Public-Key Cryptography

Public-Key Cryptography

Lecture 8

Public-Key Cryptography

Lecture 8

Public-Key Encryption from Trapdoor OWP

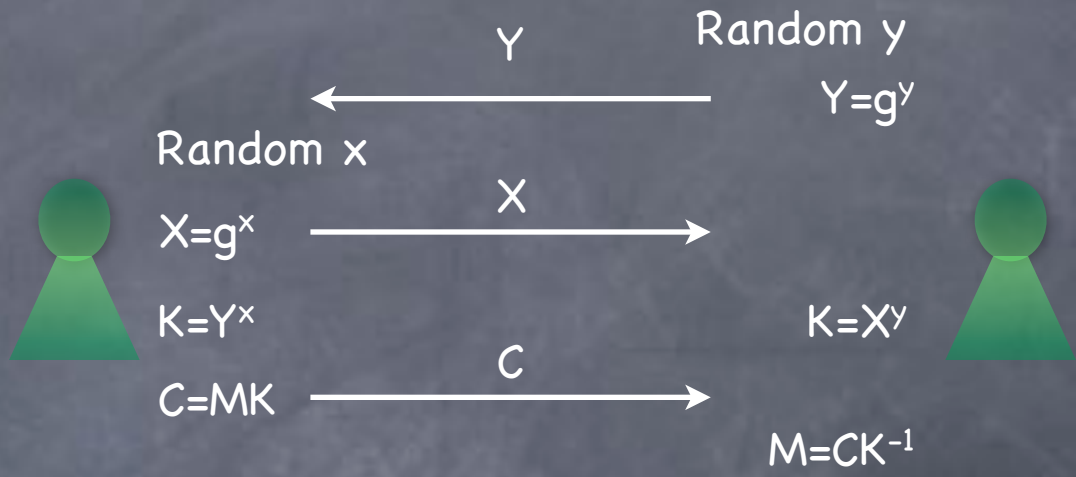
Public-Key Cryptography

Lecture 8

Public-Key Encryption from Trapdoor OWP

CCA Security

Abstracting El Gamal

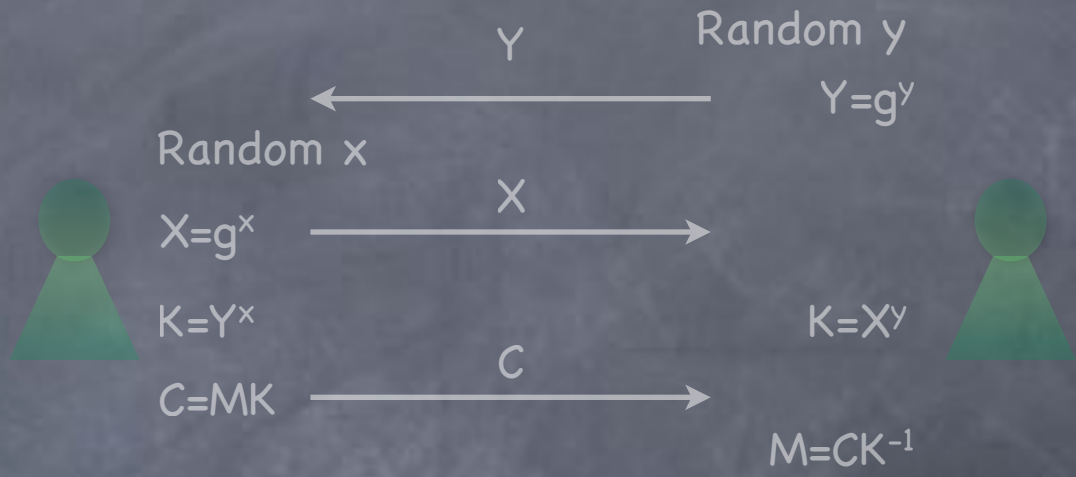


KeyGen: PK=(G,g,Y), SK=(G,g,y)

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

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

Abstracting El Gamal



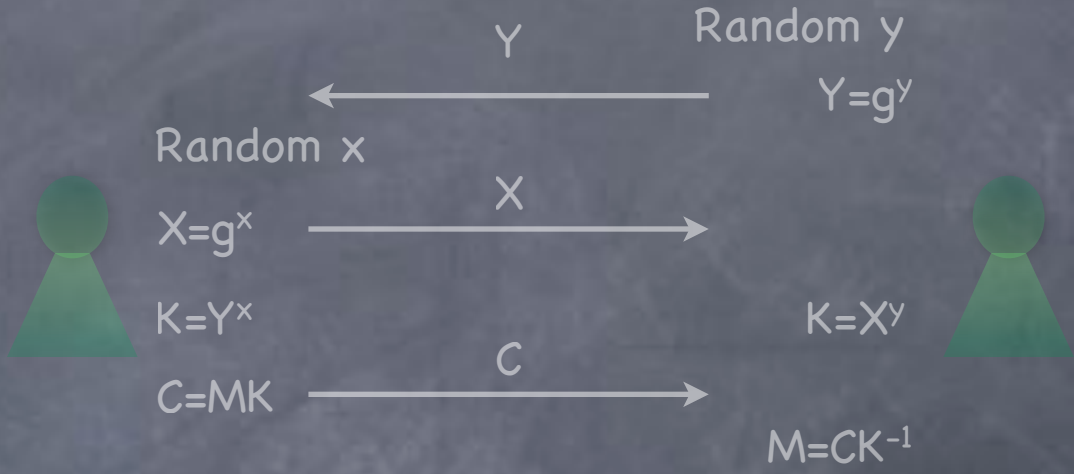
KeyGen: PK=(G,g,Y), SK=(G,g,y)

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

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

Abstracting El Gamal

- Trapdoor PRG:



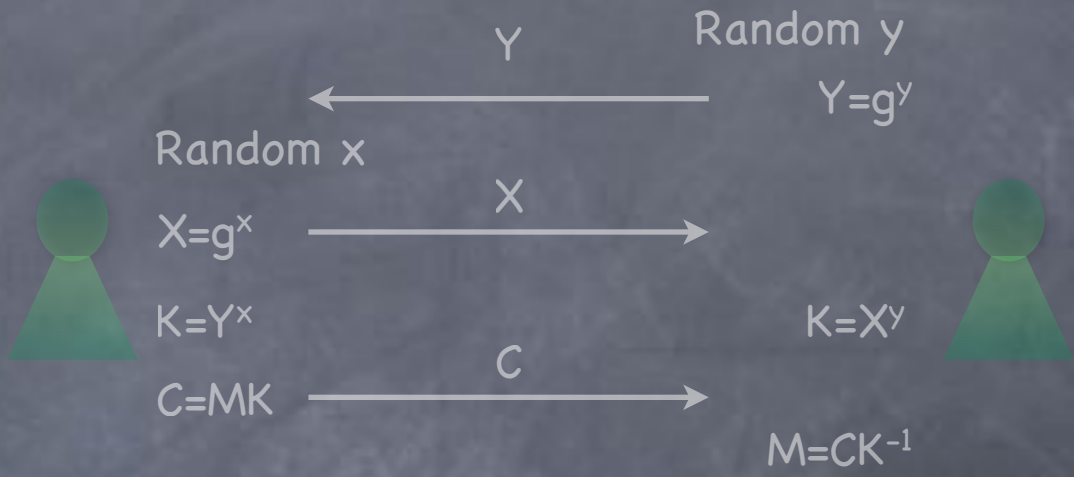
KeyGen: PK=(G,g,Y), SK=(G,g,y)

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

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

Abstracting El Gamal

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



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

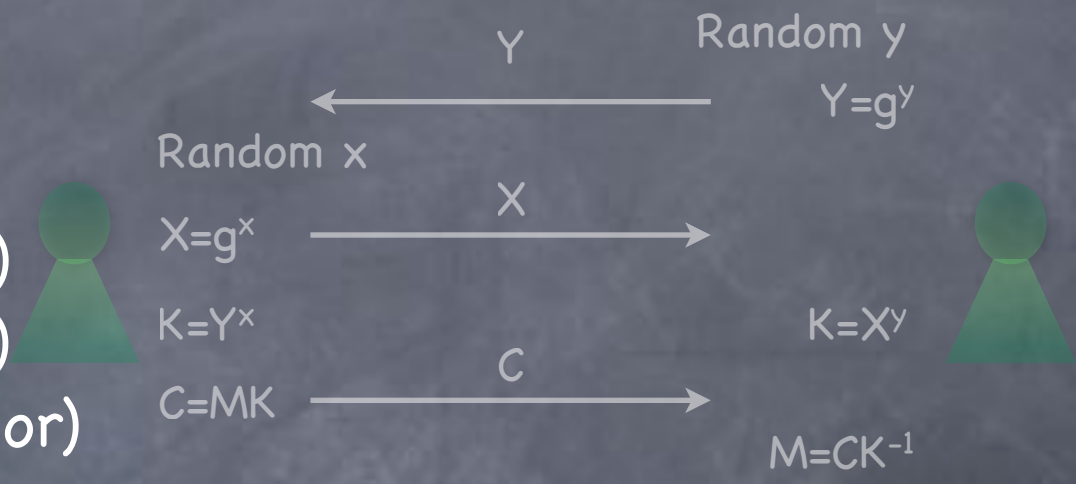
$Enc_{(G,g,Y)}(M) = (X=g^x, C=MY^x)$

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

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)

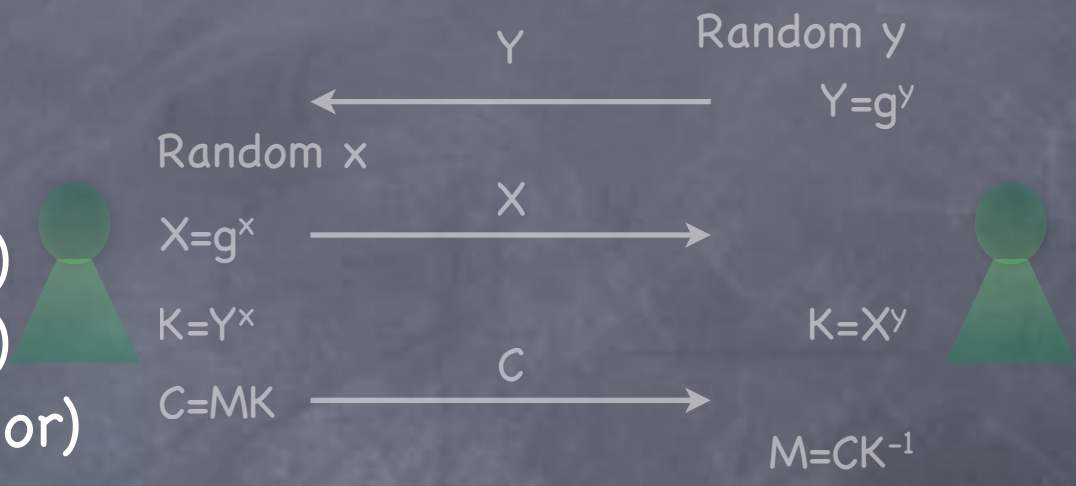
$Enc_{(G,g,Y)}(M) = (X=g^x, C=MY^x)$

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

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)



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

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

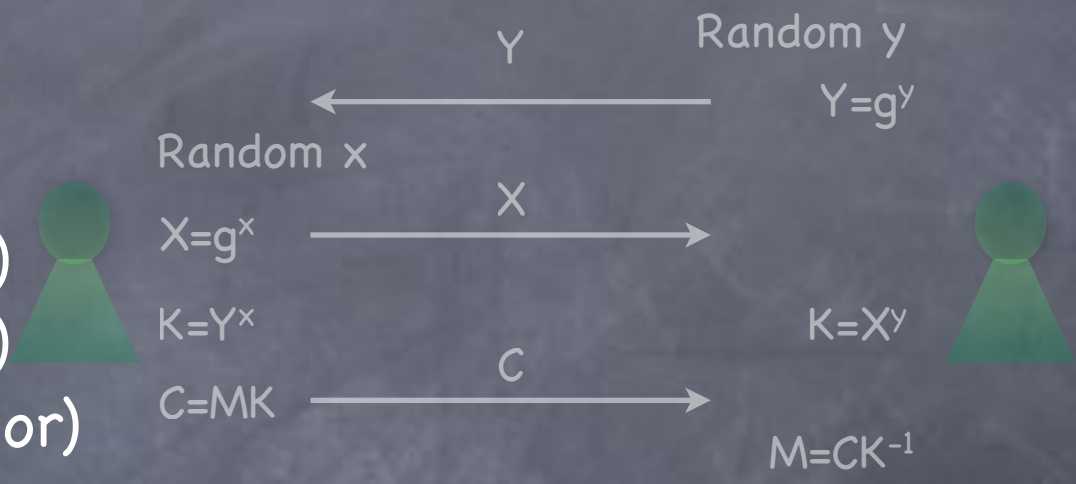
$Dec_{(G,g,y)}(X,C) = CX^{-y}$

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)

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

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

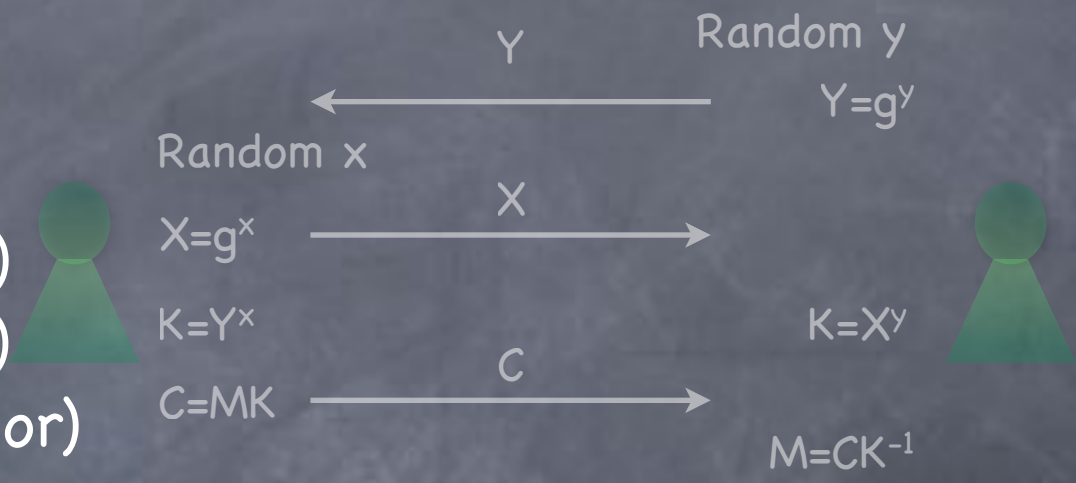
KeyGen: (PK,SK)

