Public-Key Cryptography

Public-Key Cryptography

Lecture 6

Public-Key Encryption

Public-Key Cryptography

Lecture 6

Public-Key Encryption

Diffie-Hellman Key-Exchange, El Gamal Encryption

PKE scheme

PKE scheme

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

PKE scheme

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

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

PKE scheme

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

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

PKE scheme

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

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

a.k.a asymmetric-key encryption

- PKE

PKE scheme

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

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

a.k.a asymmetric-key encryption

- PKE
  - Syntax

PKE scheme

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

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

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

a.k.a asymmetric-key encryption

PKE scheme

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

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

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

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: $\mathcal{M} \times \mathcal{K} \rightarrow \mathcal{C}$
    - Enc: $\mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$
  - Correctness
    - $\forall K \in \mathrm{Range}(\mathrm{KeyGen})$, $\mathrm{Dec}(\ \mathrm{Enc}(m,K),\ K) = m$

a.k.a asymmetric-key encryption

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

PKE scheme

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

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

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

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: $\mathcal{M} \times \mathcal{K} \rightarrow \mathcal{C}$
    - Enc: $\mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$
  - Correctness
    - $\forall K \in$ Range(KeyGen), Dec(Enc(m,K), K) = m

a.k.a asymmetric-key encryption

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

PKE scheme

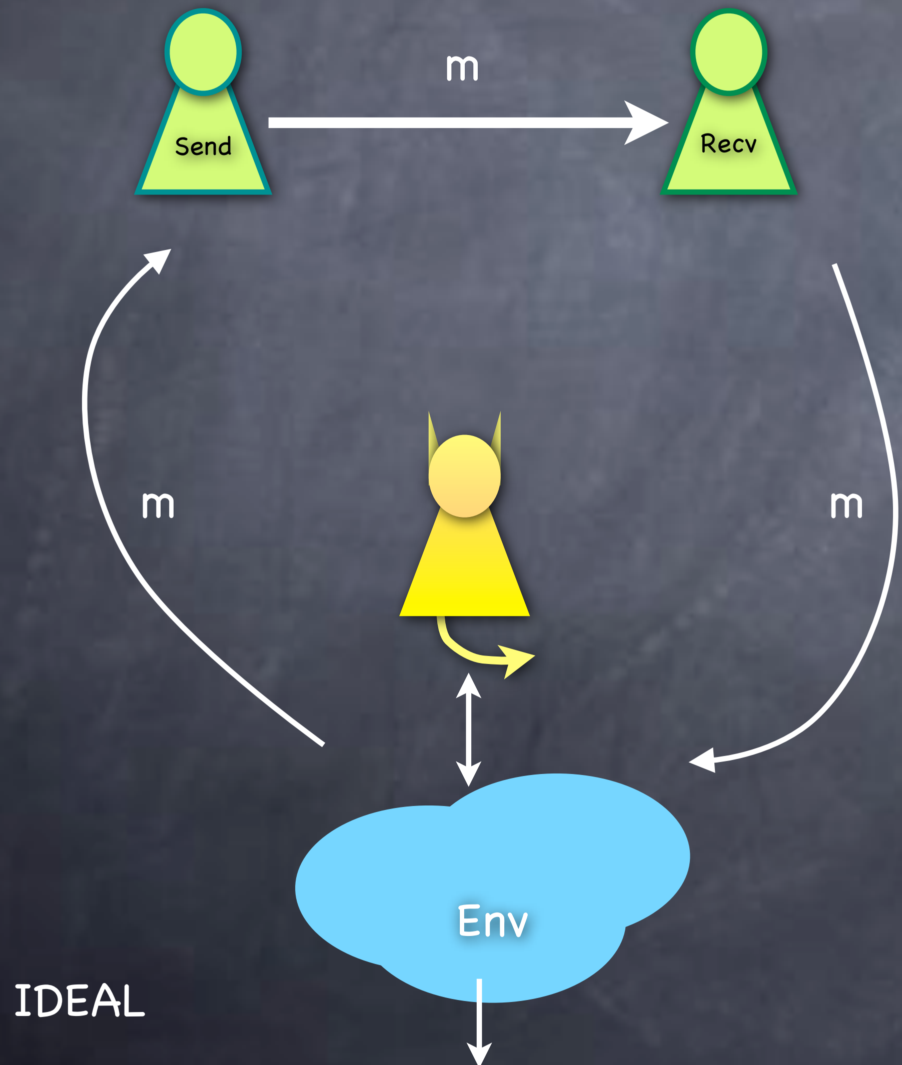
Shared/Symmetric-Key Encryption (a.k.a private-key encryption)

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

a.k.a asymmetric-key encryption

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

SIM-CPA (PKE Version)



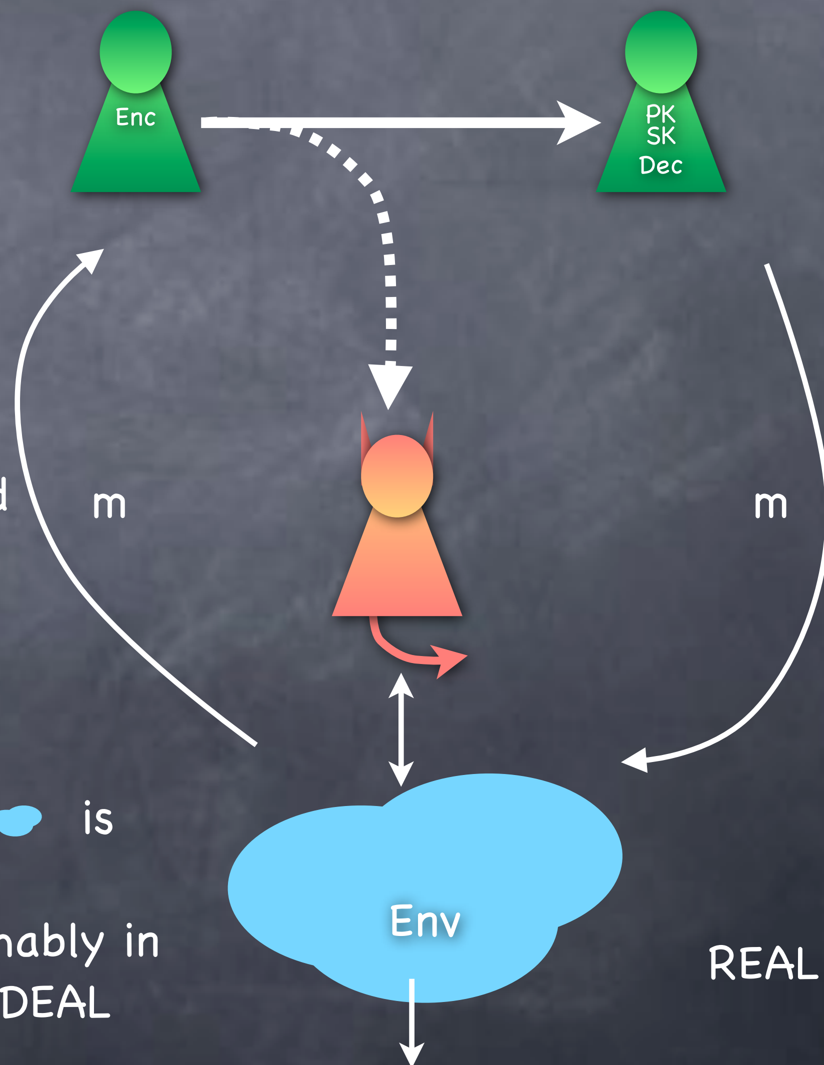
Secure (and correct) if:

∀

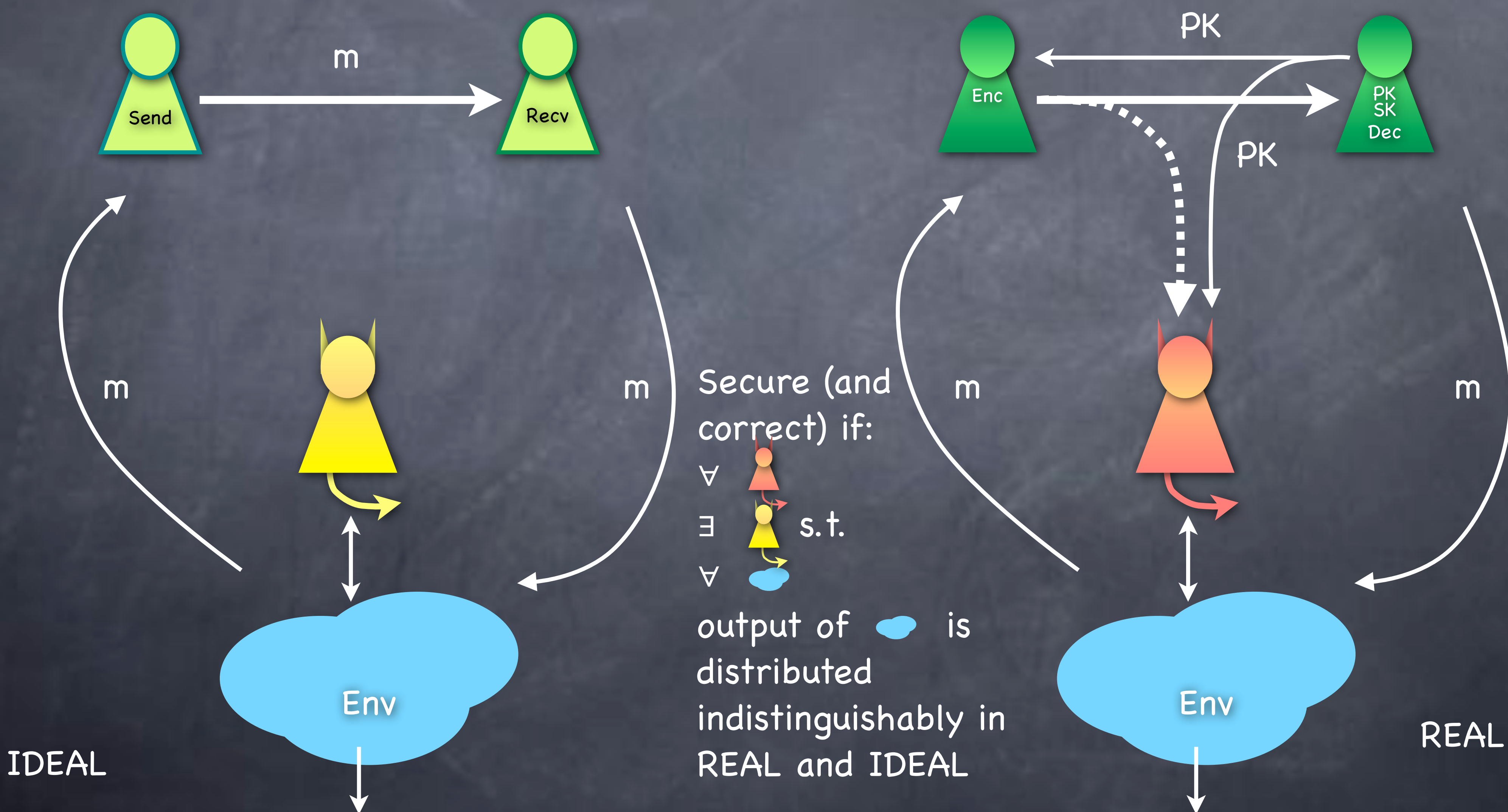
∃ s.t.

∀

output of is distributed indistinguishably in REAL and IDEAL



SIM-CPA (PKE Version)



Secure (and correct) if:

∀

∃ s.t.

∀

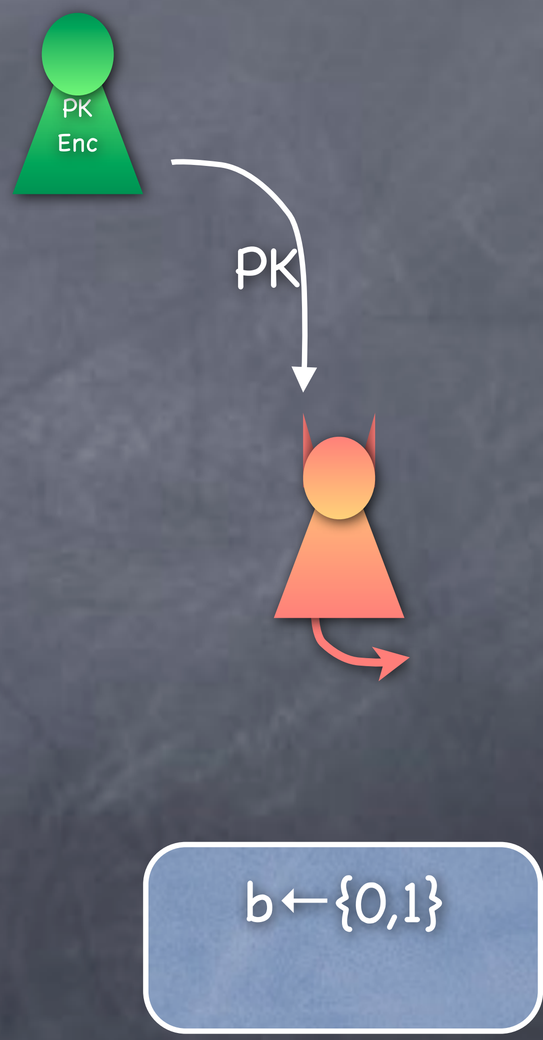
output of is distributed indistinguishably in REAL and IDEAL

IND-CPA (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 $\Pr[b'=b] - 1/2 \leq \nu(k)$

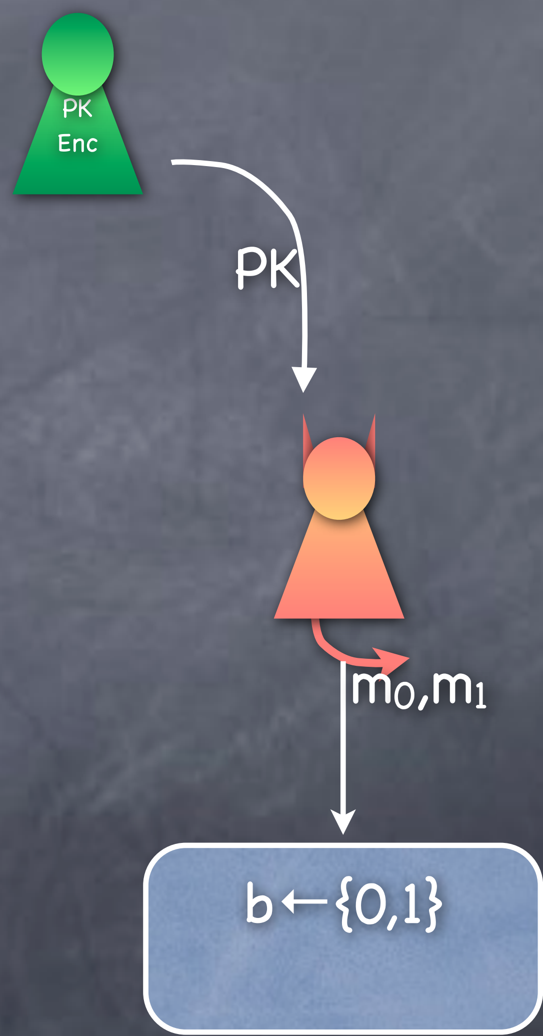
IND-CPA (PKE version)

- 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)



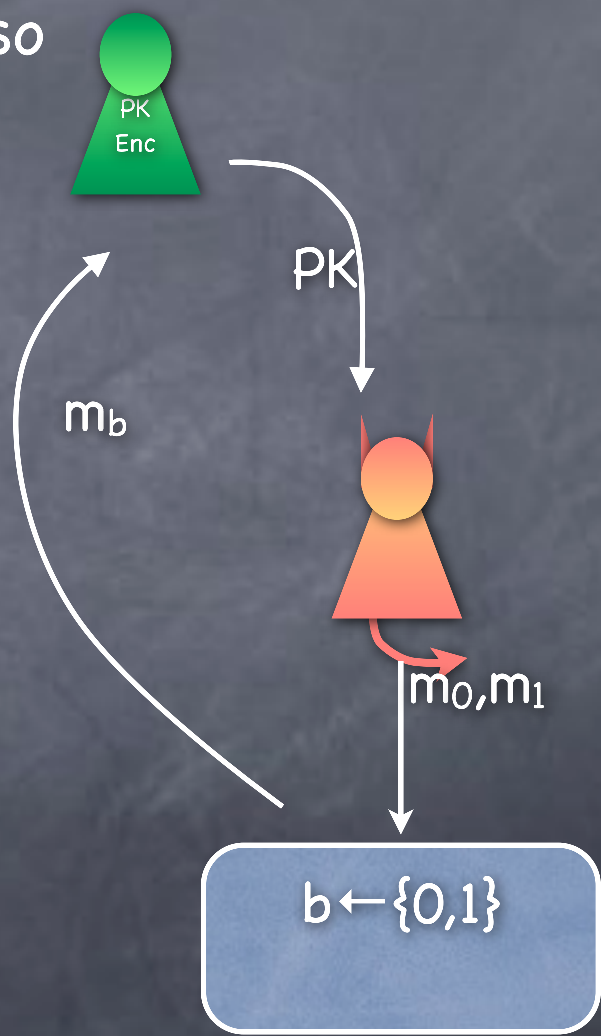
IND-CPA (PKE version)

- 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 ≤ ν(k)



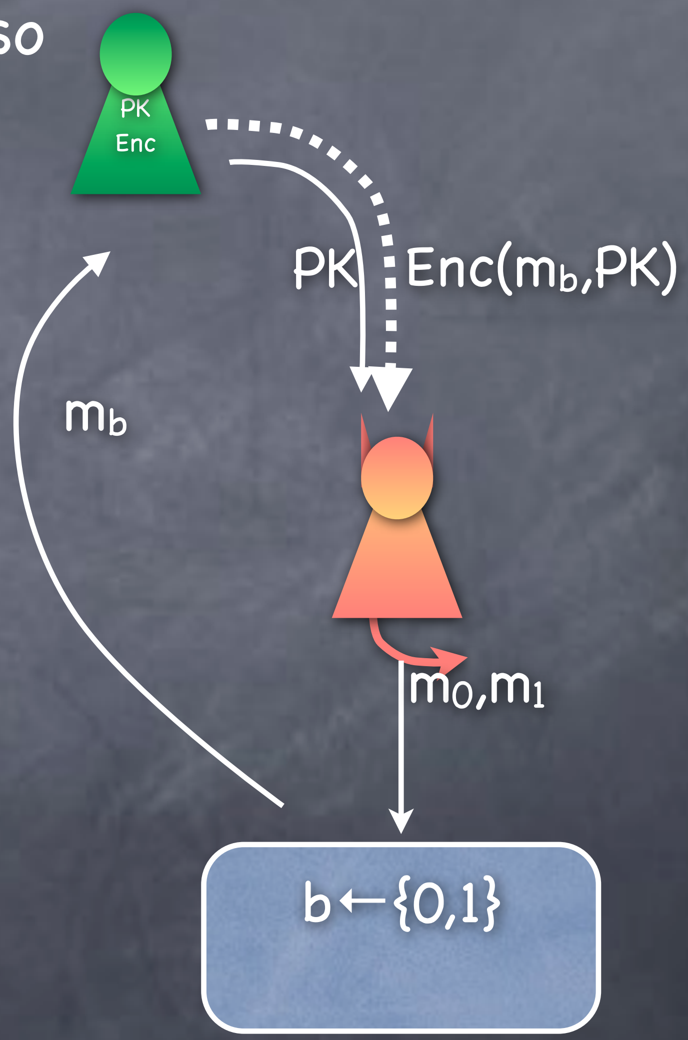
IND-CPA (PKE version)

- 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 ≤ ν(k)



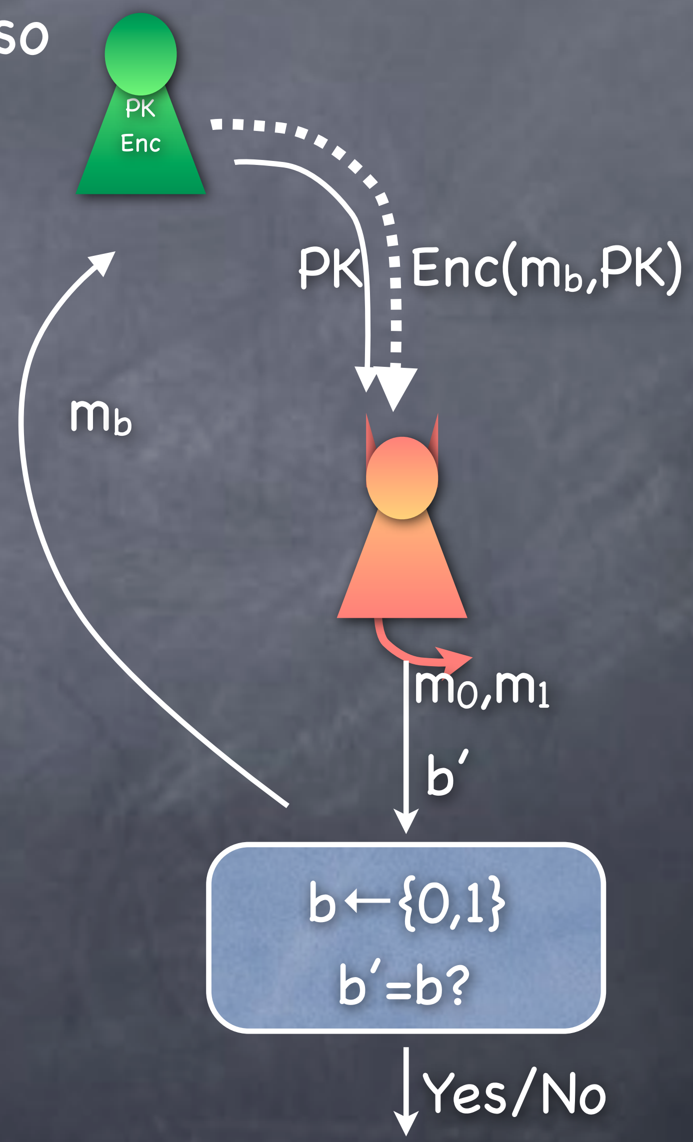
IND-CPA (PKE version)

- 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 ≤ 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 ≤ ν(k)



Perfect Secrecy?

