N$ = (integers modulo N, + mod N),

  G^n = (Cartesian product of a group G, coordinate-wise operation)

Groups, by examples

- A set G (for us finite, unless otherwise specified) and a "group operation" * that is associative, has an identity, is invertible, and (for us) commutative
- Examples: $\mathbb{Z}$ = (integers, +) (this is an infinite group),

  $\mathbb{Z}_N$ = (integers modulo N, + mod N),

  G^n = (Cartesian product of a group G, coordinate-wise operation)
- Order of a group G: |G| = number of elements in G

Groups, by examples

- A set G (for us finite, unless otherwise specified) and a "group operation" * that is associative, has an identity, is invertible, and (for us) commutative
- Examples: $\mathbb{Z}$ = (integers, +) (this is an infinite group),

  $\mathbb{Z}_N$ = (integers modulo N, + mod N),

  G^n = (Cartesian product of a group G, coordinate-wise operation)
- Order of a group G: |G| = number of elements in G
- For any $a \in G$, $a^{|G|} = a * a * \ldots * a$ (|G| times) = identity

Groups, by examples

- A set G (for us finite, unless otherwise specified) and a "group operation" * that is associative, has an identity, is invertible, and (for us) commutative
- Examples: $\mathbb{Z}$ = (integers, +) (this is an infinite group),

  $\mathbb{Z}_N$ = (integers modulo N, + mod N),

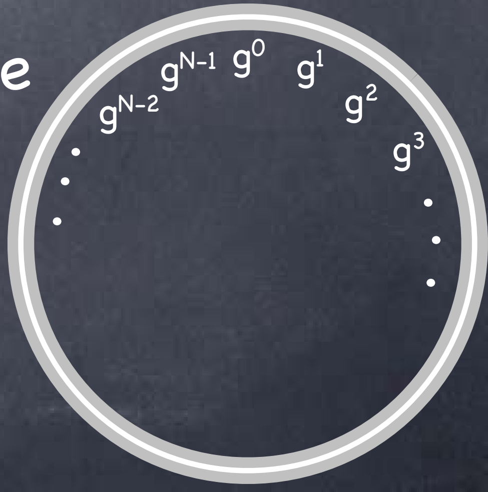
  G^n = (Cartesian product of a group G, coordinate-wise operation)
- Order of a group G: |G| = number of elements in G
- For any $a \in G$, $a^{|G|} = a * a * \ldots * a$ (|G| times) = identity

Groups, by examples

- A set G (for us finite, unless otherwise specified) and a "group operation" * that is associative, has an identity, is invertible, and (for us) commutative
- Examples: $\mathbb{Z}$ = (integers, +) (this is an infinite group),
  $\mathbb{Z}_N$ = (integers modulo N, + mod N),
  G^n = (Cartesian product of a group G, coordinate-wise operation)
- Order of a group G: |G| = number of elements in G
- For any $a \in G$, $a^{|G|} = a * a * \ldots * a$ (|G| times) = identity
- Finite Cyclic group (in multiplicative notation): there is one element g such that $G = \{g^0, g^1, g^2, \ldots g^{|G|-1}\}$

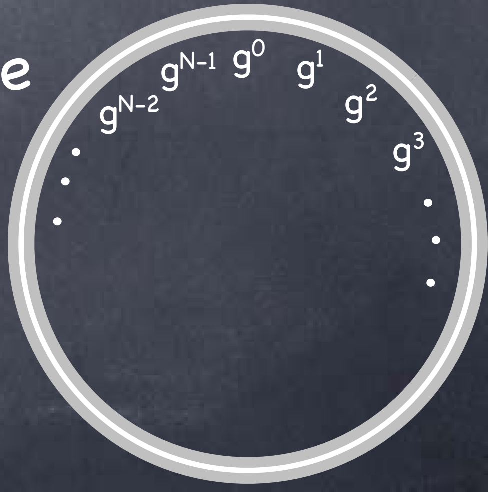
Groups, by examples

- A set G (for us finite, unless otherwise specified) and a "group operation" $*$ that is associative, has an identity, is invertible, and (for us) commutative
- Examples: $\mathbb{Z}$ = (integers, +) (this is an infinite group),
  $\mathbb{Z}_N$ = (integers modulo N, + mod N),
  G^n = (Cartesian product of a group G, coordinate-wise operation)
- Order of a group G: $|G|$ = number of elements in G
- For any $a \in G$, $a^{|G|} = a * a * \ldots * a$ ($|G|$ times) = identity
- Finite Cyclic group (in multiplicative notation): there is one element g such that $G = \{g^0, g^1, g^2, \ldots g^{|G|-1}\}$



