Public-Key Cryptography

Public-Key Cryptography

Lecture 8

Public-Key Cryptography

Lecture 8

Public-Key Encryption from Trapdoor OWP

Public-Key Cryptography

Lecture 8

Public-Key Encryption from Trapdoor OWP

CCA Security

El Gamal Encryption

El Gamal Encryption

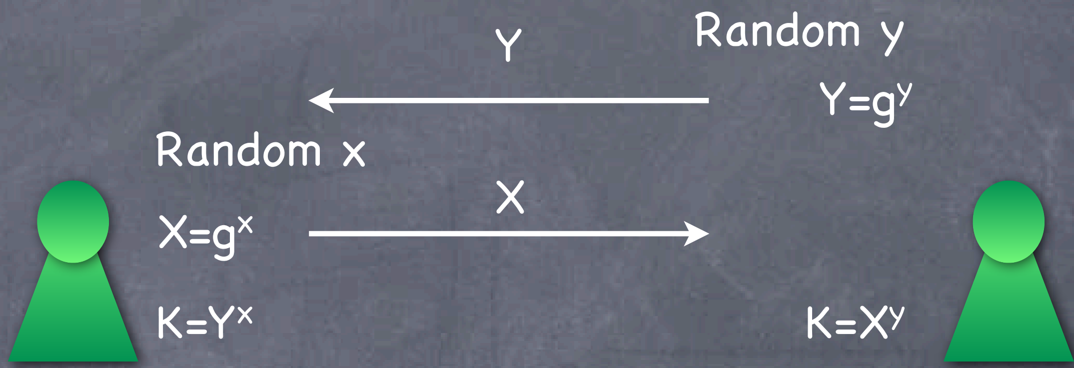
- Based on DH key-exchange

El Gamal Encryption

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

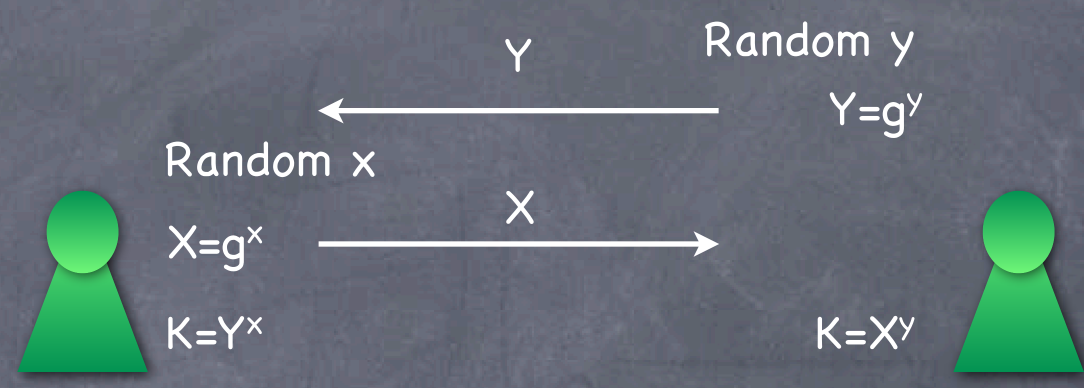
El Gamal Encryption

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



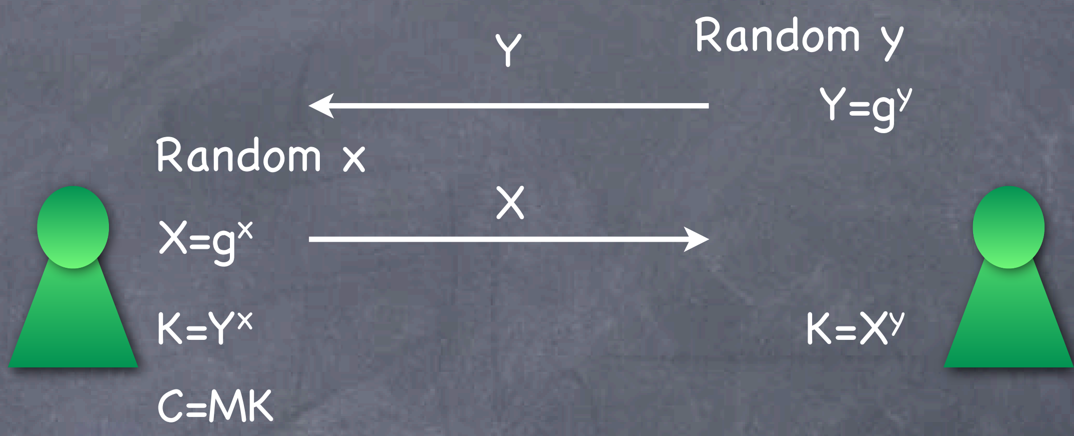
El Gamal Encryption

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



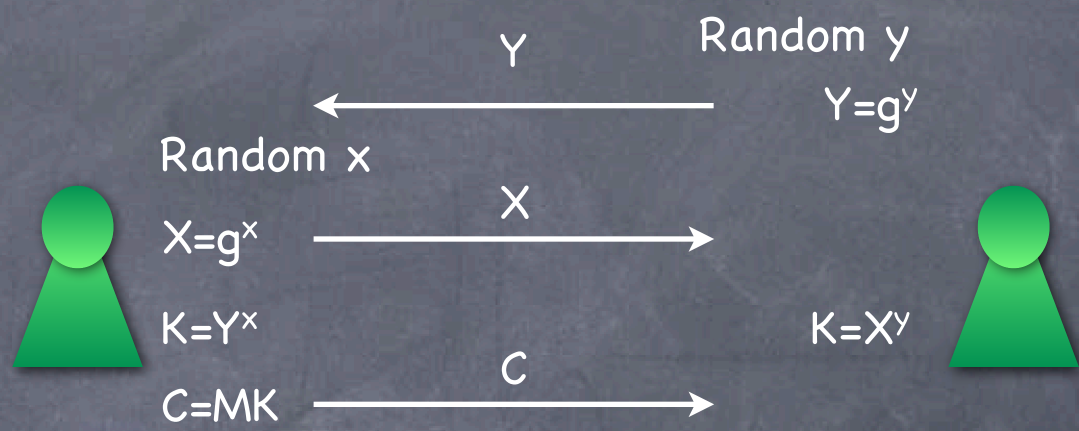
El Gamal Encryption

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



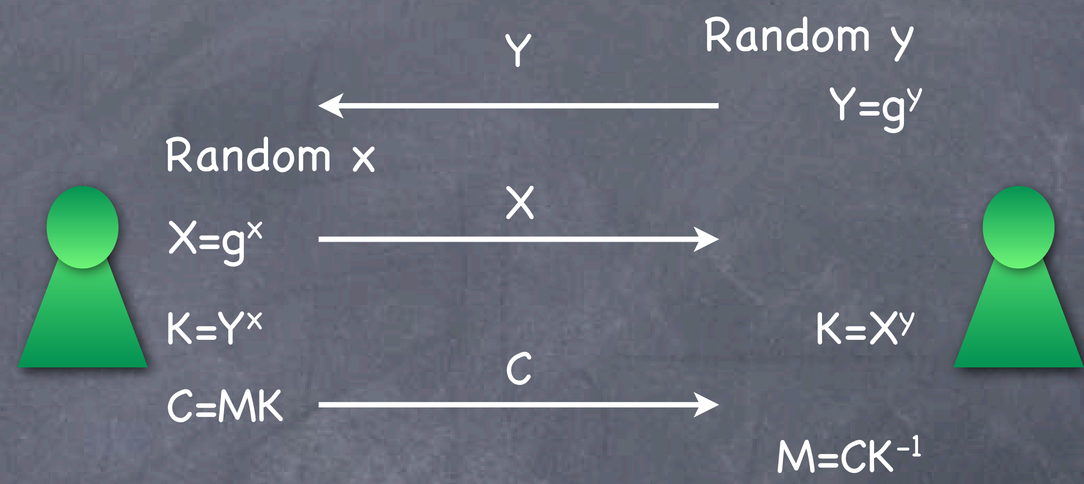
El Gamal Encryption

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

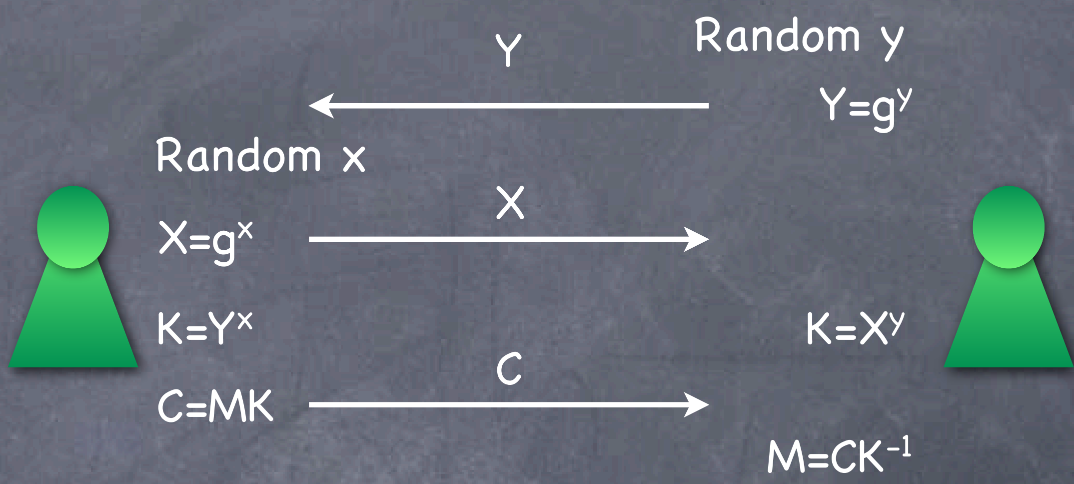


El Gamal Encryption

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



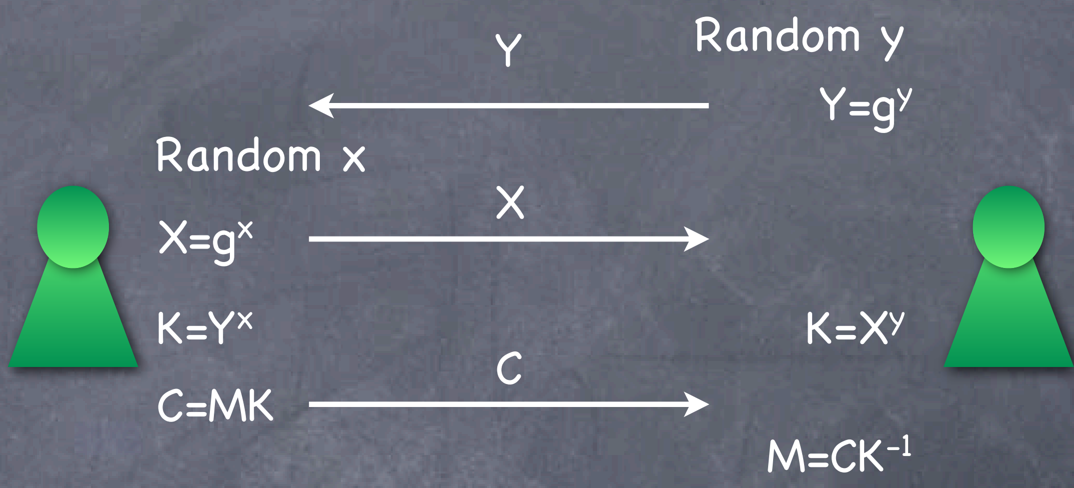
El Gamal Encryption



- 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

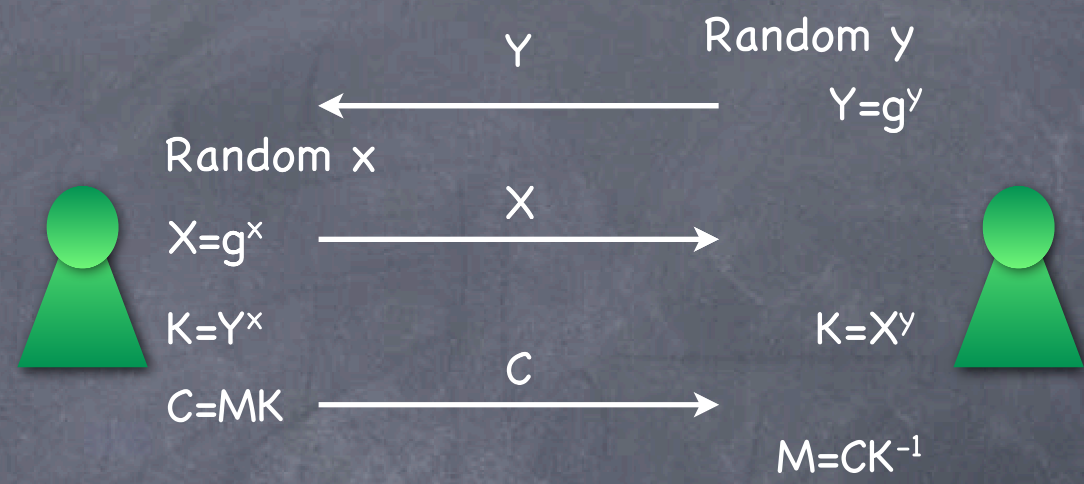
El Gamal Encryption

- 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



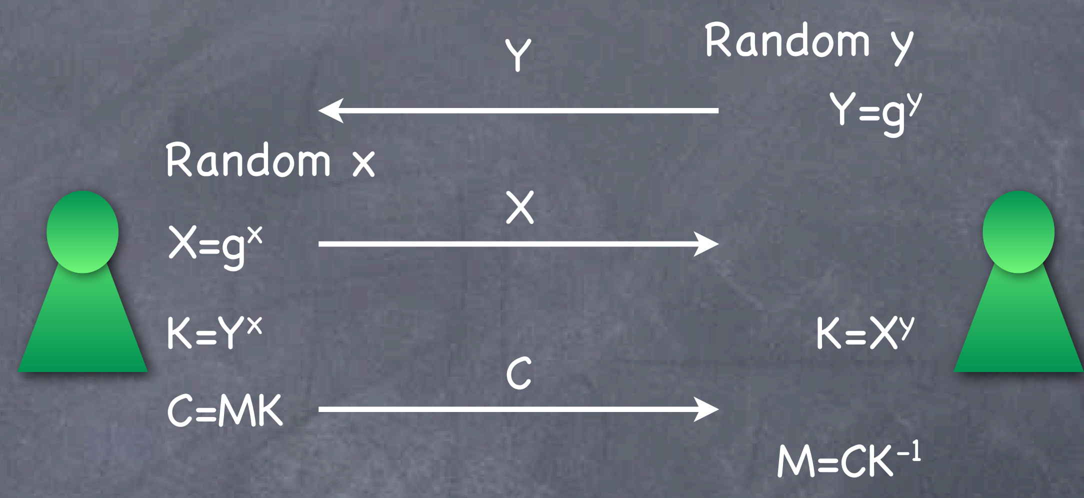
El Gamal Encryption

- 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



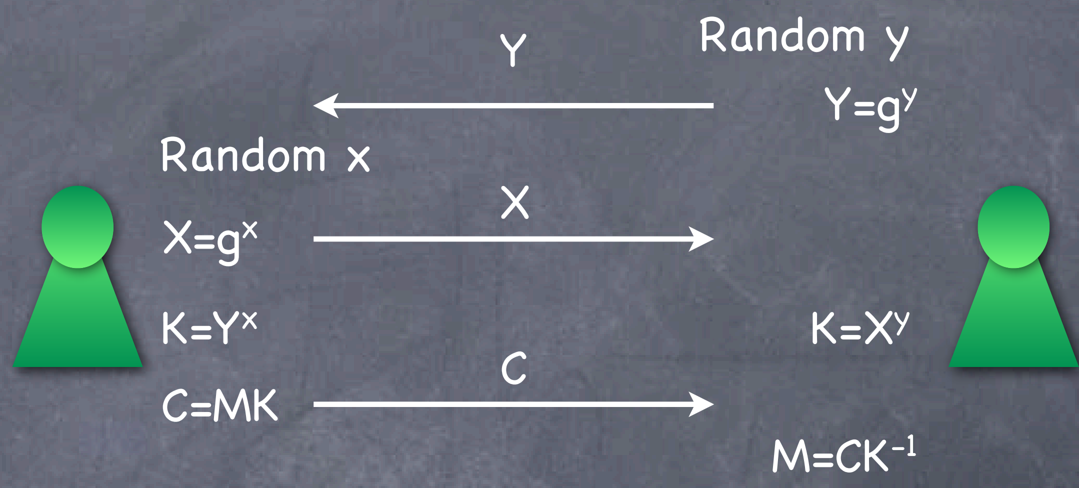
- Based on DH key-exchange
  - Alice, Bob generate a key using DH key-exchange
  - Then use it as a one-time pad
- Bob's "message" in the key-exchange is his PK
- Alice's message in the key-exchange and the ciphertext of the one-time pad together form a single ciphertext

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

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

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



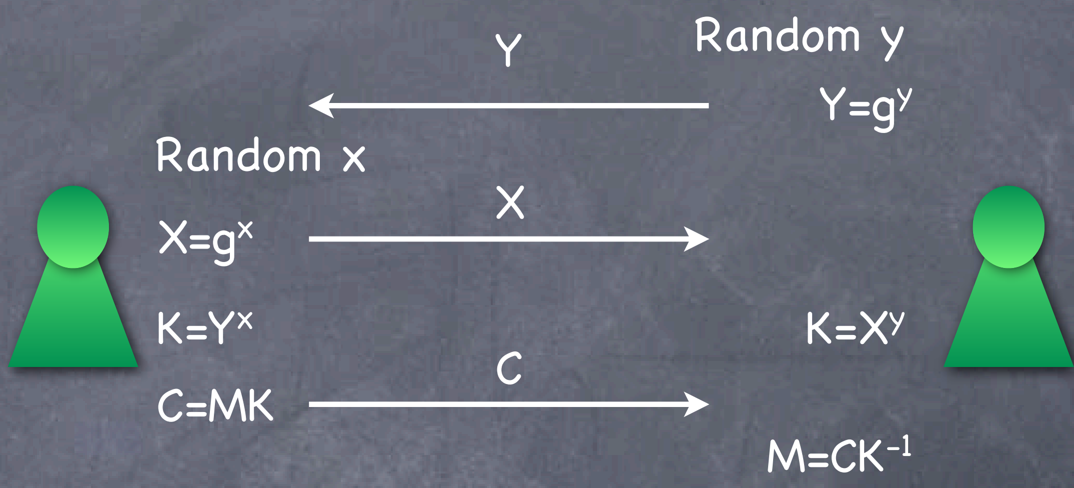
KeyGen: PK=(G,g,Y), SK=(G,g,y)

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

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

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

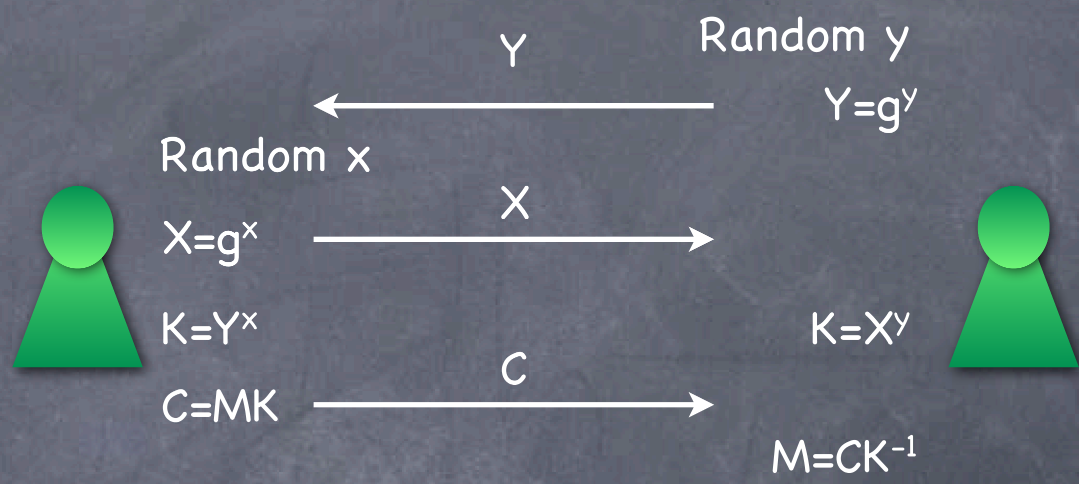
$Enc_{(G,g,Y)}(M) = (X=g^x, C=MY^x)$

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

- 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)

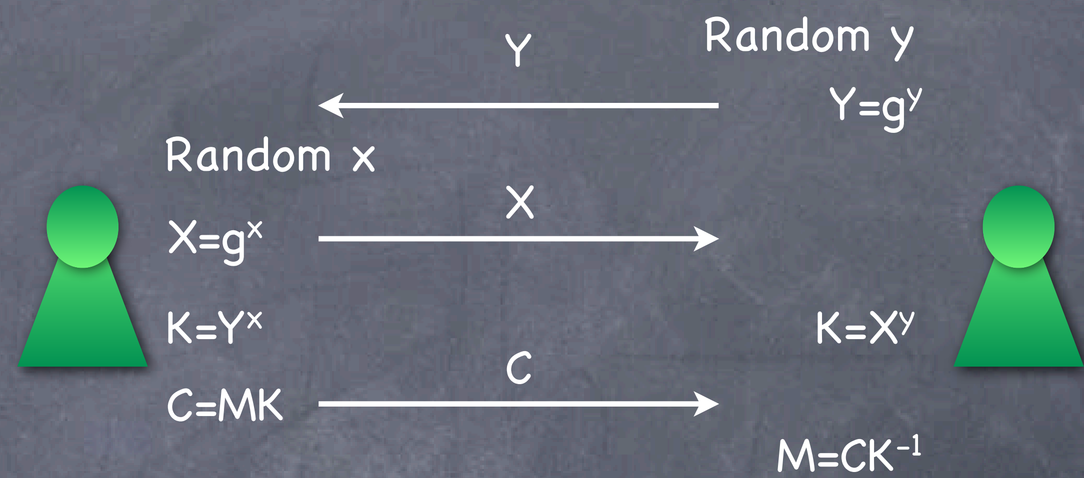
$Enc_{(G,g,Y)}(M) = (X=g^x,\ C=MY^x)$

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

- KeyGen uses GroupGen to get (G,g)
- 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 $(G,g) \leftarrow$ 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 $(G,g) \leftarrow$ GroupGen, x,y random and z=xy or random) plays the IND-CPA experiment with A:
    - But sets $PK=(G,g,g^y)$ and $Enc(M_b)=(g^x,M_b g^z)$

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')

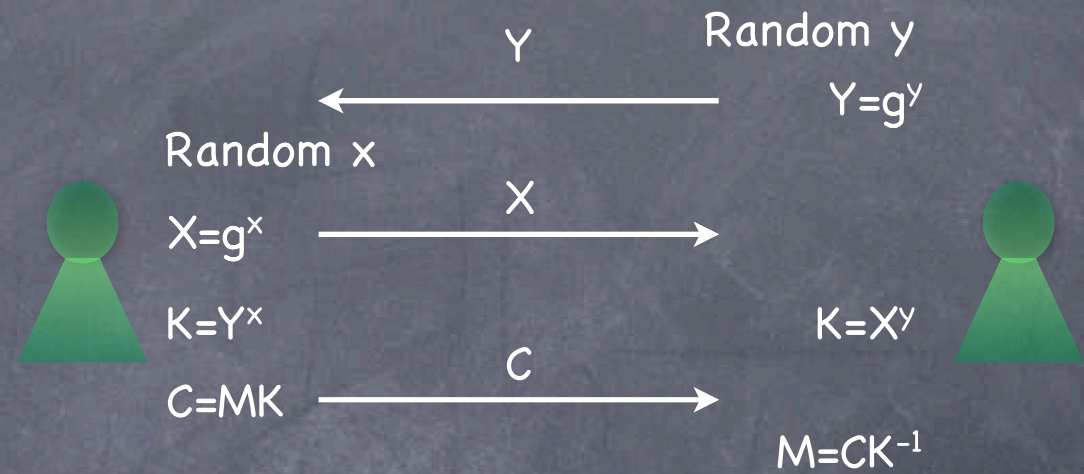
Security of El Gamal

- 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 $(G,g) \leftarrow$ 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
  - When z=xy, exactly IND-CPA experiment: A* outputs 1 with probability = 1/2 + advantage of A.