Perfect Secrecy?

- Do we have perfectly secret PKE (even allowing a new key-pair for each message)?

Perfect Secrecy?

- Do we have perfectly secret PKE (even allowing a new key-pair for each message)?
  - 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?

- Do we have perfectly secret PKE (even allowing a new key-pair for each message)?
  - 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?

- Do we have perfectly secret PKE (even allowing a new key-pair for each message)?
  - 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?

- Do we have perfectly secret PKE (even allowing a new key-pair for each message)?
  - 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?

- Do we have perfectly secret PKE (even allowing a new key-pair for each message)?
  - 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?

- Do we have perfectly secret PKE (even allowing a new key-pair for each message)?
  - 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

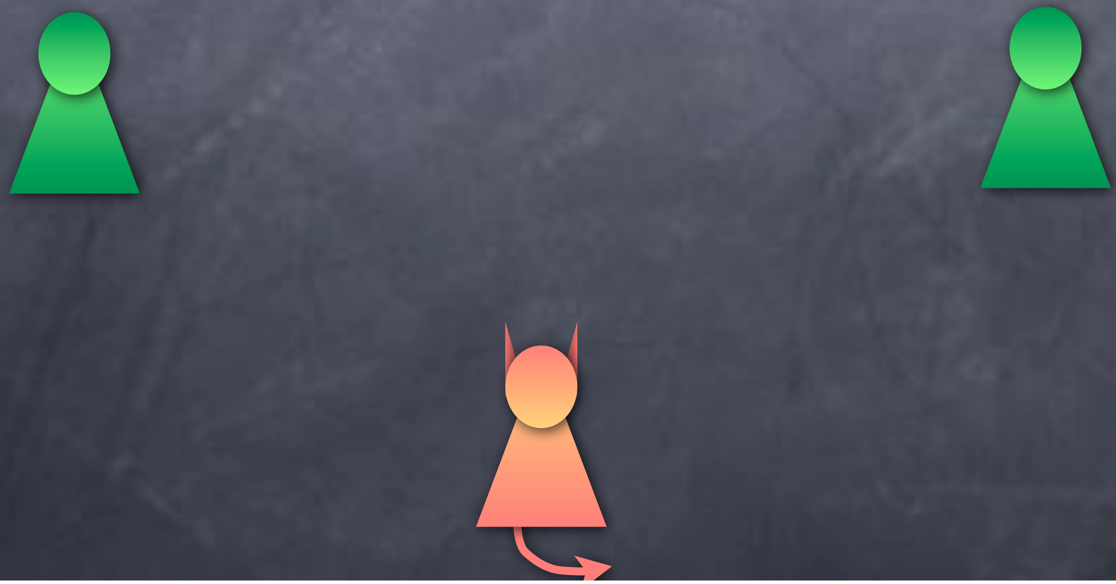
Diffie-Hellman Key-exchange

Diffie-Hellman Key-exchange

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

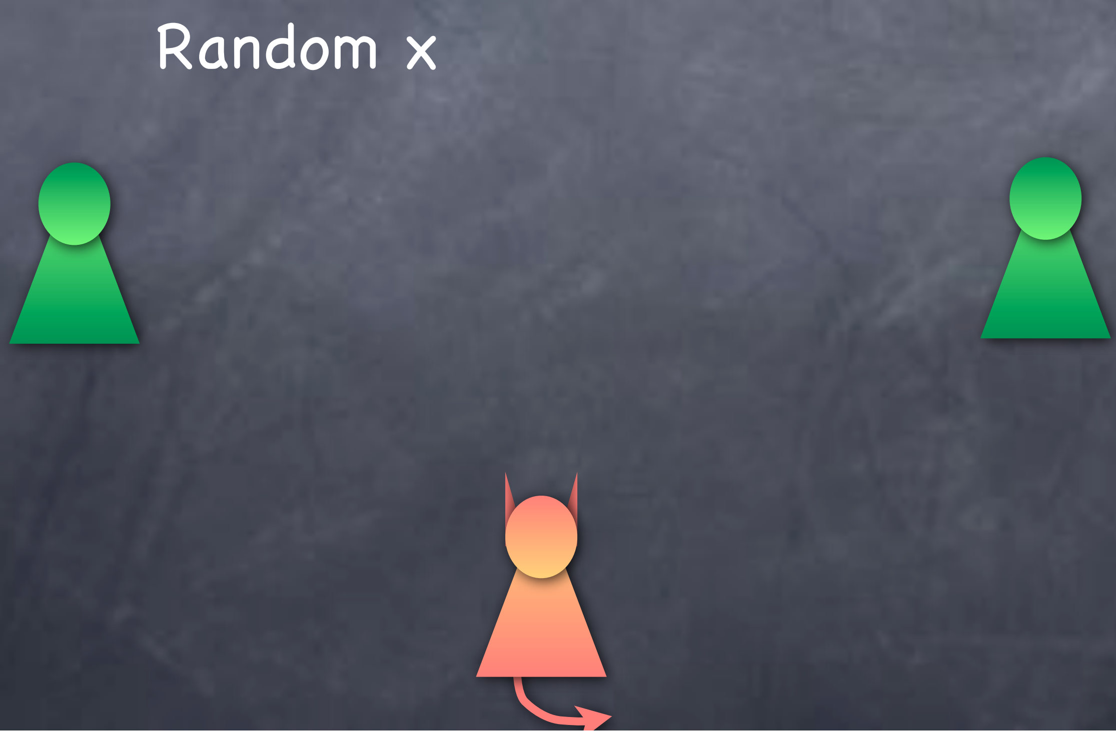
Diffie-Hellman Key-exchange

- 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

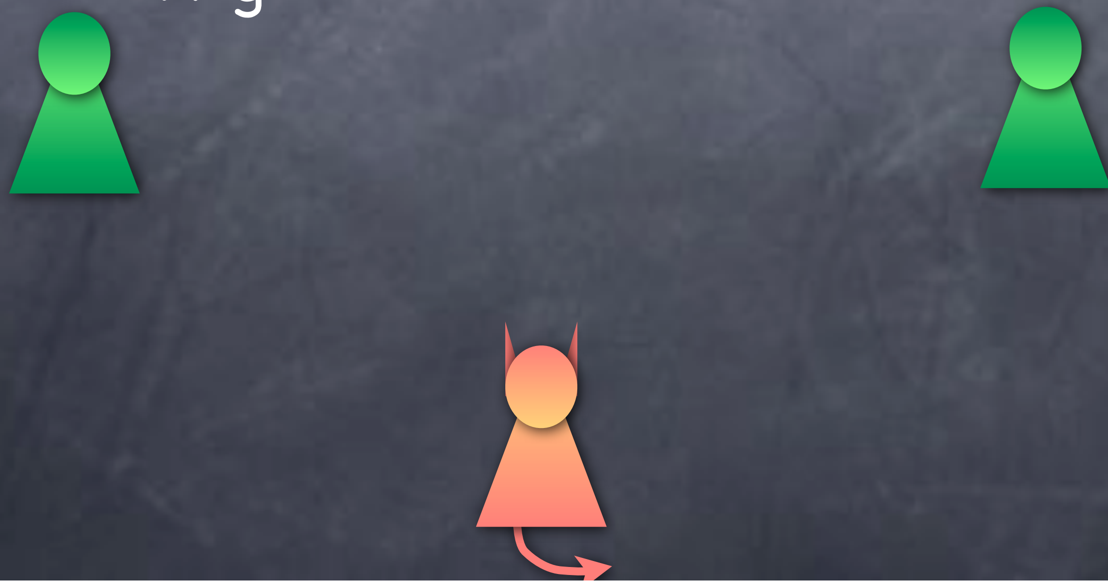


Diffie-Hellman Key-exchange

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

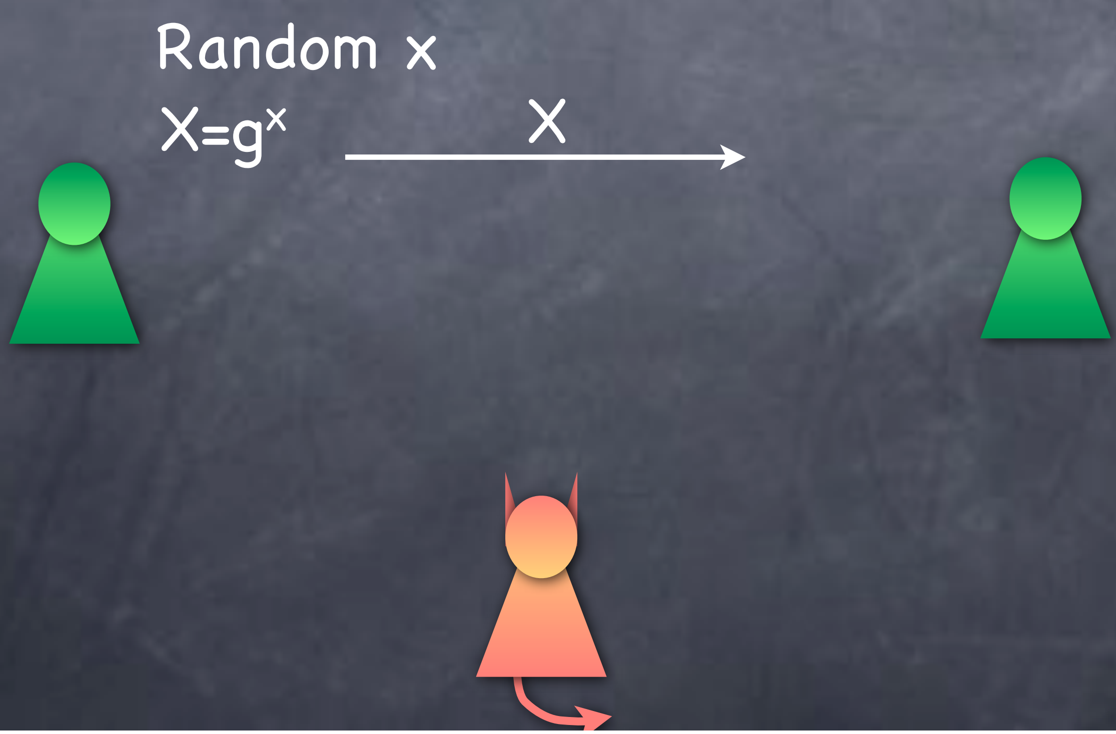
Random x

$X = g^x$



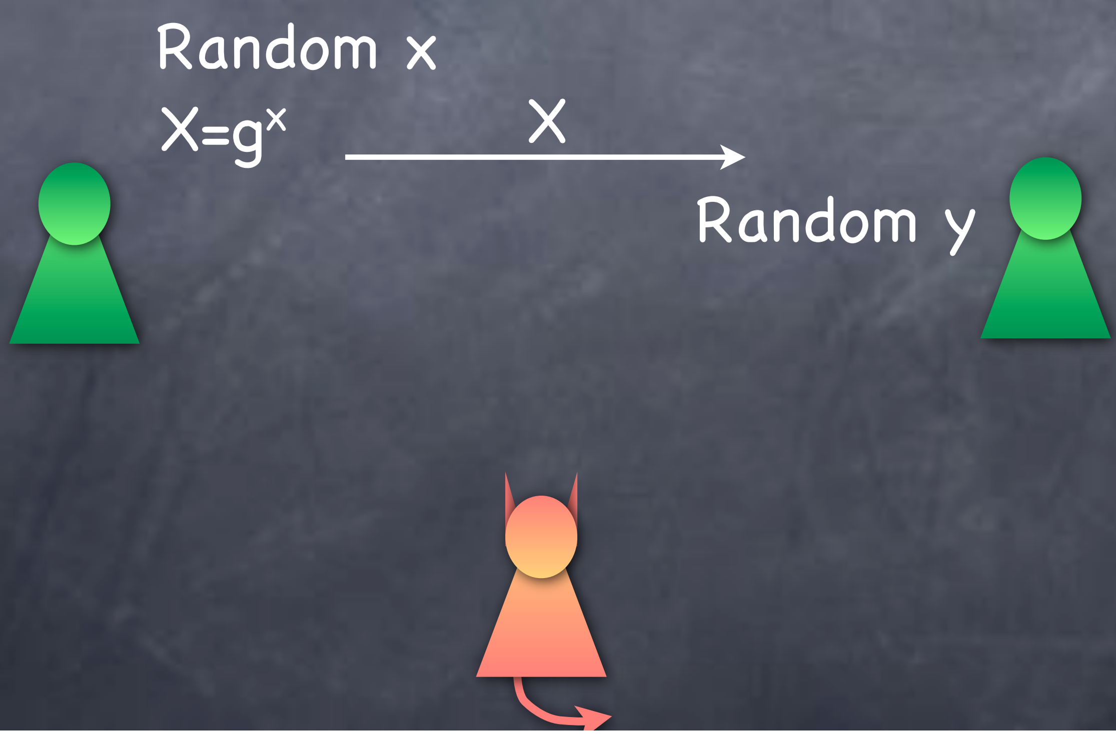
Diffie-Hellman Key-exchange

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



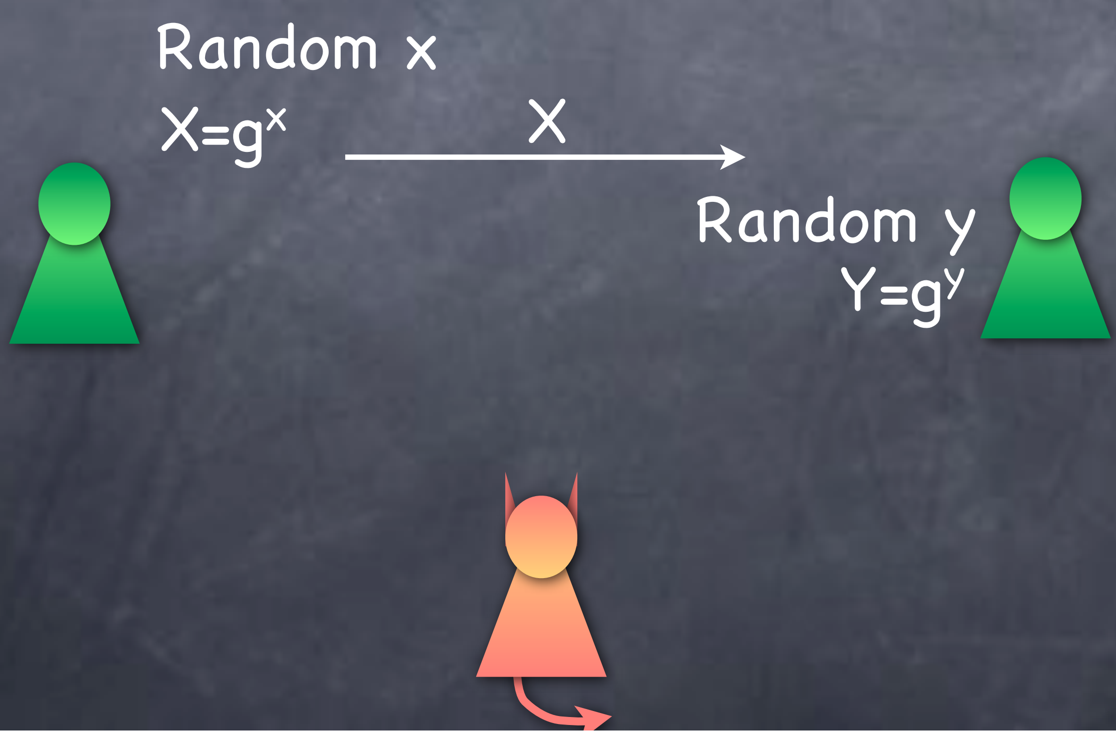
Diffie-Hellman Key-exchange

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



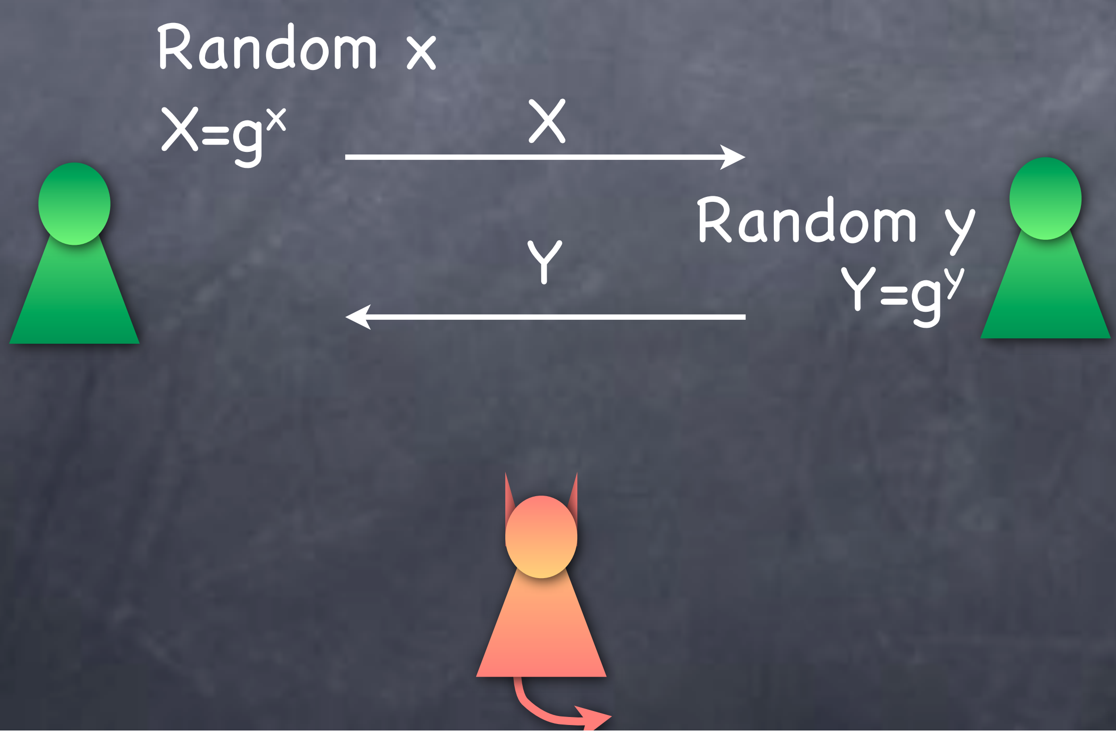
Diffie-Hellman Key-exchange

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



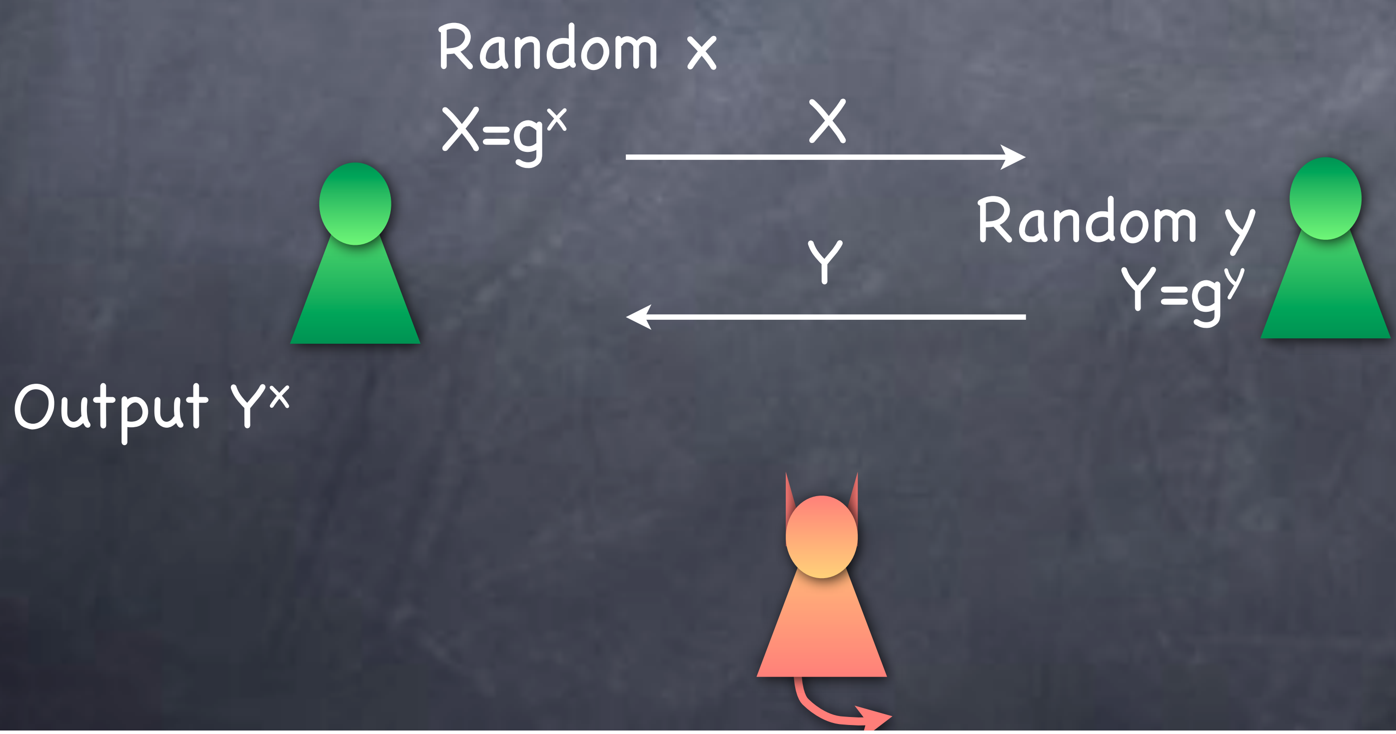
Diffie-Hellman Key-exchange

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



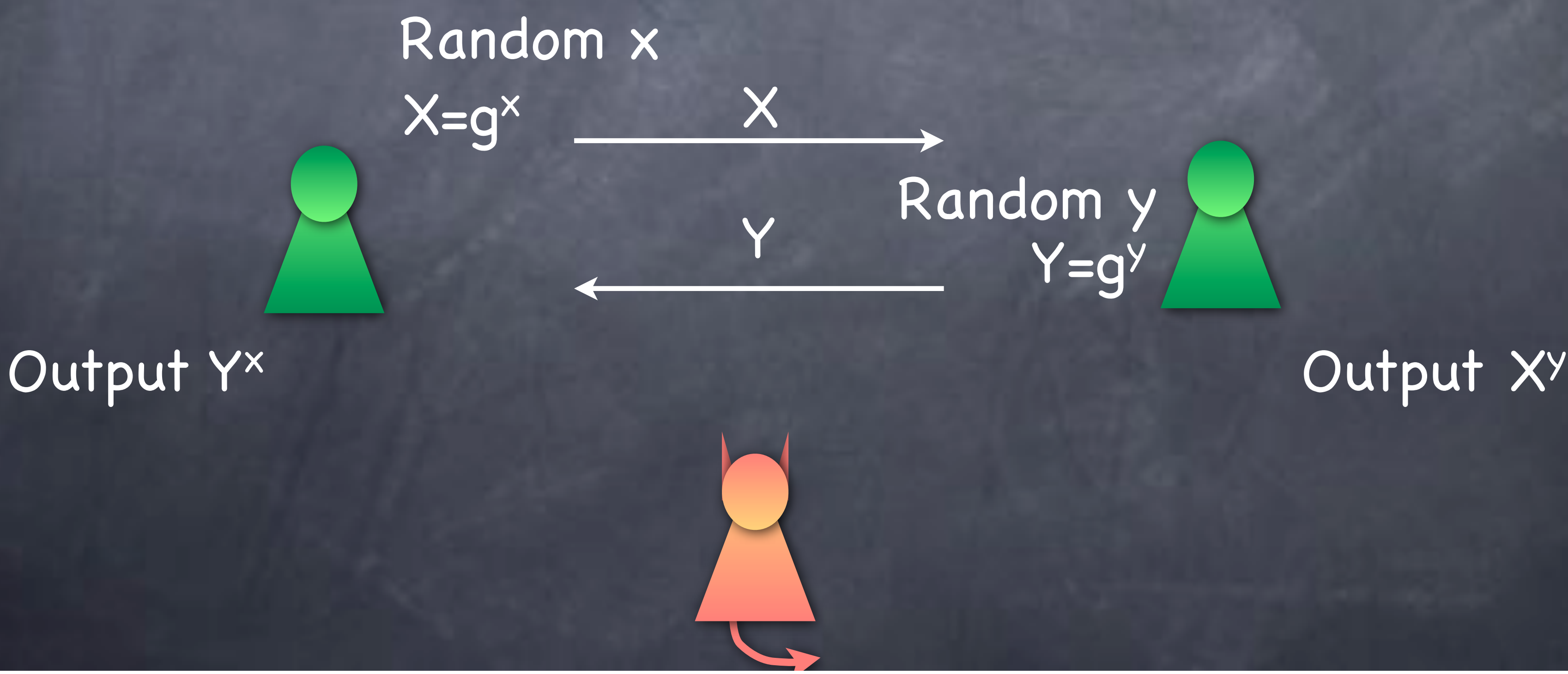
Diffie-Hellman Key-exchange

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



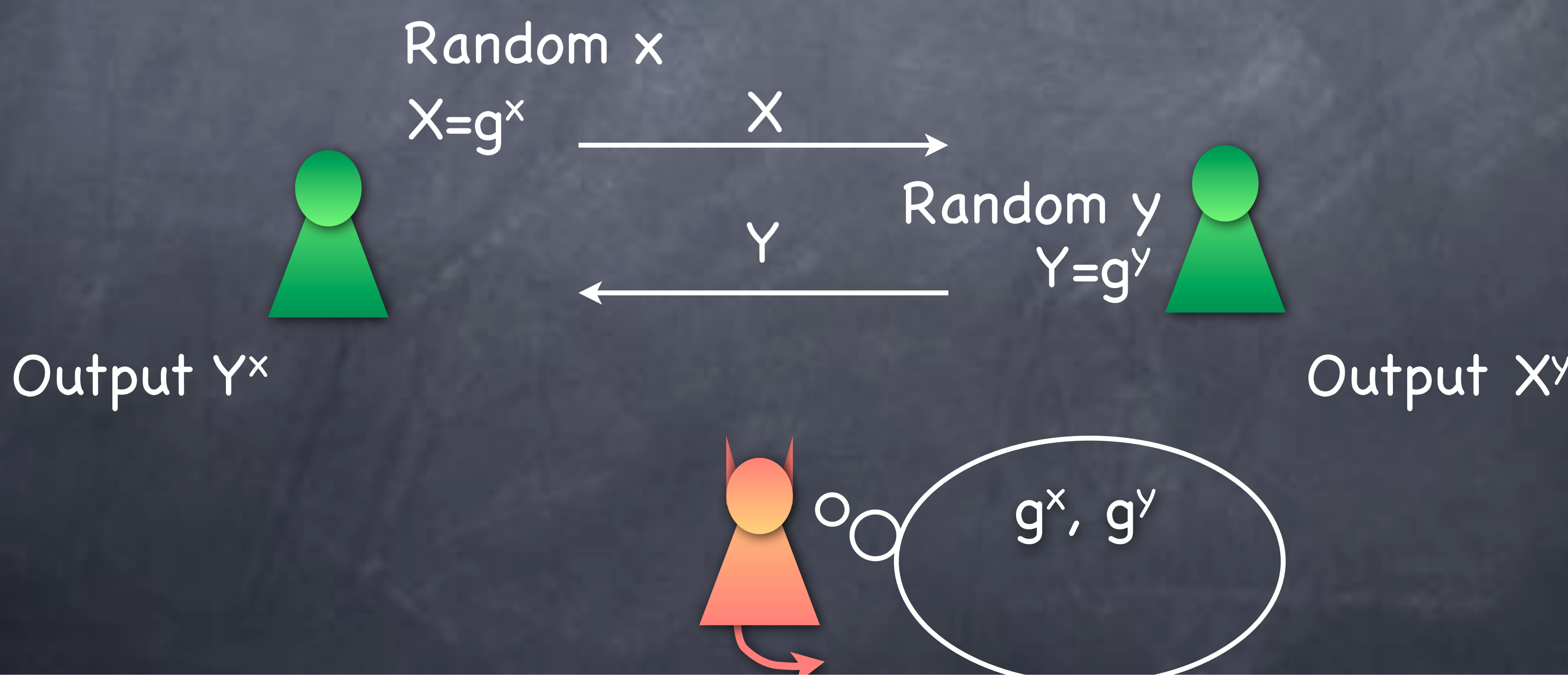
Diffie-Hellman Key-exchange

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



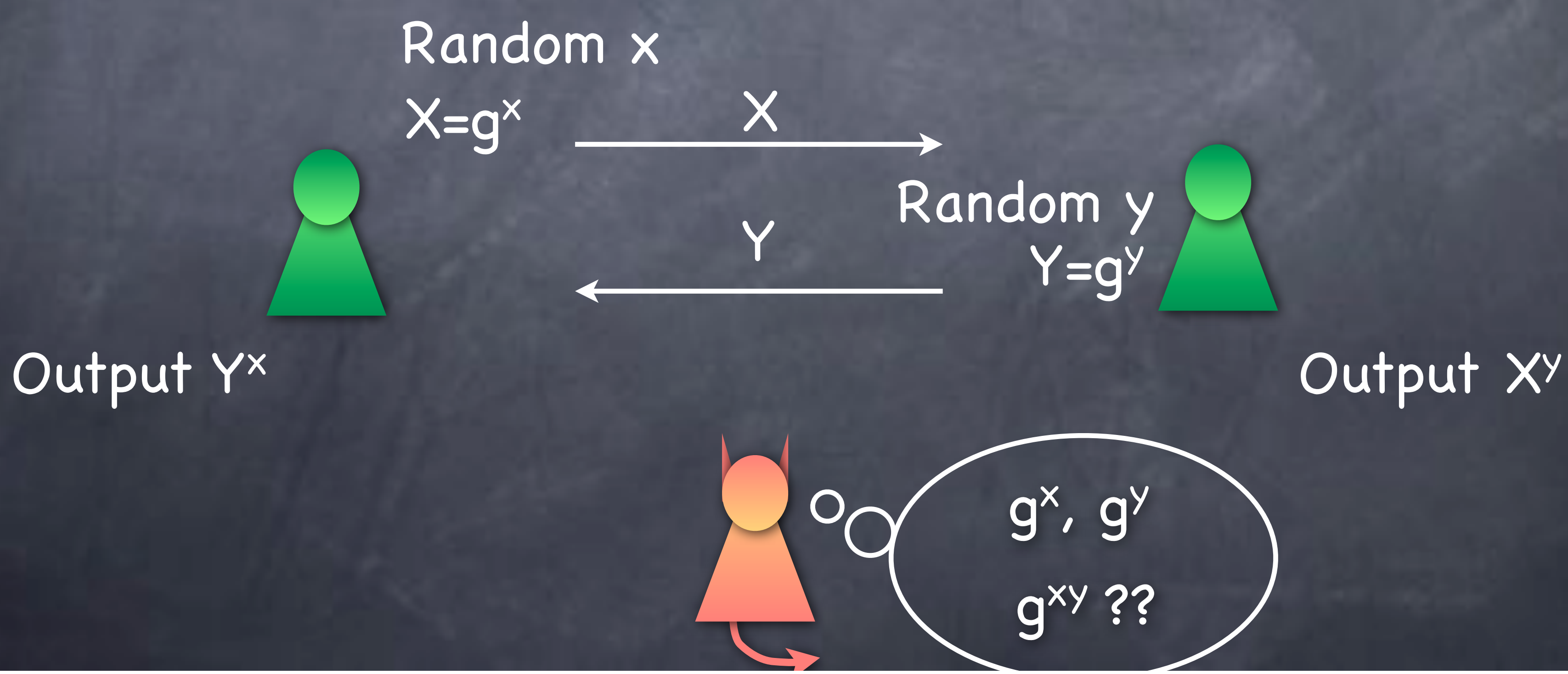
Diffie-Hellman Key-exchange

- 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, with addition operation; infinite group), $\mathbb{Z}_N$ (integers modulo N), G^n (G, a group; coordinate-wise op)

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, with addition operation; infinite group), $\mathbb{Z}_N$ (integers modulo N), G^n (G, a group; coordinate-wise op)
- 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, with addition operation; infinite group), $\mathbb{Z}_N$ (integers modulo N), G^n (G, a group; coordinate-wise op)

- Order of a group G: $|G|$ = number of elements in G