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

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

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



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

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

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

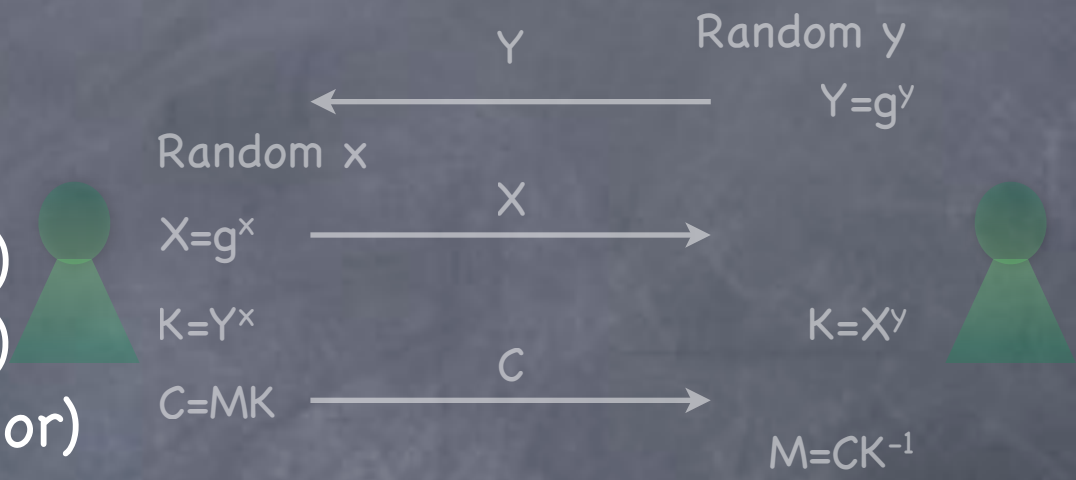
KeyGen: (PK,SK)

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

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

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



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

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

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

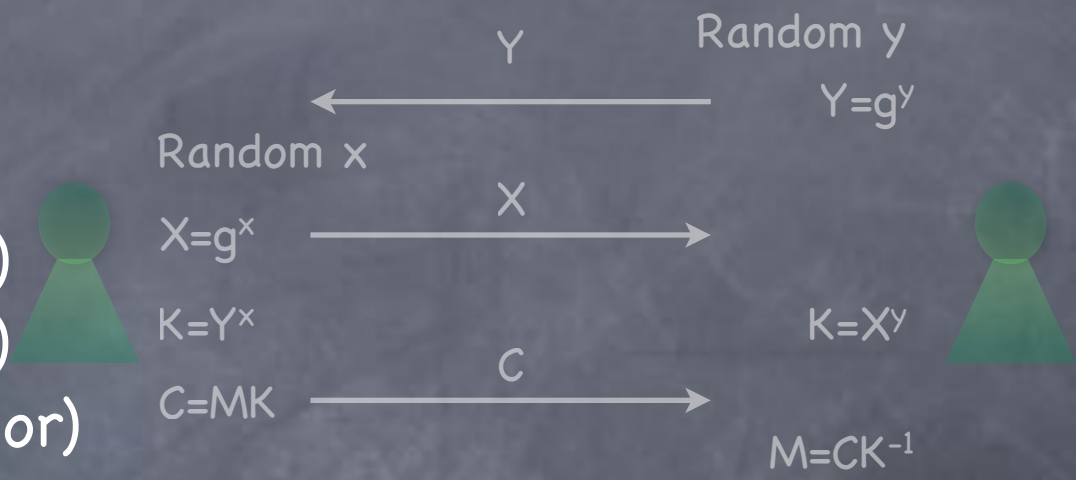
KeyGen: (PK,SK)

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

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

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)

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

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

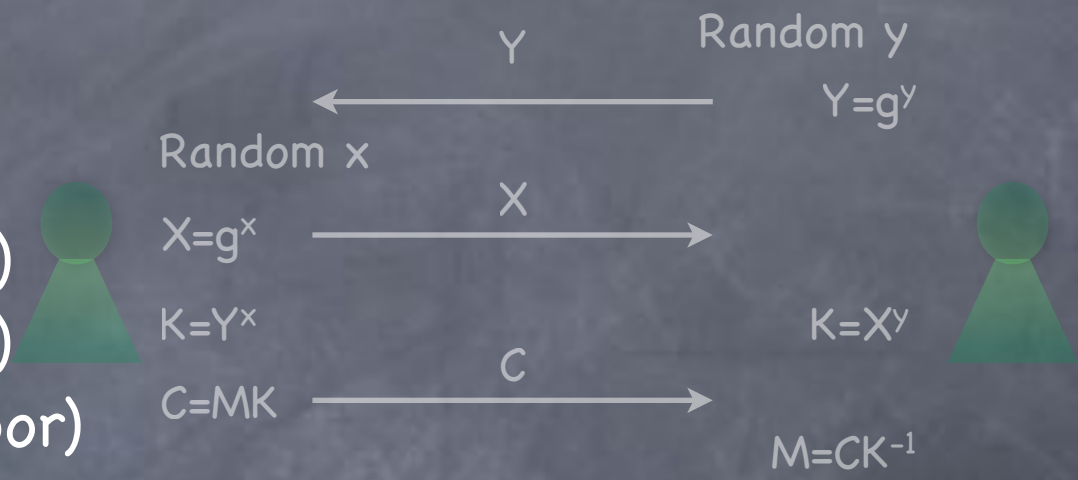
KeyGen: (PK,SK)

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

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

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)

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

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

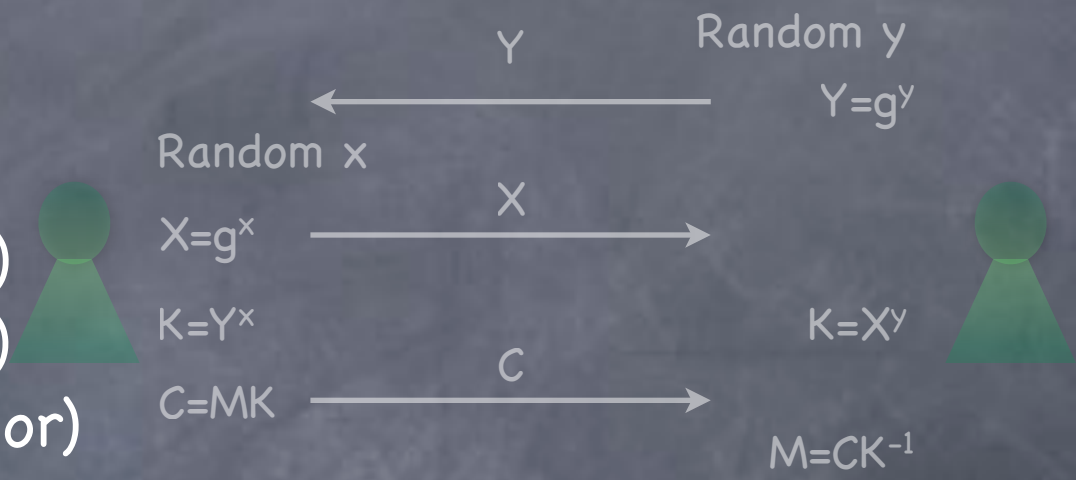
KeyGen: (PK,SK)

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

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

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)

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

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

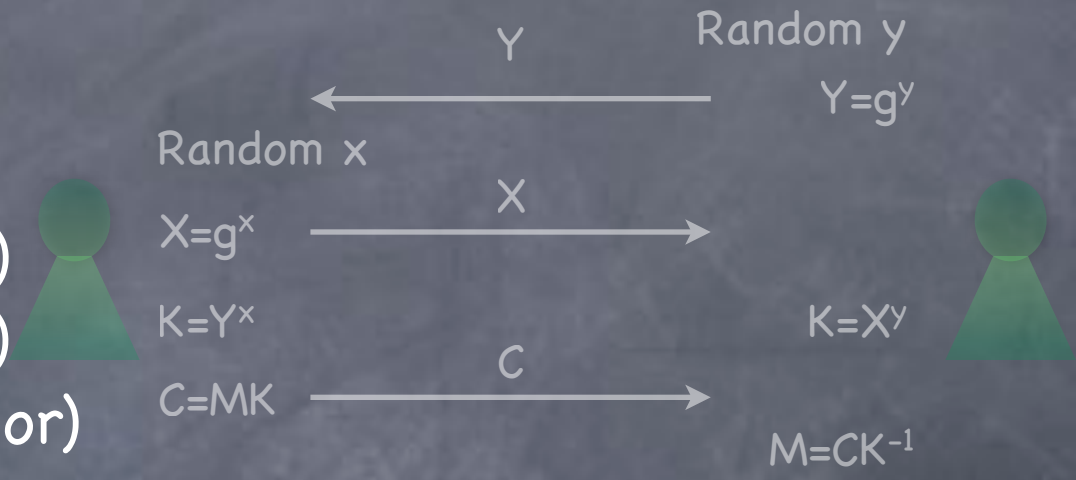
KeyGen: (PK,SK)

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

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

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)

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

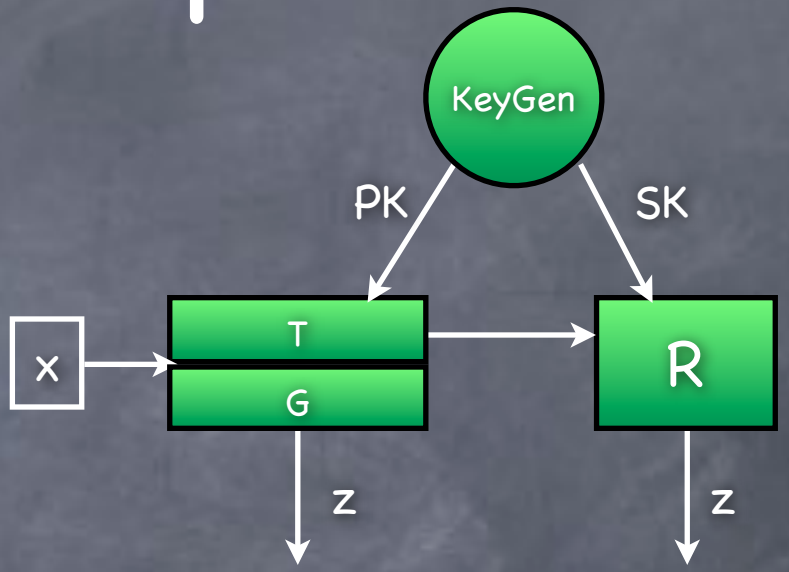
$Dec_{(G,g,y)}(X,C) = CX^{-y}$

KeyGen: (PK,SK)

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

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

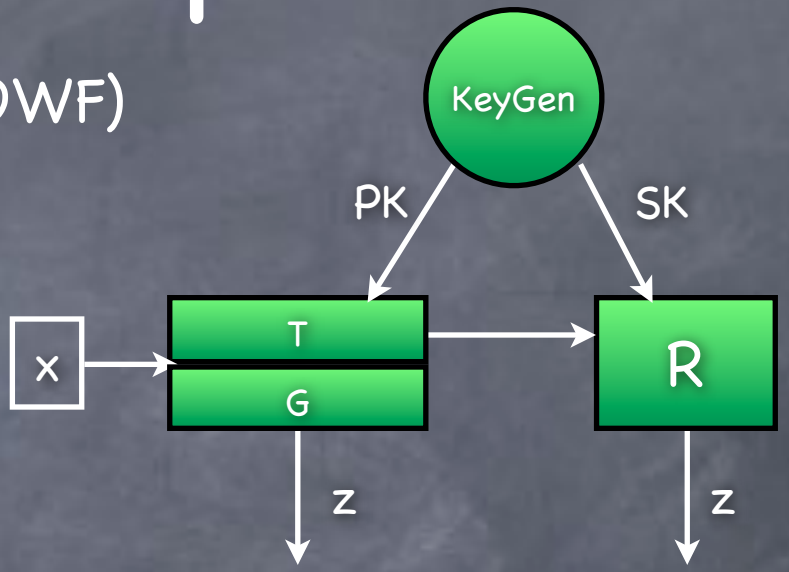
Trapdoor PRG from Generic Assumption?



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

Trapdoor PRG from Generic Assumption?

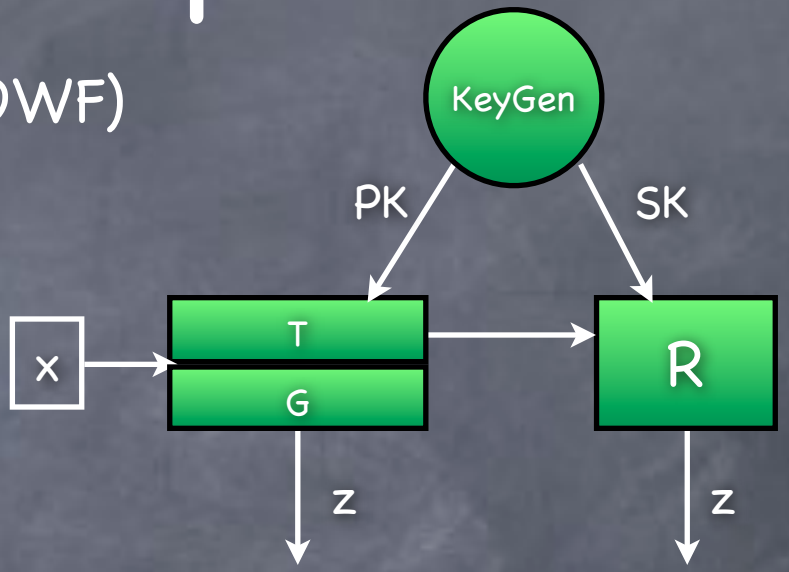
- PRG constructed from OWP (or OWF)



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

Trapdoor PRG from Generic Assumption?

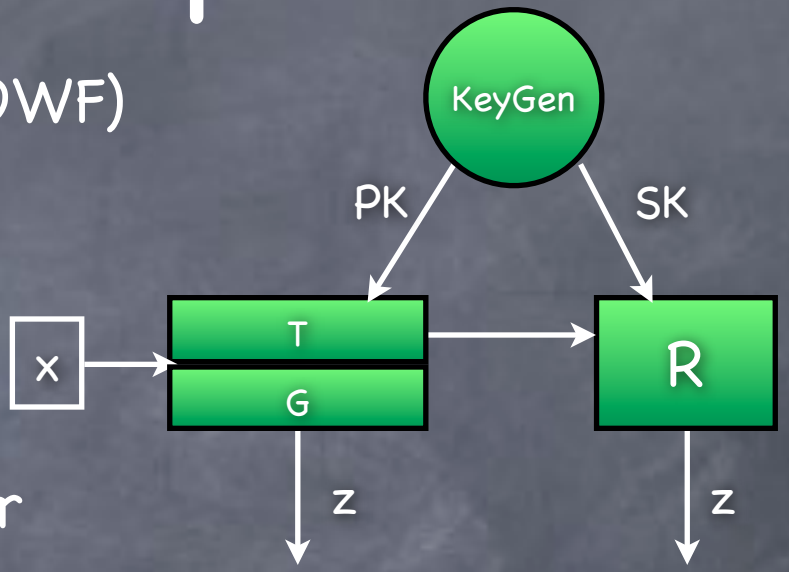
- PRG constructed from OWP (or OWF)
  - Allows us to instantiate the construction with several candidates



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

Trapdoor PRG from Generic Assumption?

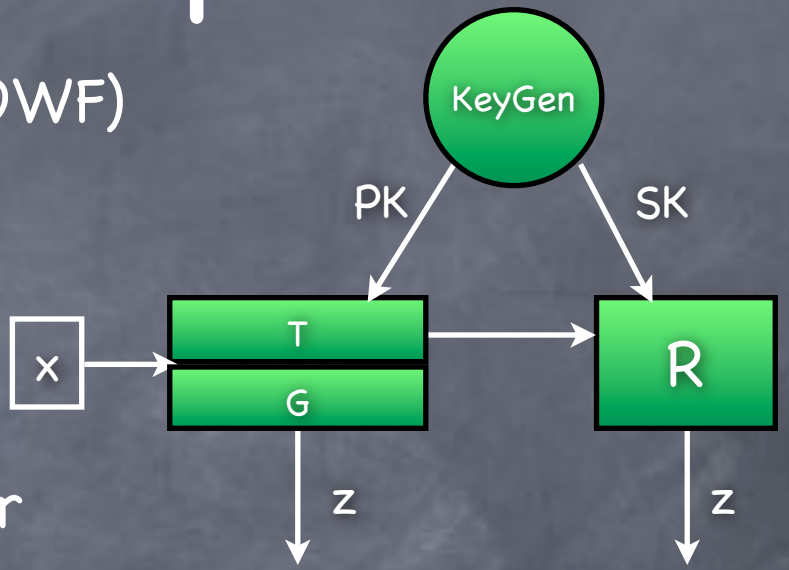
- PRG constructed from OWP (or OWF)
  - Allows us to instantiate the construction with several candidates
- Is there a similar construction for TPRG from OWP?