Abstracting El Gamal

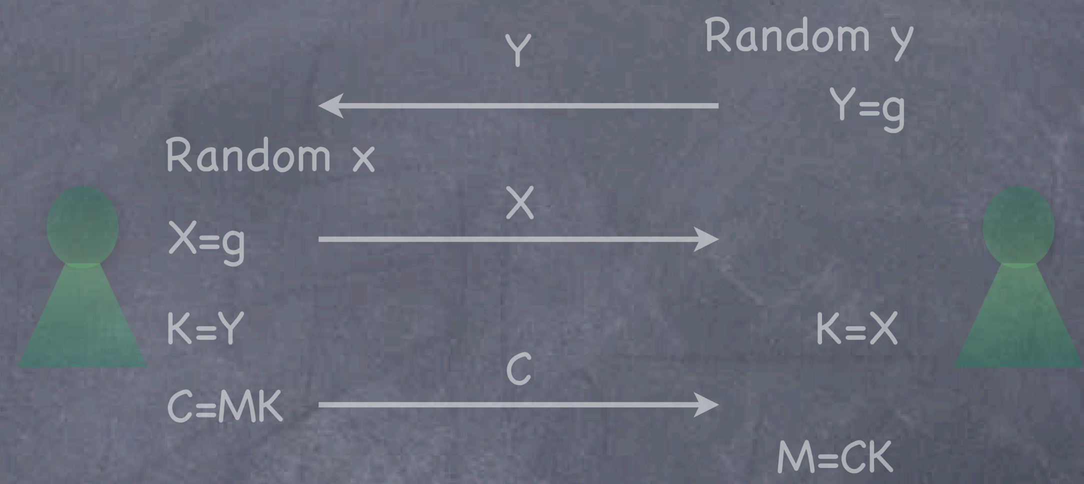


KeyGen: PK=(G,g,Y), SK=(G,g,y)

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

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

Abstracting El Gamal



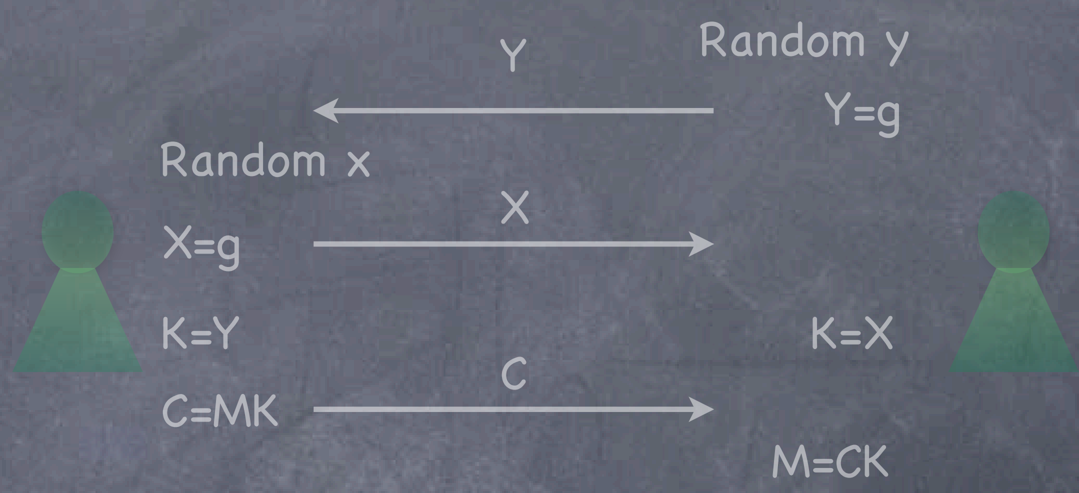
KeyGen: PK=(G,g,Y), SK=(G,g,y)

Enc

Dec

Abstracting El Gamal

- Trapdoor PRG:



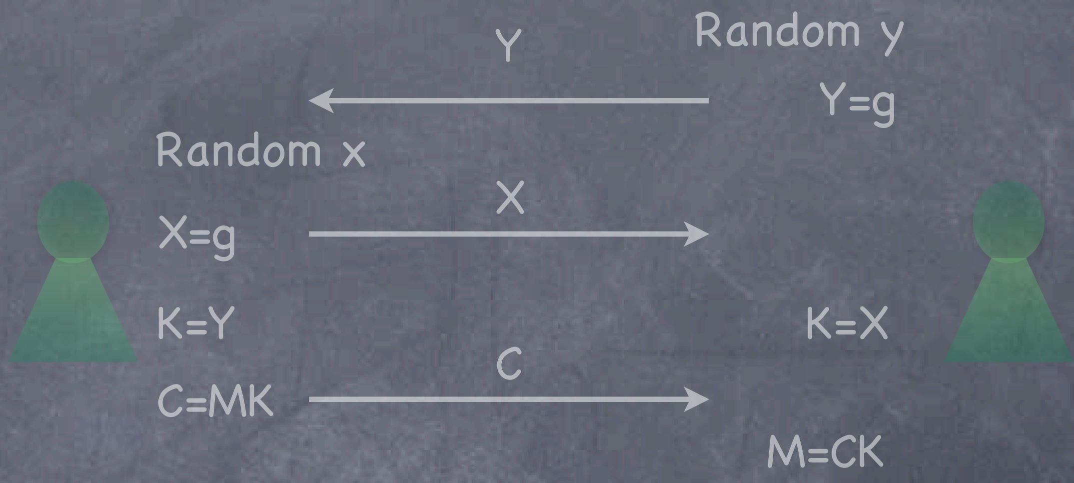
KeyGen: PK=(G,g,Y), SK=(G,g,y)

Enc

Dec

Abstracting El Gamal

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



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

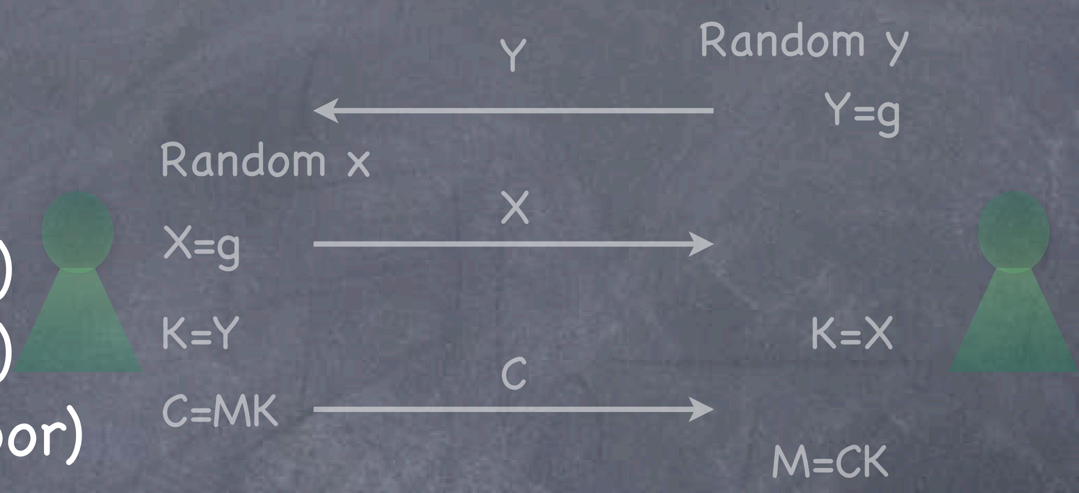
Enc

Dec

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)

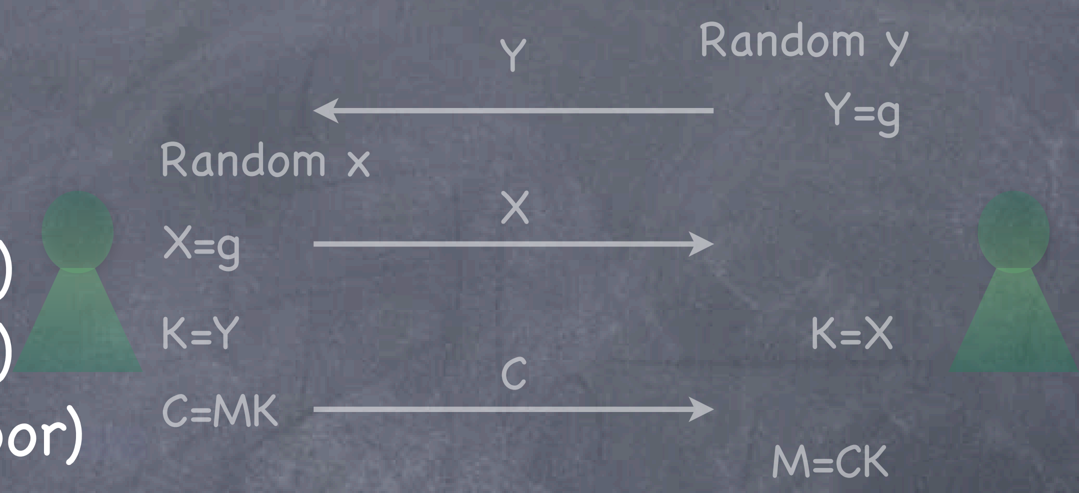
Enc

Dec

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

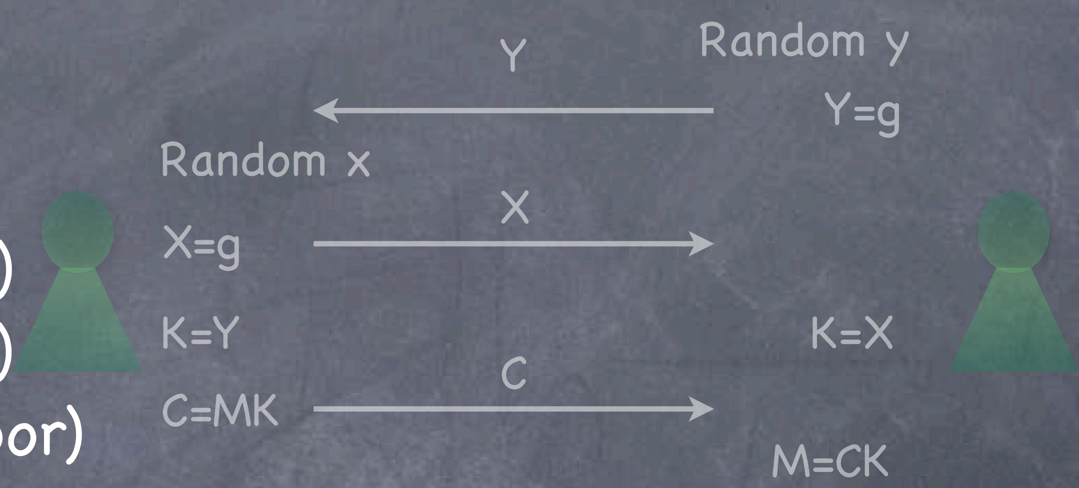
Dec

KeyGen: (PK,SK)

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

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)

Enc

Dec

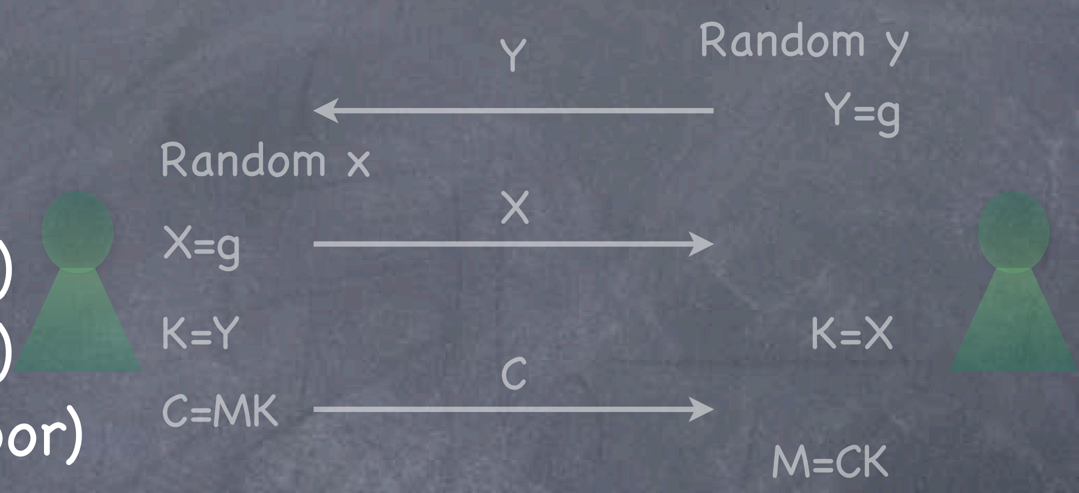
KeyGen: (PK,SK)

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

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

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



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

Enc

Dec

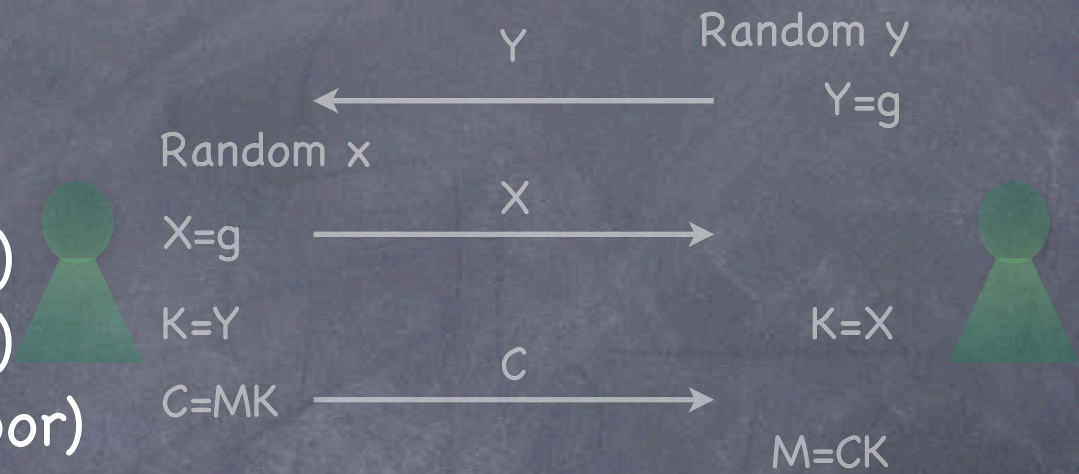
KeyGen: (PK,SK)

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

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

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



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

Enc

Dec

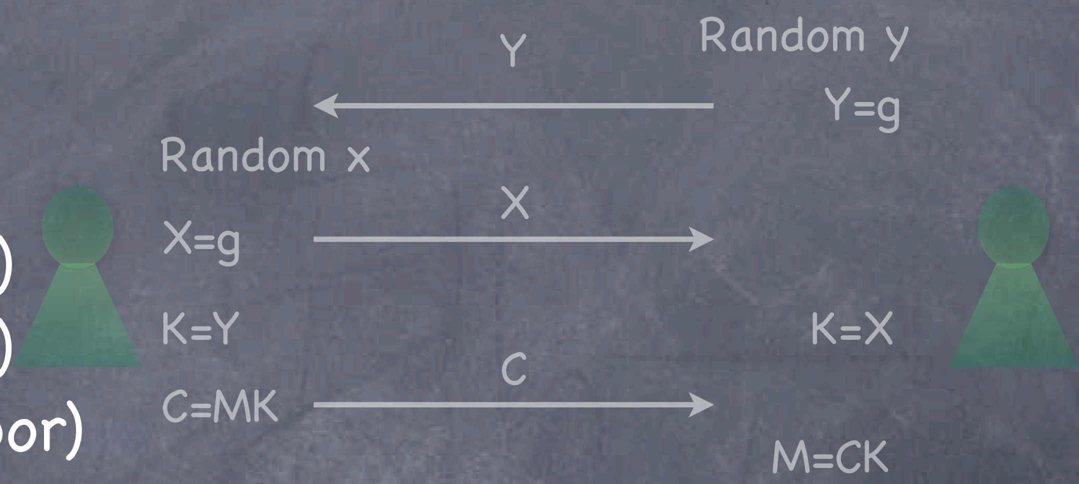
KeyGen: (PK,SK)

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

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

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

Dec

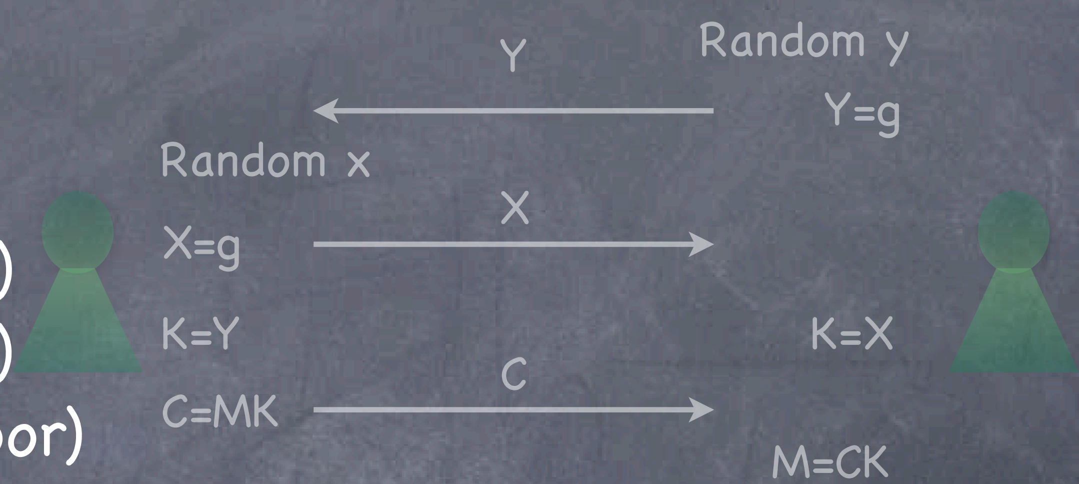
KeyGen: (PK,SK)

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

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

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

Dec

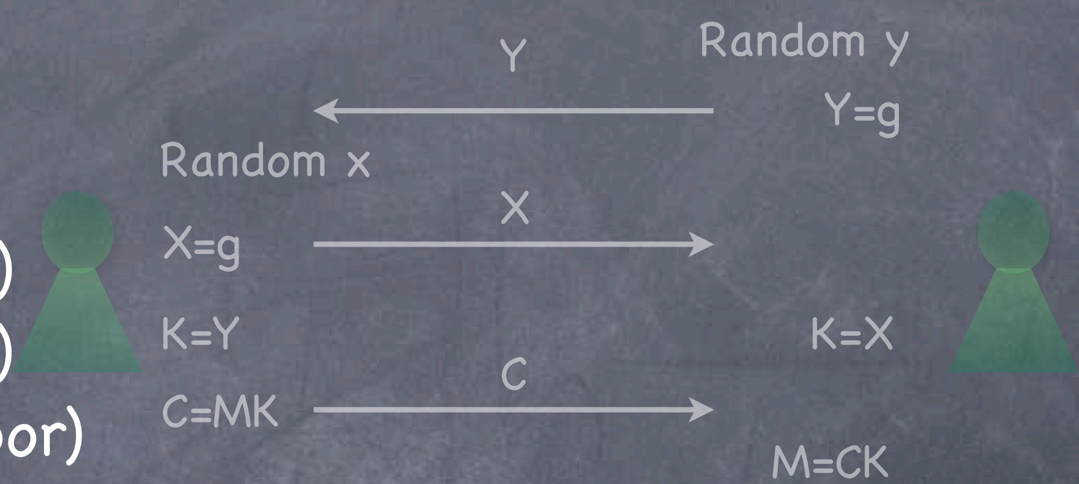
KeyGen: (PK,SK)

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

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

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

Dec

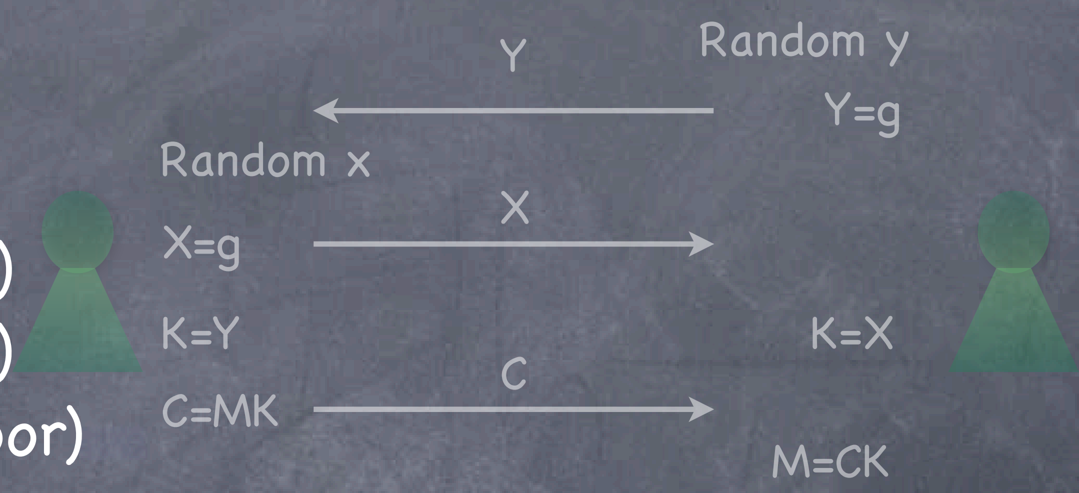
KeyGen: (PK,SK)

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

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

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

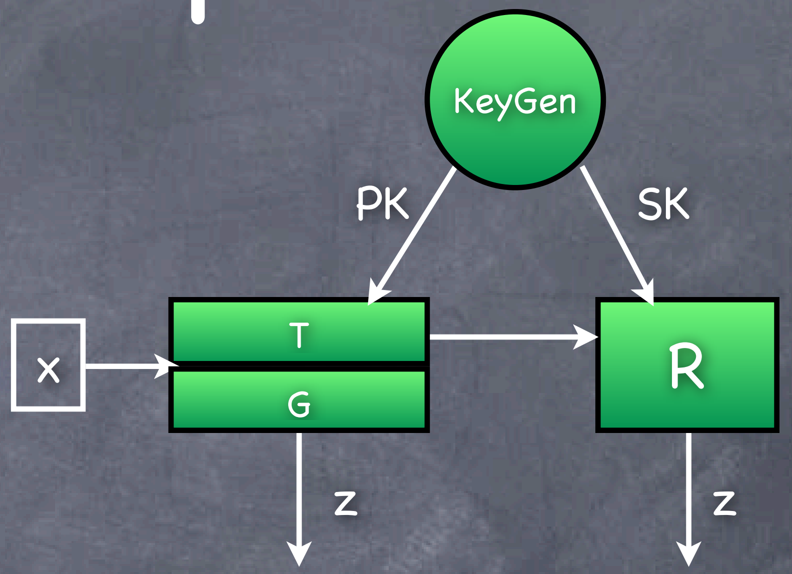
Dec

KeyGen: (PK,SK)

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

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

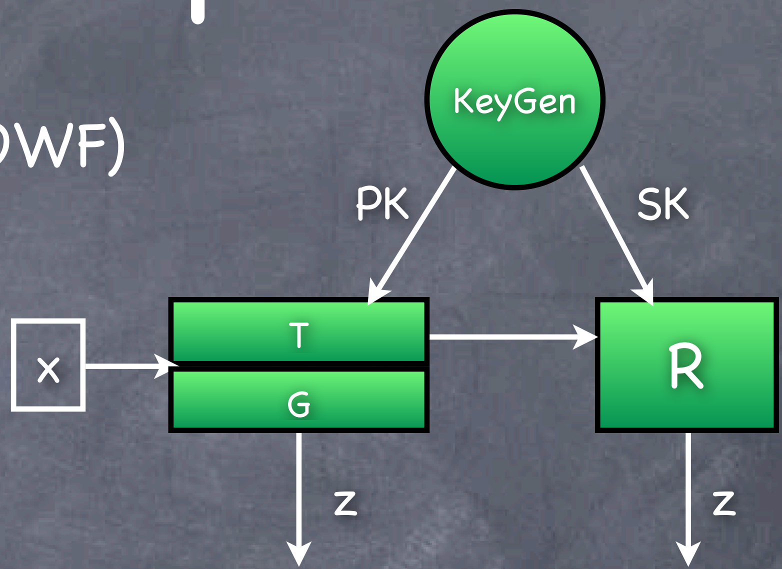
Trapdoor PRG from Generic Assumption?