- Cyclic group (finite, 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, with addition operation; infinite group), $\mathbb{Z}_N$ (integers modulo N), G^n (G, a group; coordinate-wise op)

- Order of a group G: $|G|$ = number of elements in G

- Cyclic group (finite, 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, with addition operation; infinite group), $\mathbb{Z}_N$ (integers modulo N), G^n (G, a group; coordinate-wise op)

- Order of a group G: $|G|$ = number of elements in G

- Cyclic group (finite, 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, with addition operation; infinite group), $\mathbb{Z}_N$ (integers modulo N), G^n (G, a group; coordinate-wise op)

- Order of a group G: $|G|$ = number of elements in G

- Cyclic group (finite, 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, with addition operation; infinite group), $\mathbb{Z}_N$ (integers modulo N), G^n (G, a group; coordinate-wise op)

- Order of a group G: $|G|$ = number of elements in G

- Cyclic group (finite, 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, with addition operation; infinite group), $\mathbb{Z}_N$ (integers modulo N), G^n (G, a group; coordinate-wise op)

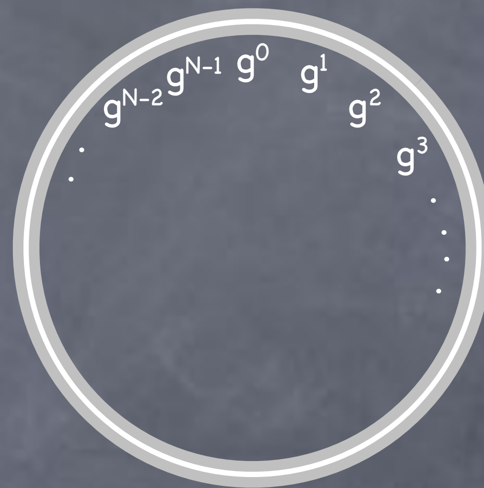
- Order of a group G: $|G|$ = number of elements in G

- Cyclic group (finite, in multiplicative notation): there is one element g such that $G = \{g^0, g^1, g^2, \ldots g^{|G|-1}\}$
  - e.g. $\mathbb{Z}_N$, with say g=1 (additive group)

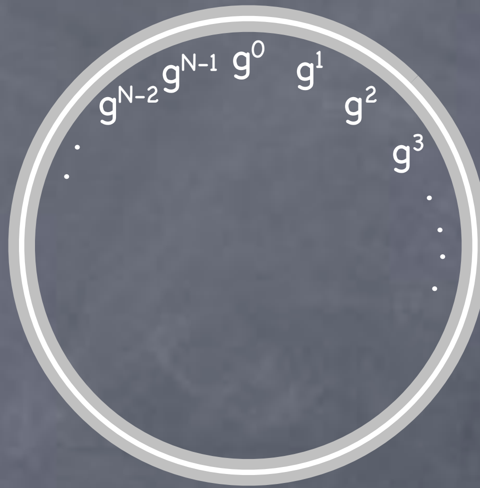
Groups, by examples

- 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, with addition operation; infinite group), $\mathbb{Z}_N$ (integers modulo N), G^n (G, a group; coordinate-wise op)
- Order of a group G: $|G|$ = number of elements in G
- Cyclic group (finite, in multiplicative notation): there is one element g such that $G = \{g^0, g^1, g^2, \ldots g^{|G|-1}\}$
  - e.g. $\mathbb{Z}_N$, with say g=1 (additive group)
    - 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)

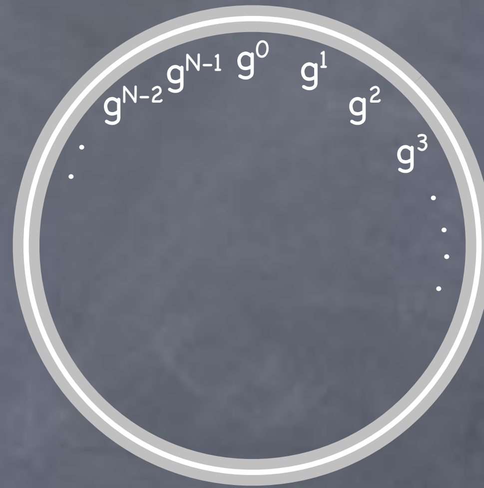
Groups, by examples

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

Groups, by examples

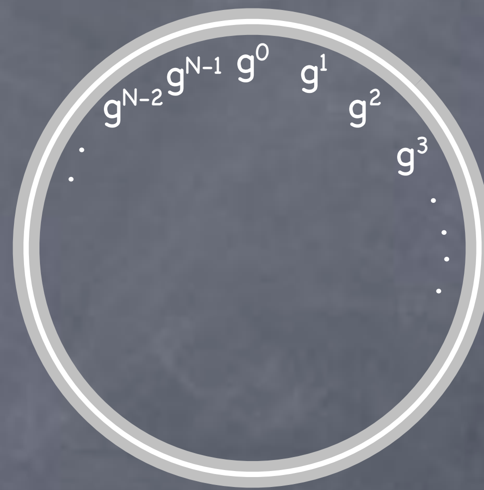
- $\mathbb{Z}_N^*$ = (generators of $\mathbb{Z}_N$, multiplication mod N)
  - Numbers which have multiplicative inverse mod N
  - If N is prime, $\mathbb{Z}_N^*$ is a cyclic group, of order N-1

Groups, by examples



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

Groups, by examples



- $\mathbb{Z}_N^*$ = (generators of $\mathbb{Z}_N$, multiplication mod N)
  - Numbers which have 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})
    - (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 (computational) group, given standard representation of g and x, can efficiently find the standard representation of $X=g^x$ (How?)