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

Trapdoor PRG from Generic Assumption?

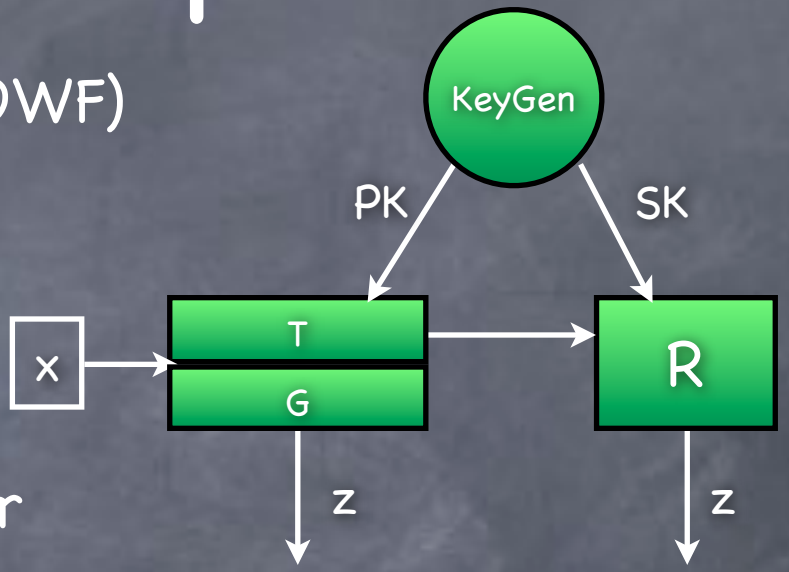
- PRG constructed from OWP (or OWF)
  - Allows us to instantiate the construction with several candidates
- Is there a similar construction for TPRG from OWP?
  - Trapdoor property seems fundamentally different: generic OWP may not offer such a property



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

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

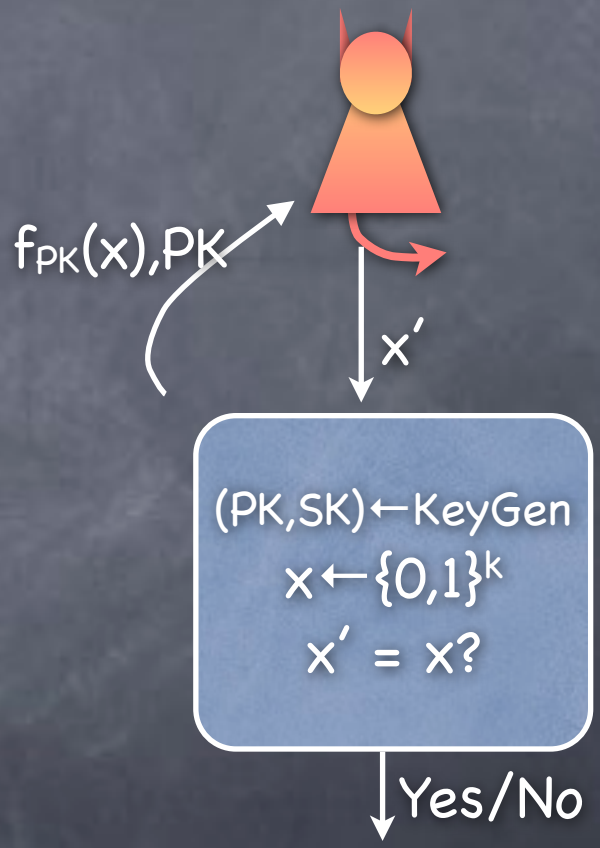
- (KeyGen,f,f') (all PPT) is a trapdoor one-way permutation (TOWP) if
  - For all (PK,SK) ←KeyGen
    - f_{PK} a permutation
    - f'_{SK} is the inverse of f_{PK}