Groups, by examples

- A set G (for us finite, unless otherwise specified) and a "group operation" * that is associative, has an identity, is invertible, and (for us) commutative
- Examples: $\mathbb{Z}$ = (integers, +) (this is an infinite group),
  $\mathbb{Z}_N$ = (integers modulo N, + mod N),
  G^n = (Cartesian product of a group G, coordinate-wise operation)
- Order of a group G: |G| = number of elements in G
- For any a∈G, $a^{|G|} = a * a * \dots * a$ (|G| times) = identity
- Finite Cyclic group (in multiplicative notation): there is one element g such that $G = \{g^0, g^1, g^2, \dots g^{|G|-1}\}$
  - Prototype: $\mathbb{Z}_N$ (additive group), with g=1

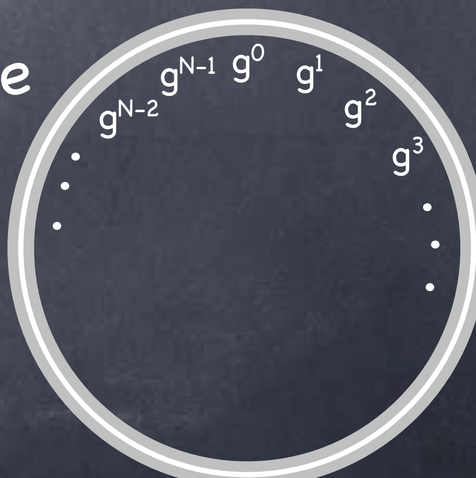


Groups, by examples

- A set G (for us finite, unless otherwise specified) and a "group operation" * that is associative, has an identity, is invertible, and (for us) commutative
- Examples: $\mathbb{Z}$ = (integers, +) (this is an infinite group),

  $\mathbb{Z}_N$ = (integers modulo N, + mod N),

  G^n = (Cartesian product of a group G, coordinate-wise operation)
- Order of a group G: |G| = number of elements in G
- For any $a \in G$, $a^{|G|} = a * a * \ldots * a$ (|G| times) = identity
- Finite Cyclic group (in multiplicative notation): there is one element g such that $G = \{g^0, g^1, g^2, \ldots g^{|G|-1}\}$
  - Prototype: $\mathbb{Z}_N$ (additive group), with g=1
    - or any g s.t. gcd(g,N) = 1



Groups, by examples

Groups, by examples

- $\mathbb{Z}_N^*$ = (generators of $\mathbb{Z}_N$, multiplication mod N)

Groups, by examples

- $\mathbb{Z}_N^*$ = (generators of $\mathbb{Z}_N$, multiplication mod N)
  - Numbers in {1,..,N-1} which have a multiplicative inverse mod N

Groups, by examples

- $\mathbb{Z}_N^*$ = (generators of $\mathbb{Z}_N$, multiplication mod N)
  - Numbers in {1,..,N-1} which have a multiplicative inverse mod N
  - If N is prime, $\mathbb{Z}_N^*$ is a cyclic group, of order N-1

Groups, by examples

- $\mathbb{Z}_N^*$ = (generators of $\mathbb{Z}_N$, multiplication mod N)
  - Numbers in {1,..,N-1} which have a multiplicative inverse mod N
  - If N is prime, $\mathbb{Z}_N^*$ is a cyclic group, of order N-1
    - e.g. $\mathbb{Z}_5^*$ = {1,2,3,4} is generated by 2 (as 1,2,4,3), and by 3 (as 1,3,4,2). But 1 and 4 are not generators.

Groups, by examples

- $\mathbb{Z}_N^*$ = (generators of $\mathbb{Z}_N$, multiplication mod N)
  - Numbers in {1,..,N-1} which have a multiplicative inverse mod N
  - If N is prime, $\mathbb{Z}_N^*$ is a cyclic group, of order N-1
    - e.g. $\mathbb{Z}_5^*$ = {1,2,3,4} is generated by 2 (as 1,2,4,3), and by 3 (as 1,3,4,2). But 1 and 4 are not generators.
    - (Also cyclic for certain other values of N)

Discrete Log Assumption

Discrete Log Assumption

- Discrete Log (w.r.t g) in a (multiplicative) cyclic group G generated by g: $DL_g(X)$:= unique x such that $X = g^x$ ($x \in \{0,1,...,|G|-1\}$)

Discrete Log Assumption

- Discrete Log (w.r.t g) in a (multiplicative) cyclic group G generated by g: $DL_g(X)$:= unique x such that $X = g^x$ ($x \in \{0,1,...,|G|-1\}$)
- In a (computationally efficient) group, given integer x and the standard representation of a group element g, can efficiently find the standard representation of $X=g^x$ (How?)