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 (computational) group, given standard representation of g and x, 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

- 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 (computational) group, given standard representation of g and x, 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; $X \leftarrow G$; $Adv(G,g,X) \rightarrow z$; $g^z=X$?

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 (computational) group, given standard representation of g and x, 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 Eve could break DLA, then Diffie-Hellman key-exchange broken

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 (computational) group, given standard representation of g and x, 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 Eve could break DLA, 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

- 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 (computational) group, given standard representation of g and x, 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 Eve could break DLA, 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

- 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 (computational) group, given standard representation of g and x, 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; $X \leftarrow G$; $\mathrm{Adv}(G,g,X) \rightarrow z$; $g^z=X$?
- If Eve could break DLA, 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
  - But could break pseudorandomness without breaking DLA too

Decisional Diffie-Hellman (DDH) Assumption

Decisional Diffie-Hellman (DDH) Assumption

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

Decisional Diffie-Hellman (DDH) Assumption

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

Decisional Diffie-Hellman (DDH) Assumption

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

Decisional Diffie-Hellman (DDH) Assumption

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

Decisional Diffie-Hellman (DDH) Assumption

- $\{(g^x, g^y, g^{xy})\}_{(G,g)\leftarrow\text{GroupGen};\ x,y\leftarrow[|G|]} \approx \{(g^x, g^y, g^r)\}_{(G,g)\leftarrow\text{GroupGen};\ x,y,r\leftarrow[|G|]}$
- At least as strong as DLA
  - If DDH assumption holds, then DLA holds [Exercise]
- 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" elements) of $\mathbb{Z}_p^*$

A Candidate DDH Group

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

A Candidate DDH Group

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

El Gamal Encryption

El Gamal Encryption

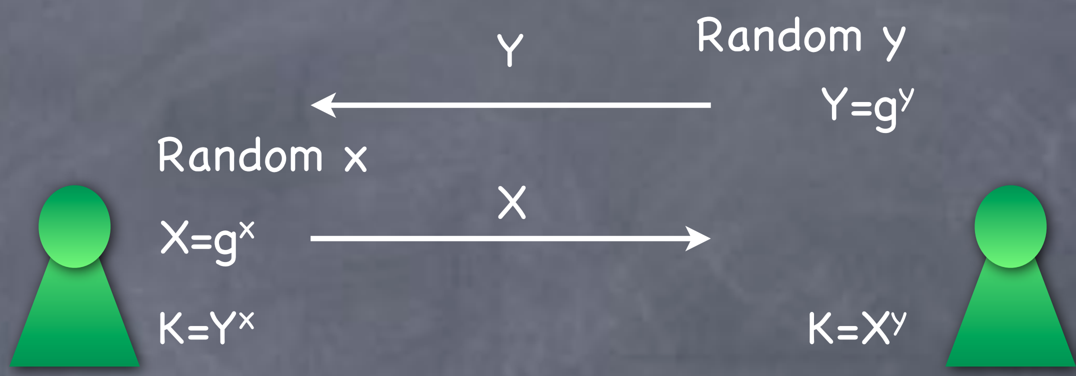
- Based on DH key-exchange

El Gamal Encryption

- Based on DH key-exchange
  - Alice, Bob generate a key using DH key-exchange

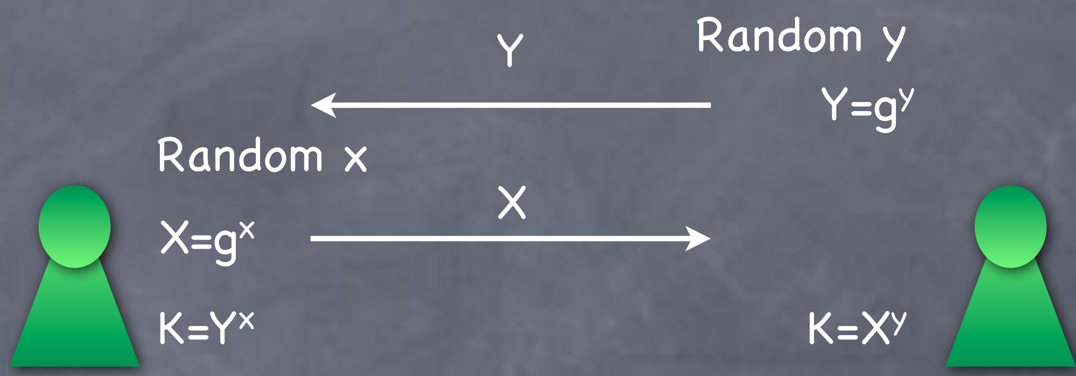
El Gamal Encryption

- Based on DH key-exchange
  - Alice, Bob generate a key using DH key-exchange



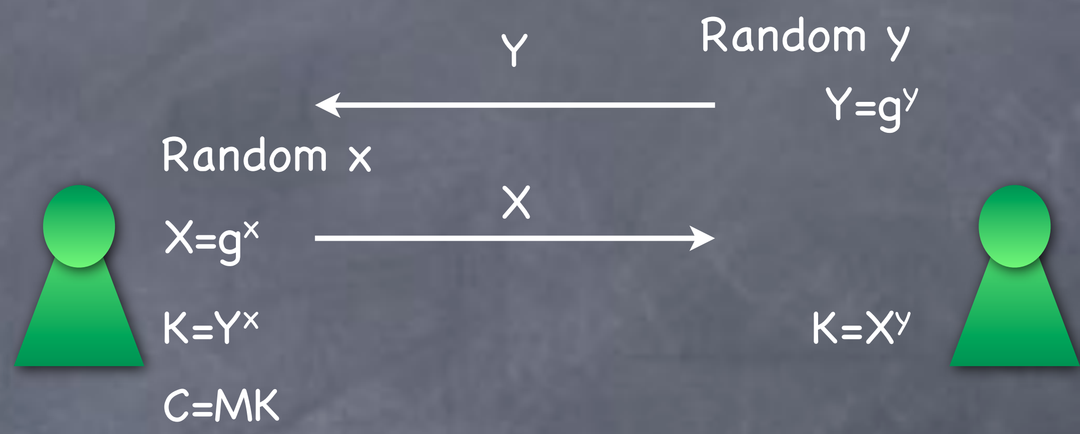
El Gamal Encryption

- Based on DH key-exchange
  - Alice, Bob generate a key using DH key-exchange
  - Then use it as a one-time pad



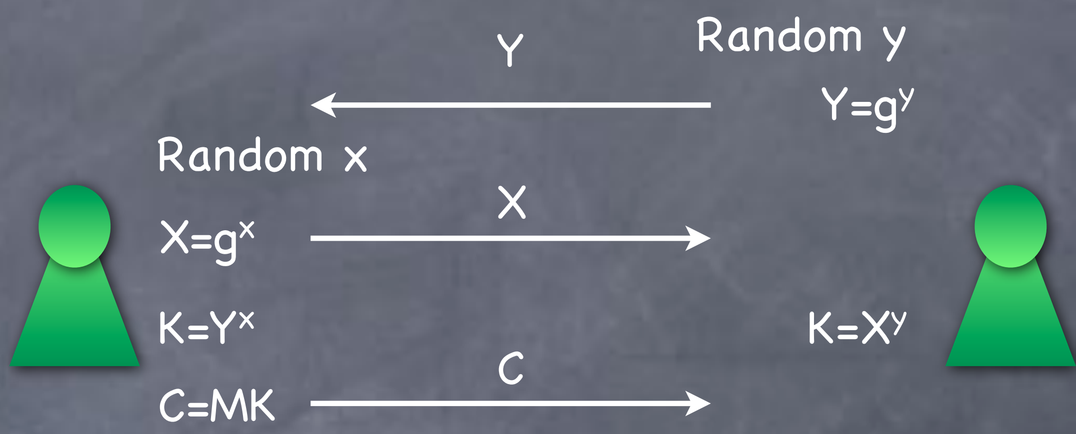
El Gamal Encryption

- Based on DH key-exchange
  - Alice, Bob generate a key using DH key-exchange
  - Then use it as a one-time pad



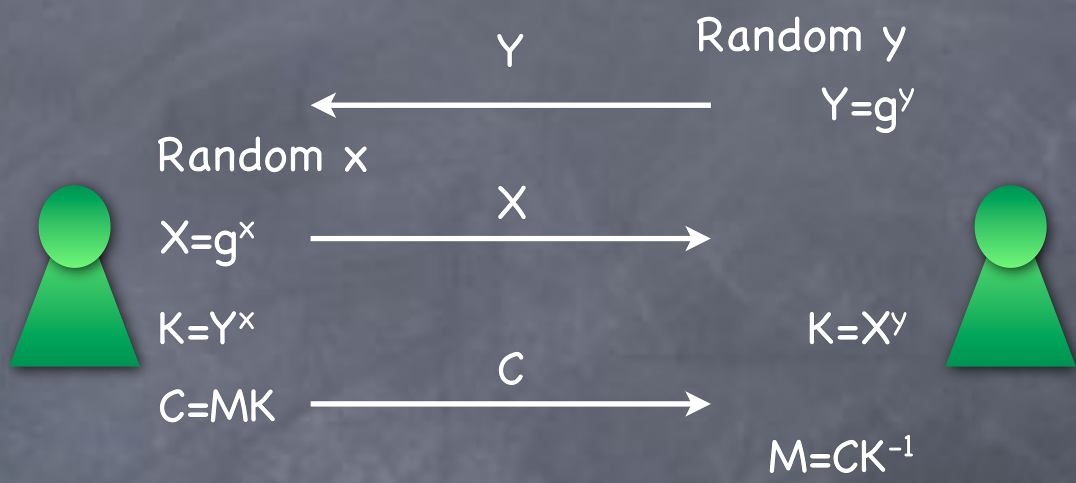
El Gamal Encryption

- Based on DH key-exchange
  - Alice, Bob generate a key using DH key-exchange
  - Then use it as a one-time pad



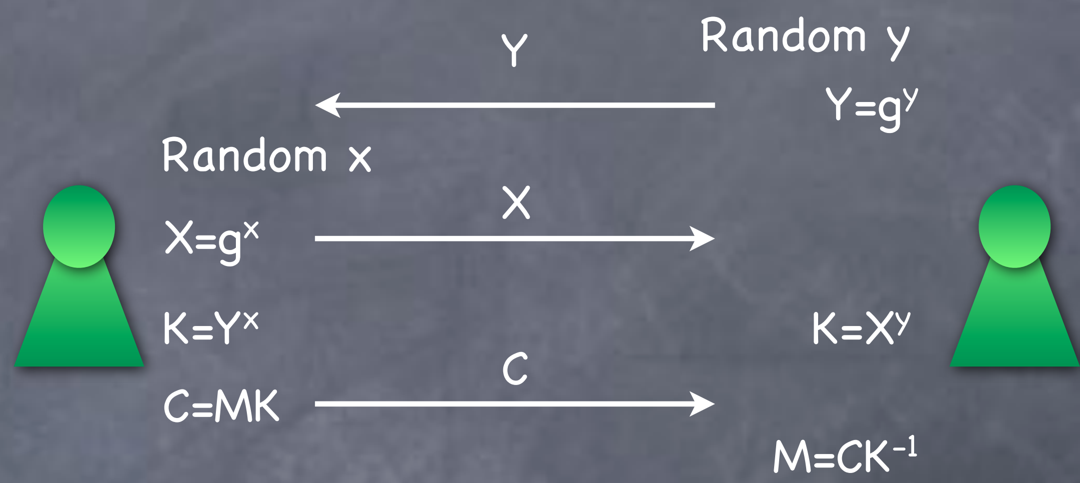
El Gamal Encryption

- Based on DH key-exchange
  - Alice, Bob generate a key using DH key-exchange
  - Then use it as a one-time pad



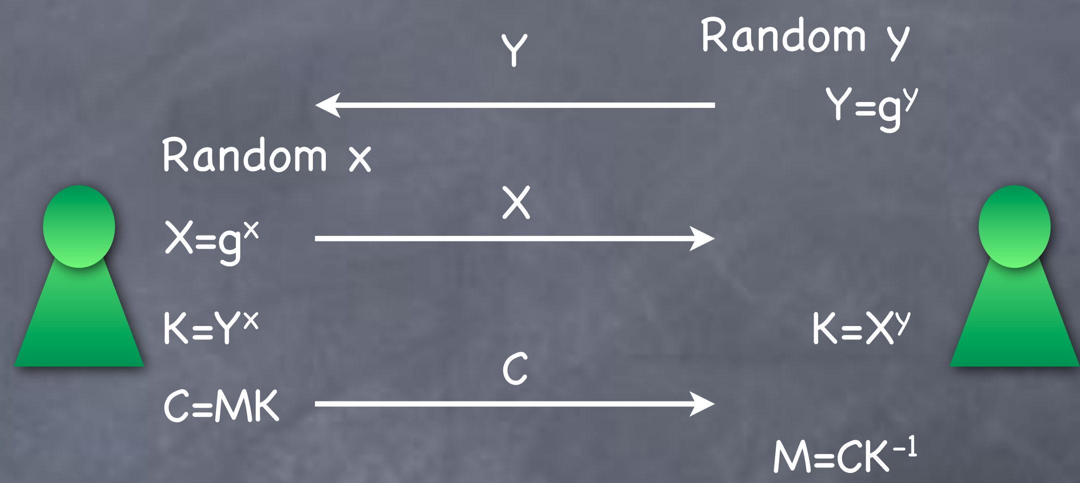
El Gamal Encryption

- Based on DH key-exchange
  - Alice, Bob generate a key using DH key-exchange
  - Then use it as a one-time pad
- Bob's "message" in the key-exchange is his PK



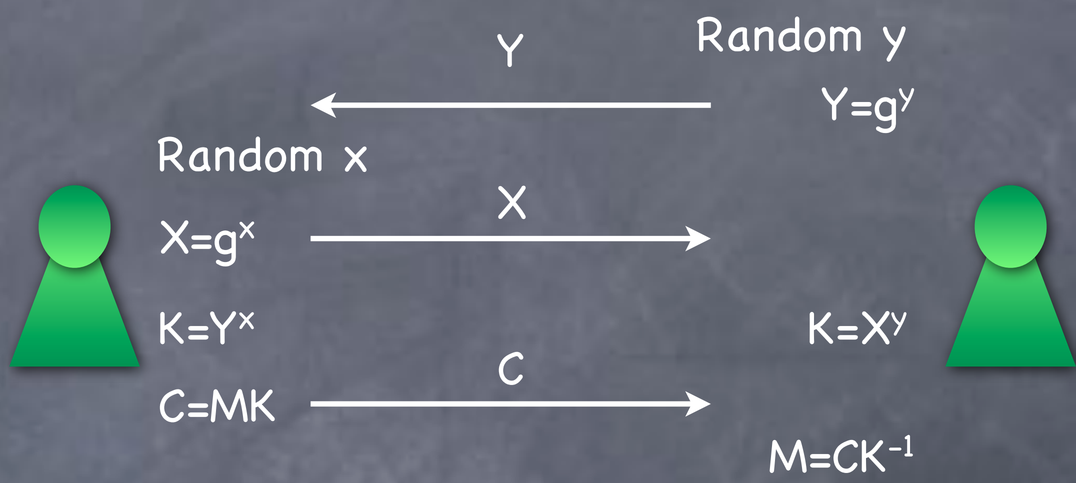
El Gamal Encryption

- Based on DH key-exchange
  - Alice, Bob generate a key using DH key-exchange
  - Then use it as a one-time pad
- Bob's "message" in the key-exchange is his PK
- Alice's message in the key-exchange and the ciphertext of the one-time pad together form a single ciphertext



El Gamal Encryption

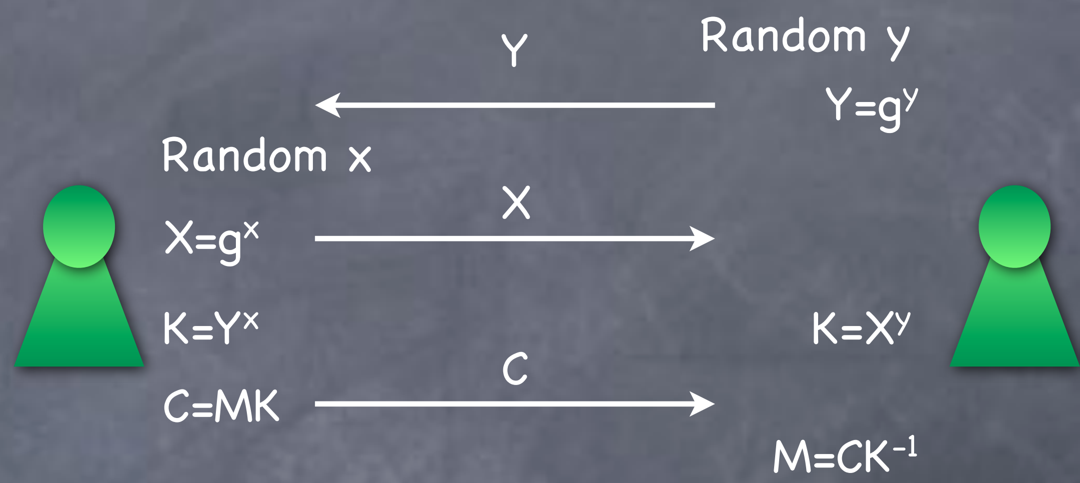
- Based on DH key-exchange
  - Alice, Bob generate a key using DH key-exchange
  - Then use it as a one-time pad
- Bob's "message" in the key-exchange is his PK
- Alice's message in the key-exchange and the ciphertext of the one-time pad together form a single ciphertext



KeyGen: PK=(G,g,Y), SK=(G,g,y)

El Gamal Encryption

- Based on DH key-exchange
  - Alice, Bob generate a key using DH key-exchange
  - Then use it as a one-time pad
- Bob's "message" in the key-exchange is his PK
- Alice's message in the key-exchange and the ciphertext of the one-time pad together form a single ciphertext