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

Trapdoor PRG from Generic Assumption?

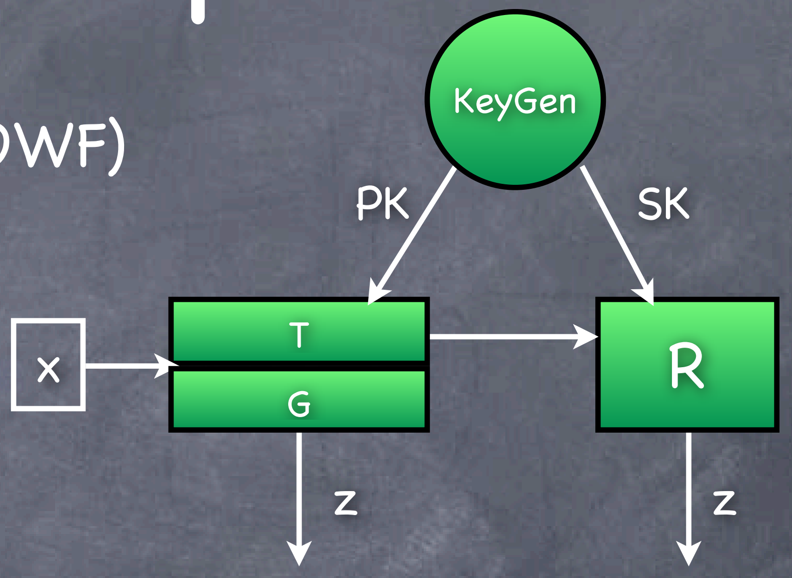
- PRG constructed from OWP (or OWF)



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

Trapdoor PRG from Generic Assumption?

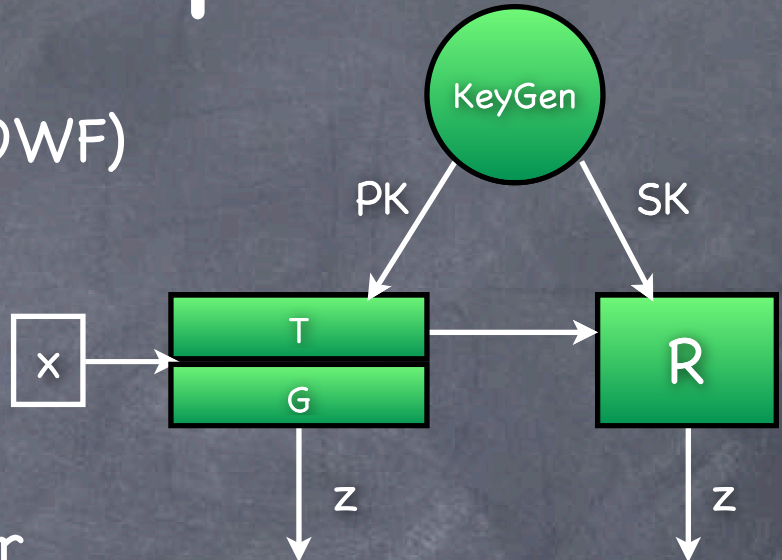
- PRG constructed from OWP (or OWF)
  - Allows us to instantiate the construction with several candidates



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

Trapdoor PRG from Generic Assumption?

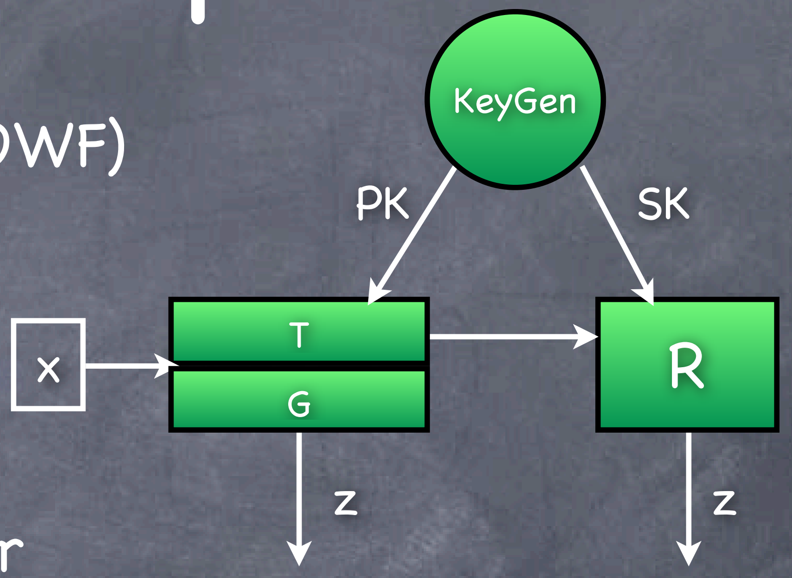
- PRG constructed from OWP (or OWF)
  - Allows us to instantiate the construction with several candidates
- Is there a similar construction for TPRG from OWP?



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

Trapdoor PRG from Generic Assumption?

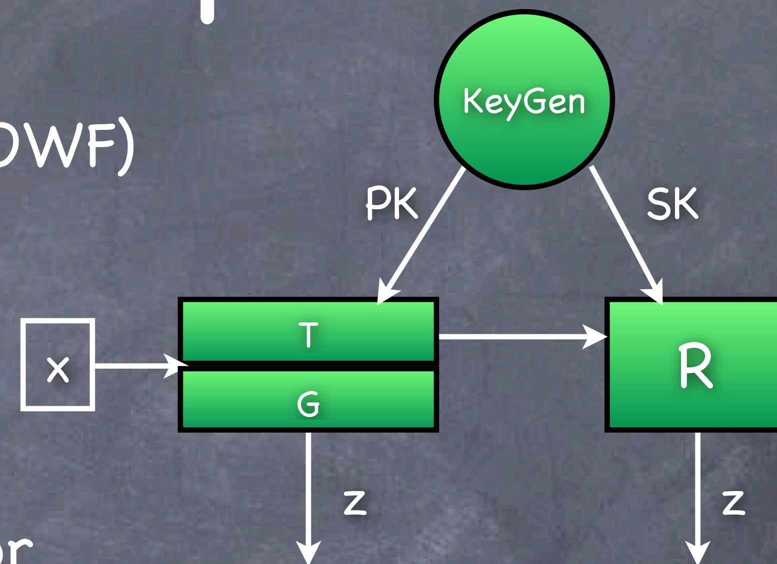
- PRG constructed from OWP (or OWF)
  - Allows us to instantiate the construction with several candidates
- Is there a similar construction for TPRG from OWP?
  - Trapdoor property seems fundamentally different: generic OWP does not suffice



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

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 does not suffice
  - 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

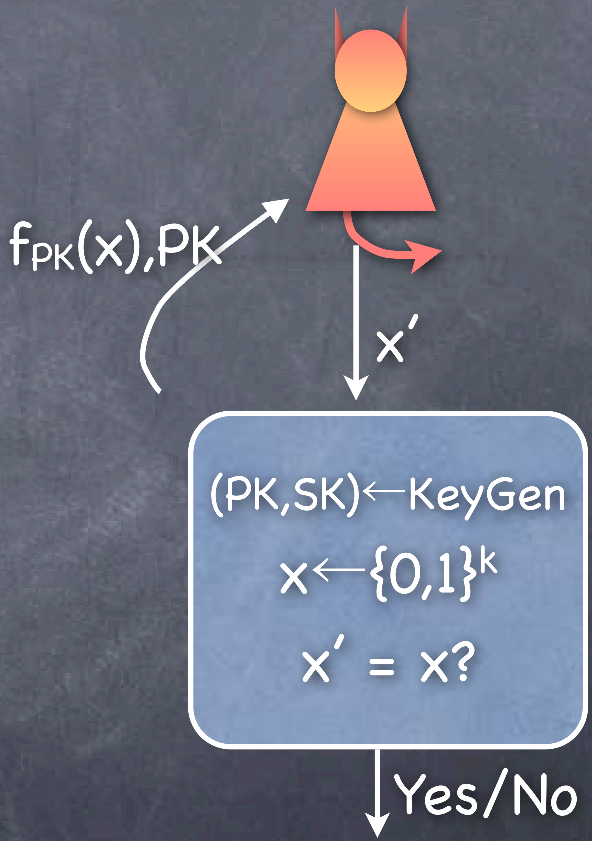
- (KeyGen,f,f') (all PPT) is a trapdoor one-way permutation (TOWP) if
  - For all (PK,SK) ←KeyGen
    - f_{PK} a permutation
    - f'_{SK} is the inverse of f_{PK}

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

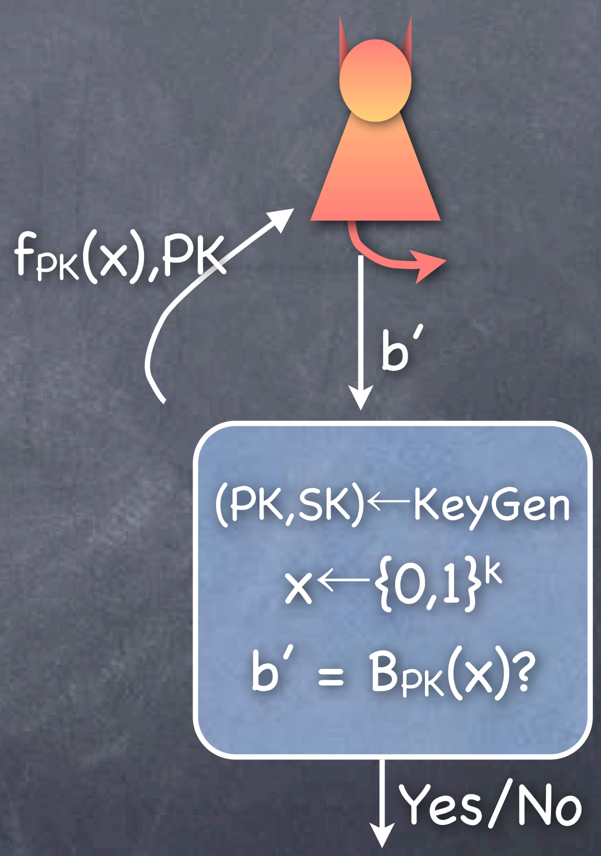
Trapdoor OWP

- (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

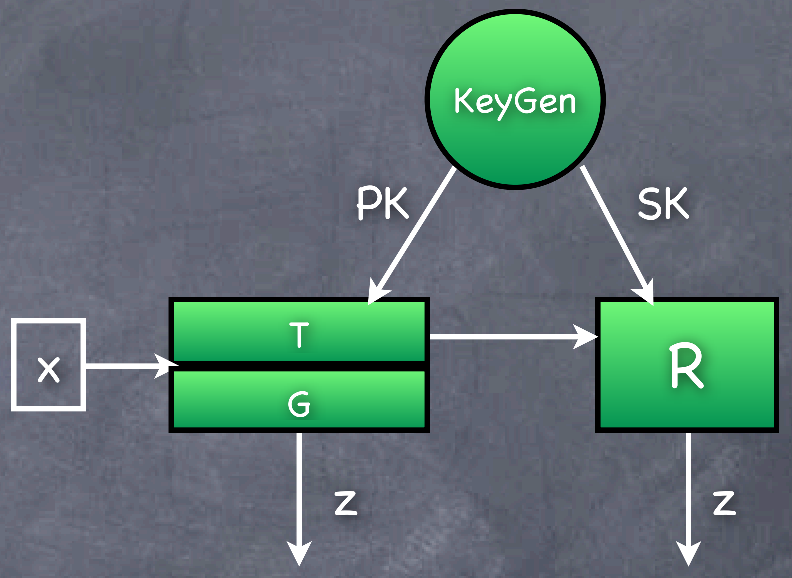


Trapdoor OWP

- (KeyGen,f,f') (all PPT) is a trapdoor one-way permutation (TOWP) if
  - For all (PK,SK) $\leftarrow$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. $(PK,f_{PK}(x),B_{PK}(x)) \approx (PK,f_{PK}(x),r)$



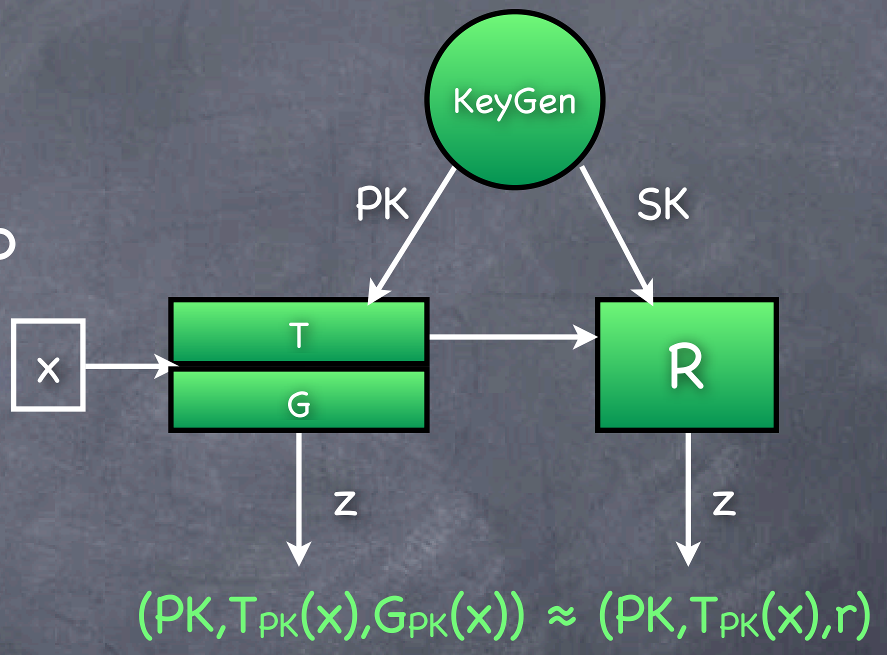
Trapdoor PRG from Trapdoor OWP



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

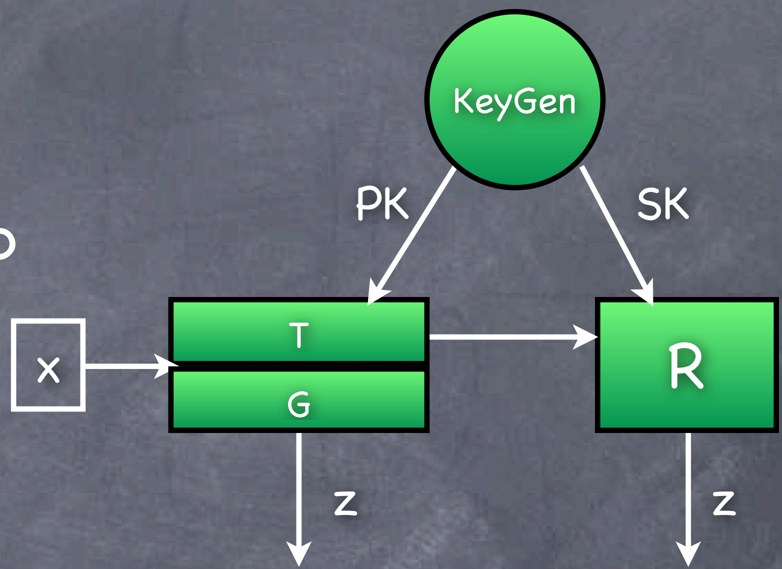
Trapdoor PRG from Trapdoor OWP

- Same construction as PRG from OWP



Trapdoor PRG from Trapdoor OWP

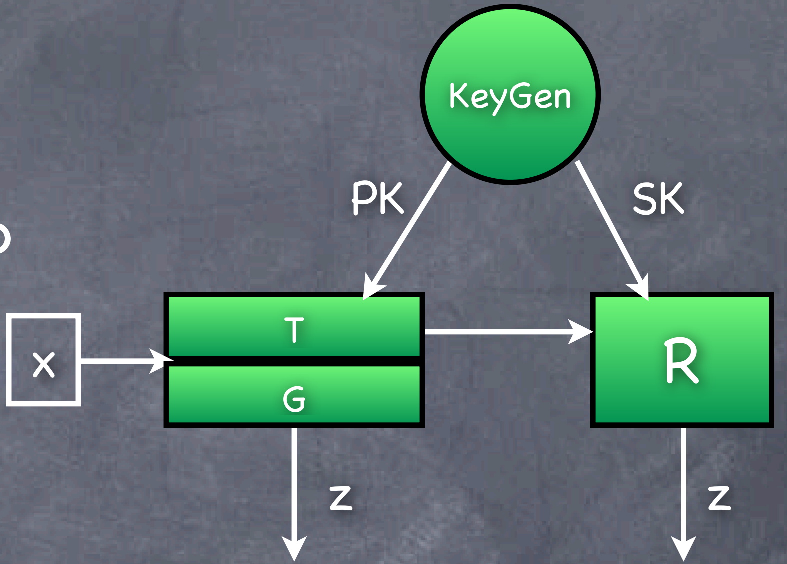
- Same construction as PRG from OWP
- One bit TPRG



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

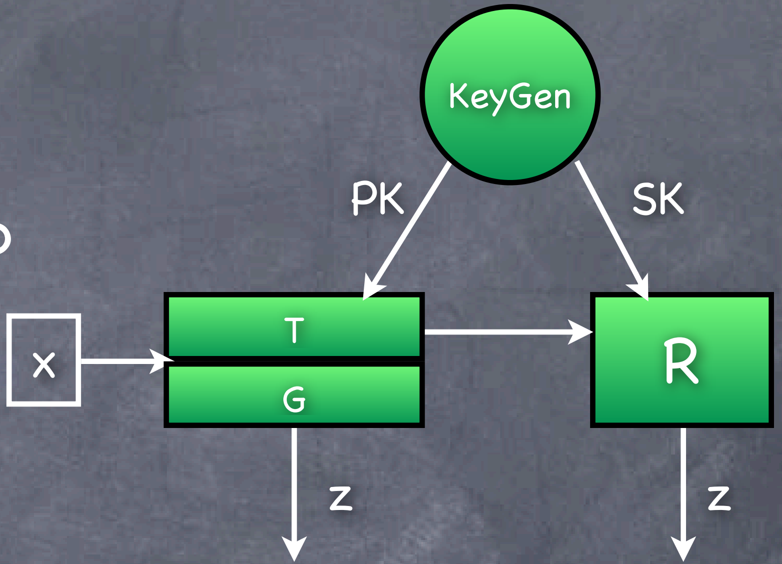
Trapdoor PRG from Trapdoor OWP

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



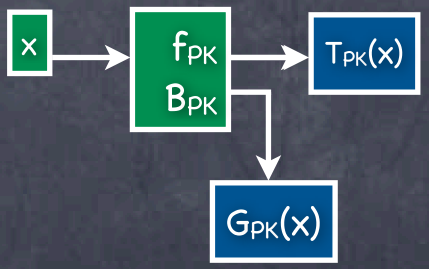
$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$

Trapdoor PRG from Trapdoor OWP

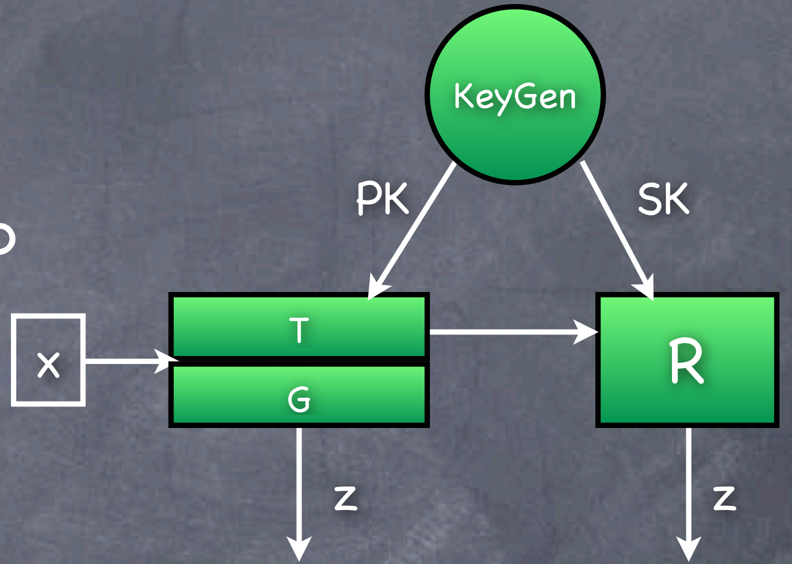


$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$

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



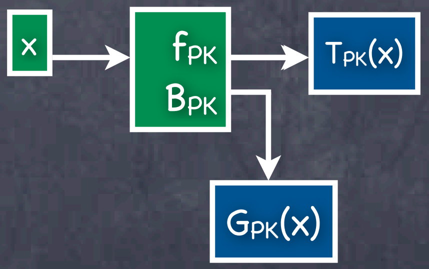
Trapdoor PRG from Trapdoor OWP



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

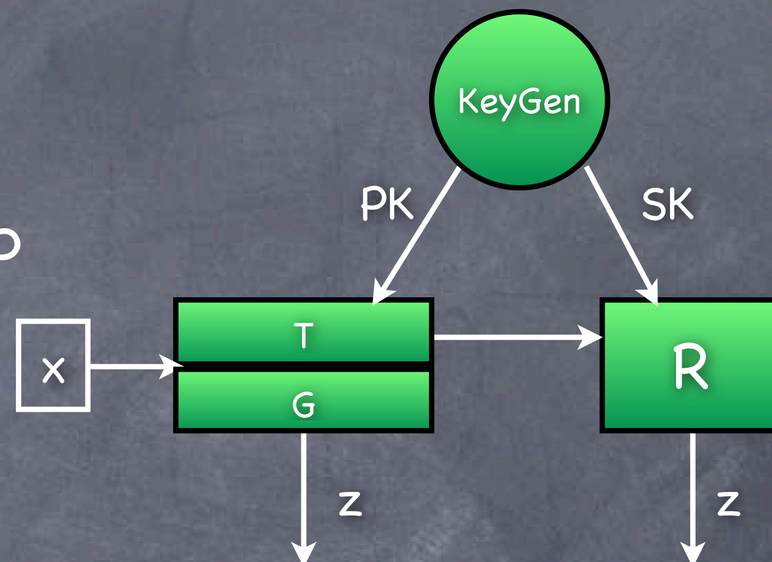
$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$

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



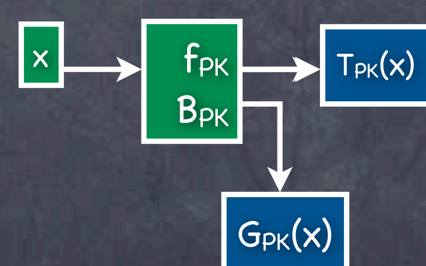
Trapdoor PRG from Trapdoor OWP

- 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)



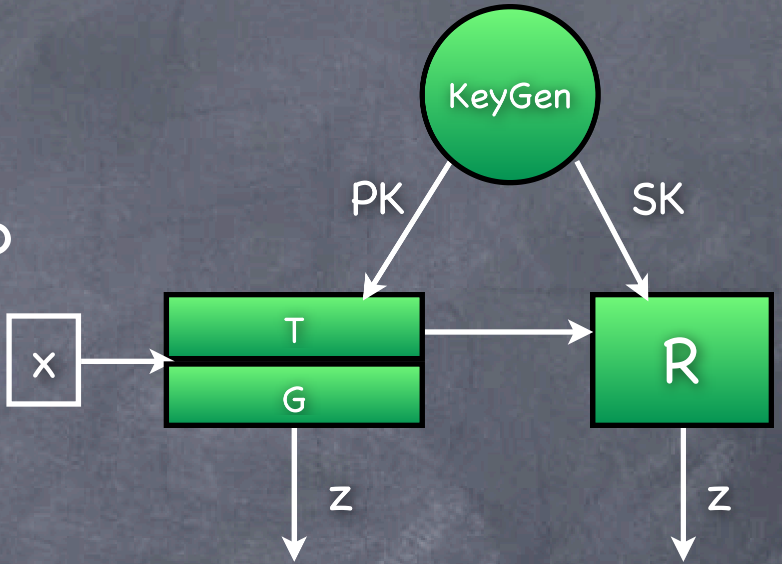
$$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$$