Discrete Log Assumption

Repeated squaring

- Discrete Log (w.r.t g) in a (multiplicative) cyclic group G generated by g: $DL_g(X)$:= unique x such that $X = g^x$ ($x \in \{0,1,...,|G|-1\}$)
- In a (computationally efficient) group, given integer x and the standard representation of a group element g, can efficiently find the standard representation of $X=g^x$ (How?)

Discrete Log Assumption

Repeated squaring

- Discrete Log (w.r.t g) in a (multiplicative) cyclic group G generated by g: $DL_g(X)$:= unique x such that $X = g^x$ $(x \in \{0,1,...,|G|-1\})$
- In a (computationally efficient) group, given integer x and the standard representation of a group element g, can efficiently find the standard representation of $X=g^x$ (How?)
  - But given X and g, may not be easy to find x (depending on G)

Discrete Log Assumption

Repeated squaring

- Discrete Log (w.r.t g) in a (multiplicative) cyclic group G generated by g: $DL_g(X)$:= unique x such that $X = g^x$ ($x \in \{0,1,...,|G|-1\}$)
- In a (computationally efficient) group, given integer x and the standard representation of a group element g, can efficiently find the standard representation of $X=g^x$ (How?)
  - But given X and g, may not be easy to find x (depending on G)
  - DLA: Every PPT Adv has negligible success probability in the DL Expt: (G,g)⟵GroupGen; X⟵G; Adv(G,g,X)⟶z; $g^z=X$?

Discrete Log Assumption

Repeated squaring

- Discrete Log (w.r.t g) in a (multiplicative) cyclic group G generated by g: $DL_g(X)$:= unique x such that $X = g^x$ $(x \in \{0,1,...,|G|-1\})$
- In a (computationally efficient) group, given integer x and the standard representation of a group element g, can efficiently find the standard representation of $X=g^x$ (How?)
  - But given X and g, may not be easy to find x (depending on G)
  - DLA: Every PPT Adv has negligible success probability in the DL Expt: $(G,g) \leftarrow$ GroupGen; $X \leftarrow G$; $Adv(G,g,X) \rightarrow z$; $g^z=X$?

OWF collection:
Raise(x;G,g)
= $(g^x;G,g)$

Discrete Log Assumption

Repeated squaring

- Discrete Log (w.r.t g) in a (multiplicative) cyclic group G generated by g: $DL_g(X)$:= unique x such that $X = g^x$ ($x \in \{0,1,...,|G|-1\}$)
- In a (computationally efficient) group, given integer x and the standard representation of a group element g, can efficiently find the standard representation of $X=g^x$ (How?)
  - But given X and g, may not be easy to find x (depending on G)
  - DLA: Every PPT Adv has negligible success probability in the DL Expt: $(G,g) \leftarrow$ GroupGen; $X \leftarrow G$; $Adv(G,g,X) \rightarrow z$; $g^z=X$?
- If DLA broken, then Diffie-Hellman key-exchange broken

OWF collection:
Raise(x;G,g) = $(g^x;G,g)$

Discrete Log Assumption

Repeated squaring

- Discrete Log (w.r.t g) in a (multiplicative) cyclic group G generated by g: $DL_g(X)$:= unique x such that $X = g^x$ ($x \in \{0,1,...,|G|-1\}$)
- In a (computationally efficient) group, given integer x and the standard representation of a group element g, can efficiently find the standard representation of $X=g^x$ (How?)
  - But given X and g, may not be easy to find x (depending on G)
  - DLA: Every PPT Adv has negligible success probability in the DL Expt: $(G,g) \leftarrow$ GroupGen; $X \leftarrow G$; $Adv(G,g,X) \rightarrow z$; $g^z=X$?

OWF collection: Raise(x;G,g) = $(g^x;G,g)$

- If DLA broken, then Diffie-Hellman key-exchange broken
  - Eve gets x, y from g^x, g^y (sometimes) and can compute g^{xy} herself

Discrete Log Assumption

Repeated squaring

- Discrete Log (w.r.t g) in a (multiplicative) cyclic group G generated by g: $DL_g(X)$:= unique x such that $X = g^x$ ($x \in \{0,1,\ldots,|G|-1\}$)
- In a (computationally efficient) group, given integer x and the standard representation of a group element g, can efficiently find the standard representation of $X=g^x$ (How?)
  - But given X and g, may not be easy to find x (depending on G)
  - DLA: Every PPT Adv has negligible success probability in the DL Expt: $(G,g) \leftarrow$ GroupGen; $X \leftarrow G$; $Adv(G,g,X) \rightarrow z$; $g^z=X$?