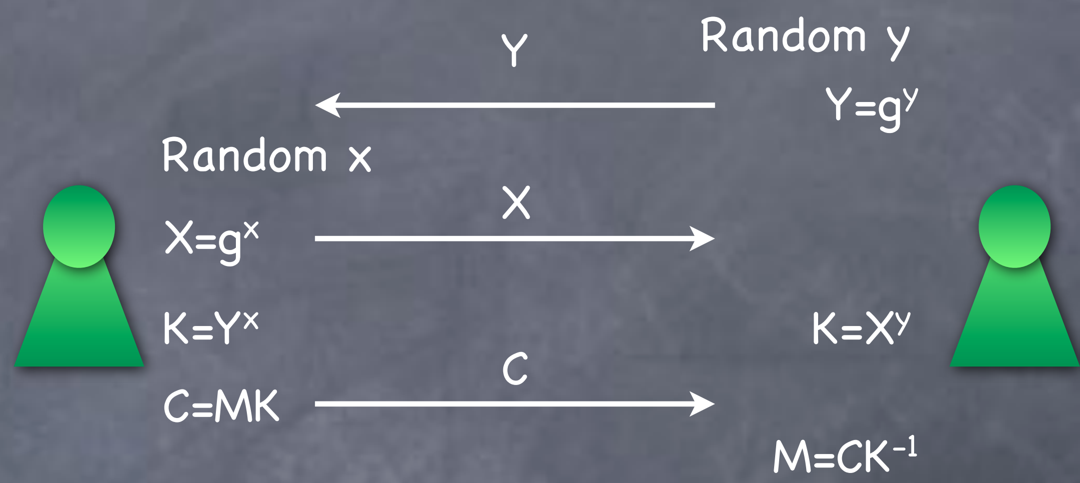


KeyGen: PK=(G,g,Y), SK=(G,g,y)

$\mathrm{Enc}_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

El Gamal Encryption

- Based on DH key-exchange
  - Alice, Bob generate a key using DH key-exchange
  - Then use it as a one-time pad
- Bob's "message" in the key-exchange is his PK
- Alice's message in the key-exchange and the ciphertext of the one-time pad together form a single ciphertext



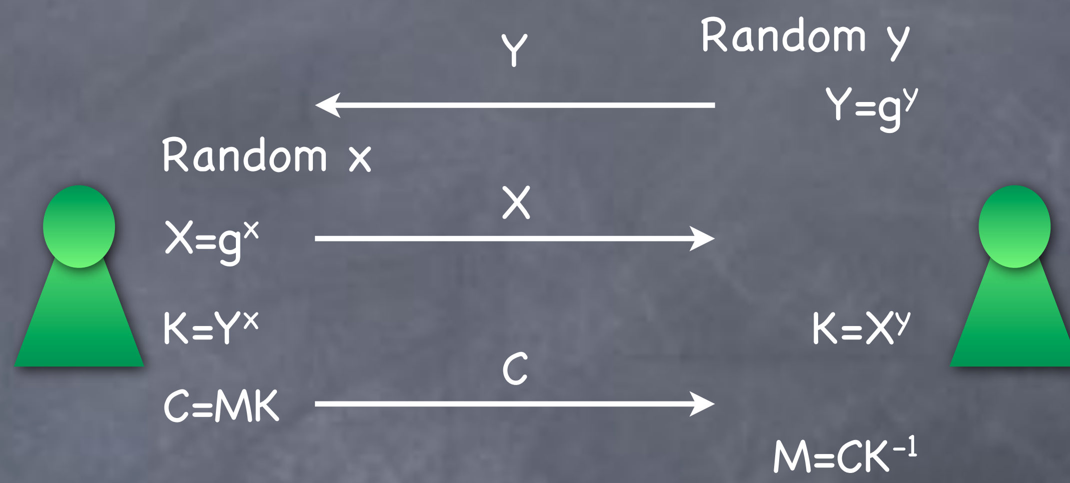
KeyGen: $PK=(G,g,Y)$, $SK=(G,g,y)$

$Enc_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

$Dec_{(G,g,y)}(X,C) = CX^{-y}$

El Gamal Encryption

- Based on DH key-exchange
  - Alice, Bob generate a key using DH key-exchange
  - Then use it as a one-time pad
- Bob's "message" in the key-exchange is his PK
- Alice's message in the key-exchange and the ciphertext of the one-time pad together form a single ciphertext



KeyGen: PK=(G,g,Y), SK=(G,g,y)

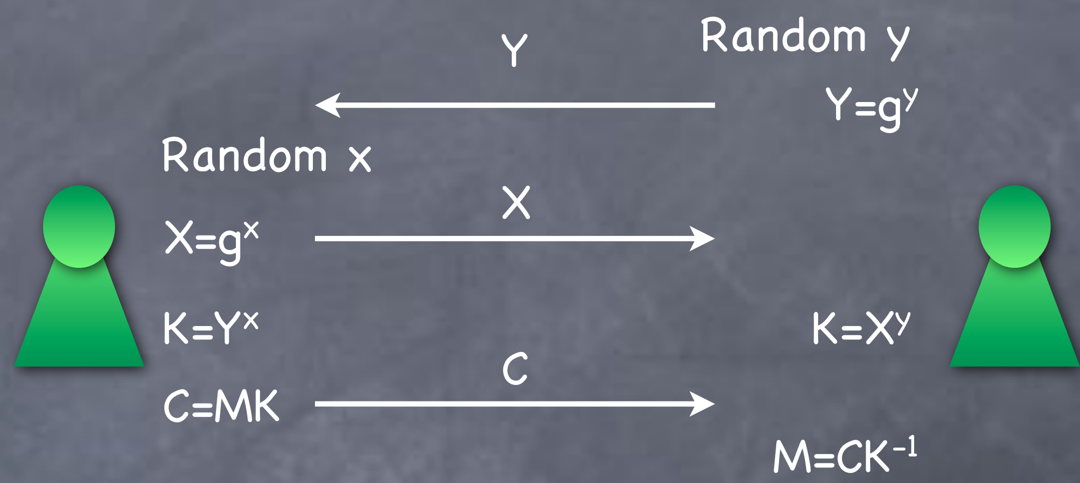
$Enc_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

$Dec_{(G,g,y)}(X,C) = CX^{-y}$

- KeyGen uses GroupGen to get (G,g)

El Gamal Encryption

- Based on DH key-exchange
  - Alice, Bob generate a key using DH key-exchange
  - Then use it as a one-time pad
- Bob's "message" in the key-exchange is his PK
- Alice's message in the key-exchange and the ciphertext of the one-time pad together form a single ciphertext



KeyGen: PK=(G,g,Y), SK=(G,g,y)

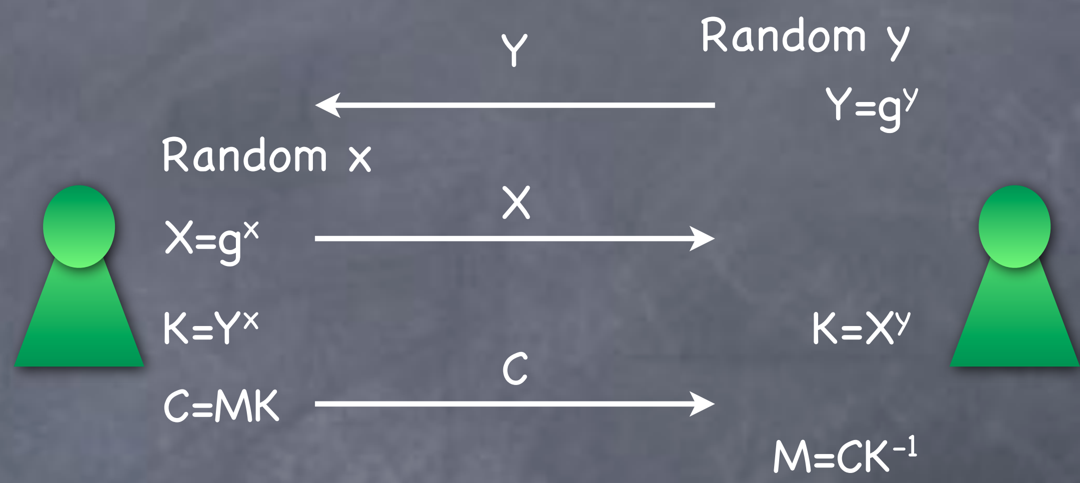
$\mathrm{Enc}_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

$\mathrm{Dec}_{(G,g,y)}(X,C) = CX^{-y}$

- KeyGen uses GroupGen to get (G,g)
- x, y uniform from [|G|]

El Gamal Encryption

- Based on DH key-exchange
  - Alice, Bob generate a key using DH key-exchange
  - Then use it as a one-time pad
- Bob's "message" in the key-exchange is his PK
- Alice's message in the key-exchange and the ciphertext of the one-time pad together form a single ciphertext



KeyGen: PK=(G,g,Y), SK=(G,g,y)

$\mathrm{Enc}_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

$\mathrm{Dec}_{(G,g,y)}(X,C) = CX^{-y}$

- KeyGen uses GroupGen to get (G,g)
- x, y uniform from [|G|]
- Message encoded into group element, and decoded

Security of El Gamal

Security of El Gamal

- El Gamal IND-CPA secure if DDH holds (for the collection of groups used)

Security of El Gamal

- El Gamal IND-CPA secure if DDH holds (for the collection of groups used)
  - Construct a DDH adversary A* given an IND-CPA adversary A

Security of El Gamal

- El Gamal IND-CPA secure if DDH holds (for the collection of groups used)
  - Construct a DDH adversary A* given an IND-CPA adversary A
  - A*(G,g; g^x,g^y,g^z) (where (G,g) ← GroupGen, x,y random and z=xy or random) plays the IND-CPA experiment with A:

Security of El Gamal

- El Gamal IND-CPA secure if DDH holds (for the collection of groups used)
  - Construct a DDH adversary A* given an IND-CPA adversary A
  - A*(G,g; g^x,g^y,g^z) (where (G,g) ← GroupGen, x,y random and z=xy or random) plays the IND-CPA experiment with A:
    - But sets PK=(G,g,g^y) and Enc(M_b)=(g^x,$M_b g^z$)

Security of El Gamal

- El Gamal IND-CPA secure if DDH holds (for the collection of groups used)
  - Construct a DDH adversary A* given an IND-CPA adversary A
  - $A^*(G,g; g^x,g^y,g^z)$ (where $(G,g) \leftarrow$ GroupGen, x,y random and z=xy or random) plays the IND-CPA experiment with A:
    - But sets $PK=(G,g,g^y)$ and $Enc(M_b)=(g^x,M_b g^z)$
    - Outputs 1 if experiment outputs 1 (i.e. if b=b')

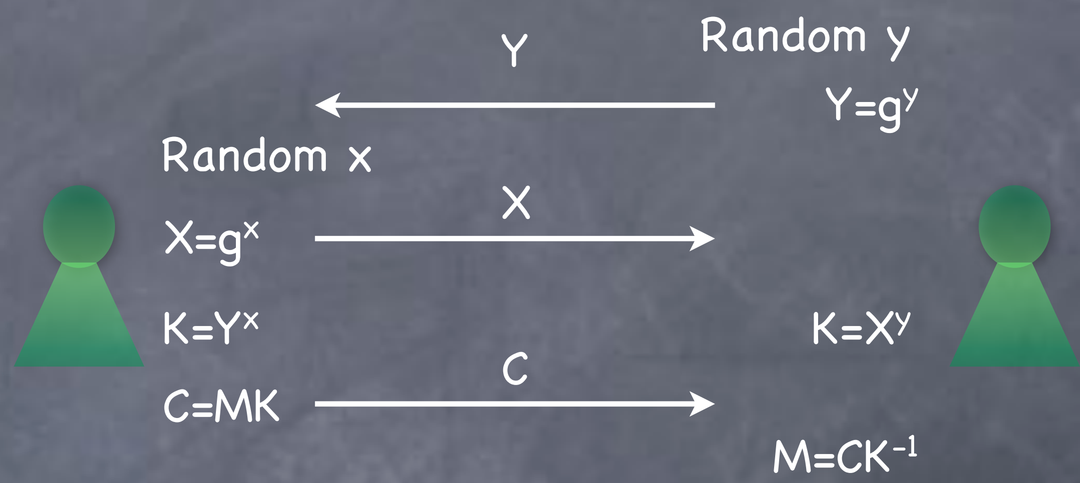
Security of El Gamal

- El Gamal IND-CPA secure if DDH holds (for the collection of groups used)
  - Construct a DDH adversary A* given an IND-CPA adversary A
  - $A^*(G,g;\ g^x,g^y,g^z)$ (where $(G,g) \leftarrow$ GroupGen, x,y random and z=xy or random) plays the IND-CPA experiment with A:
    - But sets $PK=(G,g,g^y)$ and $Enc(M_b)=(g^x,M_b g^z)$
    - Outputs 1 if experiment outputs 1 (i.e. if b=b')
  - When z=random, A* outputs 1 with probability = 1/2

Security of El Gamal

- El Gamal IND-CPA secure if DDH holds (for the collection of groups used)
  - Construct a DDH adversary A* given an IND-CPA adversary A
  - A*(G,g; g^x,g^y,g^z) (where (G,g) ← GroupGen, x,y random and z=xy or random) plays the IND-CPA experiment with A:
    - But sets PK=(G,g,g^y) and Enc(M_b)=(g^x,$M_b g^z$)
    - Outputs 1 if experiment outputs 1 (i.e. if b=b')
  - When z=random, A* outputs 1 with probability = 1/2
  - When z=xy, exactly IND-CPA experiment: A* outputs 1 with probability = 1/2 + advantage of A.

Abstracting El Gamal



KeyGen: PK=(G,g,Y), SK=(G,g,y)

$\mathrm{Enc}_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

$\mathrm{Dec}_{(G,g,y)}(X,C) = CX^{-y}$

Abstracting El Gamal

Random y

$Y=g^y$

$\xleftarrow{Y}$

Random x

$X=g^x$ $\xrightarrow{X}$

$K=Y^x$ $\qquad$ $K=X^y$

$C=MK$ $\xrightarrow{C}$

$M=CK^{-1}$

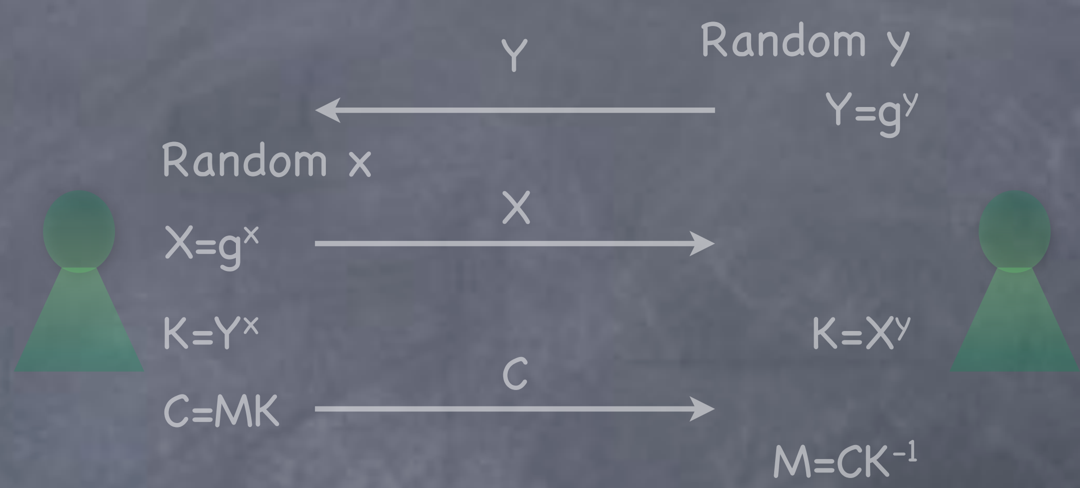
KeyGen: PK=(G,g,Y), SK=(G,g,y)

$\mathrm{Enc}_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

$\mathrm{Dec}_{(G,g,y)}(X,C) = CX^{-y}$

Abstracting El Gamal

- Trapdoor PRG:



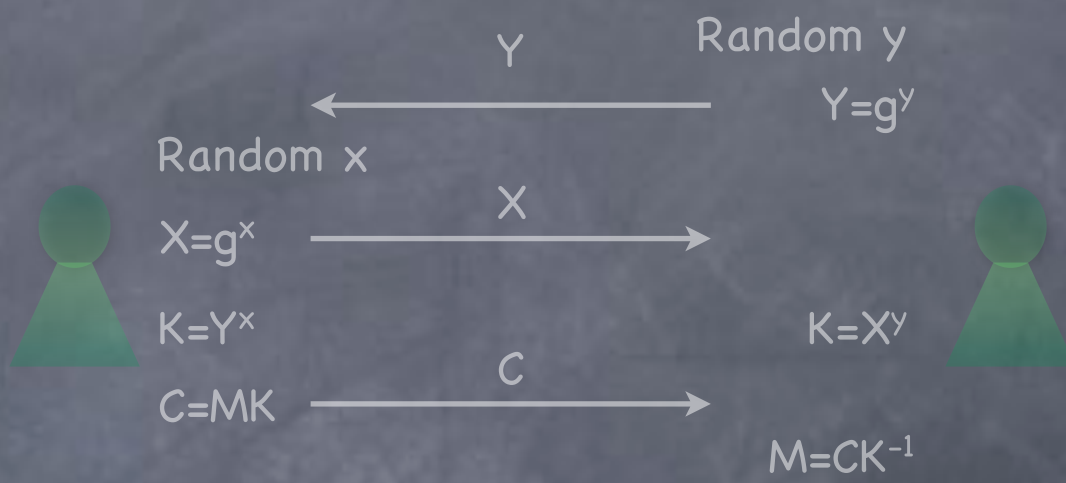
KeyGen: PK=(G,g,Y), SK=(G,g,y)

$\mathrm{Enc}_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

$\mathrm{Dec}_{(G,g,y)}(X,C) = CX^{-y}$

Abstracting El Gamal

- Trapdoor PRG:
  - KeyGen: a pair (PK,SK)



KeyGen: $PK=(G,g,Y)$, $SK=(G,g,y)$

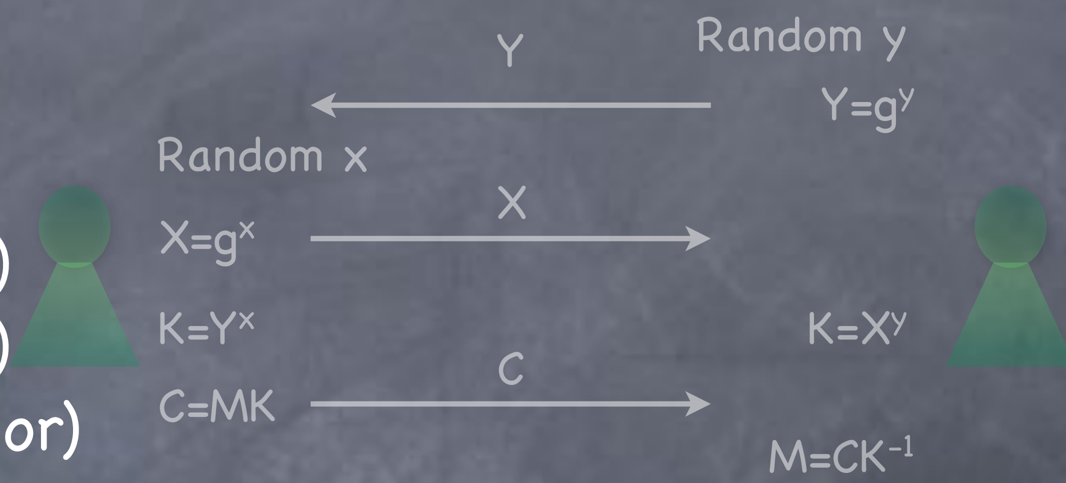
$Enc_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

$Dec_{(G,g,y)}(X,C) = CX^{-y}$

KeyGen: (PK,SK)

Abstracting El Gamal

- Trapdoor PRG:
  - KeyGen: a pair (PK,SK)
  - Three functions: $G_{PK}(.)$ (a PRG) and $T_{PK}(.)$ (make trapdoor info) and $R_{SK}(.)$ (opening the trapdoor)