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



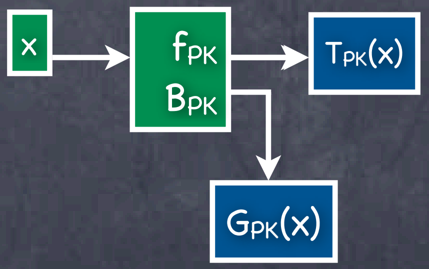
Trapdoor PRG from Trapdoor OWP

- 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}

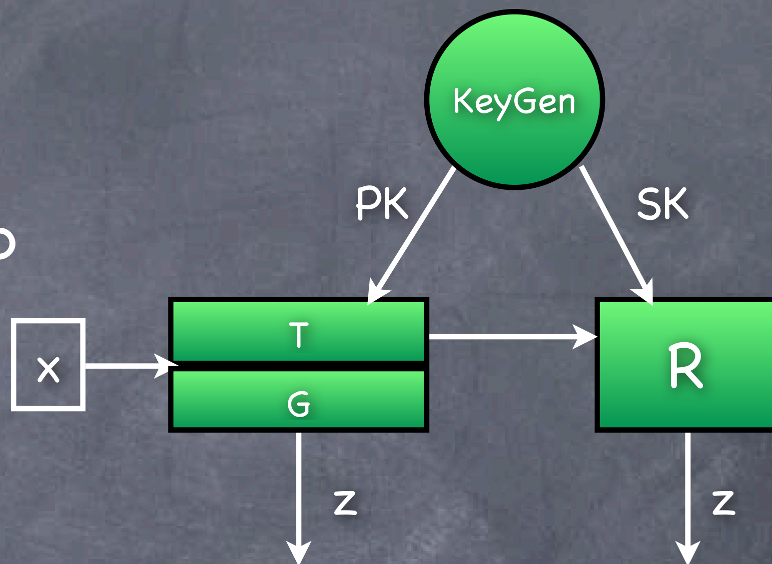


$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$

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



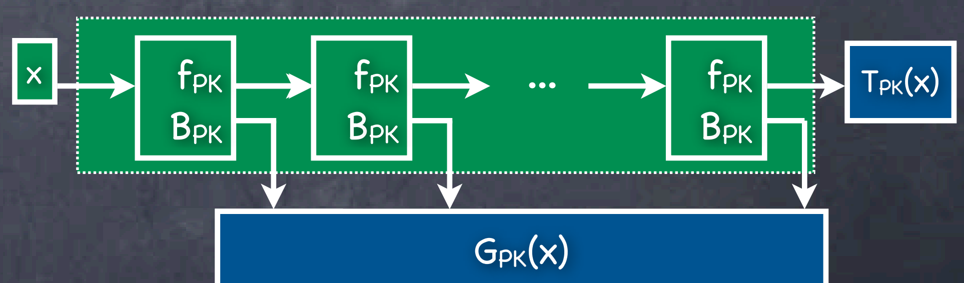
Trapdoor PRG from Trapdoor OWP



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

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

- 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}



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 $\equiv 3 \pmod 4$

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 $\equiv 3 \pmod 4$
    - 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 $\equiv 3 \pmod 4$
    - Fact: Can invert $f_{Rabin}(.; N)$ given factorization of N
  - RSA function: $f_{RSA}(x; N,e) = x^e \bmod N$ where $N=PQ$, P,Q k-bit primes, e s.t. $\gcd(e, \phi(N)) = 1$ (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 $\equiv 3 \pmod 4$
    - Fact: Can invert $f_{Rabin}(.; N)$ given factorization of N
  - RSA function: $f_{RSA}(x; N,e) = x^e \bmod N$ where $N=PQ$, P,Q k-bit primes, e s.t. $\gcd(e, \phi(N)) = 1$ (and x uniform from {0...N})
    - 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 $\equiv 3 \pmod 4$
    - Fact: Can invert $f_{Rabin}(.; N)$ given factorization of N
  - RSA function: $f_{RSA}(x; N,e) = x^e \bmod N$ where $N=PQ$, P,Q k-bit primes, e s.t. $\gcd(e, \phi(N)) = 1$ (and x uniform from $\{0...N\}$)
    - 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 $\equiv 3 \pmod 4$
    - Fact: Can invert $f_{Rabin}(.; N)$ given factorization of N
  - RSA function: $f_{RSA}(x; N,e) = x^e \bmod N$ where $N=PQ$, P,Q k-bit primes, e s.t. $\gcd(e, \phi(N)) = 1$ (and x uniform from {0...N})

see handout

    - Fact: $f_{RSA}(.; N,e)$ is a permutation
    - Fact: While picking (N,e), can also pick d s.t. $x^{ed} = x$

Recap

Recap

- CPA-secure PKE

Recap

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

Recap

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

Recap

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

Recap

- 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

Recap

- 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

Recap

- 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

Recap

- 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

Recap

- 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

Recap

- 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

CCA Secure PKE

CCA Secure PKE

- In SKE, to get CCA security, we used a MAC

CCA Secure PKE

- In SKE, to get CCA security, we used a MAC
  - Bob would accept only messages from Alice

CCA Secure PKE

- In SKE, to get CCA security, we used a MAC
  - Bob would accept only messages from Alice
- But in PKE, Bob wants to receive messages from Eve as well

CCA Secure PKE

- In SKE, to get CCA security, we used a MAC
  - Bob would accept only messages from Alice
- But in PKE, Bob wants to receive messages from Eve as well
  - Only if it is indeed Eve's own message: she should know her own message!

Chosen Ciphertext Attack

Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure

Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure

A subtle
e-mail attack

Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure

A subtle
e-mail attack

Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure

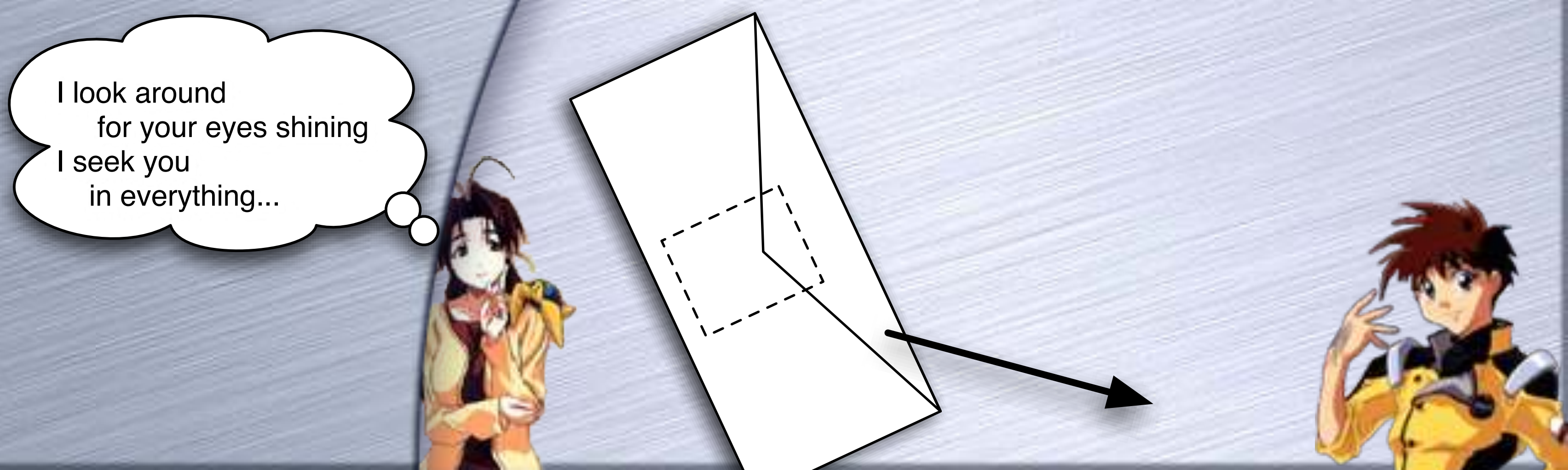
A subtle
e-mail attack

Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure

A subtle
e-mail attack

Alice → Bob: Enc(m)

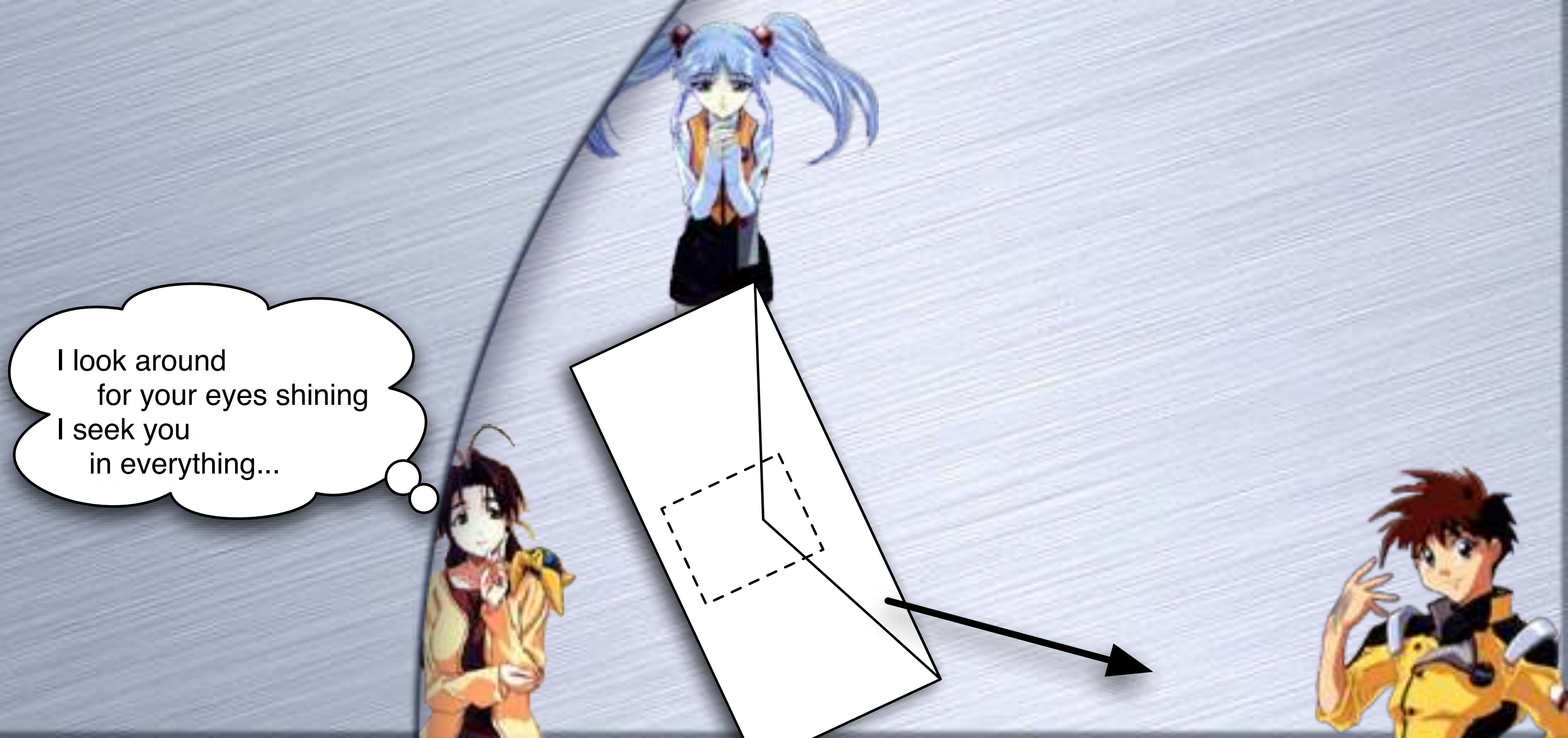


Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure

A subtle
e-mail attack

Alice → Bob: Enc(m)

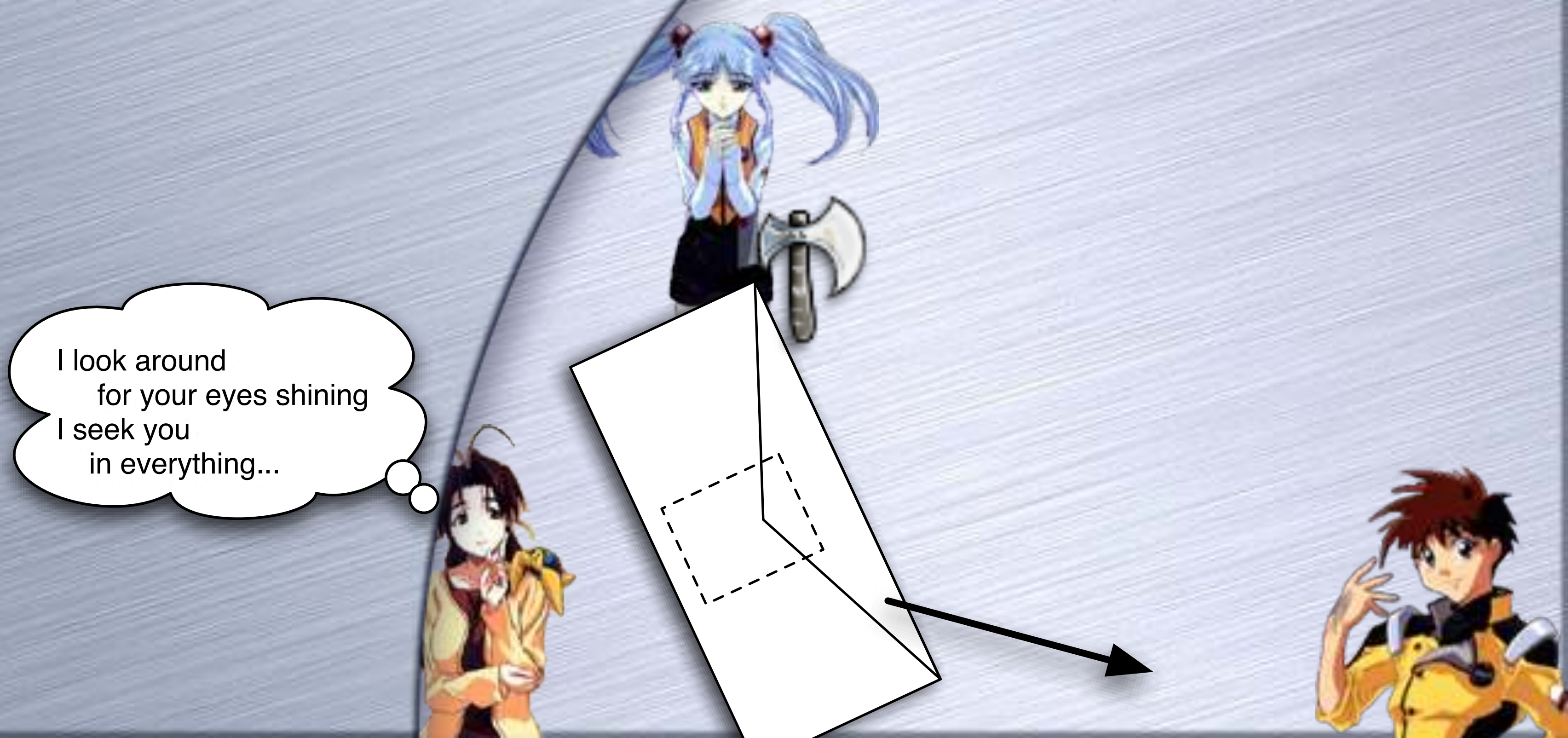


Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure

A subtle
e-mail attack

Alice → Bob: Enc(m)

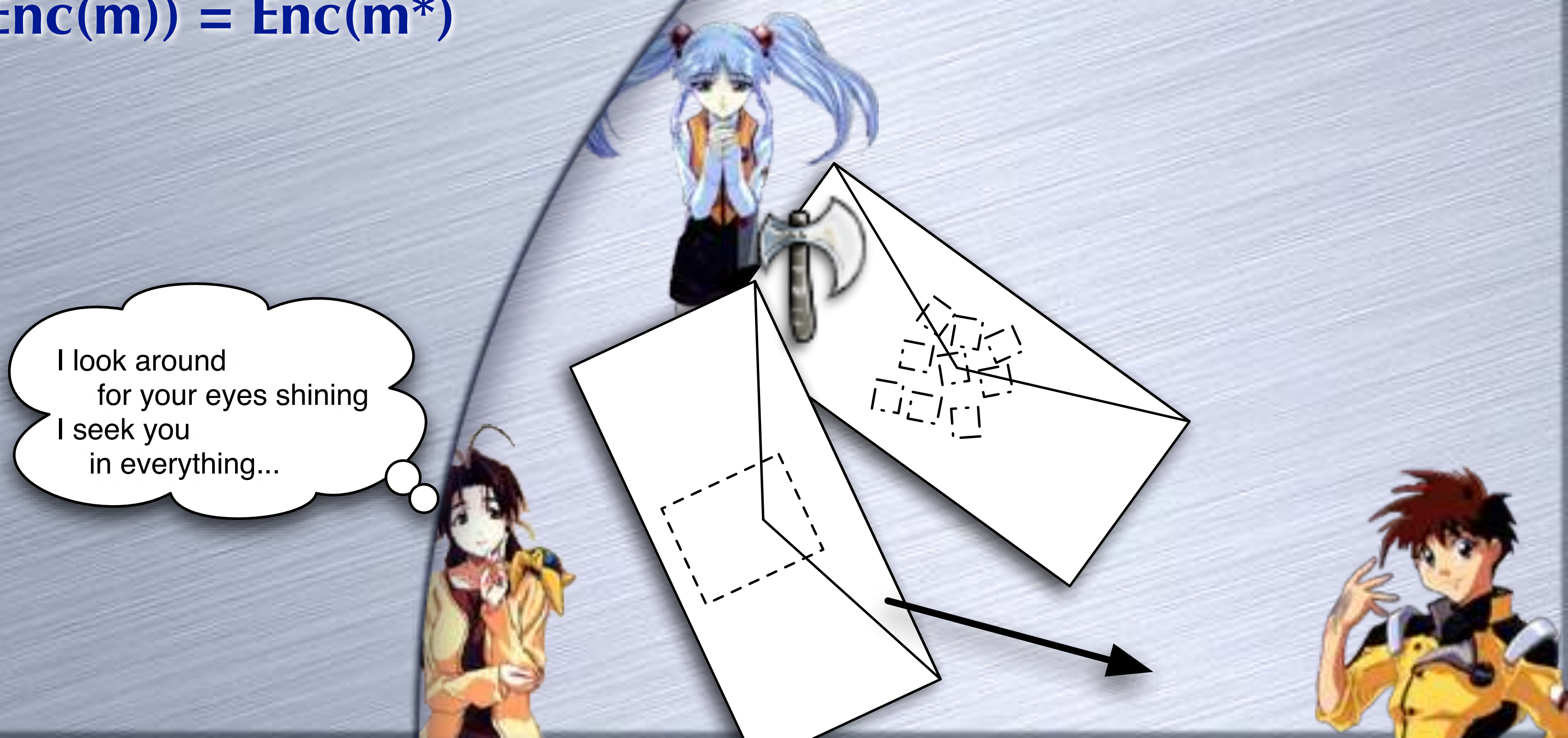


Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure

A subtle
e-mail attack

Alice → Bob: Enc(m)
Eve: Hack(Enc(m)) = Enc(m*)

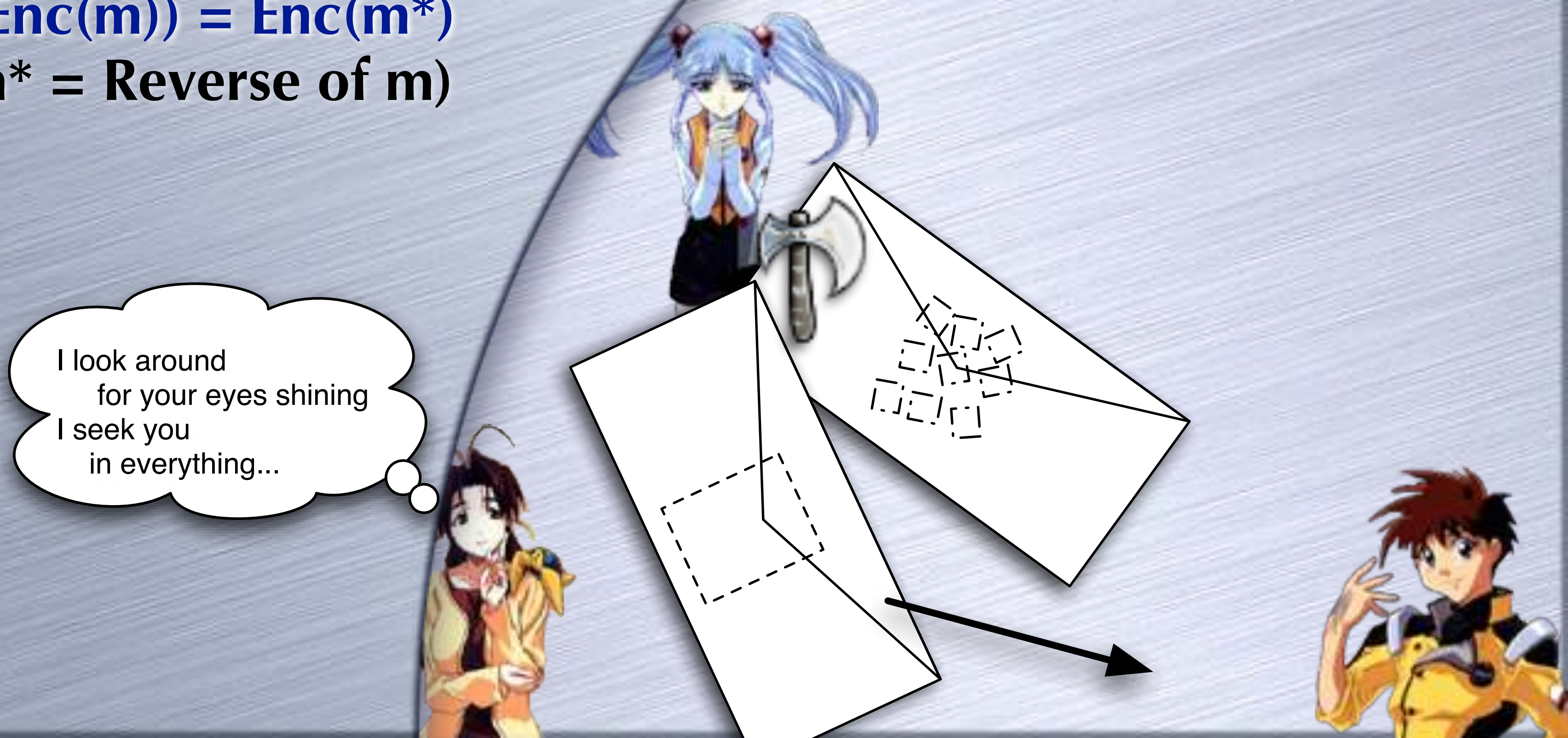


Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure

A subtle
e-mail attack

Alice → Bob: Enc(m)
Eve: Hack(Enc(m)) = Enc(m*)
(where m* = Reverse of m)