OWF collection: Raise(x;G,g) = (g^x;G,g)

- If DLA broken, then Diffie-Hellman key-exchange broken
  - Eve gets x, y from g^x, g^y (sometimes) and can compute g^{xy} herself
    - A "key-recovery" attack

Discrete Log Assumption

Repeated squaring

- Discrete Log (w.r.t g) in a (multiplicative) cyclic group G generated by g: $DL_g(X)$:= unique x such that $X = g^x$ ($x \in \{0,1,...,|G|-1\}$)
- In a (computationally efficient) group, given integer x and the standard representation of a group element g, can efficiently find the standard representation of $X=g^x$ (How?)
  - But given X and g, may not be easy to find x (depending on G)
  - DLA: Every PPT Adv has negligible success probability in the DL Expt: $(G,g) \leftarrow$ GroupGen; $X \leftarrow G$; $Adv(G,g,X) \rightarrow z$; $g^z=X$?

OWF collection: Raise(x;G,g) = (g^x;G,g)

- If DLA broken, then Diffie-Hellman key-exchange broken
  - Eve gets x, y from g^x, g^y (sometimes) and can compute g^{xy} herself
    - A "key-recovery" attack
  - Note: could potentially break pseudorandomness without breaking DLA too

Decisional Diffie-Hellman (DDH) Assumption

Decisional Diffie-Hellman (DDH) Assumption

- $\{(g^x, g^y, g^{xy})\}_{(G,g)\leftarrow\mathbf{GroupGen};\ x,y\leftarrow[|G|]} \approx \{(g^x, g^y, g^r)\}_{(G,g)\leftarrow\mathbf{GroupGen};\ x,y,r\leftarrow[|G|]}$

Decisional Diffie-Hellman (DDH) Assumption

- $\{(g^x, g^y, g^{xy})\}_{(G,g)\leftarrow\textbf{GroupGen};\ x,y\leftarrow[|G|]} \approx \{(g^x, g^y, g^r)\}_{(G,g)\leftarrow\textbf{GroupGen};\ x,y,r\leftarrow[|G|]}$
- At least as strong as DLA

Decisional Diffie-Hellman (DDH) Assumption

- $\{(g^x, g^y, g^{xy})\}_{(G,g)\leftarrow \textbf{GroupGen};\ x,y\leftarrow[|G|]} \approx \{(g^x, g^y, g^r)\}_{(G,g)\leftarrow \textbf{GroupGen};\ x,y,r\leftarrow[|G|]}$
- At least as strong as DLA
  - If DDH assumption holds, then DLA holds [Why?]

Decisional Diffie-Hellman (DDH) Assumption

- $\{(g^x, g^y, g^{xy})\}_{(G,g)\leftarrow\textbf{GroupGen};\ x,y\leftarrow[|G|]} \approx \{(g^x, g^y, g^r)\}_{(G,g)\leftarrow\textbf{GroupGen};\ x,y,r\leftarrow[|G|]}$
- At least as strong as DLA
  - If DDH assumption holds, then DLA holds [Why?]
- But possible that DLA holds and DDH assumption doesn't

Decisional Diffie-Hellman (DDH) Assumption

- $\{(g^x, g^y, g^{xy})\}_{(G,g)\leftarrow \textbf{GroupGen};\ x,y\leftarrow[|G|]} \approx \{(g^x, g^y, g^r)\}_{(G,g)\leftarrow \textbf{GroupGen};\ x,y,r\leftarrow[|G|]}$
- At least as strong as DLA
  - If DDH assumption holds, then DLA holds [Why?]
- But possible that DLA holds and DDH assumption doesn't
  - e.g.: DLA is widely assumed to hold in $\mathbb{Z}_p^*$ (p prime), but DDH assumption doesn't hold there!