KeyGen: PK=(G,g,Y), SK=(G,g,y)

$Enc_{(G,g,Y)}(M) = (X=g^x, C=MY^x)$

$Dec_{(G,g,y)}(X,C) = CX^{-y}$

KeyGen: (PK,SK)

Abstracting El Gamal

- Trapdoor PRG:
  - KeyGen: a pair (PK,SK)
  - Three functions: $G_{PK}(.)$ (a PRG) and $T_{PK}(.)$ (make trapdoor info) and $R_{SK}(.)$ (opening the trapdoor)

Random y

$Y=g^y$

$\xleftarrow{\quad Y \quad}$

Random x

$X=g^x$ $\xrightarrow{\quad X \quad}$

$K=Y^x$ $\qquad$ $K=X^y$

$C=MK$ $\xrightarrow{\quad C \quad}$

$M=CK^{-1}$

KeyGen: PK=(G,g,Y), SK=(G,g,y)

$Enc_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

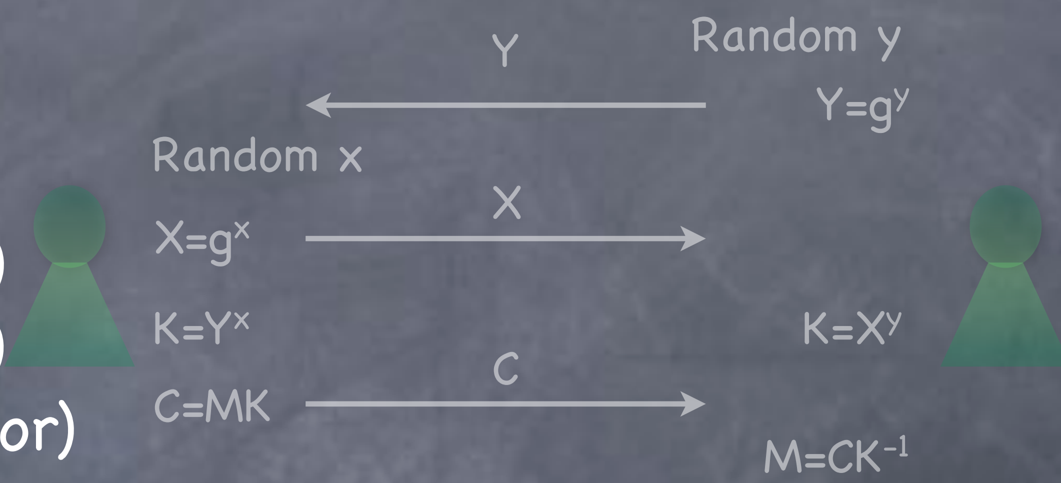
$Dec_{(G,g,y)}(X,C) = CX^{-y}$

KeyGen: (PK,SK)

$Enc_{PK}(M) = (X=T_{PK}(x),\ C=M.G_{PK}(x))$

Abstracting El Gamal

- Trapdoor PRG:
  - KeyGen: a pair (PK,SK)
  - Three functions: $G_{PK}(.)$ (a PRG) and $T_{PK}(.)$ (make trapdoor info) and $R_{SK}(.)$ (opening the trapdoor)



KeyGen: PK=(G,g,Y), SK=(G,g,y)

$\mathrm{Enc}_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

$\mathrm{Dec}_{(G,g,y)}(X,C) = CX^{-y}$

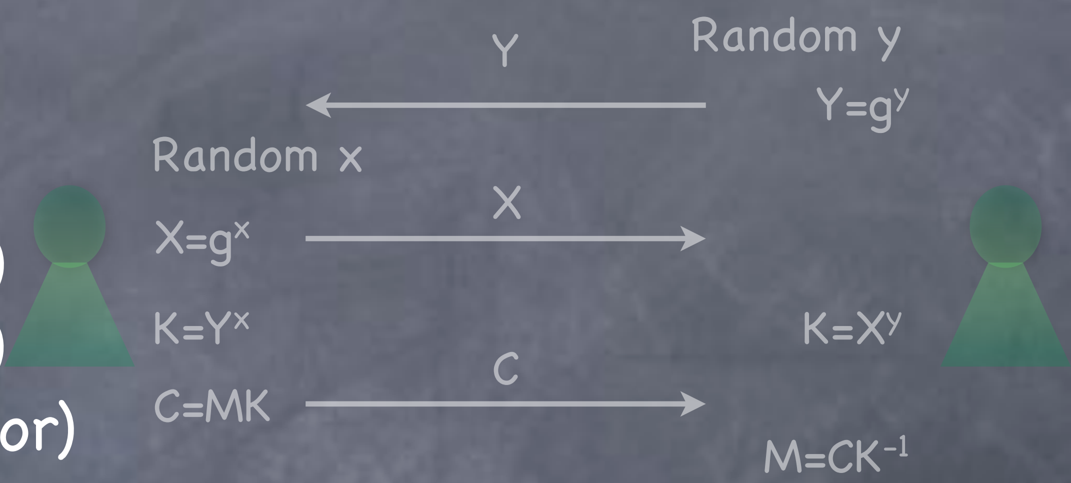
KeyGen: (PK,SK)

$\mathrm{Enc}_{PK}(M) = (X=T_{PK}(x),\ C=M.G_{PK}(x))$

$\mathrm{Dec}_{SK}(X,C) = C/R_{SK}(T_{PK}(x))$

Abstracting El Gamal

- Trapdoor PRG:
  - KeyGen: a pair (PK,SK)
  - Three functions: $G_{PK}(.)$ (a PRG) and $T_{PK}(.)$ (make trapdoor info) and $R_{SK}(.)$ (opening the trapdoor)
    - $G_{PK}(x)$ is pseudorandom even given $T_{PK}(x)$ and PK



KeyGen: PK=(G,g,Y), SK=(G,g,y)

$\mathrm{Enc}_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

$\mathrm{Dec}_{(G,g,y)}(X,C) = CX^{-y}$

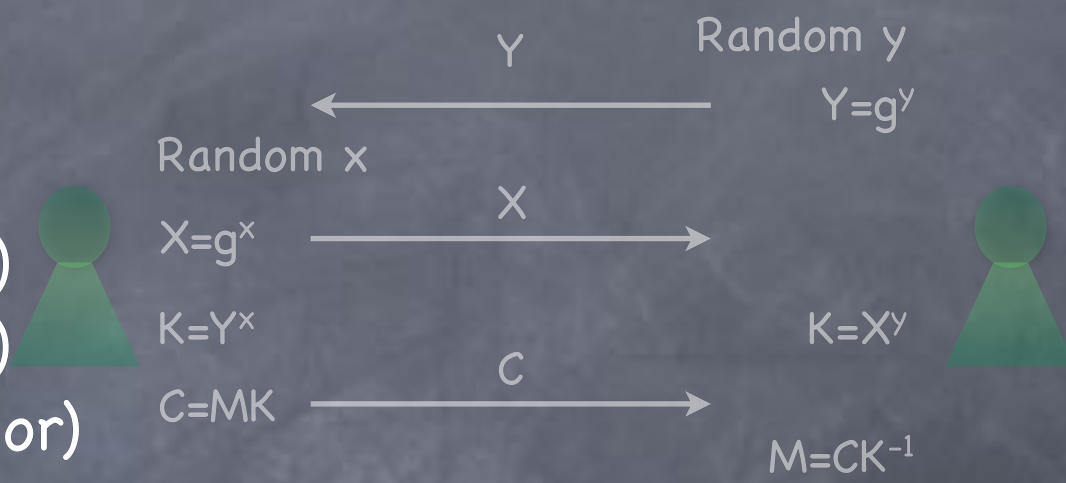
KeyGen: (PK,SK)

$\mathrm{Enc}_{PK}(M) = (X=T_{PK}(x),\ C=M.G_{PK}(x))$

$\mathrm{Dec}_{SK}(X,C) = C/R_{SK}(T_{PK}(x))$

Abstracting El Gamal

- Trapdoor PRG:
  - KeyGen: a pair (PK,SK)
  - Three functions: $G_{PK}(.)$ (a PRG) and $T_{PK}(.)$ (make trapdoor info) and $R_{SK}(.)$ (opening the trapdoor)
    - $G_{PK}(x)$ is pseudorandom even given $T_{PK}(x)$ and PK
    - $(PK,T_{PK}(x),G_{PK}(x)) \approx (PK,T_{PK}(x),r)$



KeyGen: PK=(G,g,Y), SK=(G,g,y)

$Enc_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

$Dec_{(G,g,y)}(X,C) = CX^{-y}$

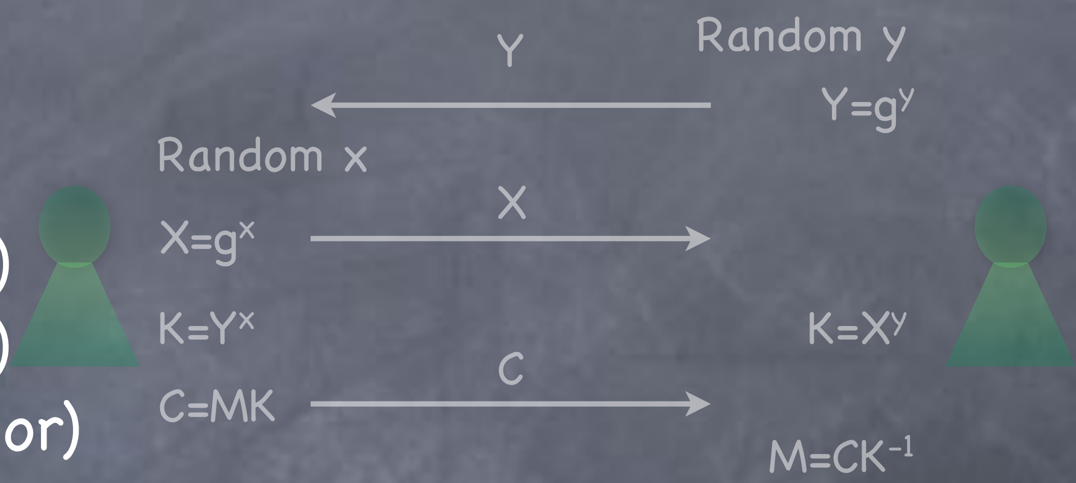
KeyGen: (PK,SK)

$Enc_{PK}(M) = (X=T_{PK}(x),\ C=M.G_{PK}(x))$

$Dec_{SK}(X,C) = C/R_{SK}(T_{PK}(x))$

Abstracting El Gamal

- Trapdoor PRG:
  - KeyGen: a pair (PK,SK)
  - Three functions: $G_{PK}(.)$ (a PRG) and $T_{PK}(.)$ (make trapdoor info) and $R_{SK}(.)$ (opening the trapdoor)
    - $G_{PK}(x)$ is pseudorandom even given $T_{PK}(x)$ and PK
    - $(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$
    - $T_{PK}(x)$ hides $G_{PK}(x)$. SK opens it.



KeyGen: PK=(G,g,Y), SK=(G,g,y)

$Enc_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

$Dec_{(G,g,y)}(X,C) = CX^{-y}$

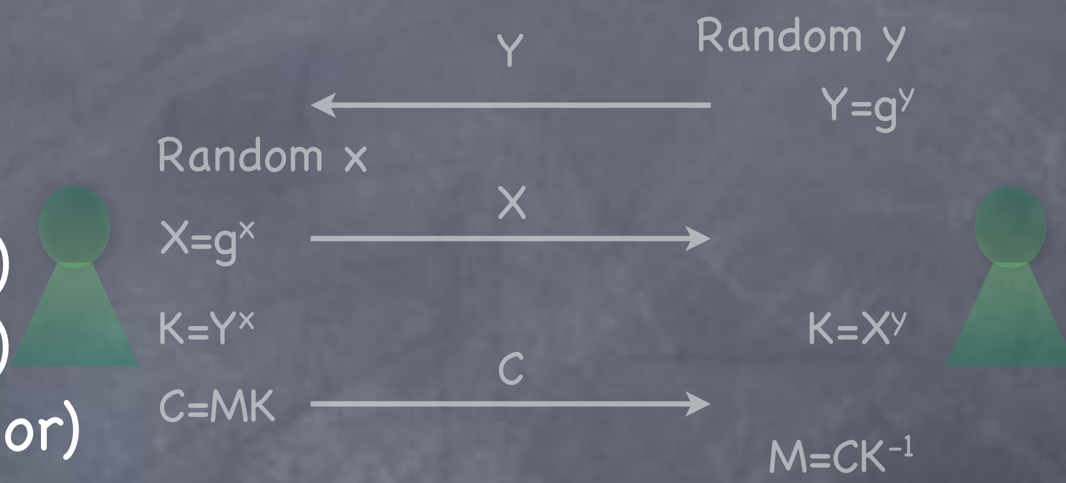
KeyGen: (PK,SK)

$Enc_{PK}(M) = (X=T_{PK}(x),\ C=M.G_{PK}(x))$

$Dec_{SK}(X,C) = C/R_{SK}(T_{PK}(x))$

Abstracting El Gamal

- Trapdoor PRG:
  - KeyGen: a pair (PK,SK)
  - Three functions: $G_{PK}(.)$ (a PRG) and $T_{PK}(.)$ (make trapdoor info) and $R_{SK}(.)$ (opening the trapdoor)
    - $G_{PK}(x)$ is pseudorandom even given $T_{PK}(x)$ and PK
    - $(PK,T_{PK}(x),G_{PK}(x)) \approx (PK,T_{PK}(x),r)$
    - $T_{PK}(x)$ hides $G_{PK}(x)$. SK opens it.
      - $R_{SK}(T_{PK}(x)) = G_{PK}(x)$



KeyGen: $PK=(G,g,Y)$, $SK=(G,g,y)$

$Enc_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

$Dec_{(G,g,y)}(X,C) = CX^{-y}$

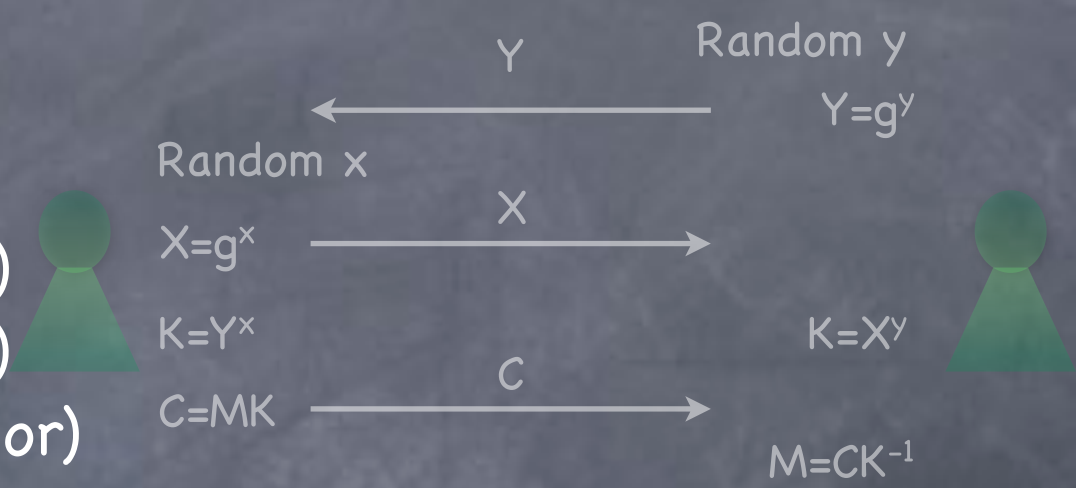
KeyGen: (PK,SK)

$Enc_{PK}(M) = (X=T_{PK}(x),\ C=M.G_{PK}(x))$

$Dec_{SK}(X,C) = C/R_{SK}(T_{PK}(x))$

Abstracting El Gamal

- Trapdoor PRG:
  - KeyGen: a pair (PK,SK)
  - Three functions: $G_{PK}(.)$ (a PRG) and $T_{PK}(.)$ (make trapdoor info) and $R_{SK}(.)$ (opening the trapdoor)
    - $G_{PK}(x)$ is pseudorandom even given $T_{PK}(x)$ and PK
    - $(PK,T_{PK}(x),G_{PK}(x)) \approx (PK,T_{PK}(x),r)$
    - $T_{PK}(x)$ hides $G_{PK}(x)$. SK opens it.
      - $R_{SK}(T_{PK}(x)) = G_{PK}(x)$
- Enough for an IND-CPA secure PKE scheme



KeyGen: PK=(G,g,Y), SK=(G,g,y)

$Enc_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

$Dec_{(G,g,y)}(X,C) = CX^{-y}$

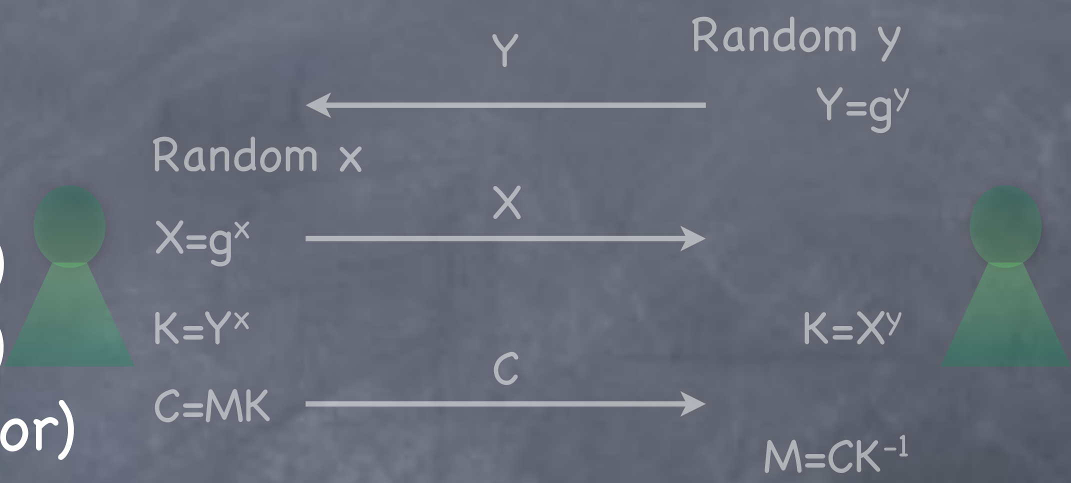
KeyGen: (PK,SK)

$Enc_{PK}(M) = (X=T_{PK}(x),\ C=M.G_{PK}(x))$

$Dec_{SK}(X,C) = C/R_{SK}(T_{PK}(x))$

Abstracting El Gamal

- Trapdoor PRG:
  - KeyGen: a pair (PK,SK)
  - Three functions: $G_{PK}(.)$ (a PRG) and $T_{PK}(.)$ (make trapdoor info) and $R_{SK}(.)$ (opening the trapdoor)
    - $G_{PK}(x)$ is pseudorandom even given $T_{PK}(x)$ and PK
    - $(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$
    - $T_{PK}(x)$ hides $G_{PK}(x)$. SK opens it.
      - $R_{SK}(T_{PK}(x)) = G_{PK}(x)$
- Enough for an IND-CPA secure PKE scheme (cf. Security of El Gamal)



KeyGen: PK=(G,g,Y), SK=(G,g,y)

$Enc_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

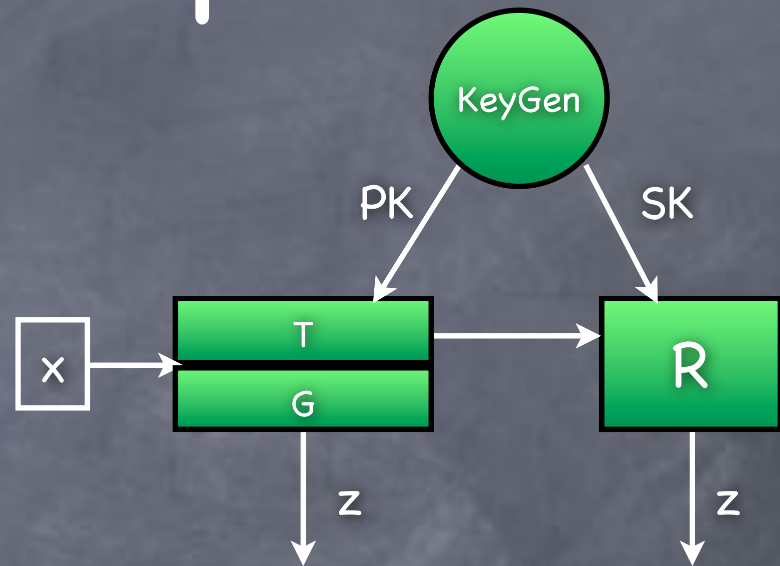
$Dec_{(G,g,y)}(X,C) = CX^{-y}$

KeyGen: (PK,SK)

$Enc_{PK}(M) = (X=T_{PK}(x),\ C=M.G_{PK}(x))$

$Dec_{SK}(X,C) = C/R_{SK}(T_{PK}(x))$

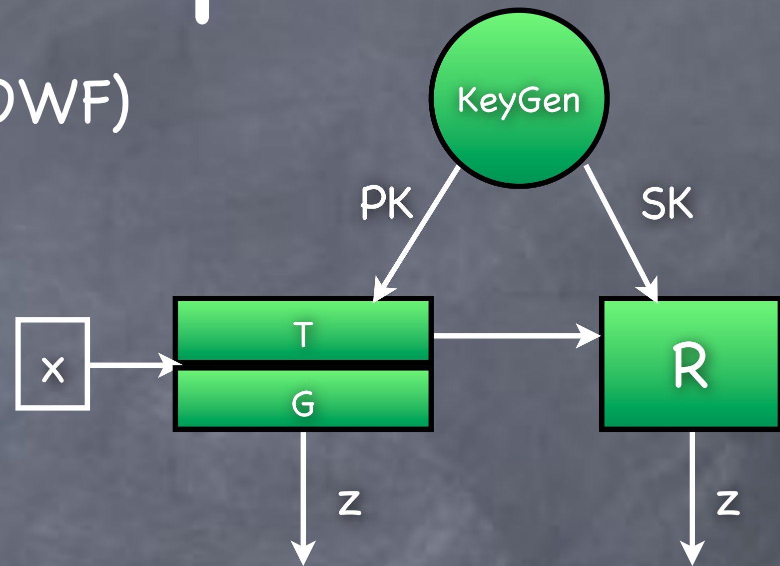
Trapdoor PRG from Generic Assumption?