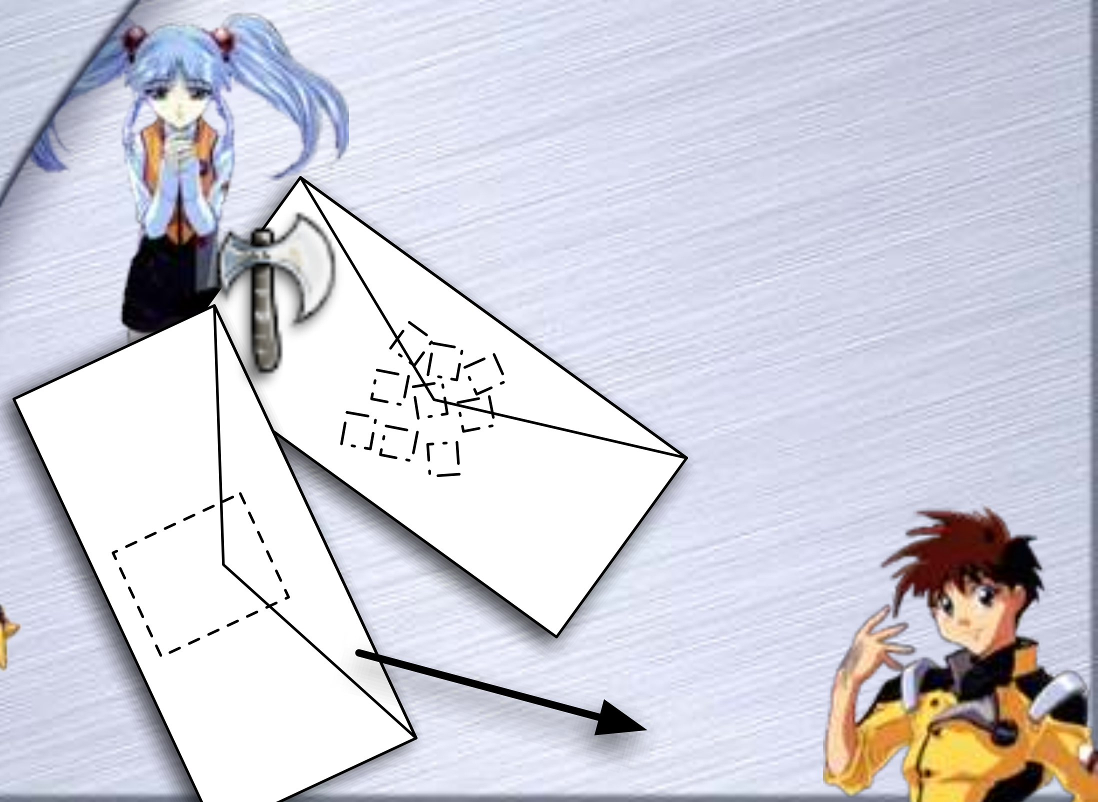


Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure
  - Suppose encrypts a character at a time (still secure)

A subtle e-mail attack

Alice → Bob: Enc(m)
Eve: Hack(Enc(m)) = Enc(m*)
(where m* = Reverse of m)

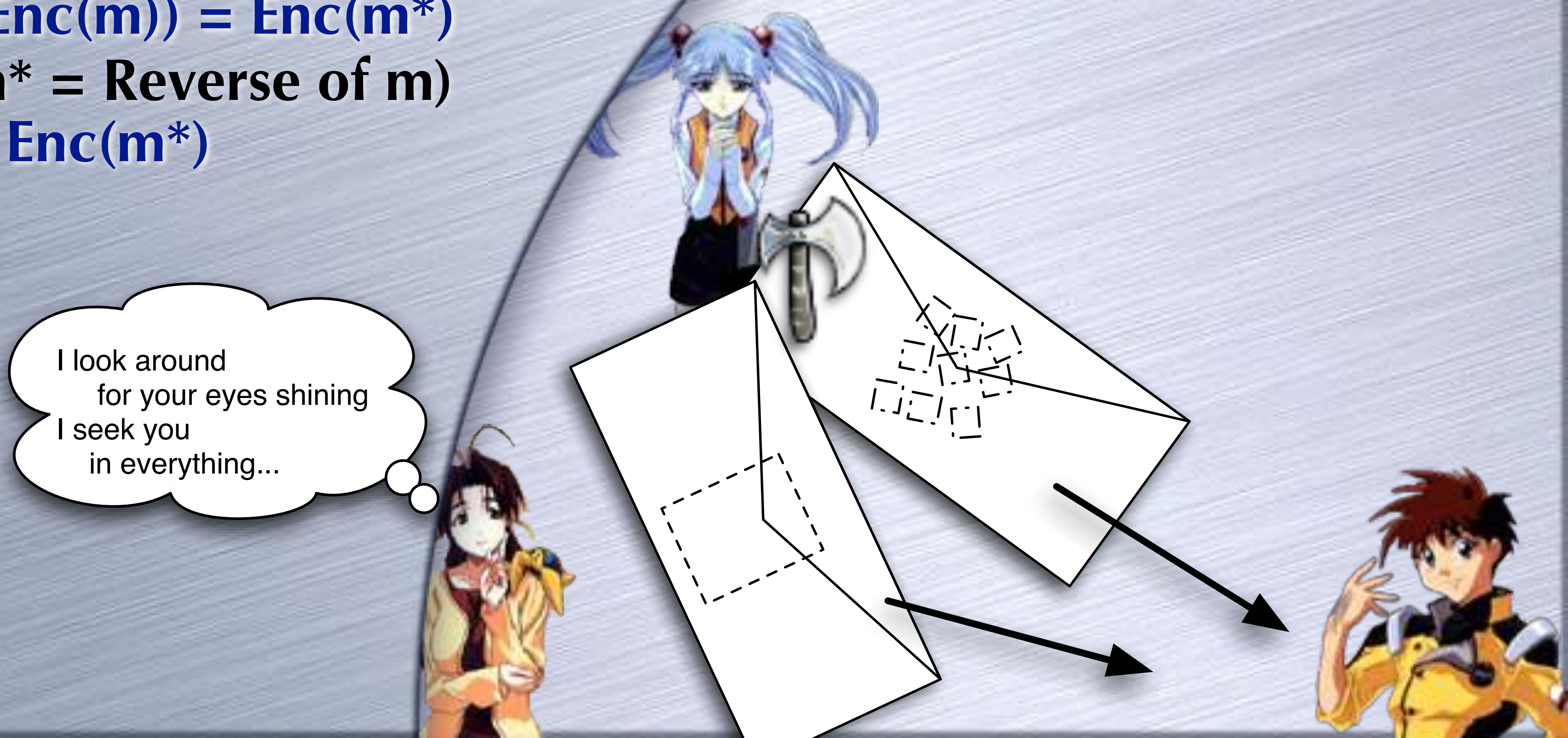


Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure
  - Suppose encrypts a character at a time (still secure)

A subtle
e-mail attack

Alice → Bob: Enc(m)
Eve: Hack(Enc(m)) = Enc(m*)
(where m* = Reverse of m)
Eve → Bob: Enc(m*)

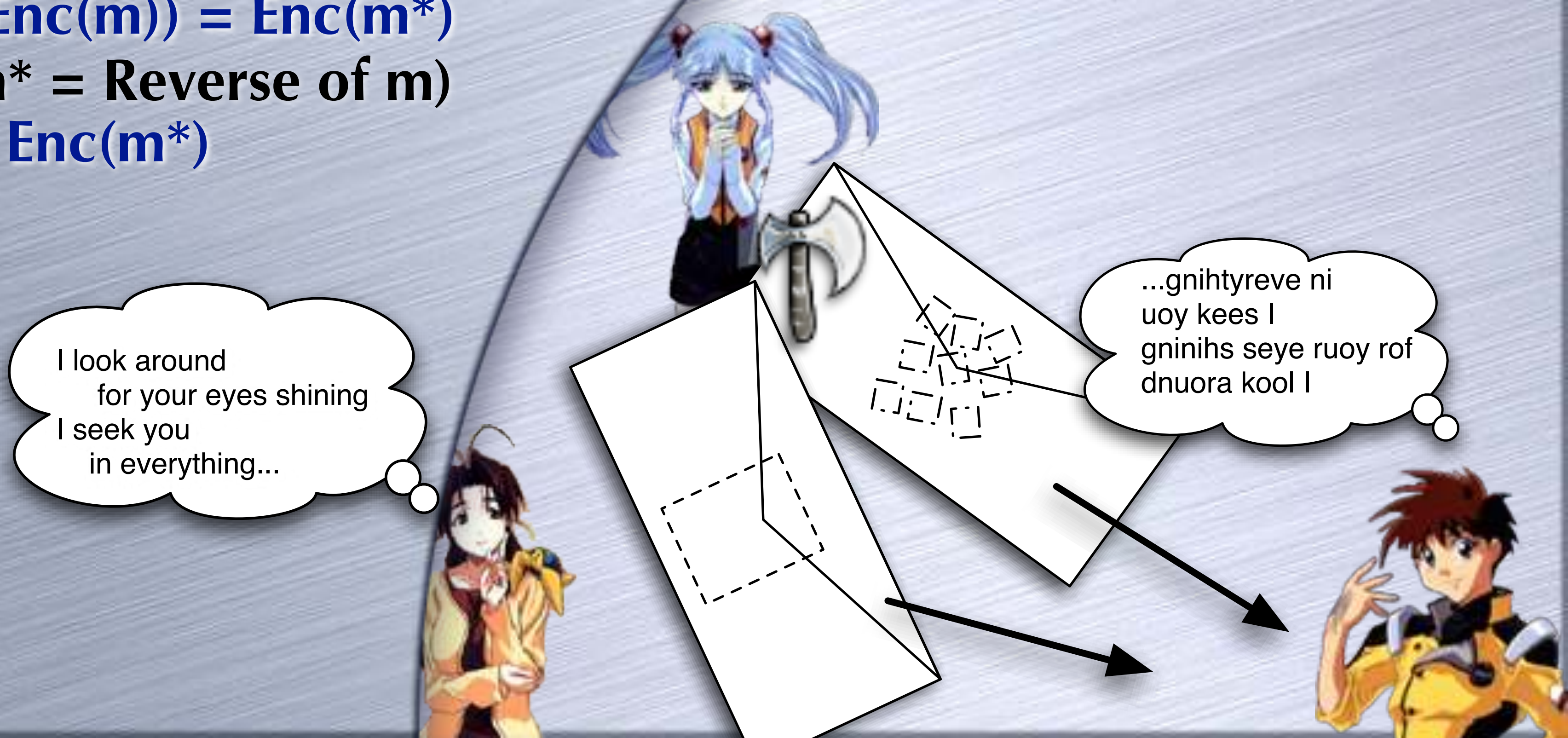


Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure
  - Suppose encrypts a character at a time (still secure)

A subtle
e-mail attack

Alice → Bob: Enc(m)
Eve: Hack(Enc(m)) = Enc(m*)
(where m* = Reverse of m)
Eve → Bob: Enc(m*)

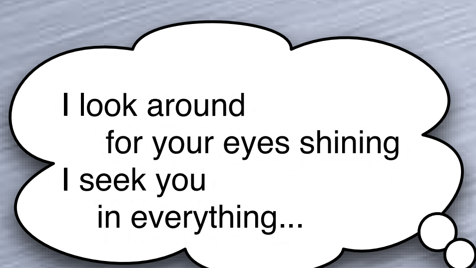


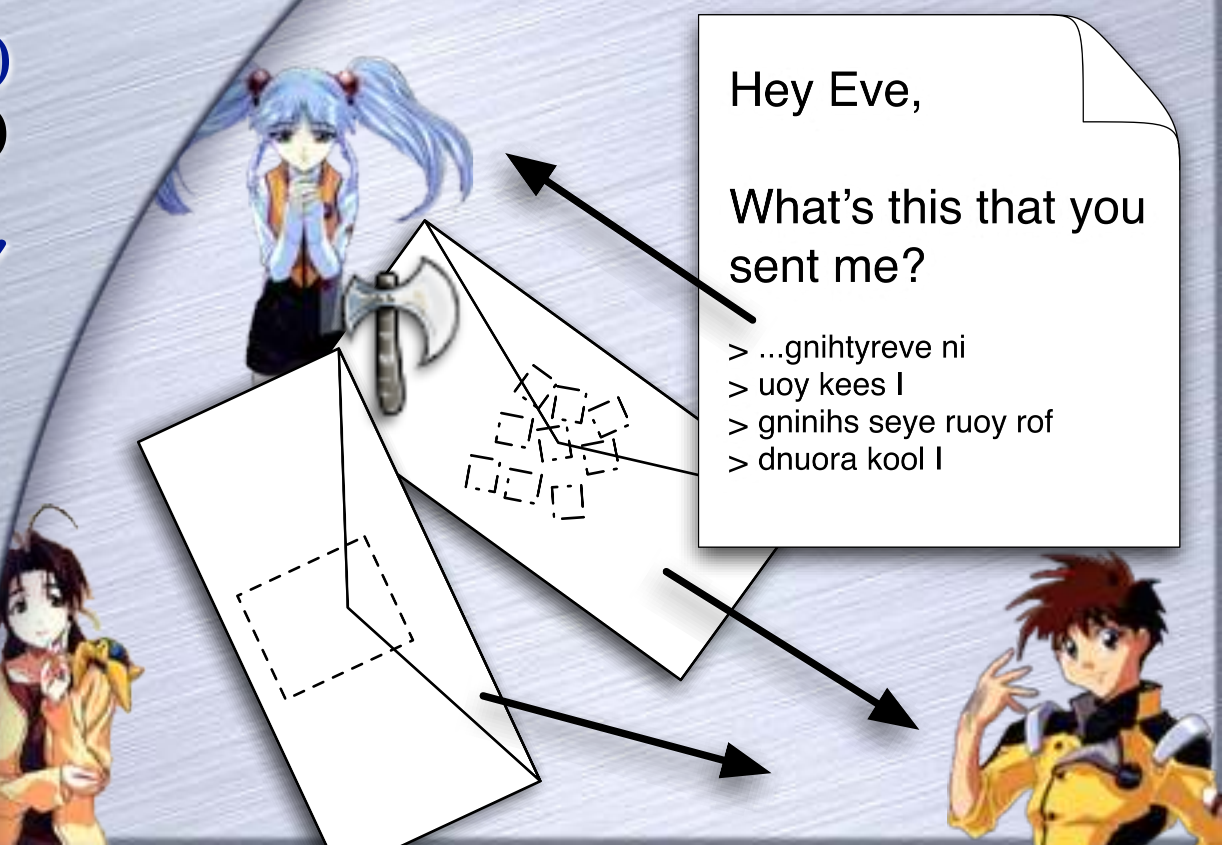
Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure
  - Suppose encrypts a character at a time (still secure)

A subtle
e-mail attack

Alice → Bob: Enc(m)
Eve: Hack(Enc(m)) = Enc(m*)
(where m* = Reverse of m)
Eve → Bob: Enc(m*)
Bob → Eve: “what’s this: m*?”



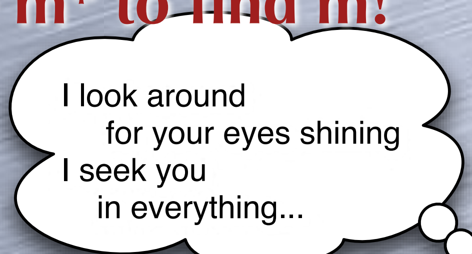


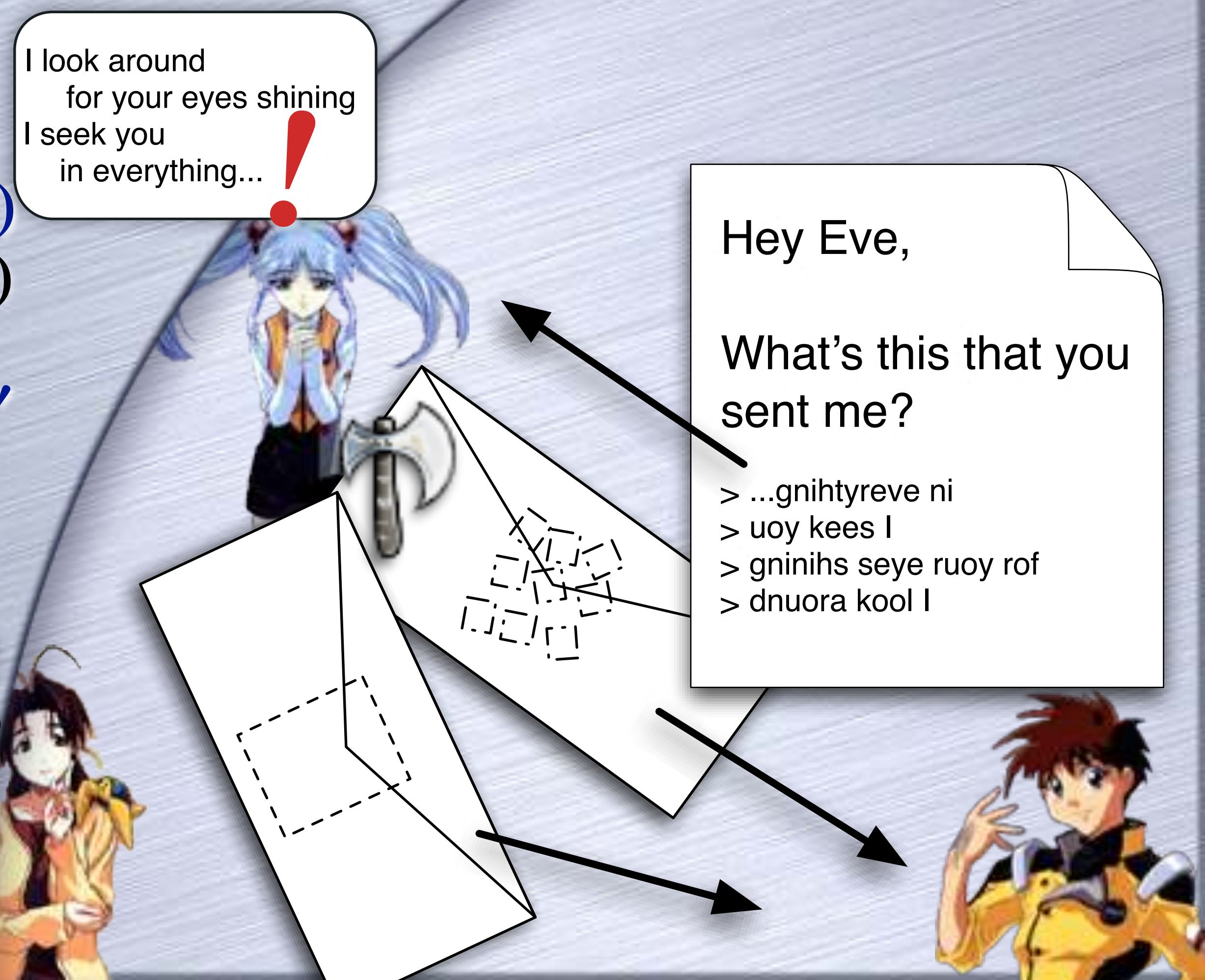
Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure
  - Suppose encrypts a character at a time (still secure)

A subtle
e-mail attack

Alice → Bob: Enc(m)
Eve: Hack(Enc(m)) = Enc(m*)
(where m* = Reverse of m)
Eve → Bob: Enc(m*)
Bob → Eve: "what's this: m*?"
Eve: Reverse m* to find m!





Malleability

Malleability

- Malleability: Eve can "malleate" a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a "related" message

Malleability

- Malleability: Eve can "malleate" a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a "related" message
- E.g.: Malleability of El Gamal

Malleability

- Malleability: Eve can "malleate" a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a "related" message
- E.g.: Malleability of El Gamal
  - Recall: $\mathrm{Enc}_{(G,g,Y)}(m) = (g^x, M.Y^x)$

Malleability

- Malleability: Eve can "malleate" a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a "related" message
- E.g.: Malleability of El Gamal
  - Recall: $\mathrm{Enc}_{(G,g,Y)}(m) = (g^x, M.Y^x)$
  - Given (X,C) change it to (X,TC): will decrypt to TM

Malleability

- Malleability: Eve can "malleate" a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a "related" message
- E.g.: Malleability of El Gamal
  - Recall: $\mathrm{Enc}_{(G,g,Y)}(m) = (g^x, M.Y^x)$
  - Given (X,C) change it to (X,TC): will decrypt to TM
  - Or change (X,C) to (X^a, C^a): will decrypt to M^a

Malleability

- Malleability: Eve can "malleate" a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a "related" message
- E.g.: Malleability of El Gamal
  - Recall: $\mathrm{Enc}_{(G,g,Y)}(m) = (g^x, M.Y^x)$
  - Given (X,C) change it to (X,TC): will decrypt to TM
  - Or change (X,C) to (X^a, C^a): will decrypt to M^a
- If chosen-ciphertext attack possible

Malleability

- Malleability: Eve can "malleate" a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a "related" message
- E.g.: Malleability of El Gamal
  - Recall: $Enc_{(G,g,Y)}(m) = (g^x, M.Y^x)$
  - Given (X,C) change it to (X,TC): will decrypt to TM
  - Or change (X,C) to (X^a, C^a): will decrypt to M^a
- If chosen-ciphertext attack possible
  - i.e., Eve can get a ciphertext of her choice decrypted

Malleability

- Malleability: Eve can "malleate" a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a "related" message
- E.g.: Malleability of El Gamal
  - Recall: $\mathrm{Enc}_{(G,g,Y)}(m) = (g^x, M.Y^x)$
  - Given (X,C) change it to (X,TC): will decrypt to TM
  - Or change (X,C) to (X^a, C^a): will decrypt to M^a
- If chosen-ciphertext attack possible
  - i.e., Eve can get a ciphertext of her choice decrypted
  - Then Eve can exploit malleability to learn something "related to" Alice's messages

Malleability

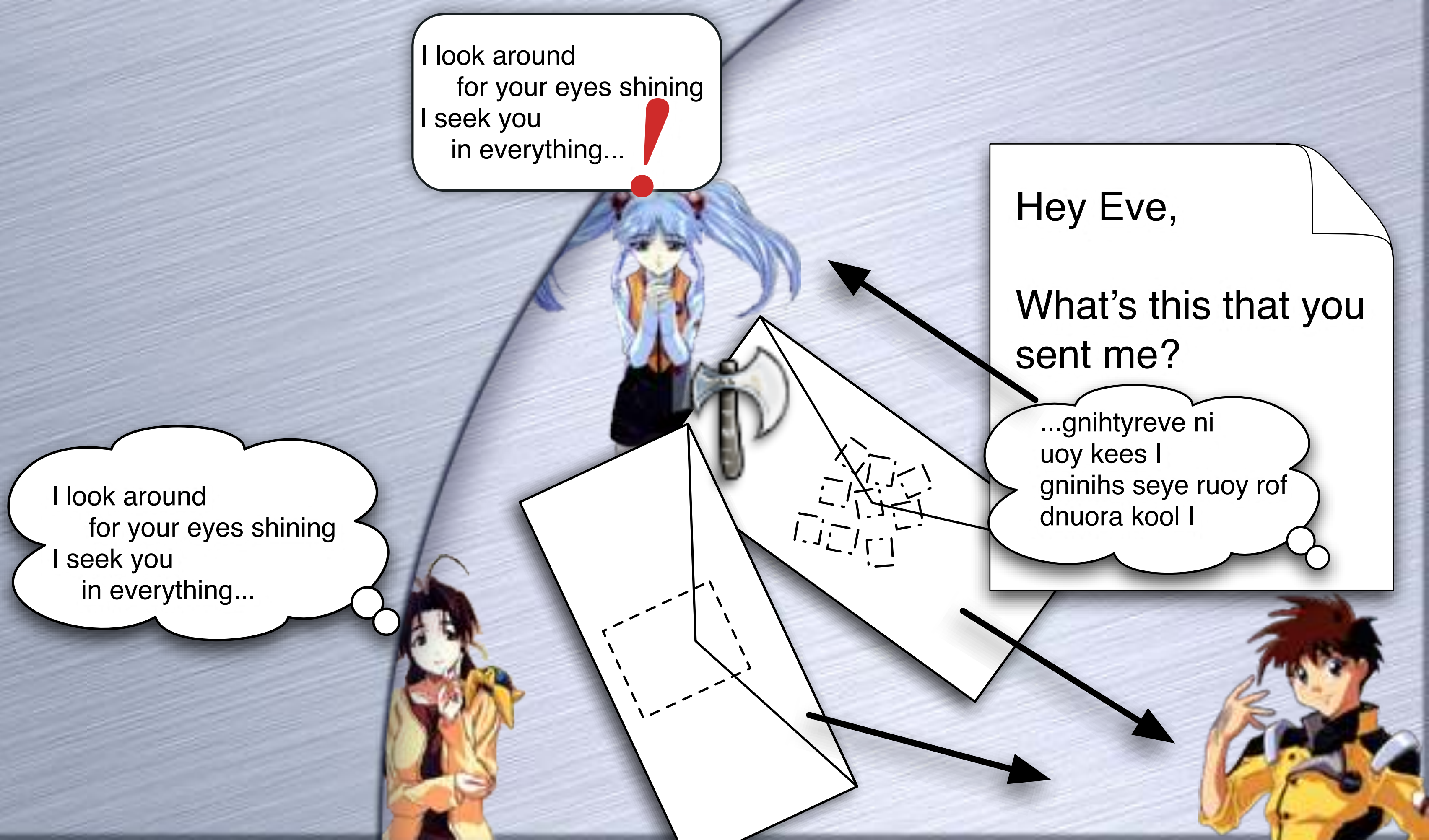
- Malleability: Eve can "malleate" a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a "related" message