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

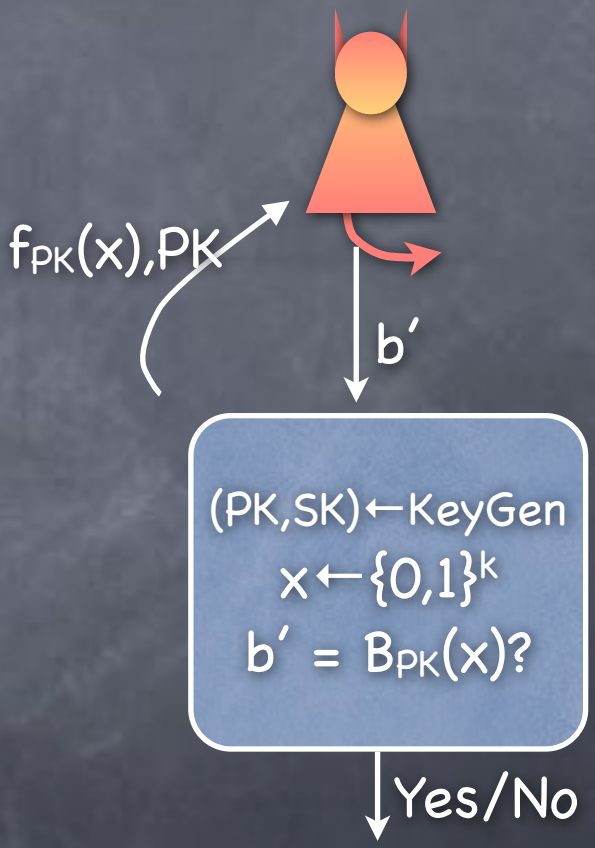
Trapdoor OWP

- (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

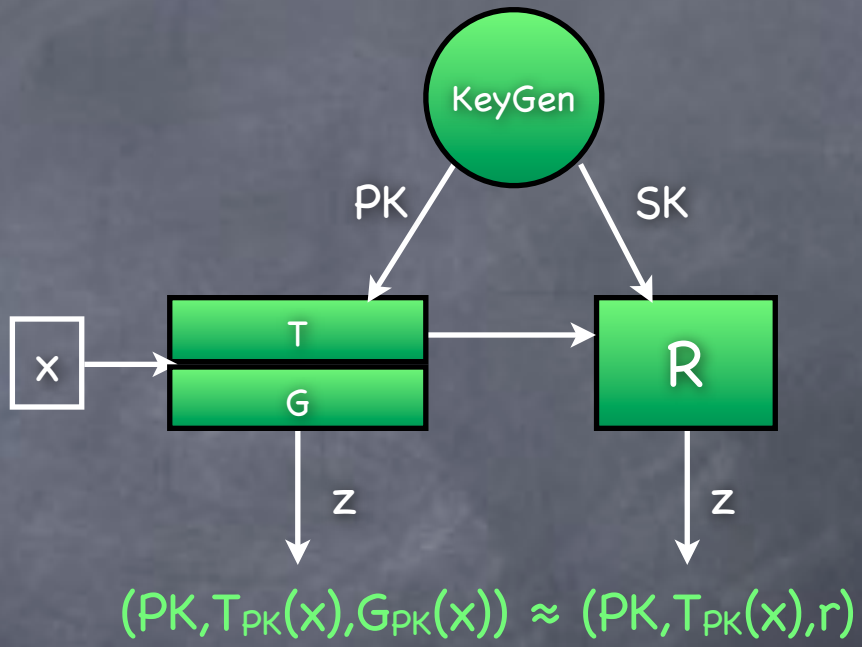


Trapdoor OWP

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

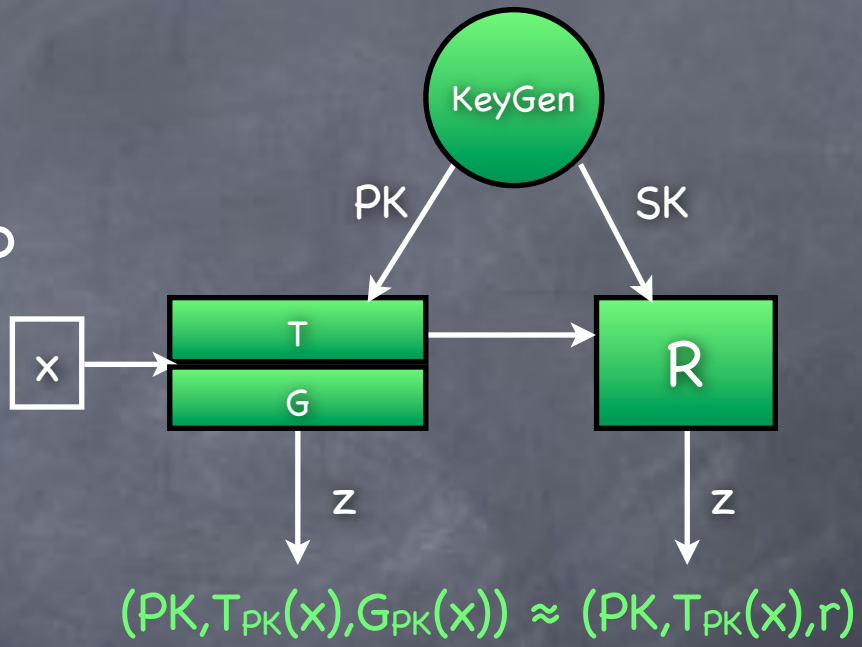


Trapdoor PRG from Trapdoor OWP



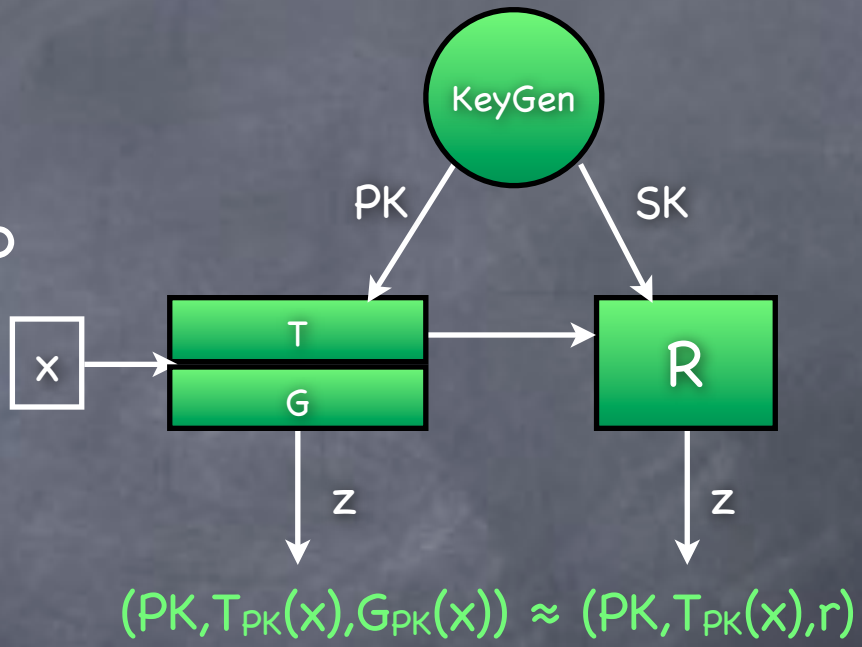
Trapdoor PRG from Trapdoor OWP

- Same construction as PRG from OWP



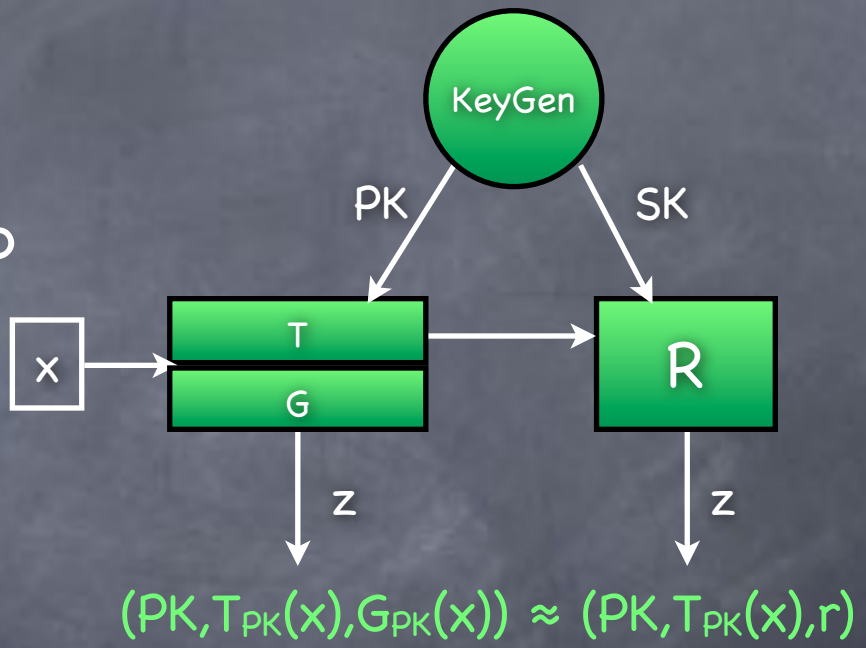
Trapdoor PRG from Trapdoor OWP

- Same construction as PRG from OWP
- One bit TPRG



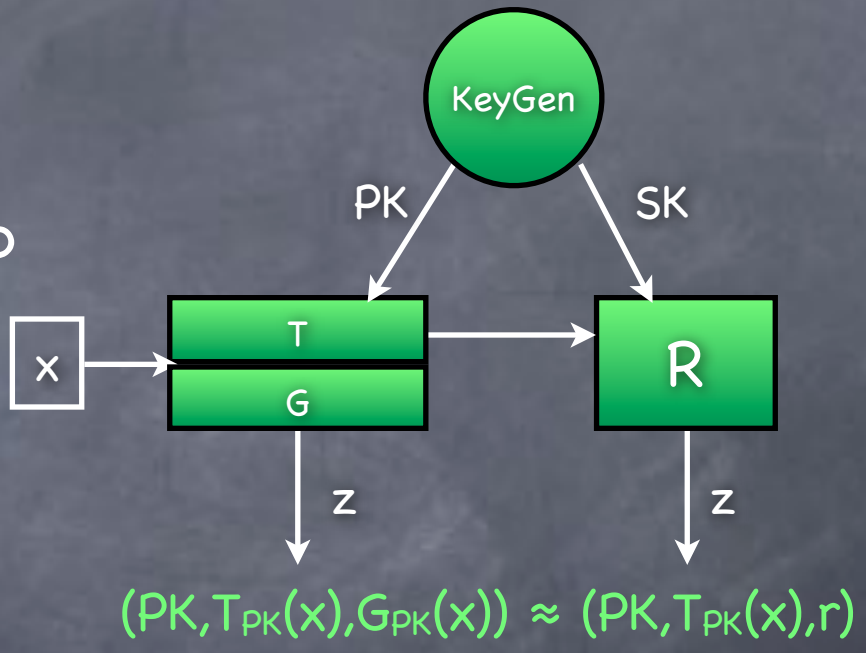
Trapdoor PRG from Trapdoor OWP

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



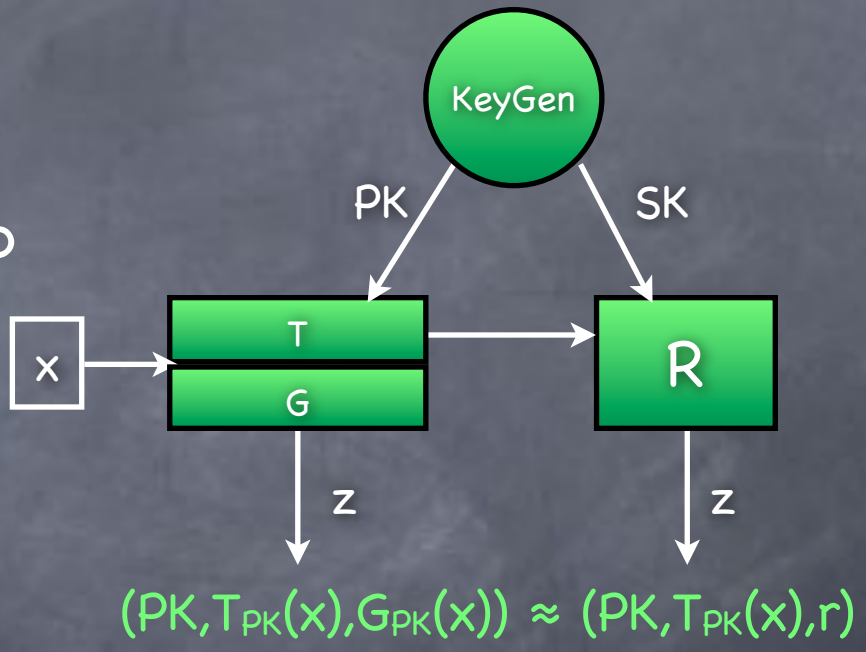
Trapdoor PRG from Trapdoor OWP

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



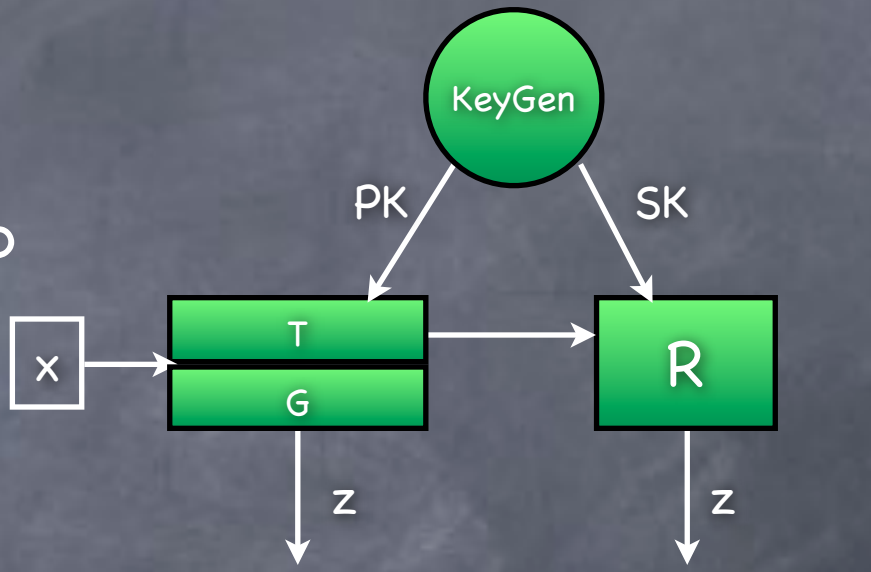
Trapdoor PRG from Trapdoor OWP

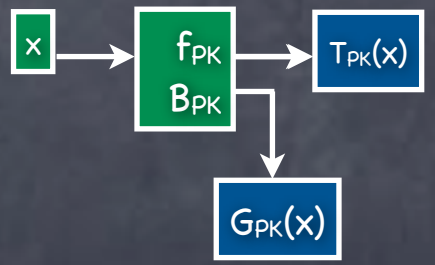
- Same construction as PRG from OWP
- One bit TPRG
  - KeyGen same as TOWP's KeyGen
  - $G_{PK}(x) := B_{PK}(x)$. $T_{PK}(x) := f_{PK}(x)$. $R_{SK}(y) := G_{PK}(f'_{SK}(y))$
    - (SK assumed to contain PK)



Trapdoor PRG from Trapdoor OWP

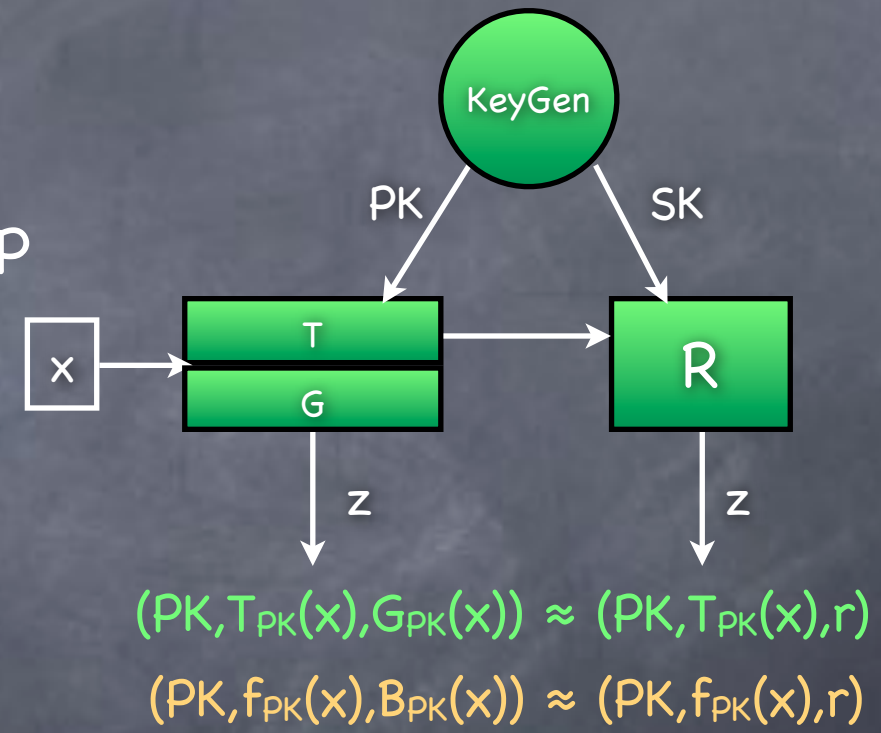
- Same construction as PRG from OWP
- One bit TPRG
  - KeyGen same as TOWP's KeyGen
  - $G_{PK}(x) := B_{PK}(x)$. $T_{PK}(x) := f_{PK}(x)$. $R_{SK}(y) := G_{PK}(f'_{SK}(y))$
    - (SK assumed to contain PK)
- More generally, last permutation output serves as T_{PK}

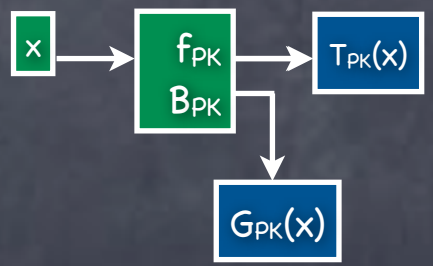




Trapdoor PRG from Trapdoor OWP

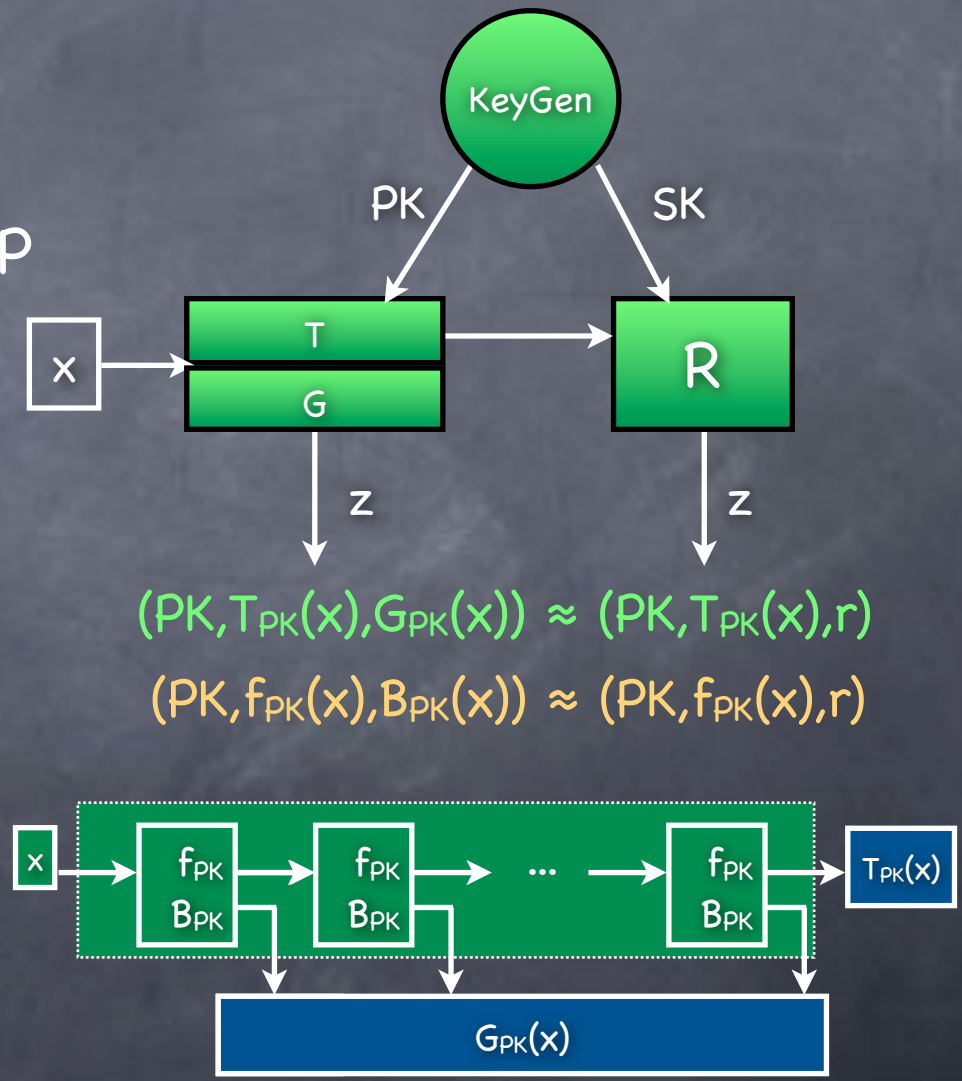
- Same construction as PRG from OWP
- One bit TPRG
  - KeyGen same as TOWP's KeyGen
  - $G_{PK}(x) := B_{PK}(x)$. $T_{PK}(x) := f_{PK}(x)$. $R_{SK}(y) := G_{PK}(f'_{SK}(y))$
    - (SK assumed to contain PK)
- More generally, last permutation output serves as T_{PK}





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}



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. $\gcd(e,\varphi(N)) = 1$ (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. $\gcd(e,\varphi(N)) = 1$ (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. $\gcd(e,\varphi(N)) = 1$ (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 $P, Q \text{ are } \equiv 3 \pmod 4$
    - 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. $\gcd(e,\varphi(N)) = 1$ (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. $x^{ed} = x$

see handout

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

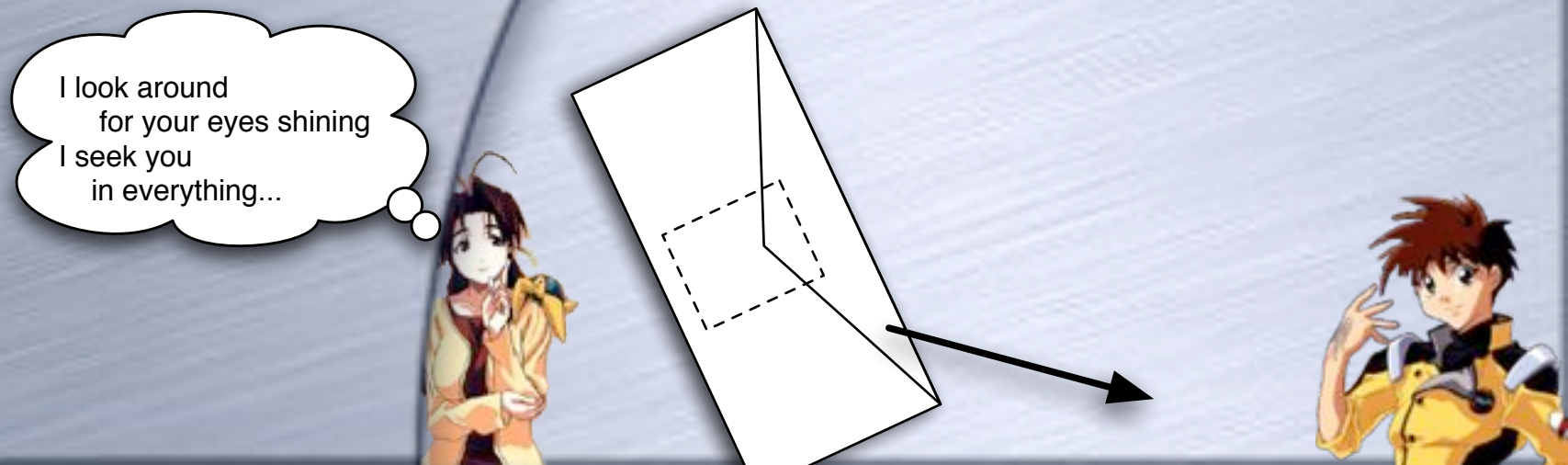
A subtle
e-mail attack

Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure

A subtle
e-mail attack

Alice → Bob: Enc(m)

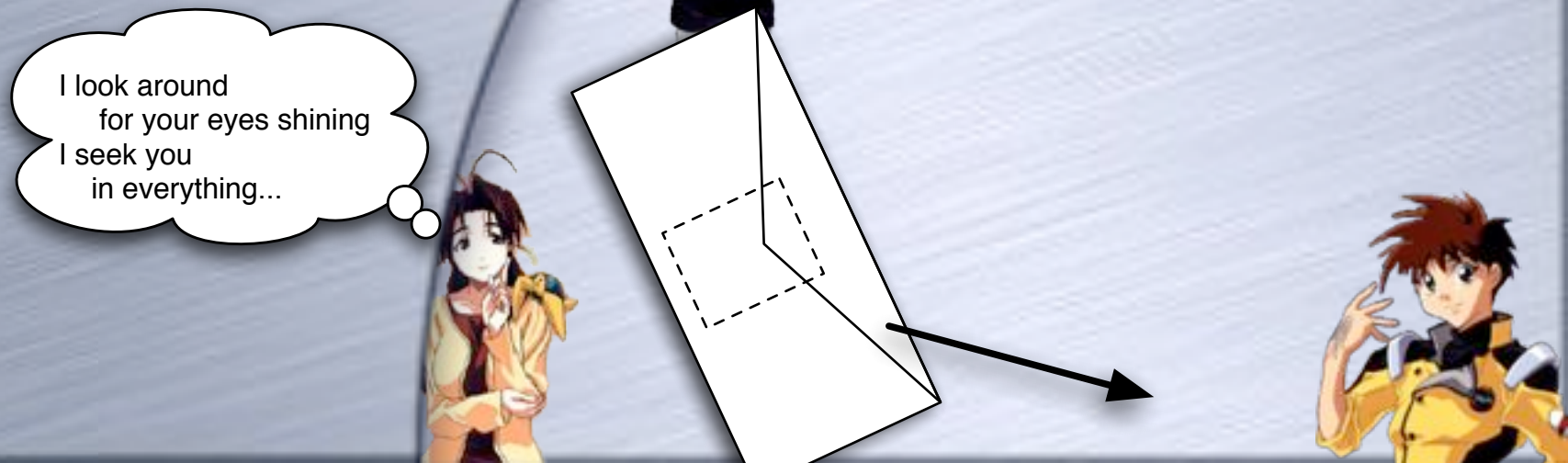


Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure

A subtle
e-mail attack

Alice → Bob: Enc(m)