$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$

Trapdoor PRG from Generic Assumption?

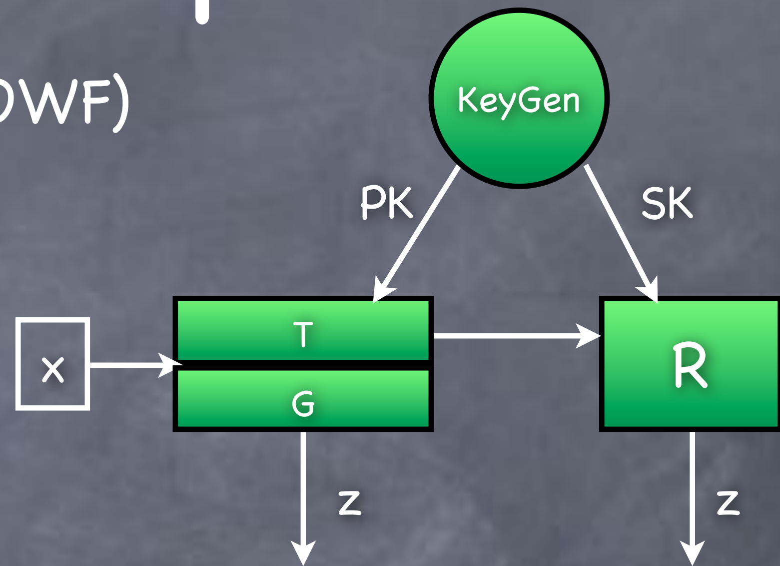
- PRG constructed from OWP (or OWF)



$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$

Trapdoor PRG from Generic Assumption?

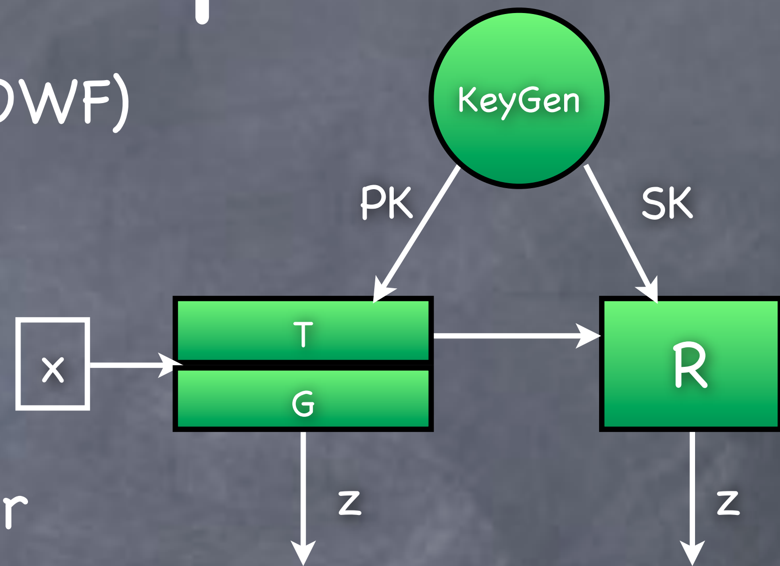
- PRG constructed from OWP (or OWF)
  - Allows us to instantiate the construction with several candidates



$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$

Trapdoor PRG from Generic Assumption?

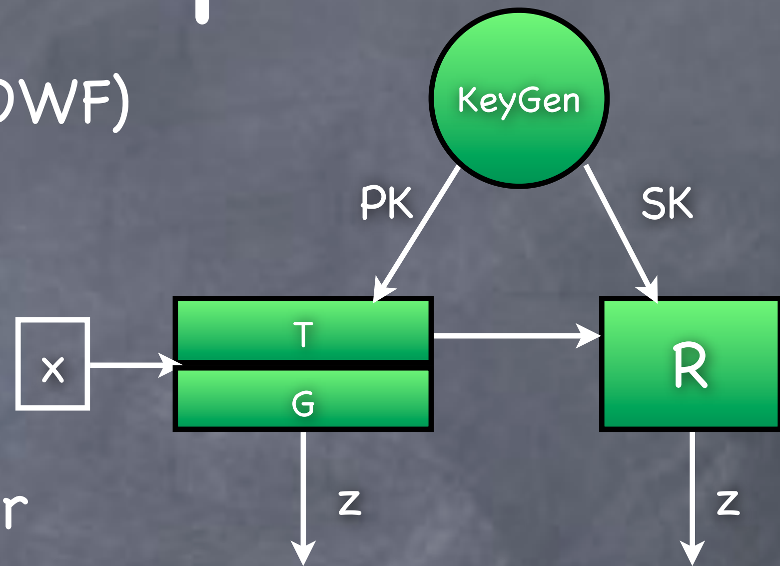
- PRG constructed from OWP (or OWF)
  - Allows us to instantiate the construction with several candidates
- Is there a similar construction for TPRG from OWP?



$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$

Trapdoor PRG from Generic Assumption?

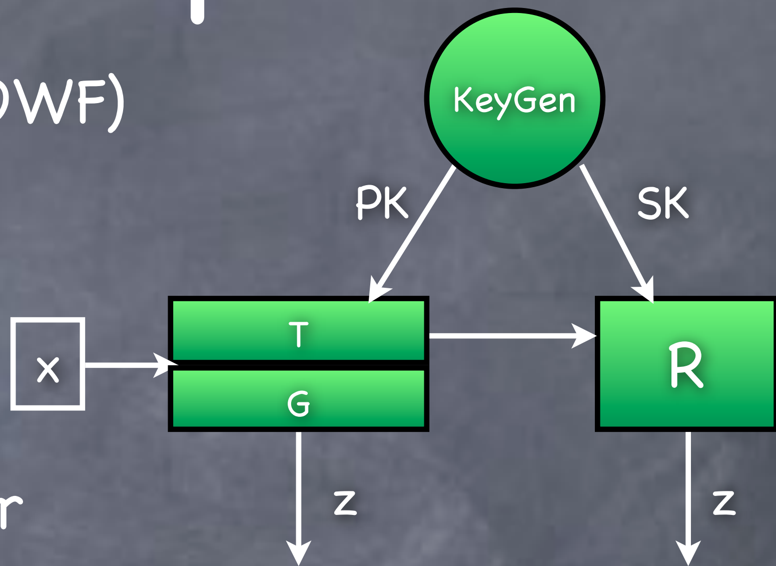
- PRG constructed from OWP (or OWF)
  - Allows us to instantiate the construction with several candidates
- Is there a similar construction for TPRG from OWP?
  - Trapdoor property seems fundamentally different: generic OWP may not offer such a property



$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$

Trapdoor PRG from Generic Assumption?

- PRG constructed from OWP (or OWF)
  - Allows us to instantiate the construction with several candidates
- Is there a similar construction for TPRG from OWP?
  - Trapdoor property seems fundamentally different: generic OWP may not offer such a property
  - Will start with "Trapdoor OWP"



$$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$$

Trapdoor OWP

Trapdoor OWP

- (KeyGen,f,f') (all PPT) is a trapdoor one-way permutation (TOWP) if

Trapdoor OWP

- (KeyGen,f,f') (all PPT) is a trapdoor one-way permutation (TOWP) if
  - For all (PK,SK) ←KeyGen

Trapdoor OWP

- (KeyGen,f,f') (all PPT) is a trapdoor one-way permutation (TOWP) if
  - For all (PK,SK) ←KeyGen
    - f_{PK} a permutation

Trapdoor OWP

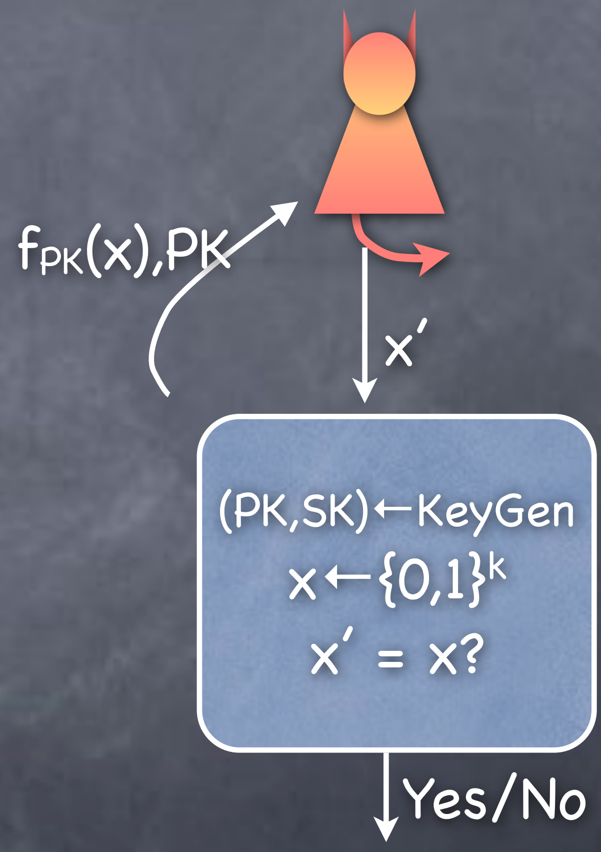
- (KeyGen,f,f') (all PPT) is a trapdoor one-way permutation (TOWP) if
  - For all (PK,SK) ←KeyGen
    - f_{PK} a permutation
    - f'_{SK} is the inverse of f_{PK}

Trapdoor OWP

- (KeyGen,f,f') (all PPT) is a trapdoor one-way permutation (TOWP) if
  - For all (PK,SK) ←KeyGen
    - f_{PK} a permutation
    - f'_{SK} is the inverse of f_{PK}
  - For all PPT adversary, probability of success in the TOWP experiment is negligible

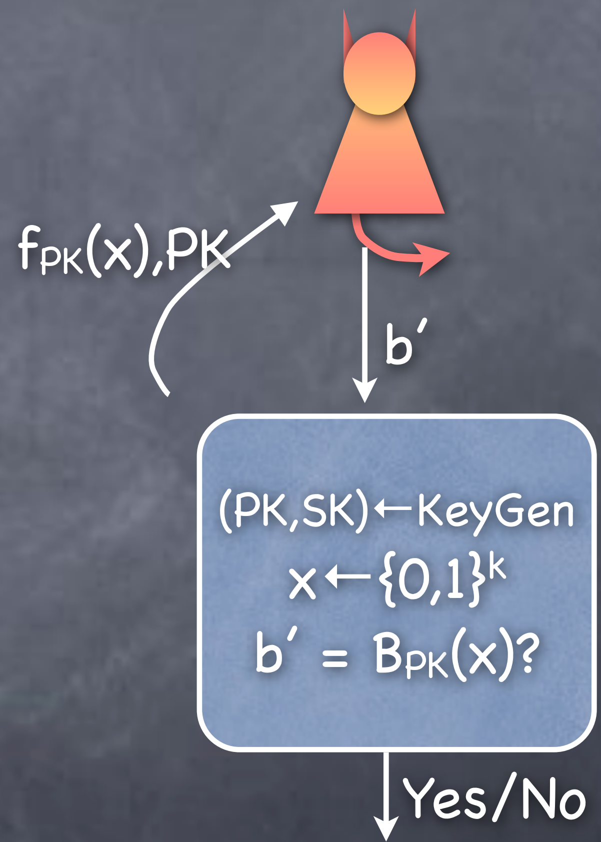
Trapdoor OWP

- (KeyGen,f,f') (all PPT) is a trapdoor one-way permutation (TOWP) if
  - For all (PK,SK) ←KeyGen
    - f_{PK} a permutation
    - f'_{SK} is the inverse of f_{PK}
  - For all PPT adversary, probability of success in the TOWP experiment is negligible

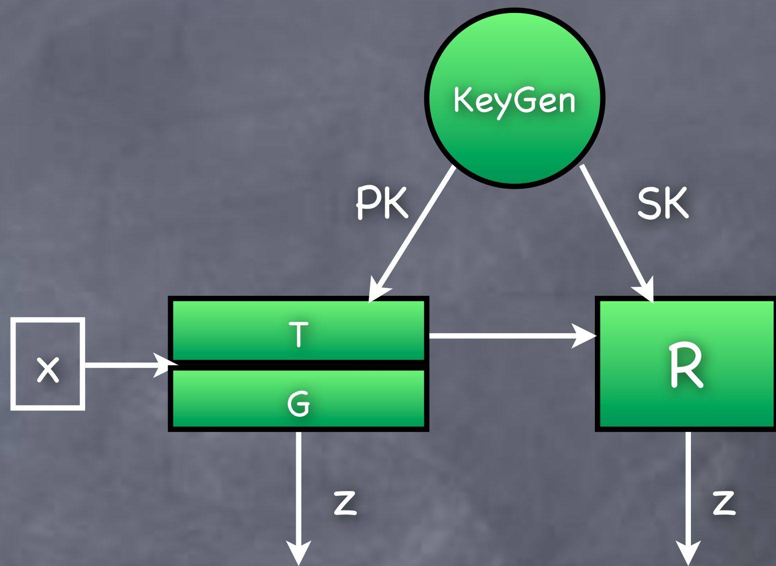


Trapdoor OWP

- (KeyGen,f,f') (all PPT) is a trapdoor one-way permutation (TOWP) if
  - For all (PK,SK) ←KeyGen
    - f_{PK} a permutation
    - f'_{SK} is the inverse of f_{PK}
  - For all PPT adversary, probability of success in the TOWP experiment is negligible
  - Hardcore predicate:
    - B_{PK} s.t. $(PK,f_{PK}(x),B_{PK}(x)) \approx (PK,f_{PK}(x),r)$



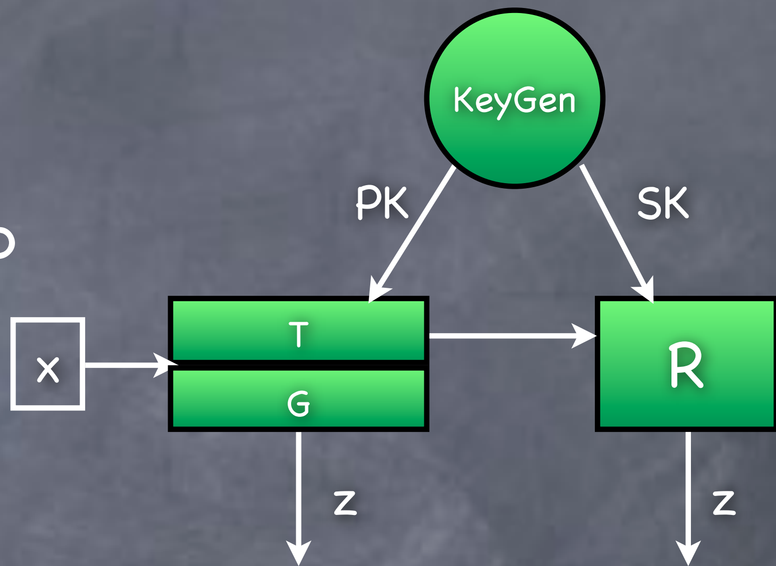
Trapdoor PRG from Trapdoor OWP



$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$

Trapdoor PRG from Trapdoor OWP

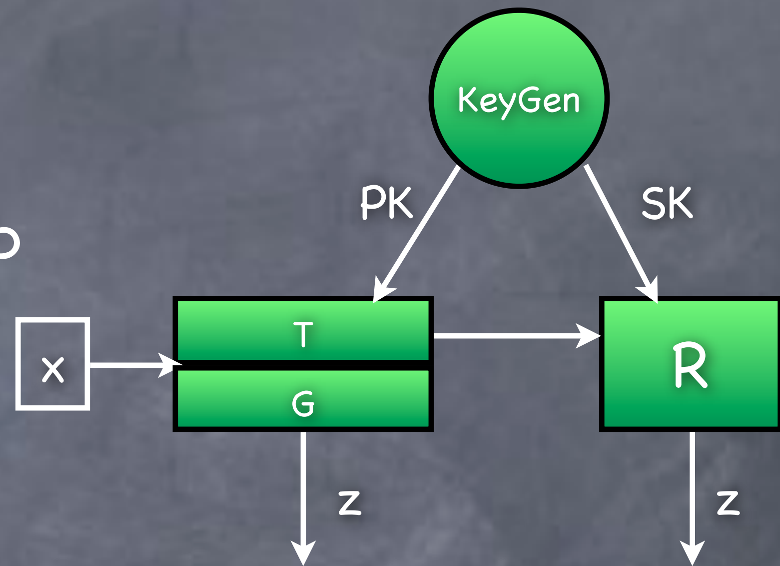
- Same construction as PRG from OWP



$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$

Trapdoor PRG from Trapdoor OWP

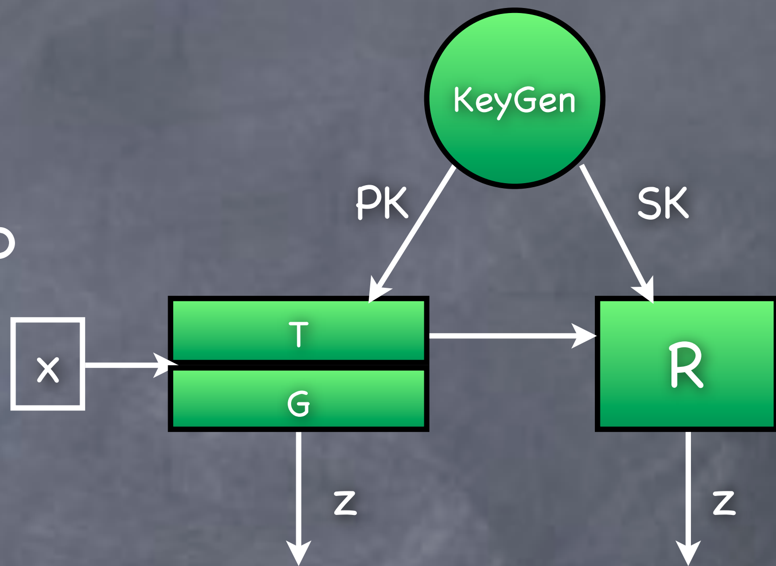
- Same construction as PRG from OWP
- One bit TPRG