More subtly, the 1 bit - valid or invalid - may leak information on message or SK

- E.g.: Malleability of El Gamal
  - Recall: $Enc_{(G,g,Y)}(m) = (g^x, M.Y^x)$
  - Given (X,C) change it to (X,TC): will decrypt to TM
  - Or change (X,C) to (X^a, C^a): will decrypt to M^a
- If chosen-ciphertext attack possible
  - i.e., Eve can get a ciphertext of her choice decrypted
  - Then Eve can exploit malleability to learn something "related to" Alice's messages

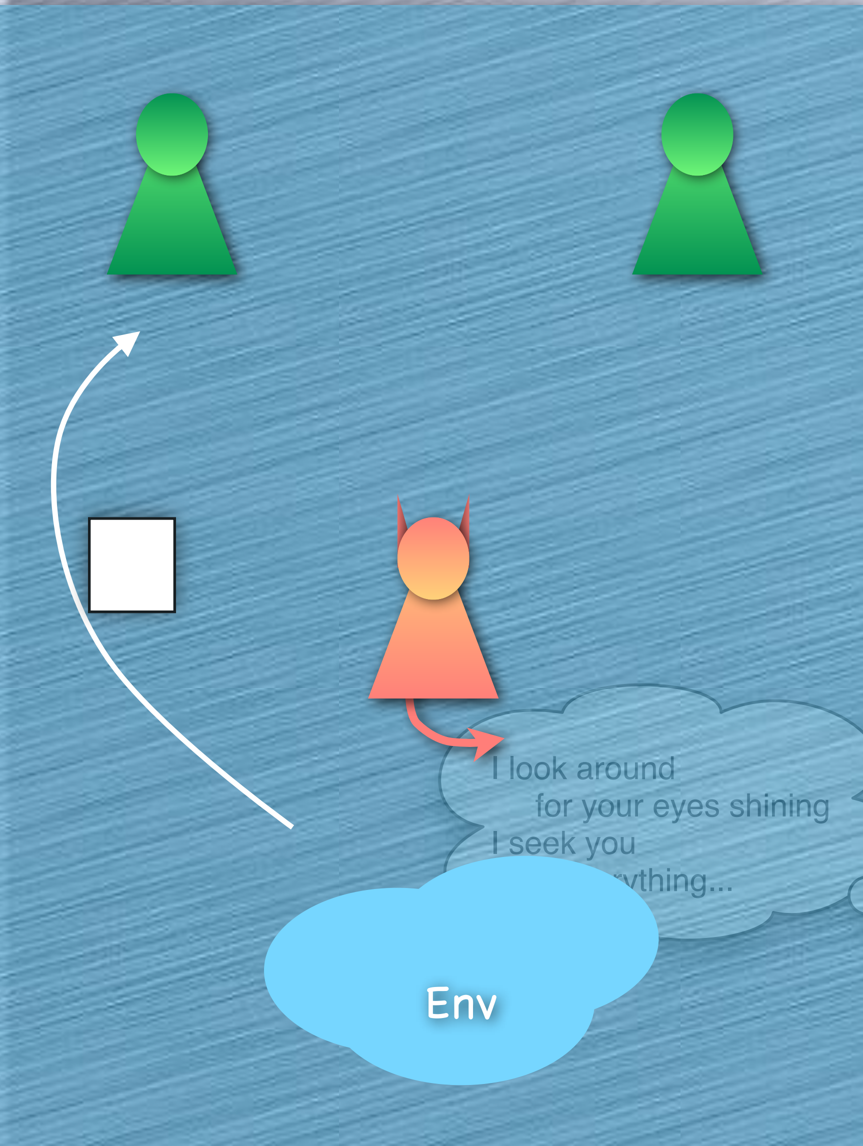
Chosen Ciphertext Attack

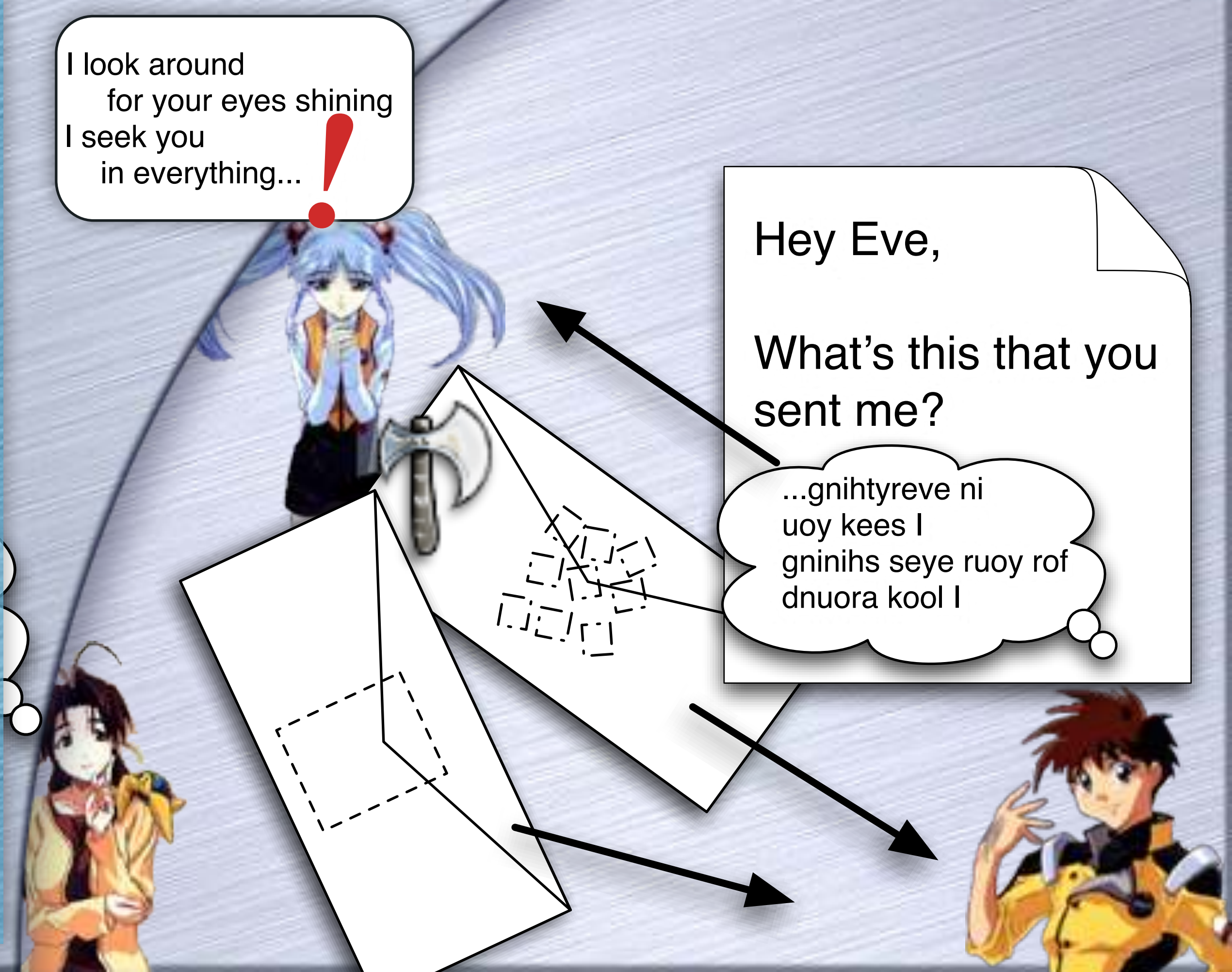
- SIM-CCA: does capture this attack



Chosen Ciphertext Attack

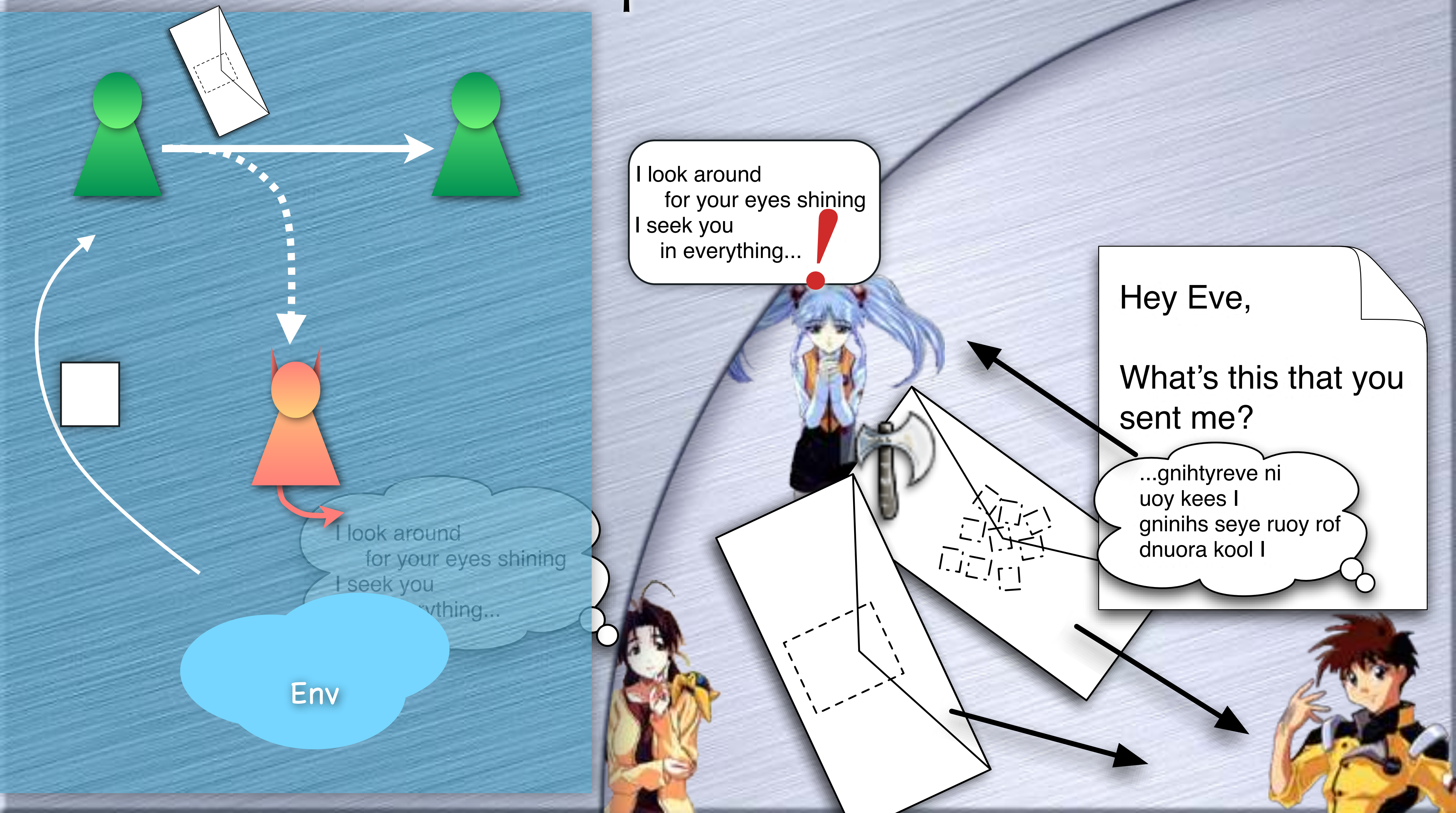
- SIM-CCA: does capture this attack





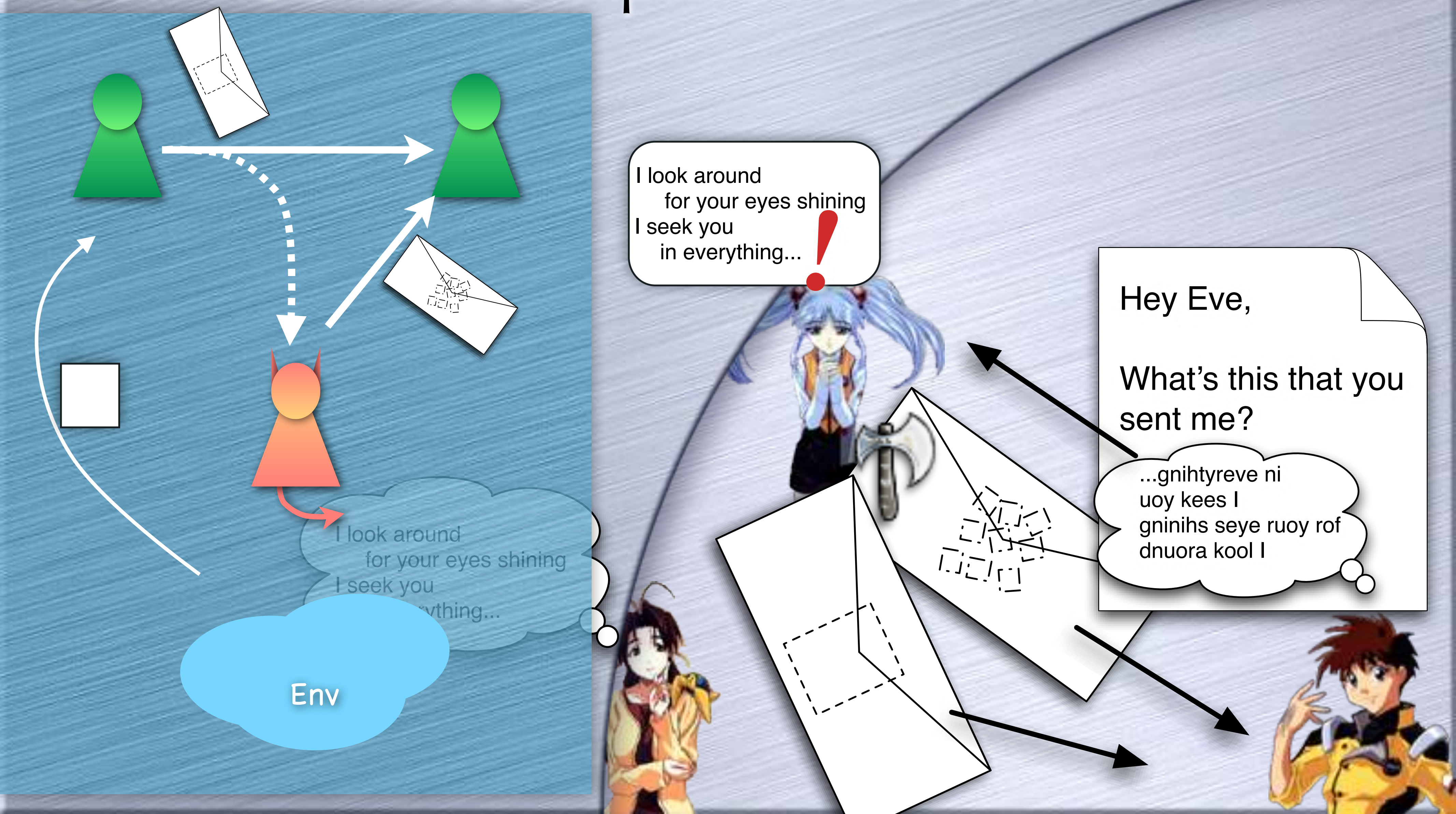
Chosen Ciphertext Attack