A Candidate DDH Group

A Candidate DDH Group

- Consider $\mathbb{QR}_p^*$: subgroup of Quadratic Residues ("even power" elements) of $\mathbb{Z}_p^*$

A Candidate DDH Group

- Consider $\mathbb{QR}_p{}^*$: subgroup of Quadratic Residues ("even power" elements) of $\mathbb{Z}_p{}^*$

A Candidate DDH Group

- Consider $\mathbb{QR}_p^*$: subgroup of Quadratic Residues ("even power" elements) of $\mathbb{Z}_p^*$
- Easy to check if an element is a QR or not: check if raising to |G|/2 gives 1 (identity element)

A Candidate DDH Group

- Consider $\mathbb{QR}_p^*$: subgroup of Quadratic Residues ("even power" elements) of $\mathbb{Z}_p^*$

- Easy to check if an element is a QR or not: check if raising to |G|/2 gives 1 (identity element)
- DDH does not hold in $\mathbb{Z}_p^*$: g^{xy} is a QR w/ prob. 3/4; g^z is QR only w/ prob. 1/2.

A Candidate DDH Group

- Consider $\mathbb{QR}_p^*$: subgroup of Quadratic Residues ("even power" elements) of $\mathbb{Z}_p^*$
- Easy to check if an element is a QR or not: check if raising to |G|/2 gives 1 (identity element)
- DDH does not hold in $\mathbb{Z}_p^*$: g^{xy} is a QR w/ prob. 3/4; g^z is QR only w/ prob. 1/2.
- How about in $\mathbb{QR}_p^*$?

A Candidate DDH Group

- Consider $\mathbb{QR}_p^*$: subgroup of Quadratic Residues ("even power" elements) of $\mathbb{Z}_p^*$

- Easy to check if an element is a QR or not: check if raising to |G|/2 gives 1 (identity element)
- DDH does not hold in $\mathbb{Z}_p^*$: g^{xy} is a QR w/ prob. 3/4; g^z is QR only w/ prob. 1/2.
- How about in $\mathbb{QR}_p^*$?
  - Could check if cubic residue in $\mathbb{Z}_p^*$!

A Candidate DDH Group

- Consider $\mathbb{QR}_P^*$: subgroup of Quadratic Residues ("even power" elements) of $\mathbb{Z}_P^*$
- Easy to check if an element is a QR or not: check if raising to $|G|/2$ gives 1 (identity element)
- DDH does not hold in $\mathbb{Z}_P^*$: g^{xy} is a QR w/ prob. 3/4; g^z is QR only w/ prob. 1/2.
- How about in $\mathbb{QR}_P^*$?
    - Could check if cubic residue in $\mathbb{Z}_P^*$!
        - But if (P-1) is not divisible by 3, all elements in $\mathbb{Z}_P^*$ are cubic residues!

A Candidate DDH Group

- Consider $\mathbb{QR}_P^*$: subgroup of Quadratic Residues ("even power" elements) of $\mathbb{Z}_P^*$
- Easy to check if an element is a QR or not: check if raising to $|G|/2$ gives 1 (identity element)
- DDH does not hold in $\mathbb{Z}_P^*$: g^{xy} is a QR w/ prob. 3/4; g^z is QR only w/ prob. 1/2.
- How about in $\mathbb{QR}_P^*$?
    - Could check if cubic residue in $\mathbb{Z}_P^*$!
        - But if (P-1) is not divisible by 3, all elements in $\mathbb{Z}_P^*$ are cubic residues!
    - "Safe" if (P-1)/2 is also prime: P called a safe-prime

A Candidate DDH Group

- Consider $\mathbb{QR}_P{}^*$: subgroup of Quadratic Residues ("even power" elements) of $\mathbb{Z}_P{}^*$
- Easy to check if an element is a QR or not: check if raising to |G|/2 gives 1 (identity element)
- DDH does not hold in $\mathbb{Z}_P{}^*$: g^{xy} is a QR w/ prob. 3/4; g^z is QR only w/ prob. 1/2.
- How about in $\mathbb{QR}_P{}^*$?
  - Could check if cubic residue in $\mathbb{Z}_P{}^*$!
    - But if (P-1) is not divisible by 3, all elements in $\mathbb{Z}_P{}^*$ are cubic residues!
  - "Safe" if (P-1)/2 is also prime: P called a safe-prime

DDH Candidate:
$\mathbb{QR}_P{}^*$
where P is a safe-prime

Today

Today

- Public Key Encryption
  - CPA security
  - Diffie-Hellman Key Exchange
  - DDH Assumption
    - Candidate group: $\mathbb{QR}_P^*$ where P is a "safe prime"

Today

- Public Key Encryption
  - CPA security
  - Diffie-Hellman Key Exchange
  - DDH Assumption
    - Candidate group: $\mathbb{QR}_P^*$ where P is a "safe prime"
- Next: El Gamal encryption (DH Key-Exchange used for encryption). Building CPA secure PKE, more generally. CCA security for PKE.