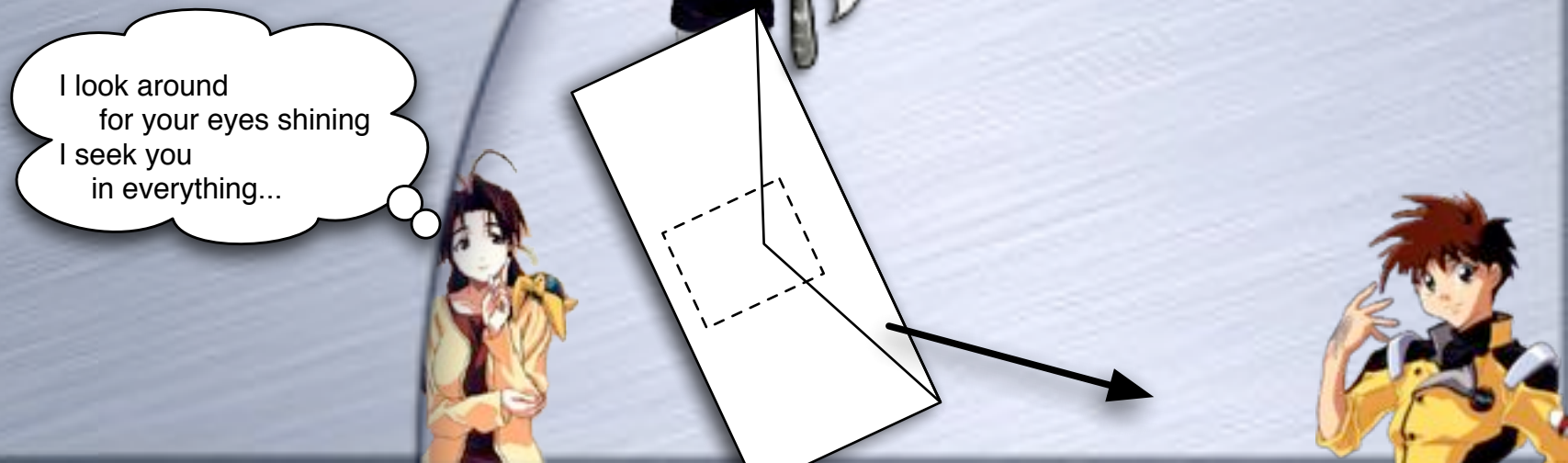


Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure

A subtle
e-mail attack

Alice → Bob: Enc(m)

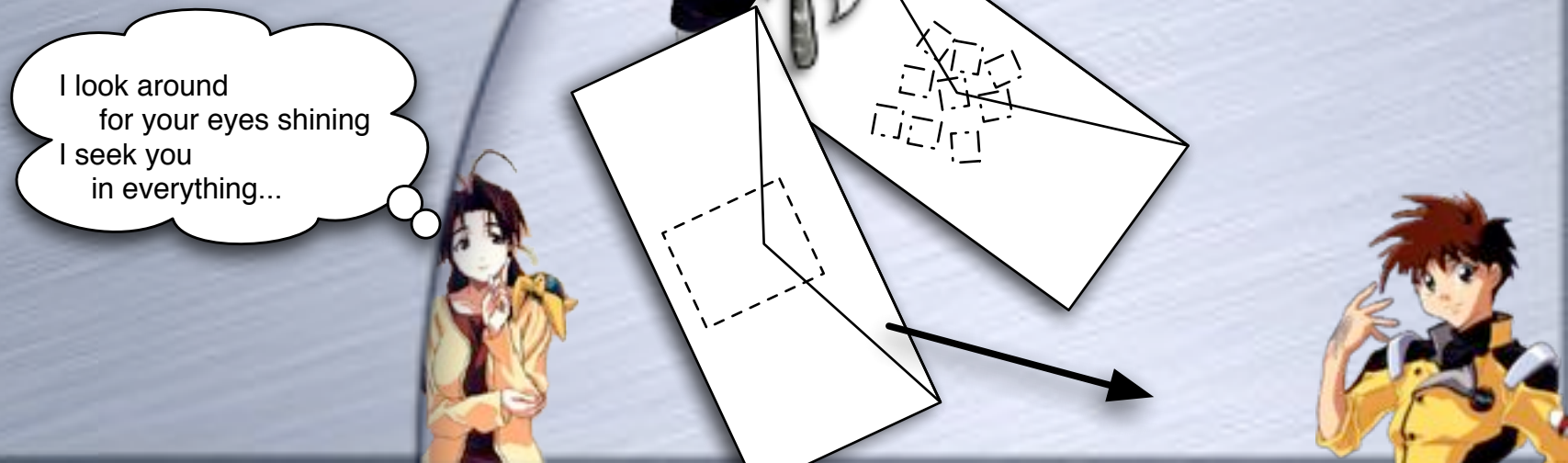


Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure

A subtle
e-mail attack

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

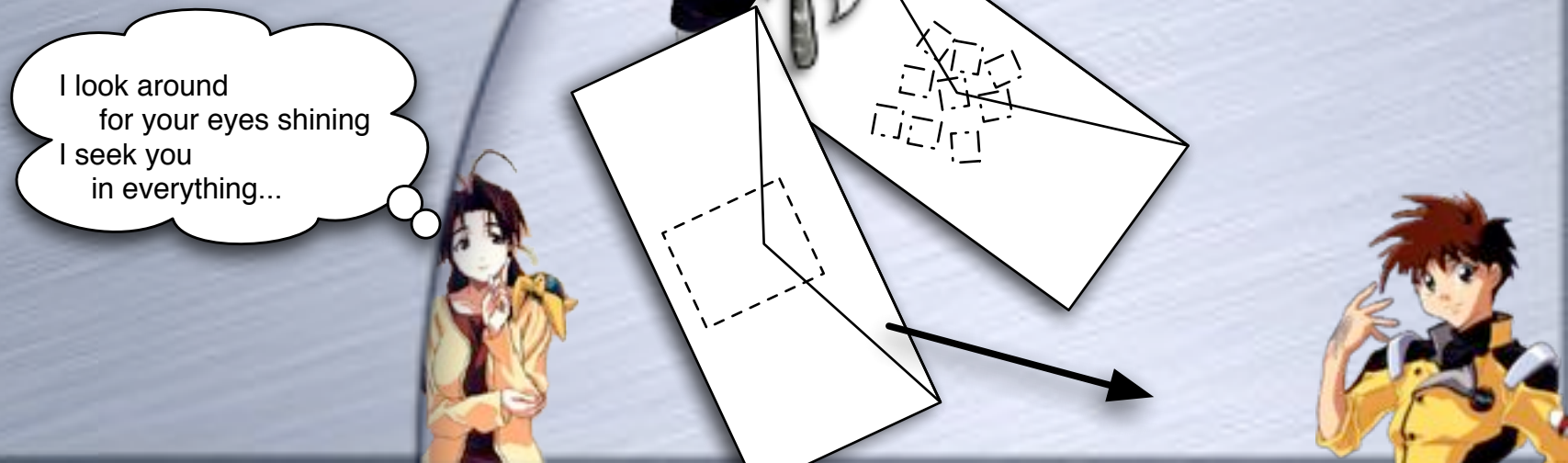


Chosen Ciphertext Attack

- 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)

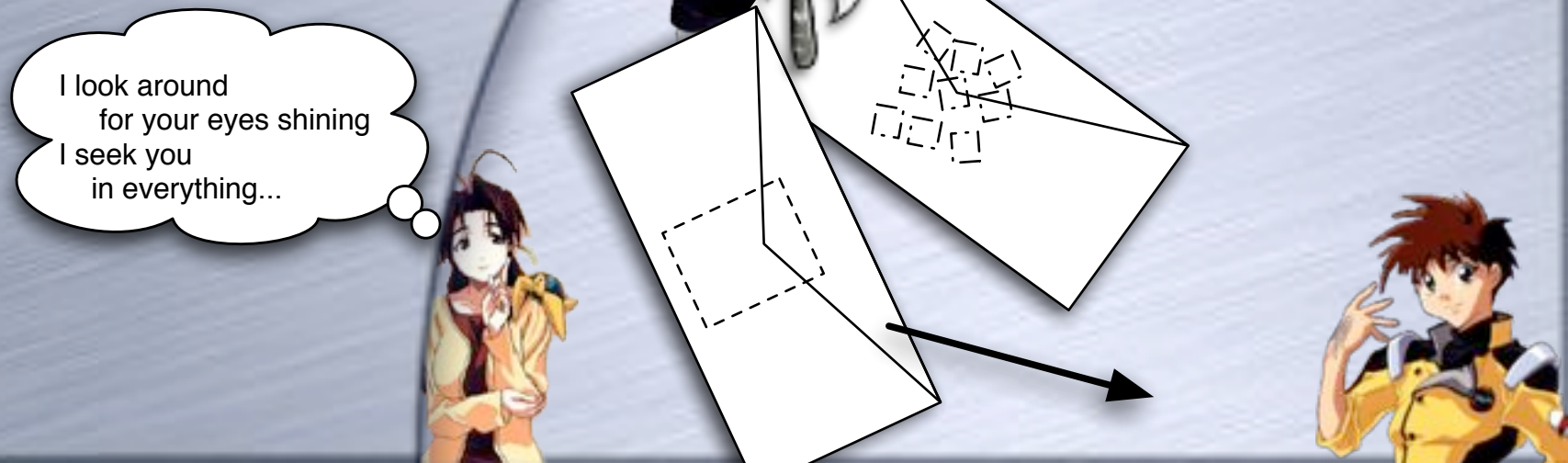


Chosen Ciphertext Attack

- 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)

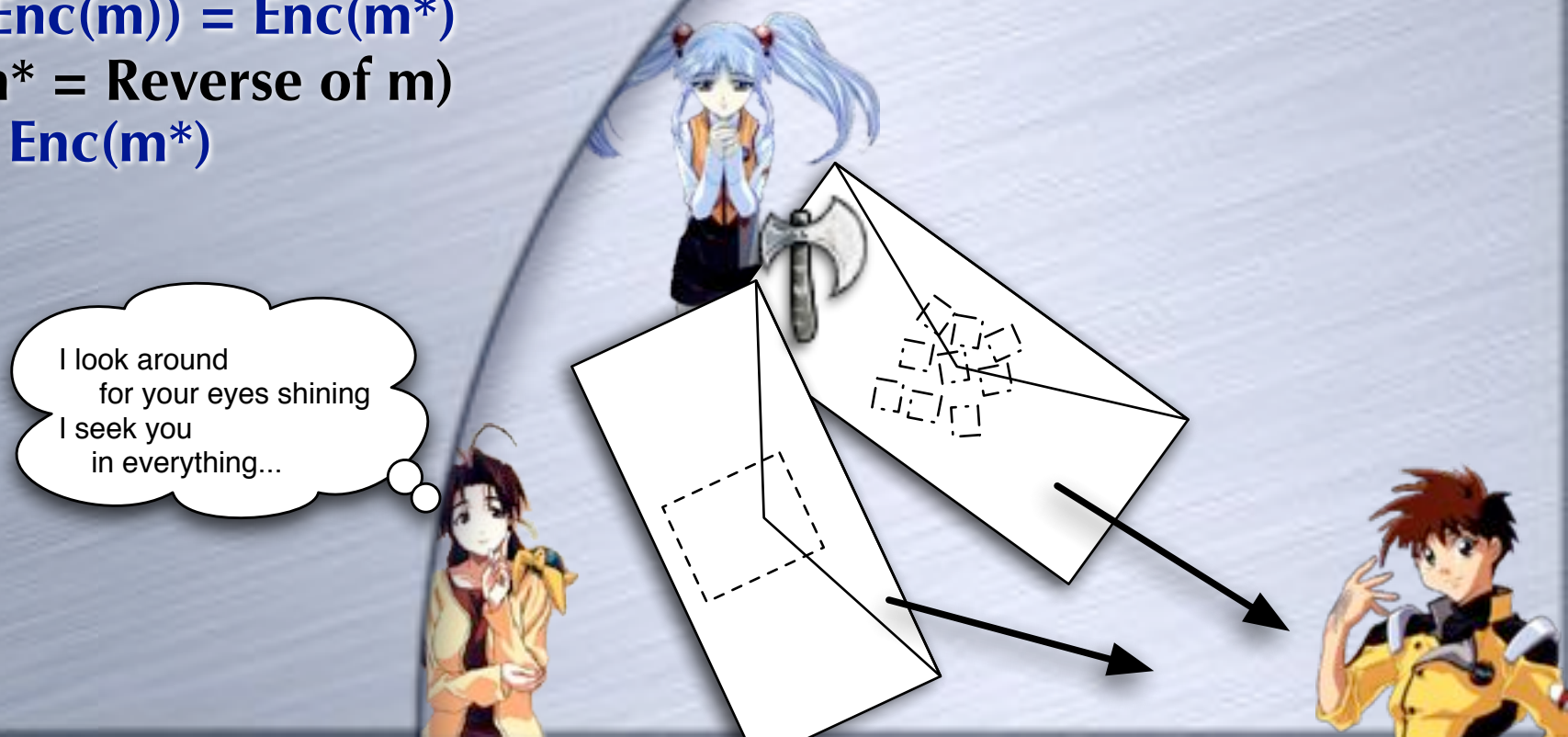


Chosen Ciphertext Attack

- 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*)