- SIM-CCA: does capture this attack



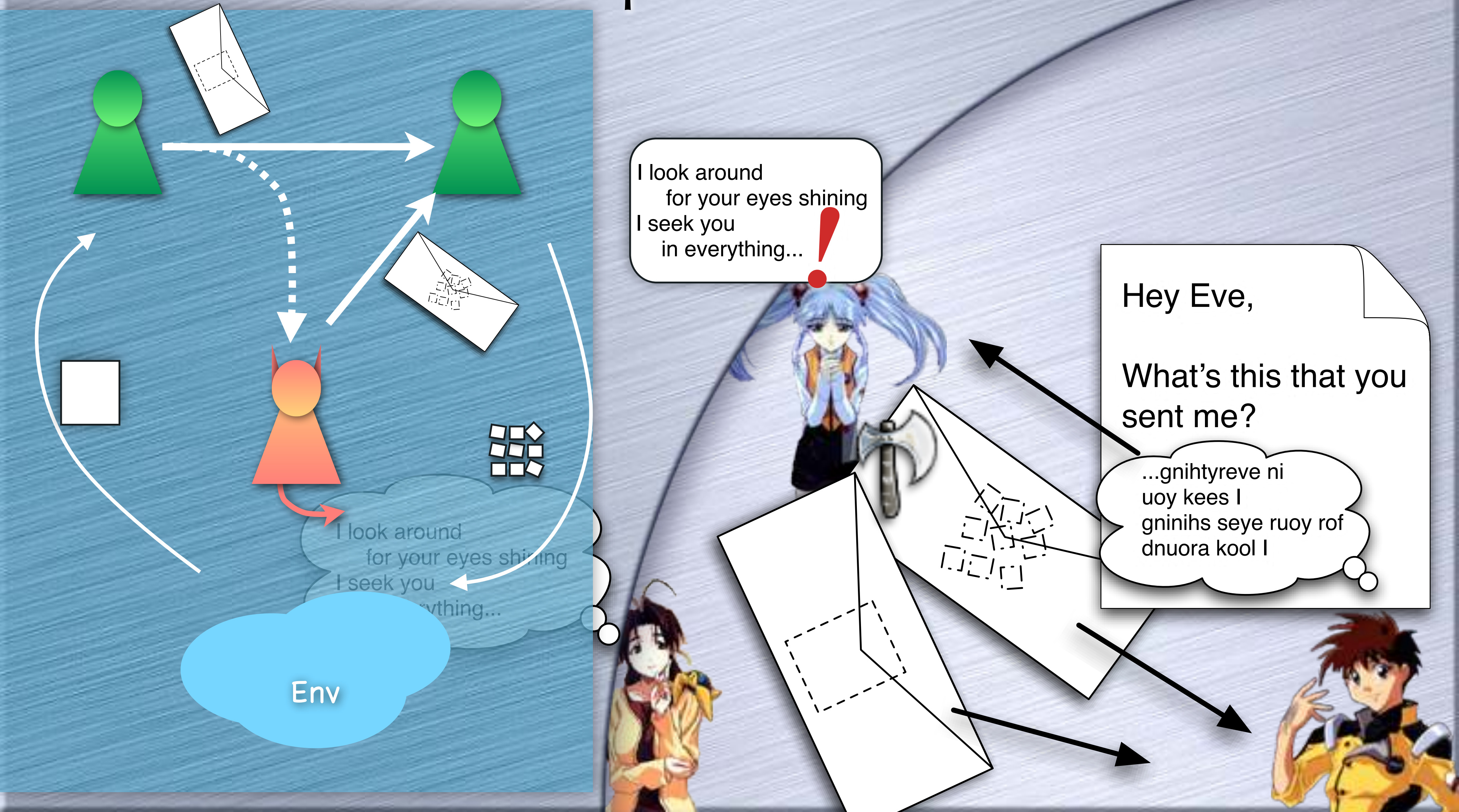
Chosen Ciphertext Attack

- SIM-CCA: does capture this attack



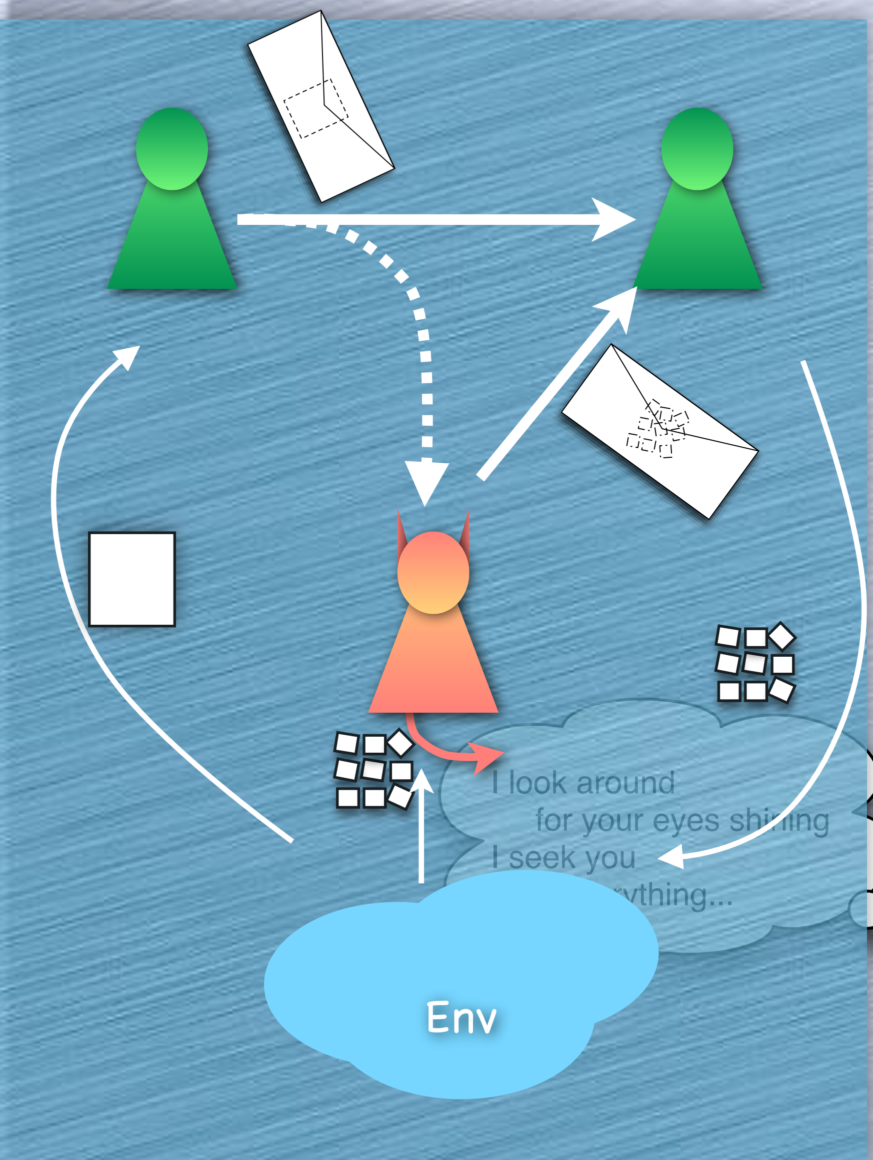
Chosen Ciphertext Attack

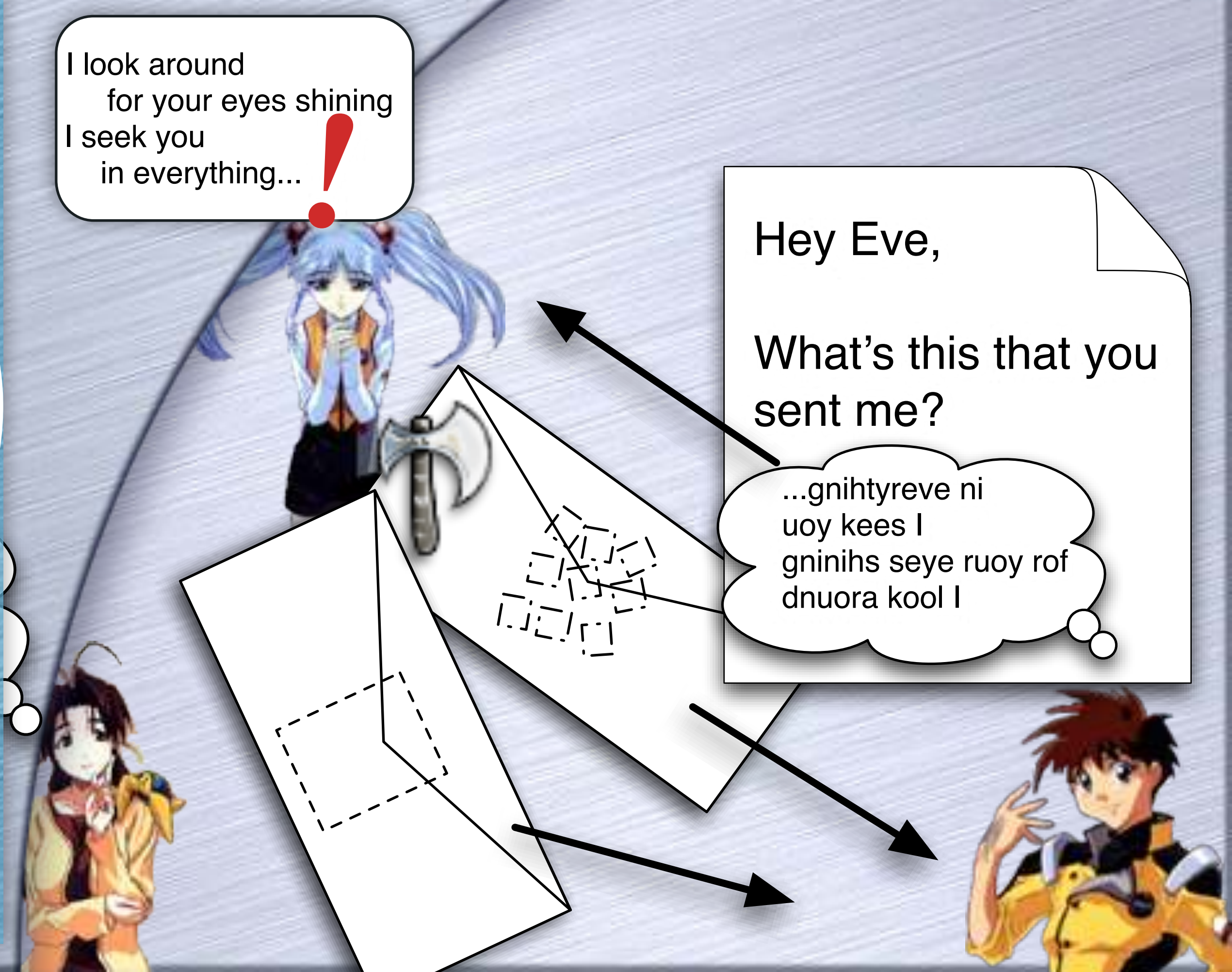
- SIM-CCA: does capture this attack



Chosen Ciphertext Attack

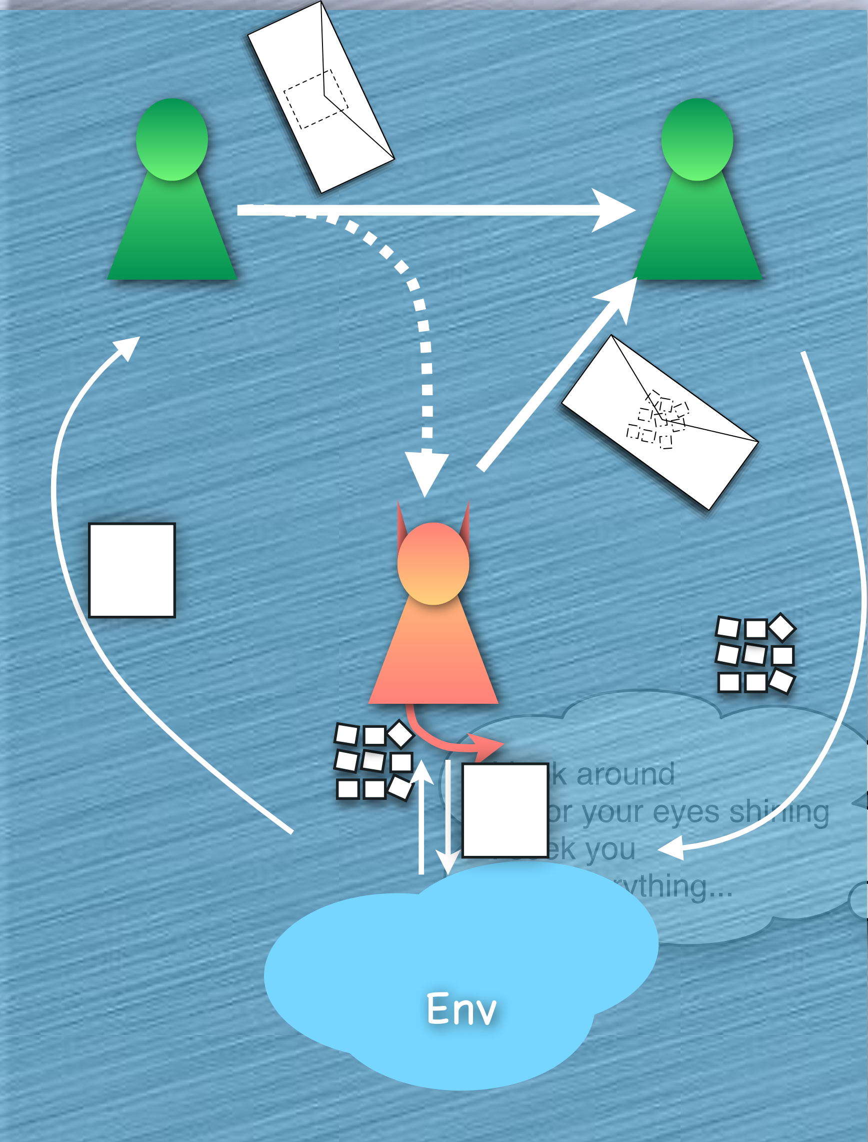
- SIM-CCA: does capture this attack

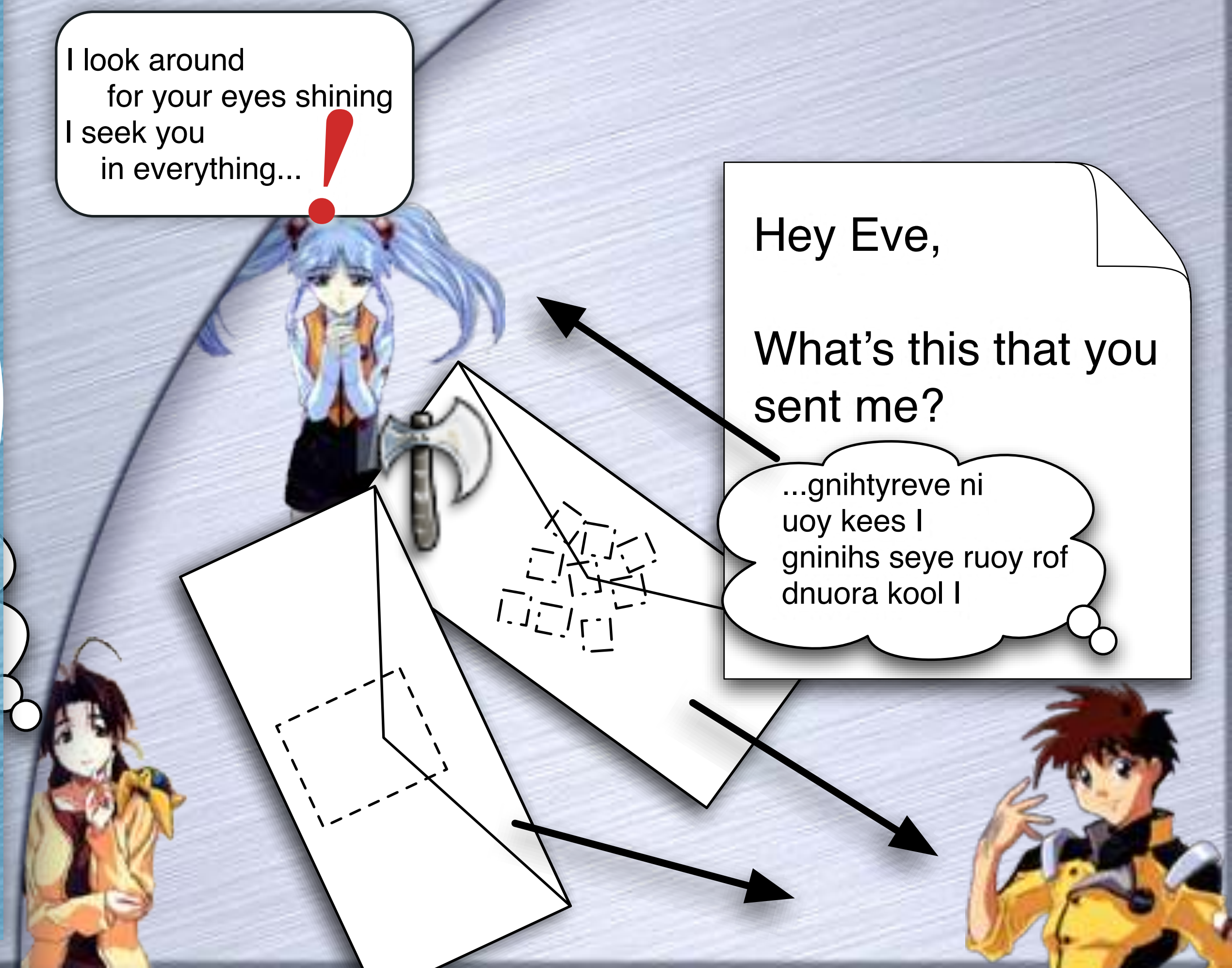




Chosen Ciphertext Attack

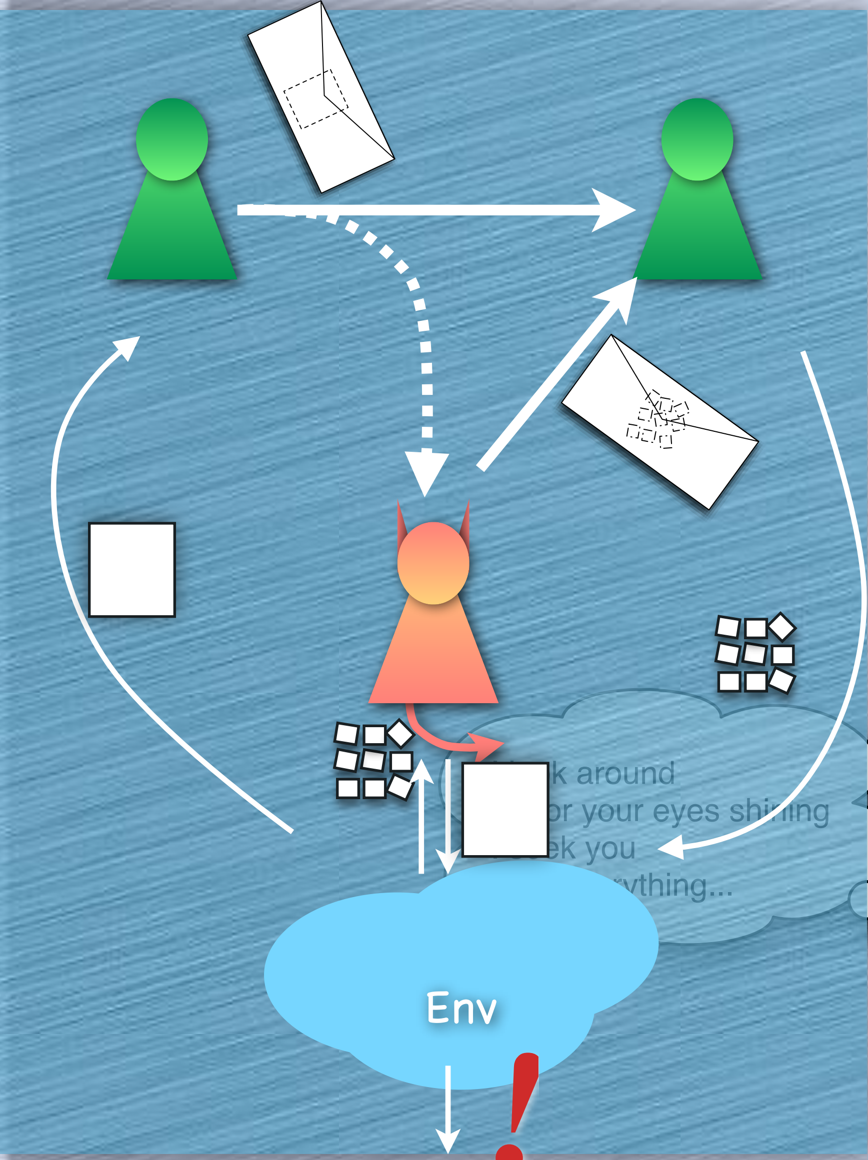
- SIM-CCA: does capture this attack

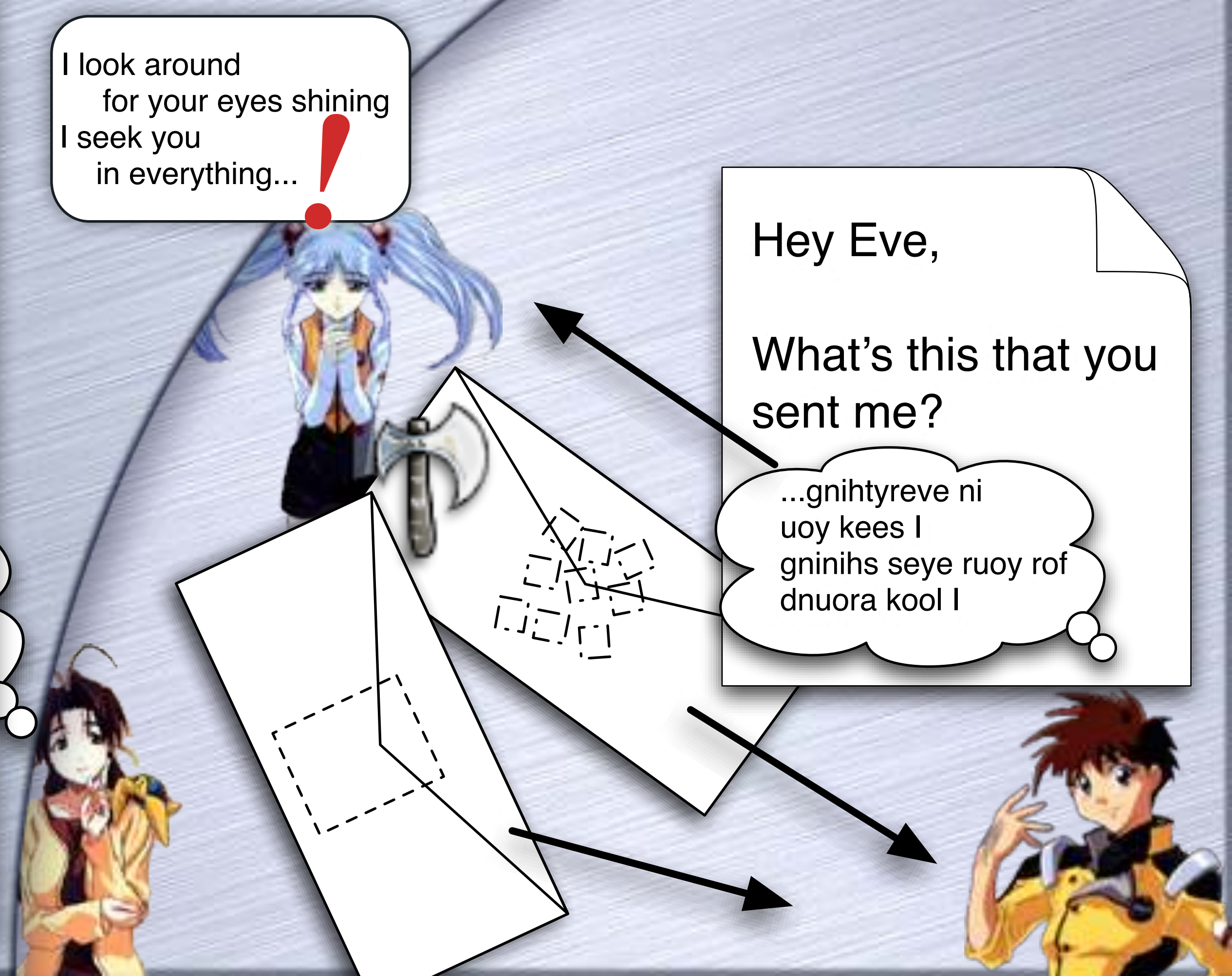




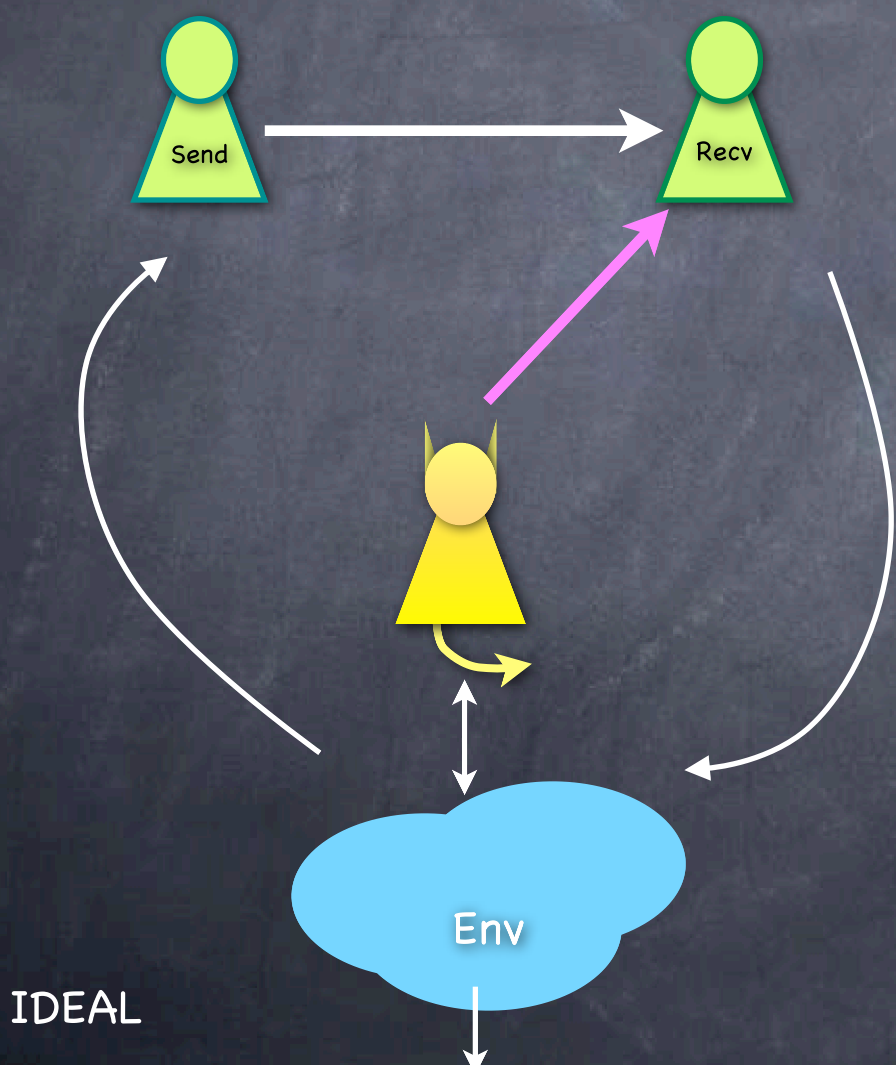
Chosen Ciphertext Attack

- SIM-CCA: does capture this attack





SIM-CCA Security (PKE)



Secure (and correct) if:

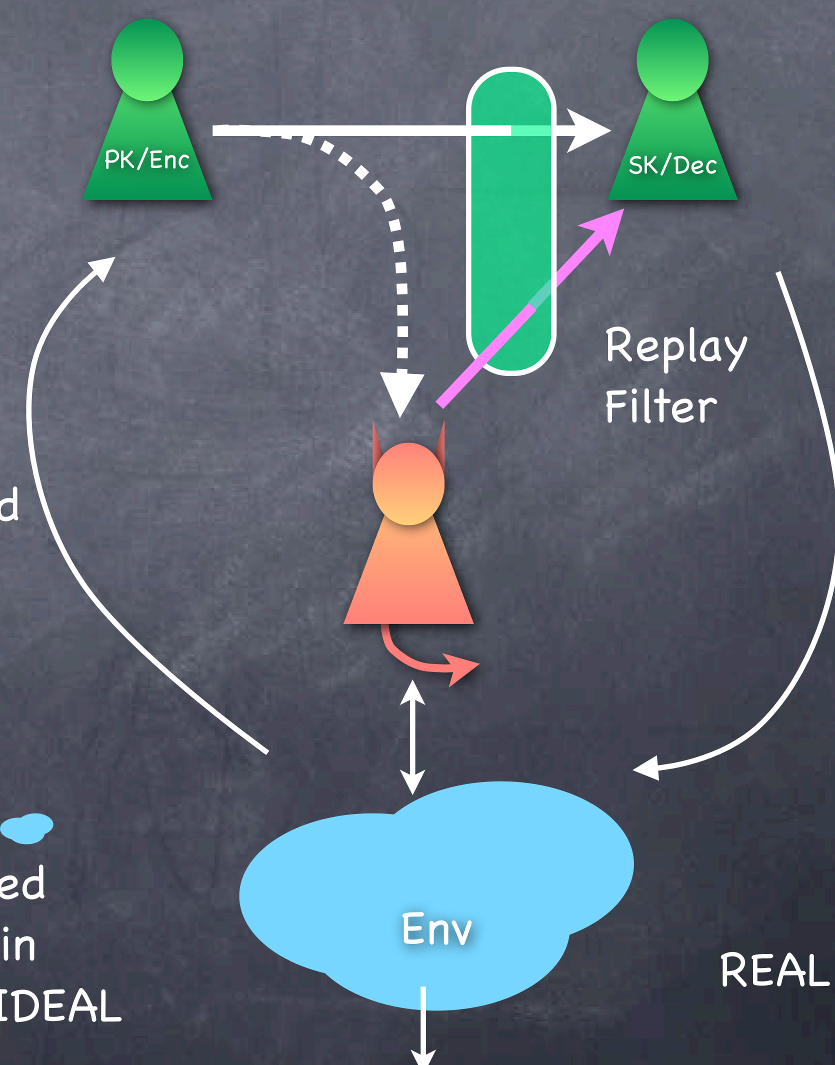
∀

∃ s.t.