$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$

Trapdoor PRG from Trapdoor OWP

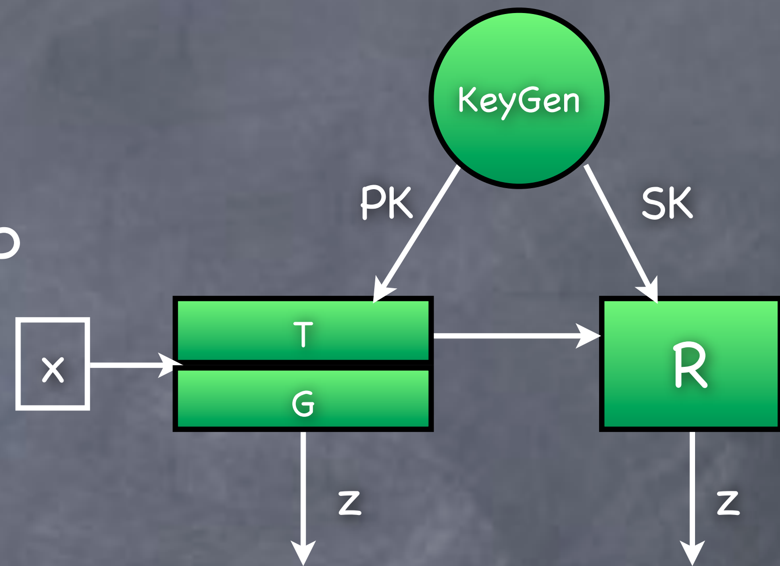
- Same construction as PRG from OWP
- One bit TPRG
  - KeyGen same as TOWP's KeyGen



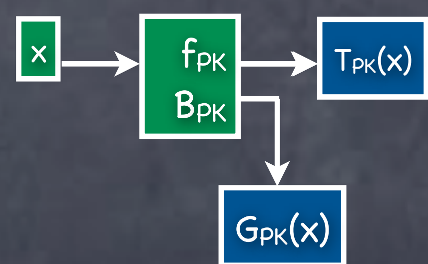
$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$

Trapdoor PRG from Trapdoor OWP

- Same construction as PRG from OWP
- One bit TPRG
  - KeyGen same as TOWP's KeyGen
  - $G_{PK}(x) := B_{PK}(x)$. $T_{PK}(x) := f_{PK}(x)$. $R_{SK}(y) := G_{PK}(f'_{SK}(y))$

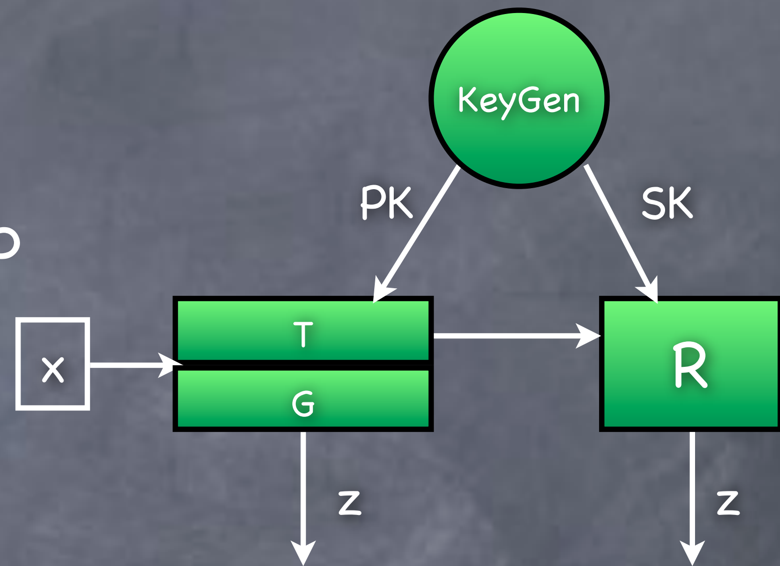


$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$

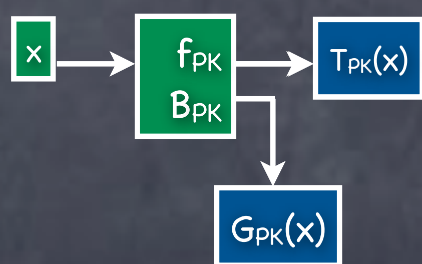


Trapdoor PRG from Trapdoor OWP

- Same construction as PRG from OWP
- One bit TPRG
  - KeyGen same as TOWP's KeyGen
  - $G_{PK}(x) := B_{PK}(x)$. $T_{PK}(x) := f_{PK}(x)$. $R_{SK}(y) := G_{PK}(f'_{SK}(y))$
    - (SK assumed to contain PK)

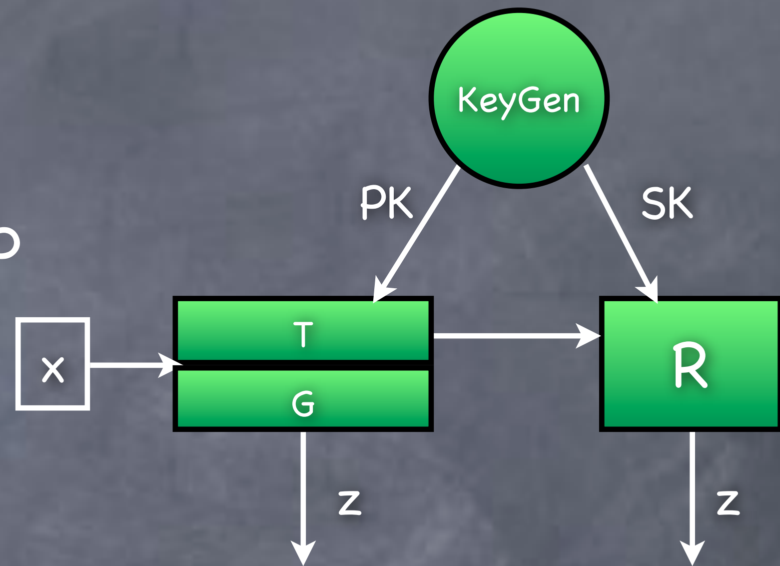


$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$



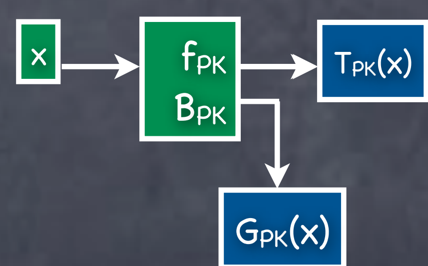
Trapdoor PRG from Trapdoor OWP

- Same construction as PRG from OWP
- One bit TPRG
  - KeyGen same as TOWP's KeyGen
  - $G_{PK}(x) := B_{PK}(x)$. $T_{PK}(x) := f_{PK}(x)$. $R_{SK}(y) := G_{PK}(f'_{SK}(y))$
    - (SK assumed to contain PK)



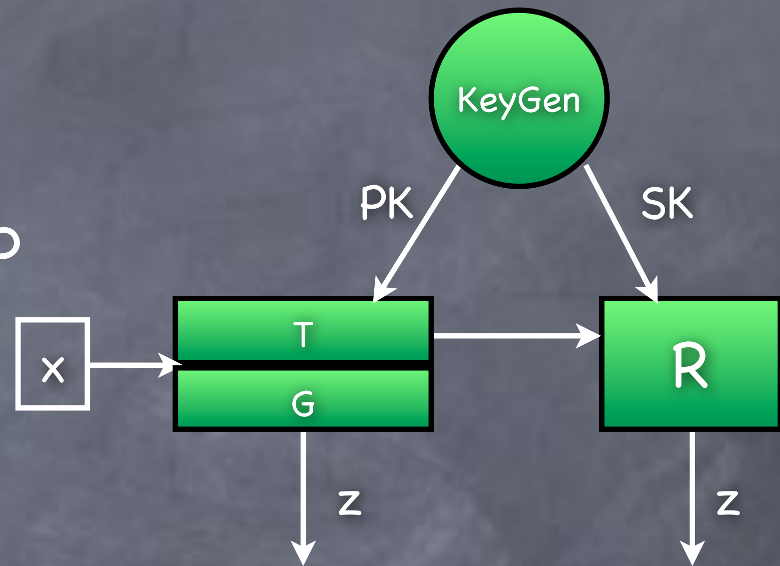
$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$

$(PK, f_{PK}(x), B_{PK}(x)) \approx (PK, f_{PK}(x), r)$



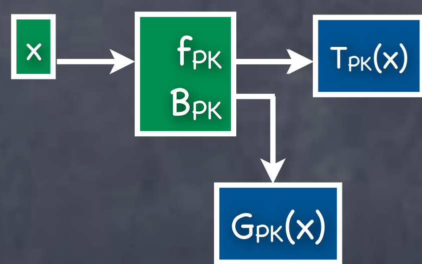
Trapdoor PRG from Trapdoor OWP

- Same construction as PRG from OWP
- One bit TPRG
  - KeyGen same as TOWP's KeyGen
  - $G_{PK}(x) := B_{PK}(x)$. $T_{PK}(x) := f_{PK}(x)$. $R_{SK}(y) := G_{PK}(f'_{SK}(y))$
    - (SK assumed to contain PK)
- More generally, last permutation output serves as T_{PK}



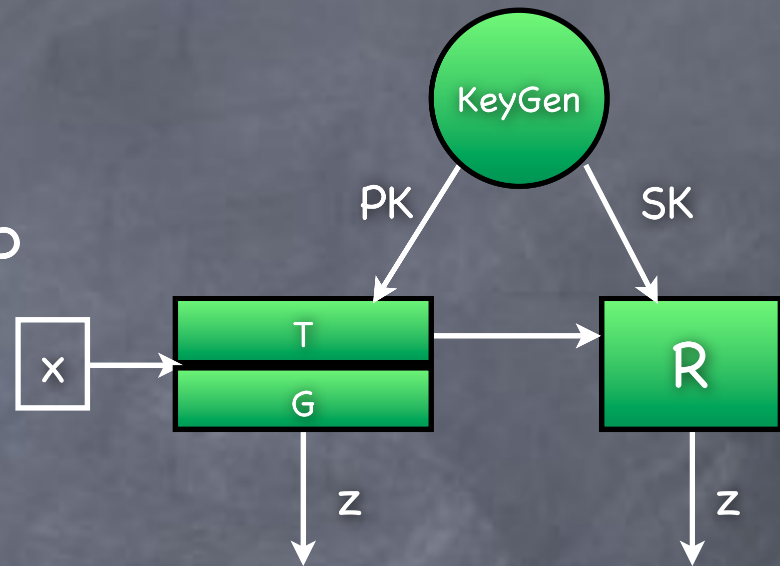
$(PK,T_{PK}(x),G_{PK}(x)) \approx (PK,T_{PK}(x),r)$

$(PK,f_{PK}(x),B_{PK}(x)) \approx (PK,f_{PK}(x),r)$



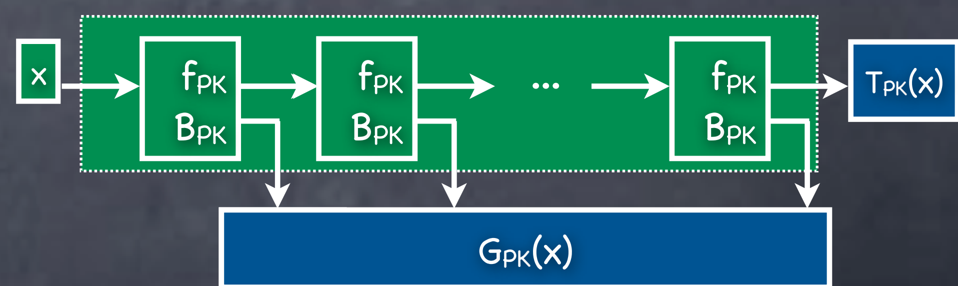
Trapdoor PRG from Trapdoor OWP

- Same construction as PRG from OWP
- One bit TPRG
  - KeyGen same as TOWP's KeyGen
  - $G_{PK}(x) := B_{PK}(x)$. $T_{PK}(x) := f_{PK}(x)$. $R_{SK}(y) := G_{PK}(f'_{SK}(y))$
    - (SK assumed to contain PK)
- More generally, last permutation output serves as T_{PK}



$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$

$(PK, f_{PK}(x), B_{PK}(x)) \approx (PK, f_{PK}(x), r)$



Candidate TOWPs

Candidate TOWPs

- From some (candidate) OWP collections, with index as public-key

Candidate TOWPs

- From some (candidate) OWP collections, with index as public-key
- Recall candidate OWF collections

Candidate TOWPs

- From some (candidate) OWP collections, with index as public-key
- Recall candidate OWF collections
  - Rabin OWF: $f_{Rabin}(x; N) = x^2 \bmod N$, where $N = PQ$, and P, Q are k-bit primes, and x uniform from $\{0...N\}$

Candidate TOWPs

- From some (candidate) OWP collections, with index as public-key
- Recall candidate OWF collections
  - Rabin OWF: $f_{Rabin}(x; N) = x^2 \bmod N$, where $N = PQ$, and P, Q are k-bit primes, and x uniform from {0...N}
    - Fact: $f_{Rabin}(.; N)$ is a permutation among quadratic residues, when P, Q are safe primes

Candidate TOWPs

- From some (candidate) OWP collections, with index as public-key
- Recall candidate OWF collections
  - Rabin OWF: $f_{Rabin}(x; N) = x^2 \bmod N$, where $N = PQ$, and P, Q are k-bit primes, and x uniform from {0…N}
    - Fact: $f_{Rabin}(.; N)$ is a permutation among quadratic residues, when P, Q are safe primes
    - Fact: Can invert $f_{Rabin}(.; N)$ given factorization of N

Candidate TOWPs

- From some (candidate) OWP collections, with index as public-key
- Recall candidate OWF collections
  - Rabin OWF: $f_{Rabin}(x; N) = x^2 \bmod N$, where N = PQ, and P, Q are k-bit primes, and x uniform from {0...N}
    - Fact: $f_{Rabin}(.; N)$ is a permutation among quadratic residues, when P, Q are safe primes
    - Fact: Can invert $f_{Rabin}(.; N)$ given factorization of N
  - RSA function: $f_{RSA}(x; N,e) = x^e \bmod N$ (uses index: N=PQ, and an exponent e s.t. $\gcd(e,\varphi(N)) = 1$)

Candidate TOWPs

- From some (candidate) OWP collections, with index as public-key
- Recall candidate OWF collections
  - Rabin OWF: $f_{Rabin}(x; N) = x^2 \bmod N$, where $N = PQ$, and P, Q are k-bit primes, and x uniform from {0...N}
    - Fact: $f_{Rabin}(.; N)$ is a permutation among quadratic residues, when P, Q are safe primes
    - Fact: Can invert $f_{Rabin}(.; N)$ given factorization of N
  - RSA function: $f_{RSA}(x; N,e) = x^e \bmod N$ (uses index: $N=PQ$, and an exponent e s.t. $\gcd(e,\varphi(N)) = 1$)
    - Fact: $f_{RSA}(.; N,e)$ is a permutation

Candidate TOWPs

- From some (candidate) OWP collections, with index as public-key
- Recall candidate OWF collections
  - Rabin OWF: $f_{Rabin}(x; N) = x^2 \bmod N$, where N = PQ, and P, Q are k-bit primes, and x uniform from {0…N}
    - Fact: $f_{Rabin}(.; N)$ is a permutation among quadratic residues, when P, Q are safe primes
    - Fact: Can invert $f_{Rabin}(.; N)$ given factorization of N
  - RSA function: $f_{RSA}(x; N,e) = x^e \bmod N$ (uses index: N=PQ, and an exponent e s.t. $\gcd(e,\varphi(N)) = 1$)
    - Fact: $f_{RSA}(.; N,e)$ is a permutation
    - Fact: While picking (N,e), can also pick d s.t. $x^{ed} = x$

Candidate TOWPs

- From some (candidate) OWP collections, with index as public-key
- Recall candidate OWF collections
  - Rabin OWF: $f_{Rabin}(x; N) = x^2 \bmod N$, where $N = PQ$, and P, Q are k-bit primes, and x uniform from {0...N}
    - Fact: $f_{Rabin}(.; N)$ is a permutation among quadratic residues, when P, Q are safe primes
    - Fact: Can invert $f_{Rabin}(.; N)$ given factorization of N
  - RSA function: $f_{RSA}(x; N,e) = x^e \bmod N$ (uses index: N=PQ, and an exponent e s.t. $\gcd(e,\varphi(N)) = 1$)
    - Fact: $f_{RSA}(.; N,e)$ is a permutation
    - Fact: While picking (N,e), can also pick d s.t. $x^{ed} = x$

see handout

Today

Today

- CPA-secure PKE

Today

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption

Today

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption
- Trapdoor PRG

Today

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption
- Trapdoor PRG
  - Abstracts what DDH gives for El Gamal

Today

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption
- Trapdoor PRG
  - Abstracts what DDH gives for El Gamal
    - With a secret-key, trapdoor information can also yield the pseudorandom string

Today

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption
- Trapdoor PRG
  - Abstracts what DDH gives for El Gamal
    - With a secret-key, trapdoor information can also yield the pseudorandom string
  - Can be used to get IND-CPA secure PKE scheme

Today

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption
- Trapdoor PRG
    - Abstracts what DDH gives for El Gamal
        - With a secret-key, trapdoor information can also yield the pseudorandom string
    - Can be used to get IND-CPA secure PKE scheme
- Trapdoor OWP

Today

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption
- Trapdoor PRG
  - Abstracts what DDH gives for El Gamal
    - With a secret-key, trapdoor information can also yield the pseudorandom string
  - Can be used to get IND-CPA secure PKE scheme
- Trapdoor OWP
  - With a secret-key, invert the OWP

Today

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption
- Trapdoor PRG
  - Abstracts what DDH gives for El Gamal
    - With a secret-key, trapdoor information can also yield the pseudorandom string
  - Can be used to get IND-CPA secure PKE scheme
- Trapdoor OWP
  - With a secret-key, invert the OWP
  - Can be used to construct Trapdoor PRG

Today

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption
- Trapdoor PRG
  - Abstracts what DDH gives for El Gamal
    - With a secret-key, trapdoor information can also yield the pseudorandom string
  - Can be used to get IND-CPA secure PKE scheme
- Trapdoor OWP
  - With a secret-key, invert the OWP
  - Can be used to construct Trapdoor PRG
- Next: CCA secure PKE