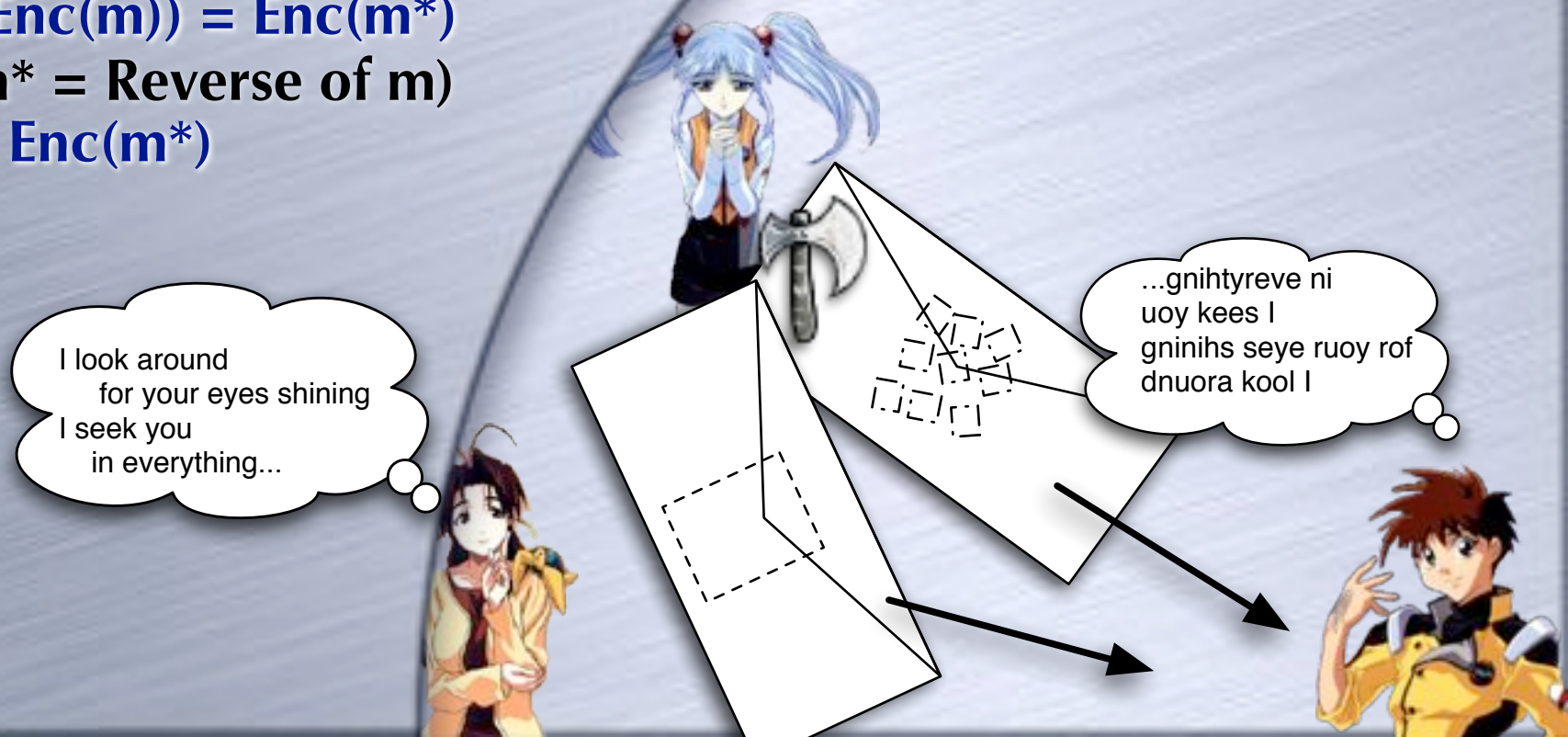


Chosen Ciphertext Attack

- 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*)

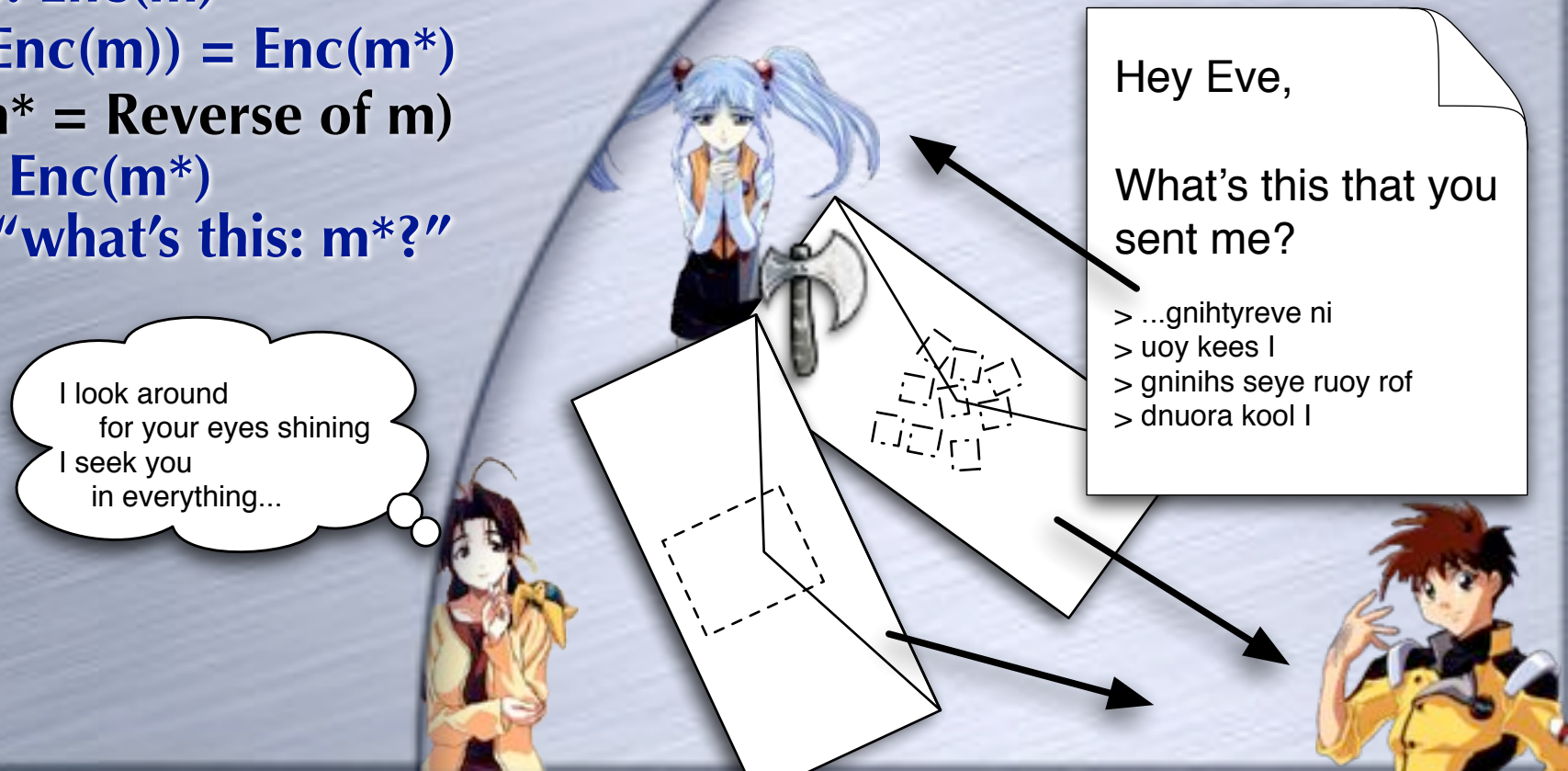


Chosen Ciphertext Attack

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

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*?"

A subtle e-mail attack

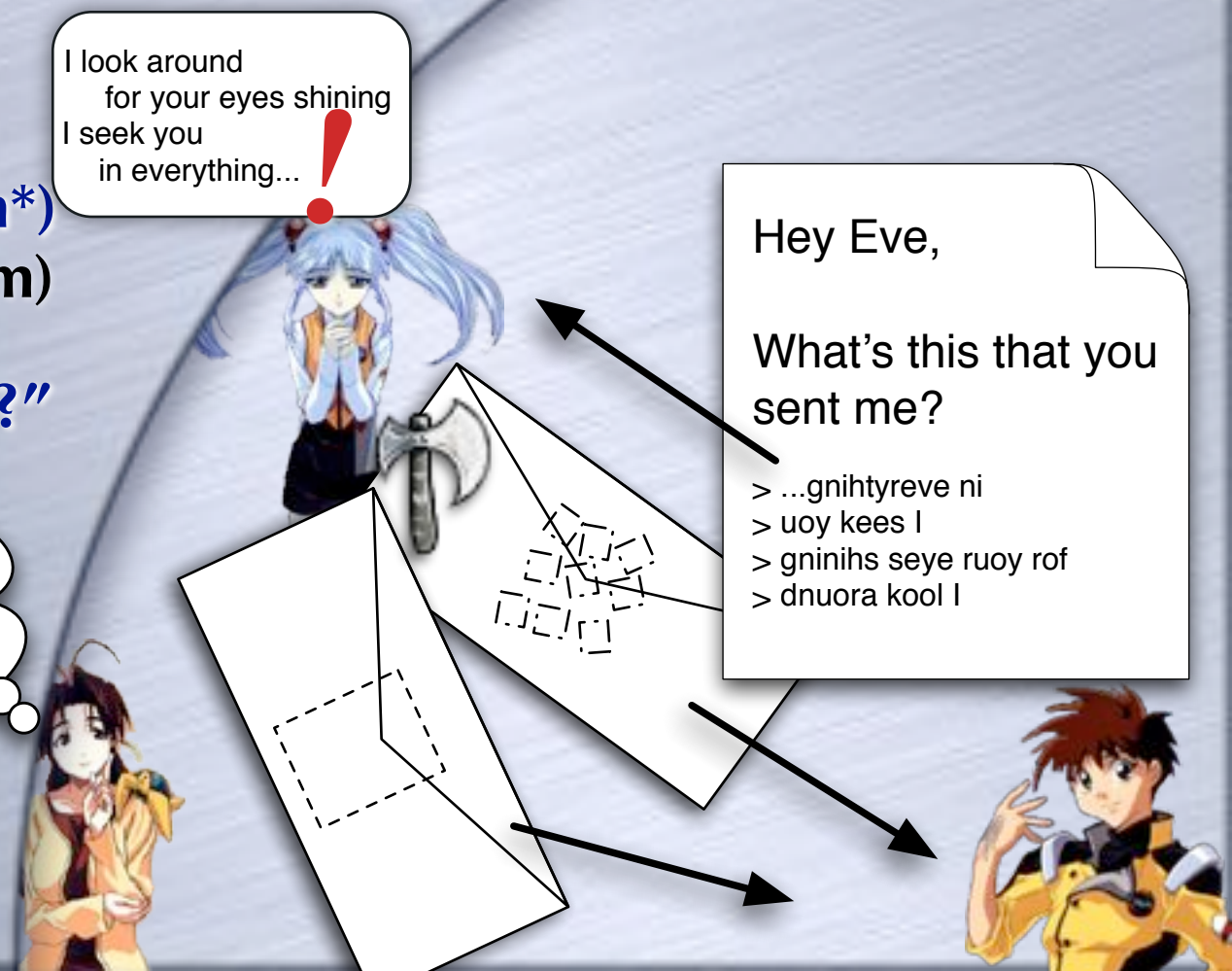


Chosen Ciphertext Attack

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

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!

A subtle e-mail attack



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: $Enc_{(G,g,Y)}(m) = (g^x, M.Y^x)$

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: $Enc_{(G,g,Y)}(m) = (g^x, M.Y^x)$
  - 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: $Enc_{(G,g,Y)}(m) = (g^x, M.Y^x)$
  - 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: $Enc_{(G,g,Y)}(m) = (g^x, M.Y^x)$
  - 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

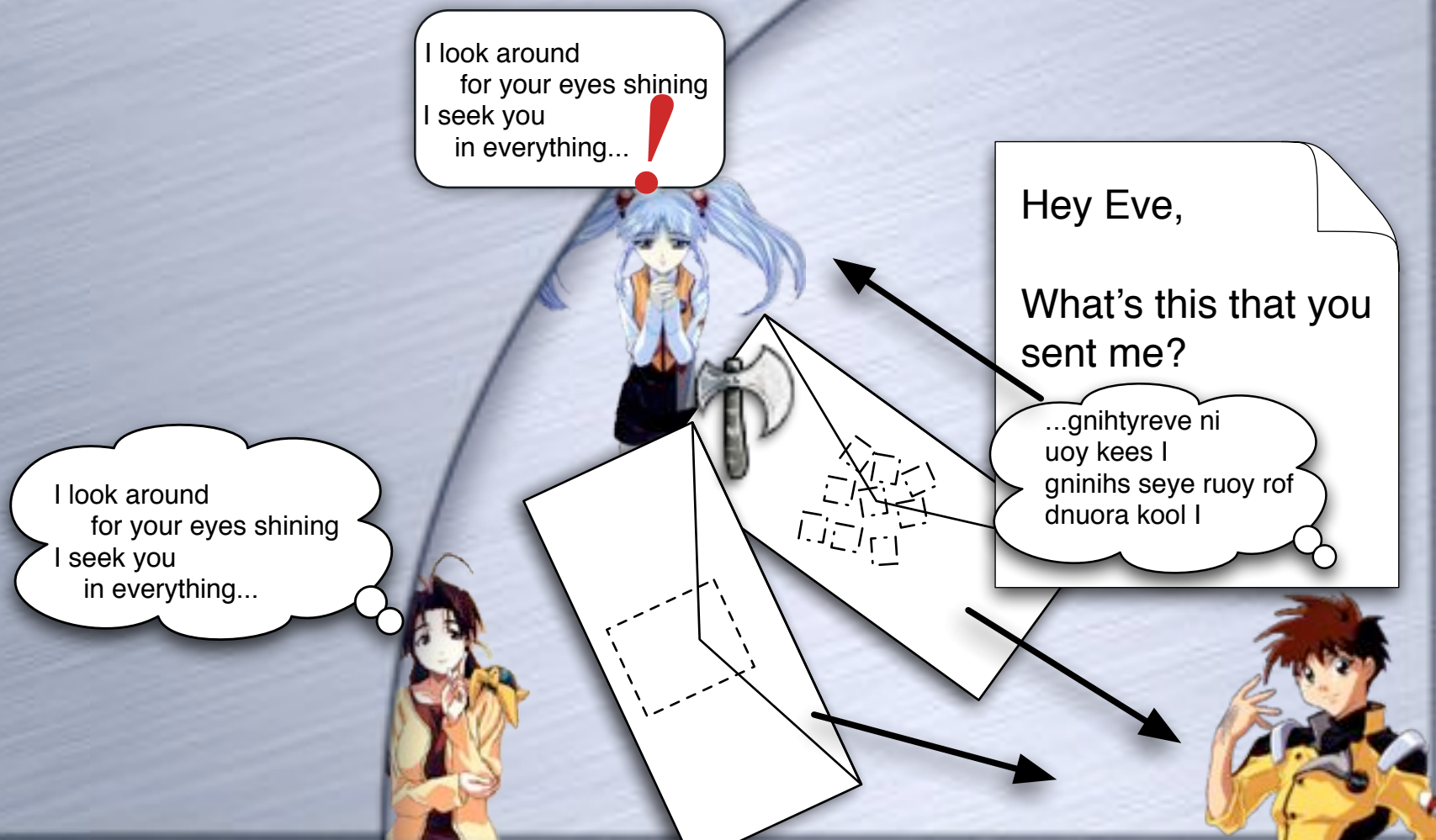
- Malleability: Eve can "malleate" a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a "related" message
- E.g.: Malleability of El Gamal
  - Recall: $Enc_{(G,g,Y)}(m) = (g^x, M.Y^x)$
  - Given (X,C) change it to (X,TC): will decrypt to TM
  - Or change (X,C)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