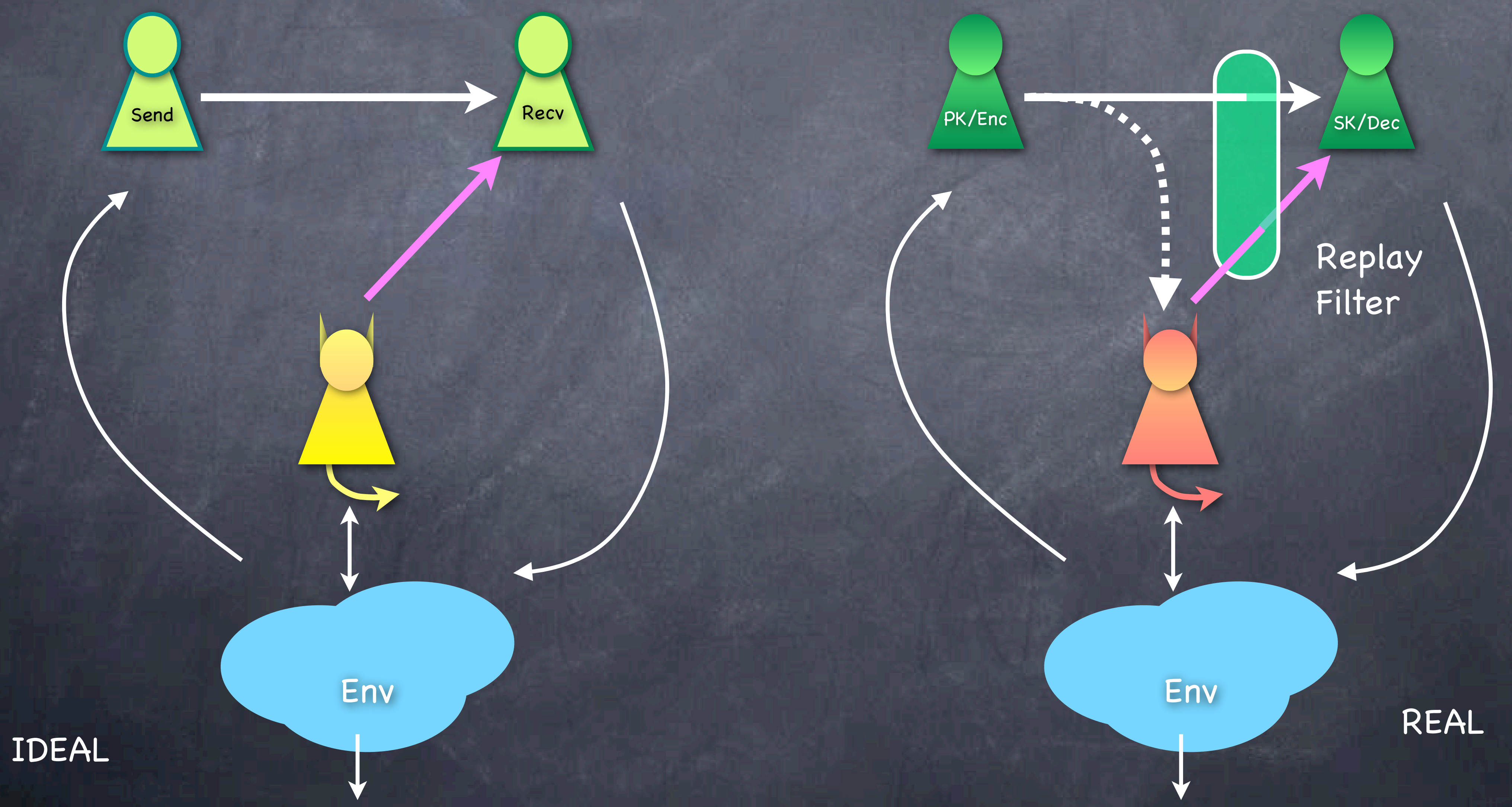
∀

output of

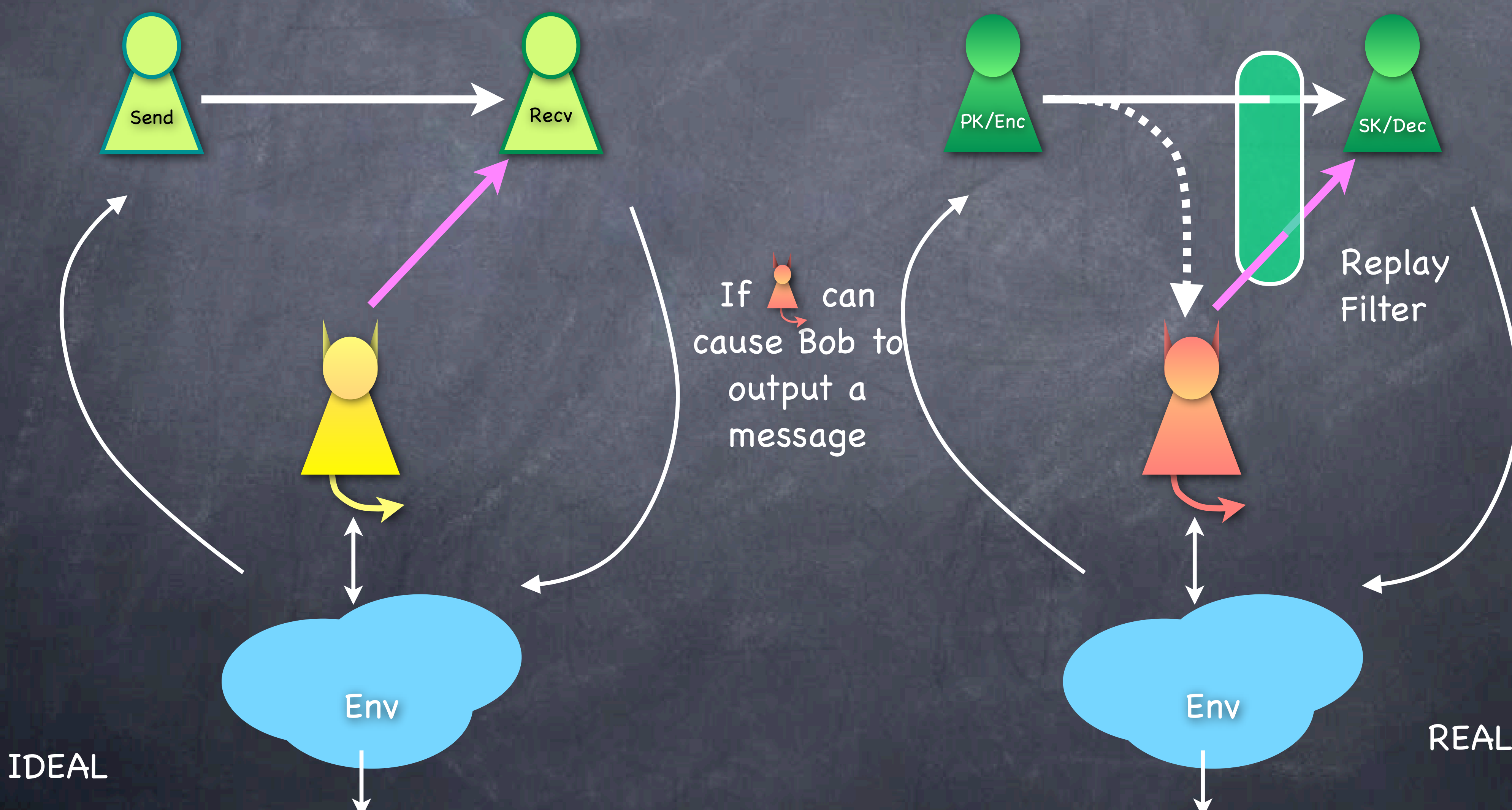
is distributed identically in REAL and IDEAL



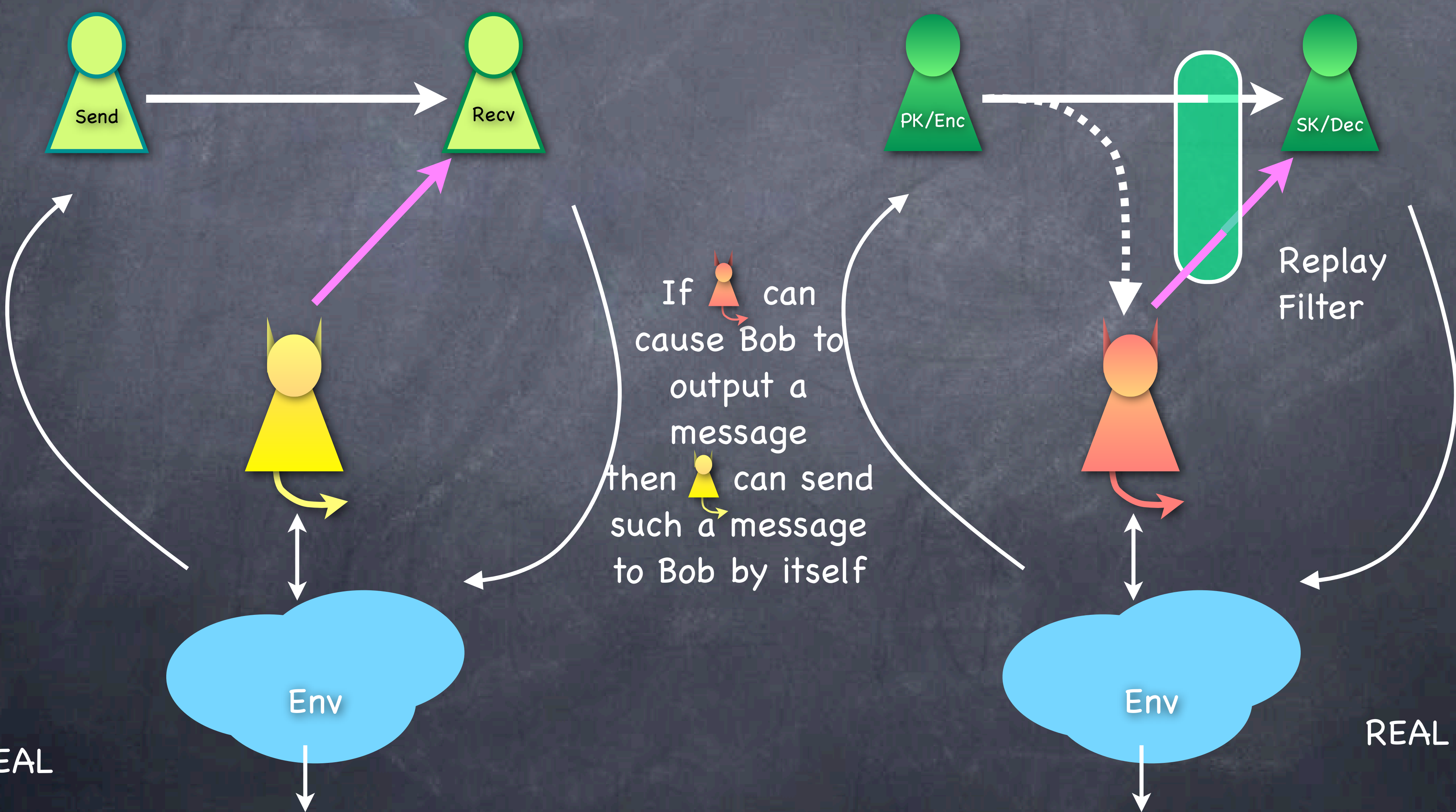
SIM-CCA Security and Malleability



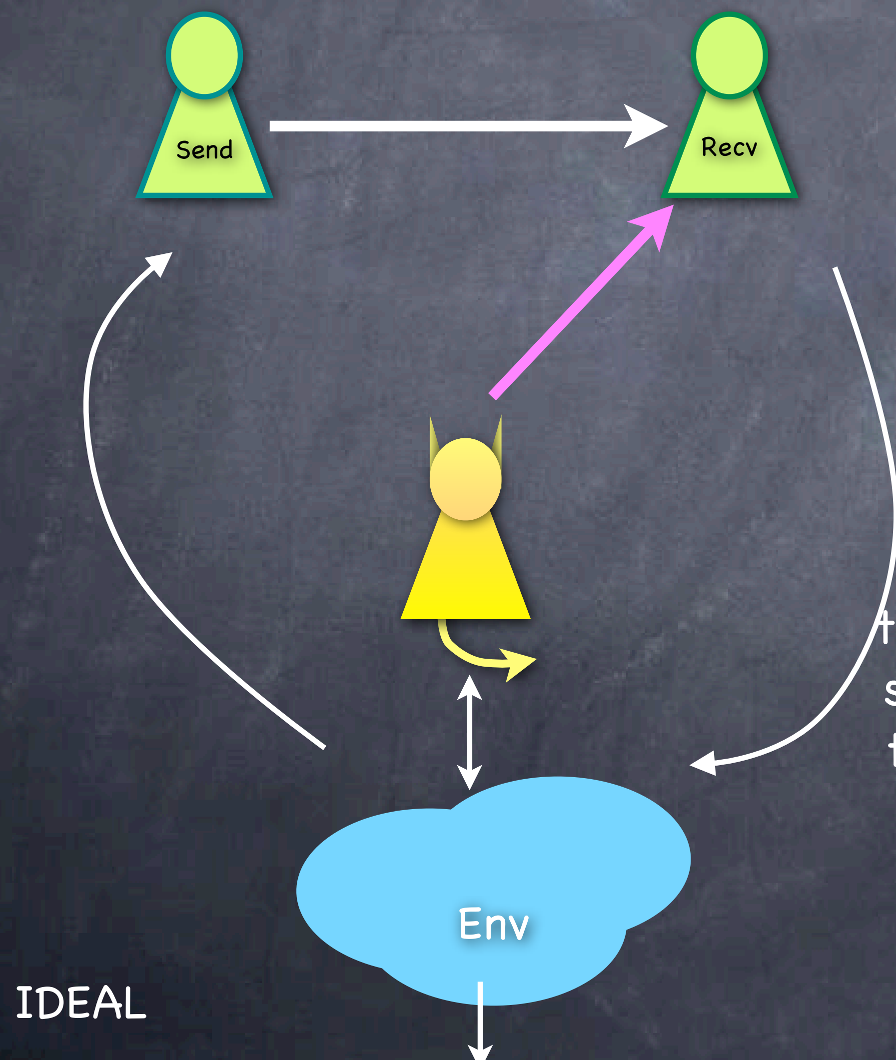
SIM-CCA Security and Malleability



SIM-CCA Security and Malleability

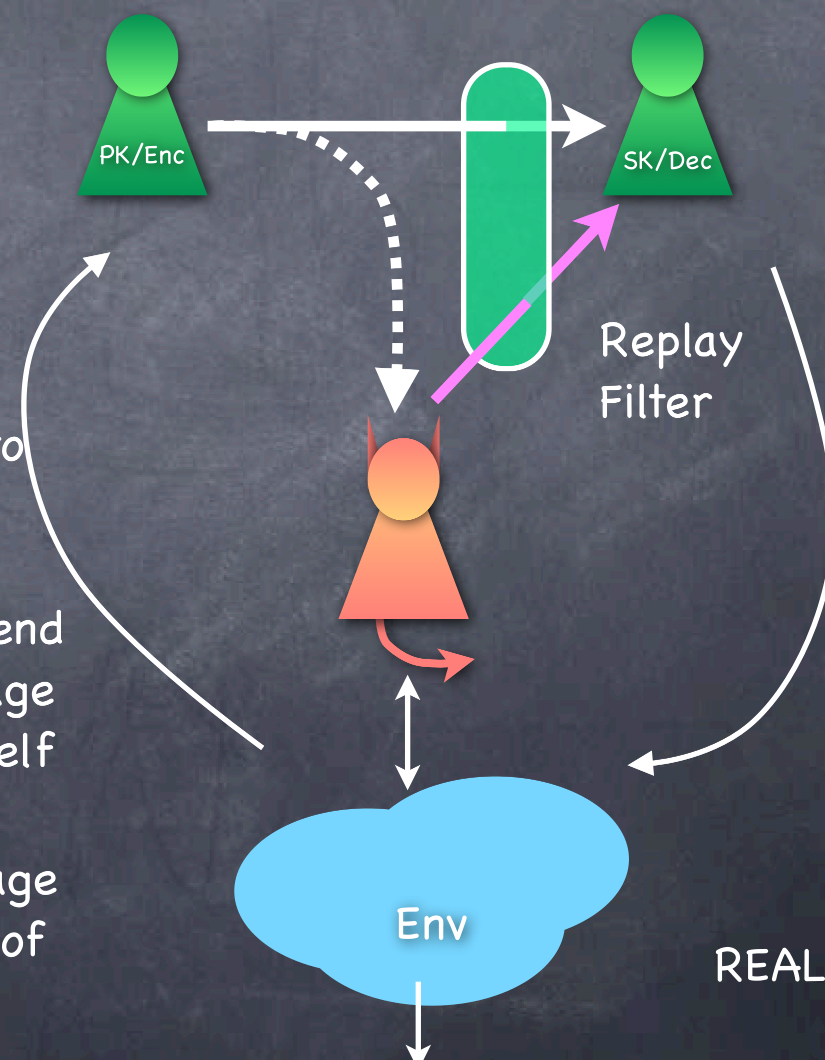


SIM-CCA Security and Malleability



If can cause Bob to output a message then can send such a message to Bob by itself

Hence message not a result of malleating



Constructing CCA Secure PKEs

Constructing CCA Secure PKEs

- Possible from generic assumptions

Constructing CCA Secure PKEs

- Possible from generic assumptions
  - e.g. Enhanced T-OWP, Lossy T-OWF, Correlation-secure T-OWF, Adaptive T-OWF/relation, ...

Constructing CCA Secure PKEs

- Possible from generic assumptions
  - e.g. Enhanced T-OWP, Lossy T-OWF, Correlation-secure T-OWF, Adaptive T-OWF/relation, ...
    - e.g. Using a CPA secure PKE to create two ciphertexts and a "Non-Interactive Zero Knowledge proof" of consistency

Constructing CCA Secure PKEs

- Possible from generic assumptions
  - e.g. Enhanced T-OWP, Lossy T-OWF, Correlation-secure T-OWF, Adaptive T-OWF/relation, ...
    - e.g. Using a CPA secure PKE to create two ciphertexts and a "Non-Interactive Zero Knowledge proof" of consistency
    - e.g. Include a "NIZK proof of knowledge" of the plaintext

Constructing CCA Secure PKEs

- Possible from generic assumptions
  - e.g. Enhanced T-OWP, Lossy T-OWF, Correlation-secure T-OWF, Adaptive T-OWF/relation, ...
    - e.g. Using a CPA secure PKE to create two ciphertexts and a "Non-Interactive Zero Knowledge proof" of consistency
    - e.g. Include a "NIZK proof of knowledge" of the plaintext
- Much more efficient from specific number theoretic/algebraic assumptions

Constructing CCA Secure PKEs

- Possible from generic assumptions
  - e.g. Enhanced T-OWP, Lossy T-OWF, Correlation-secure T-OWF, Adaptive T-OWF/relation, ...
    - e.g. Using a CPA secure PKE to create two ciphertexts and a "Non-Interactive Zero Knowledge proof" of consistency
    - e.g. Include a "NIZK proof of knowledge" of the plaintext
- Much more efficient from specific number theoretic/algebraic assumptions
- Even more efficient in the "Random Oracle Model"

Constructing CCA Secure PKEs

- Possible from generic assumptions
  - e.g. Enhanced T-OWP, Lossy T-OWF, Correlation-secure T-OWF, Adaptive T-OWF/relation, ...
    - e.g. Using a CPA secure PKE to create two ciphertexts and a "Non-Interactive Zero Knowledge proof" of consistency
    - e.g. Include a "NIZK proof of knowledge" of the plaintext
- Much more efficient from specific number theoretic/algebraic assumptions
- Even more efficient in the "Random Oracle Model"
- Significant efficiency gain using "Hybrid Encryption"

Hybrid Encryption

Hybrid Encryption

- PKE is far less efficient compared to SKE (CCA- or CPA-secure)

Hybrid Encryption

- PKE is far less efficient compared to SKE (CCA- or CPA-secure)
  - SKE using Block Ciphers (e.g. AES) and MAC is very fast

Hybrid Encryption

- PKE is far less efficient compared to SKE (CCA- or CPA-secure)
  - SKE using Block Ciphers (e.g. AES) and MAC is very fast
  - El Gamal uses exponentiations (CCA-secure versions even more)

Hybrid Encryption

- PKE is far less efficient compared to SKE (CCA- or CPA-secure)
  - SKE using Block Ciphers (e.g. AES) and MAC is very fast
  - El Gamal uses exponentiations (CCA-secure versions even more)
- Hybrid encryption: Use (CCA secure) PKE to transfer a key (or key generation material) for the (CCA secure) SKE. Use SKE with this key for sending data

Hybrid Encryption

- PKE is far less efficient compared to SKE (CCA- or CPA-secure)
  - SKE using Block Ciphers (e.g. AES) and MAC is very fast
  - El Gamal uses exponentiations (CCA-secure versions even more)
- Hybrid encryption: Use (CCA secure) PKE to transfer a key (or key generation material) for the (CCA secure) SKE. Use SKE with this key for sending data
  - Hopefully the combination remains CCA secure

Hybrid Encryption

- PKE is far less efficient compared to SKE (CCA- or CPA-secure)
  - SKE using Block Ciphers (e.g. AES) and MAC is very fast
  - El Gamal uses exponentiations (CCA-secure versions even more)
- Hybrid encryption: Use (CCA secure) PKE to transfer a key (or key generation material) for the (CCA secure) SKE. Use SKE with this key for sending data
  - Hopefully the combination remains CCA secure
  - PKE used to encrypt only a (short) key for the SKE

Hybrid Encryption

- PKE is far less efficient compared to SKE (CCA- or CPA-secure)
  - SKE using Block Ciphers (e.g. AES) and MAC is very fast
  - El Gamal uses exponentiations (CCA-secure versions even more)
- Hybrid encryption: Use (CCA secure) PKE to transfer a key (or key generation material) for the (CCA secure) SKE. Use SKE with this key for sending data
  - Hopefully the combination remains CCA secure
  - PKE used to encrypt only a (short) key for the SKE
    - Relatively low overhead on top of the (fast) SKE encryption

Hybrid Encryption

Hybrid Encryption

- Hybrid Encryption: KEM/DEM paradigm

Hybrid Encryption

- Hybrid Encryption: KEM/DEM paradigm
  - Key Encapsulation Method: a public-key scheme to transfer a key

Hybrid Encryption

- Hybrid Encryption: KEM/DEM paradigm
  - Key Encapsulation Method: a public-key scheme to transfer a key

Or to generate a key

Hybrid Encryption

Or to generate a key

- Hybrid Encryption: KEM/DEM paradigm
  - Key Encapsulation Method: a public-key scheme to transfer a key
  - Data Encapsulation Method: a shared-key scheme (using the key transferred using KEM)

Hybrid Encryption

Or to generate a key

- Hybrid Encryption: KEM/DEM paradigm
  - Key Encapsulation Method: a public-key scheme to transfer a key
  - Data Encapsulation Method: a shared-key scheme (using the key transferred using KEM)
- For what KEM/DEM is a hybrid encryption scheme CCA secure?

Hybrid Encryption

Or to generate a key

- Hybrid Encryption: KEM/DEM paradigm
  - Key Encapsulation Method: a public-key scheme to transfer a key
  - Data Encapsulation Method: a shared-key scheme (using the key transferred using KEM)
- For what KEM/DEM is a hybrid encryption scheme CCA secure?
  - Works if KEM is a SIM-CCA secure PKE scheme and DEM is a SIM-CCA secure SKE scheme

Hybrid Encryption

Or to generate a key

- Hybrid Encryption: KEM/DEM paradigm
  - Key Encapsulation Method: a public-key scheme to transfer a key
  - Data Encapsulation Method: a shared-key scheme (using the key transferred using KEM)
- For what KEM/DEM is a hybrid encryption scheme CCA secure?
  - Works if KEM is a SIM-CCA secure PKE scheme and DEM is a SIM-CCA secure SKE scheme
    - Easy to prove using "composition" properties of the SIM definition

Hybrid Encryption

Or to generate a key

- Hybrid Encryption: KEM/DEM paradigm
  - Key Encapsulation Method: a public-key scheme to transfer a key
  - Data Encapsulation Method: a shared-key scheme (using the key transferred using KEM)
- For what KEM/DEM is a hybrid encryption scheme CCA secure?
  - Works if KEM is a SIM-CCA secure PKE scheme and DEM is a SIM-CCA secure SKE scheme
    - Easy to prove using "composition" properties of the SIM definition
  - Less security sufficient: KEM used to transfer a random key; DEM uses a new key every time.

Today

Today

- CPA secure PKE: Constructions

Today

- CPA secure PKE: Constructions
  - El Gamal Encryption

Today

- CPA secure PKE: Constructions
  - El Gamal Encryption
  - TPRG and TOWP

Today

- CPA secure PKE: Constructions
  - El Gamal Encryption
  - TPRG and TOWP
- CCA secure PKE

Today

- CPA secure PKE: Constructions
  - El Gamal Encryption
  - TPRG and TOWP
- CCA secure PKE
  - Motivating problem: Malleability

Today

- CPA secure PKE: Constructions
  - El Gamal Encryption
  - TPRG and TOWP
- CCA secure PKE
  - Motivating problem: Malleability
  - Hybrid Encryption: KEM/DEM

Today

- CPA secure PKE: Constructions
  - El Gamal Encryption
  - TPRG and TOWP
- CCA secure PKE
  - Motivating problem: Malleability
  - Hybrid Encryption: KEM/DEM
    - Given a basic (CCA secure) PKE, improves efficiency by combining with (CCA secure) SKE

Today

- CPA secure PKE: Constructions
  - El Gamal Encryption
  - TPRG and TOWP
- CCA secure PKE
  - Motivating problem: Malleability
  - Hybrid Encryption: KEM/DEM
    - Given a basic (CCA secure) PKE, improves efficiency by combining with (CCA secure) SKE
  - Next: Constructions for CCA secure PKE