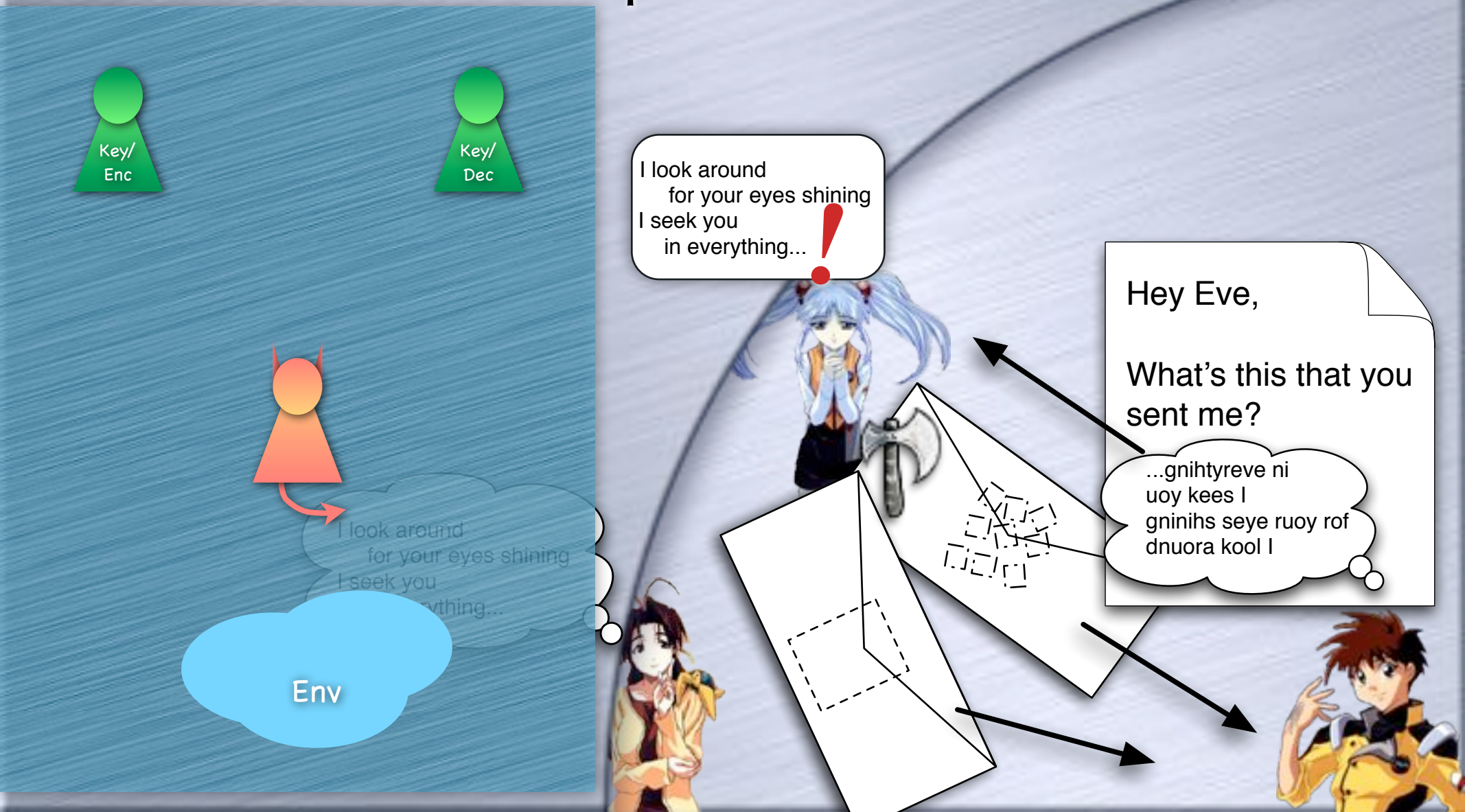
Chosen Ciphertext Attack

- SIM-CCA: does capture this attack



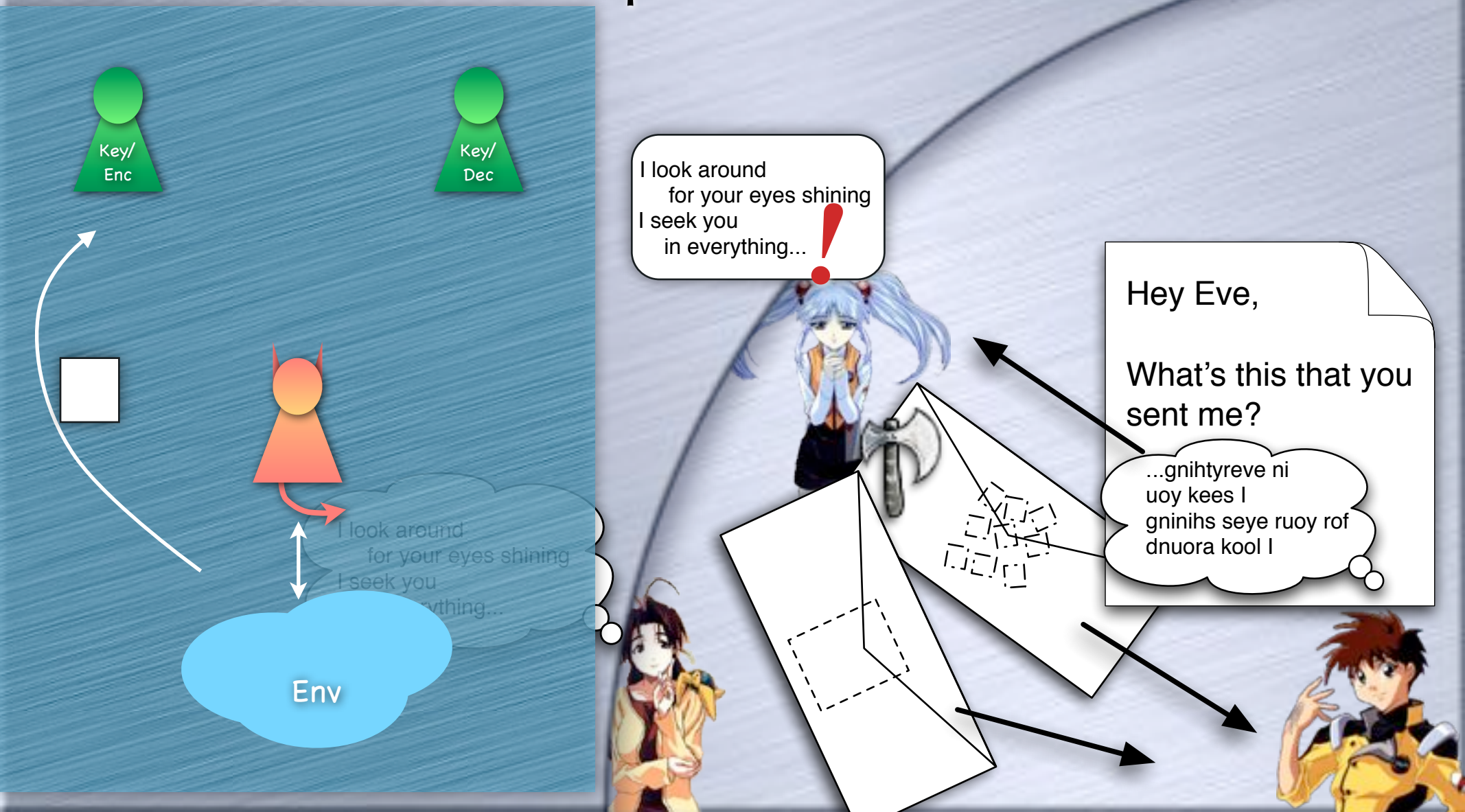
Chosen Ciphertext Attack

- SIM-CCA: does capture this attack



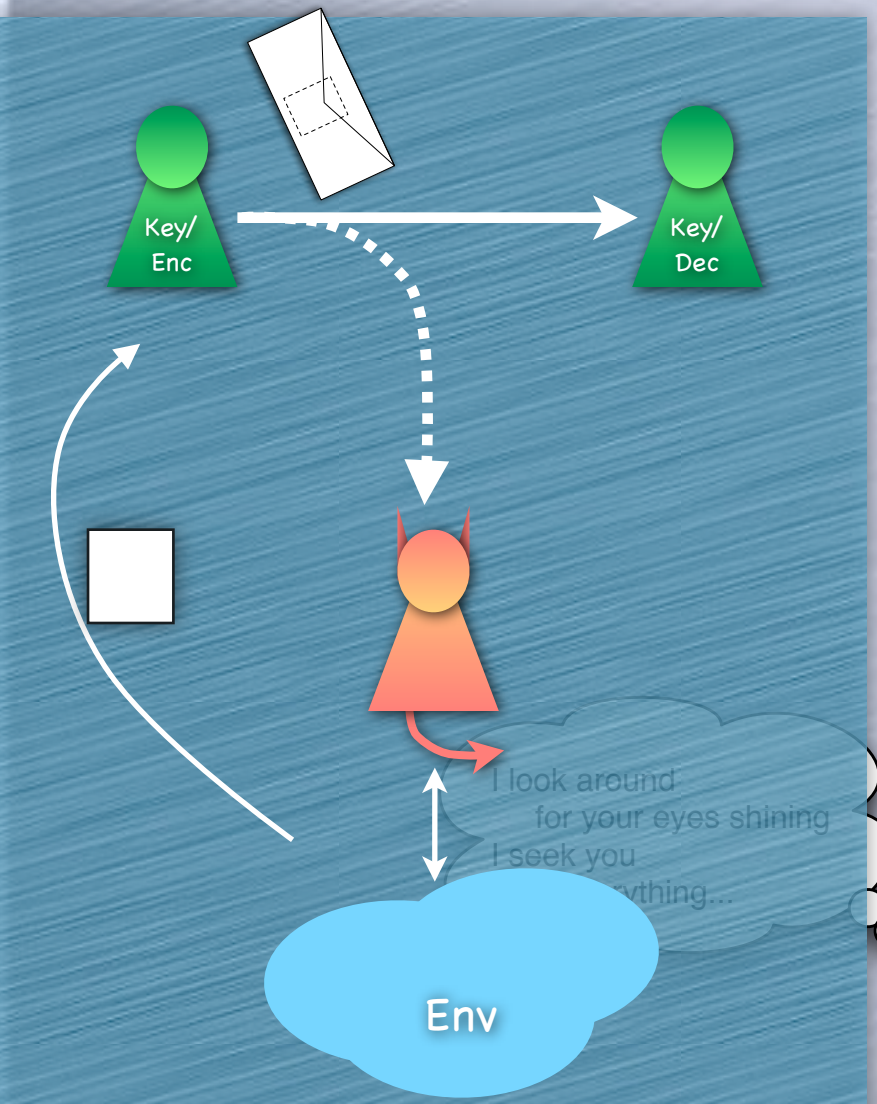
Chosen Ciphertext Attack

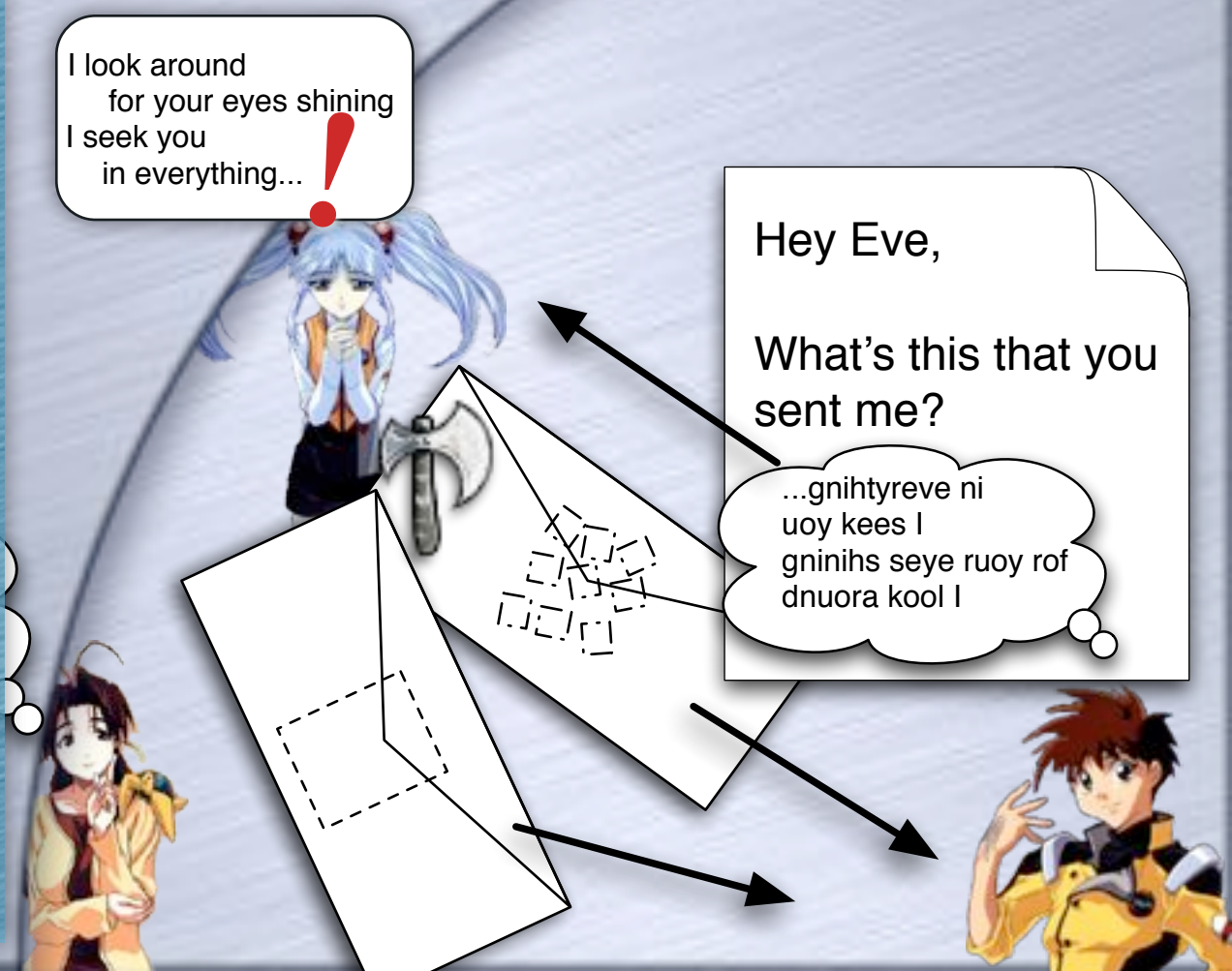
- SIM-CCA: does capture this attack



Chosen Ciphertext Attack

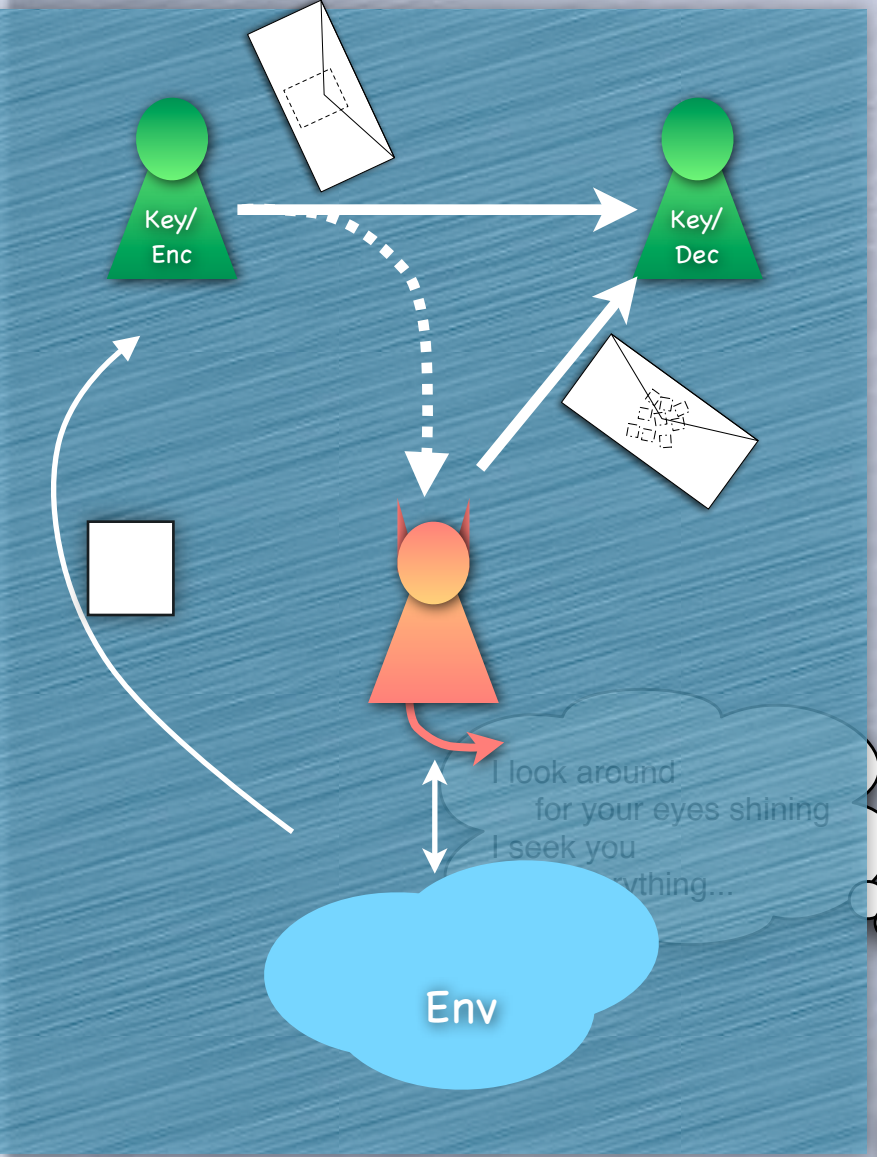
- SIM-CCA: does capture this attack

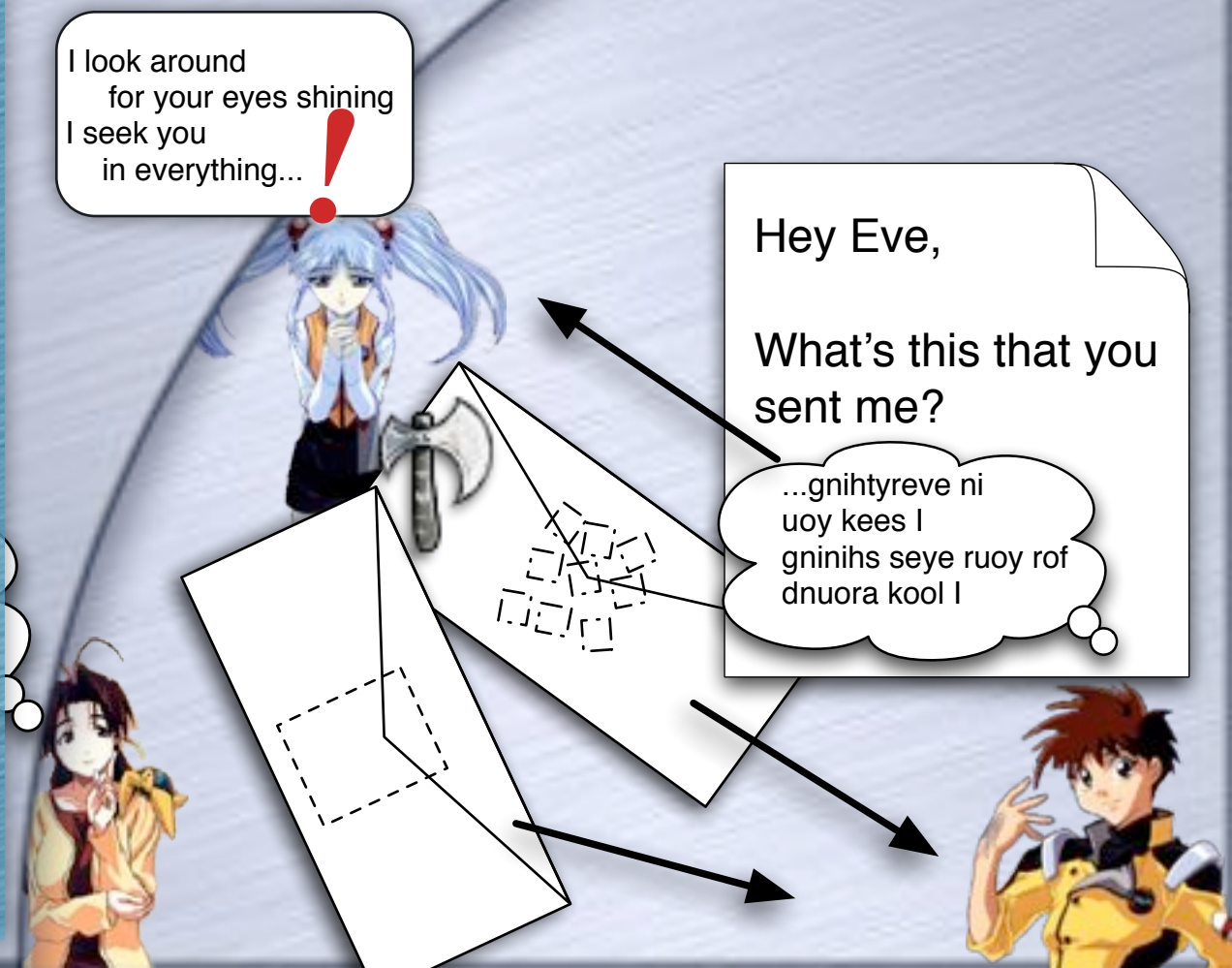




Chosen Ciphertext Attack

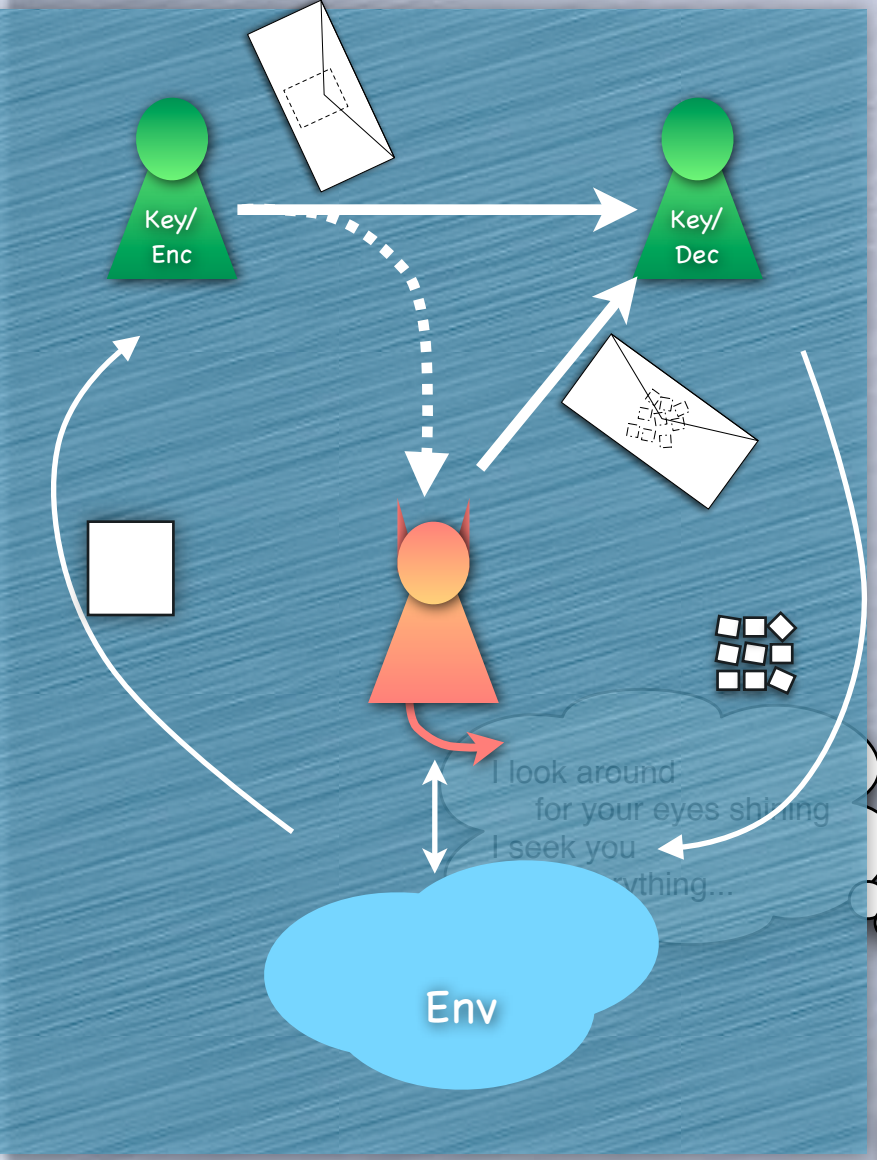
- SIM-CCA: does capture this attack

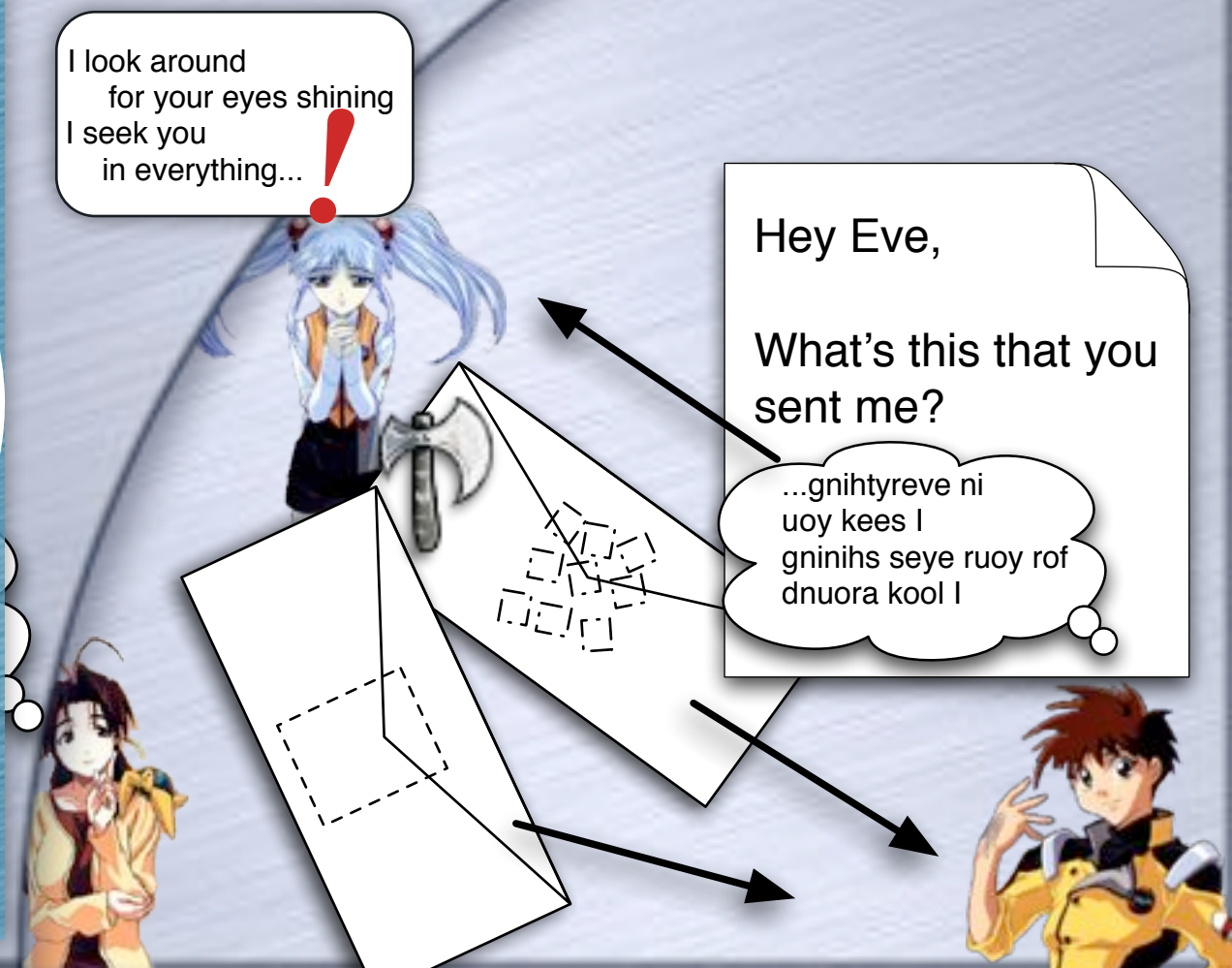




Chosen Ciphertext Attack

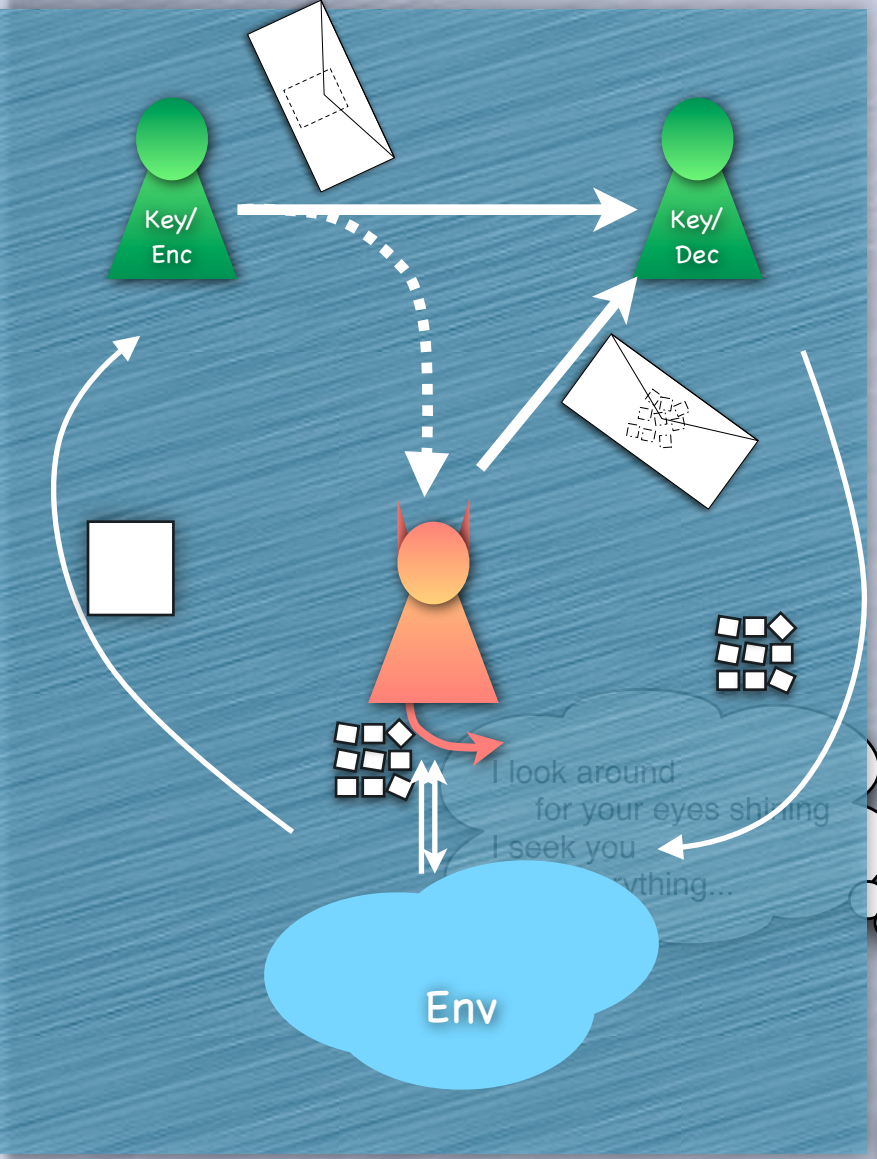
- SIM-CCA: does capture this attack

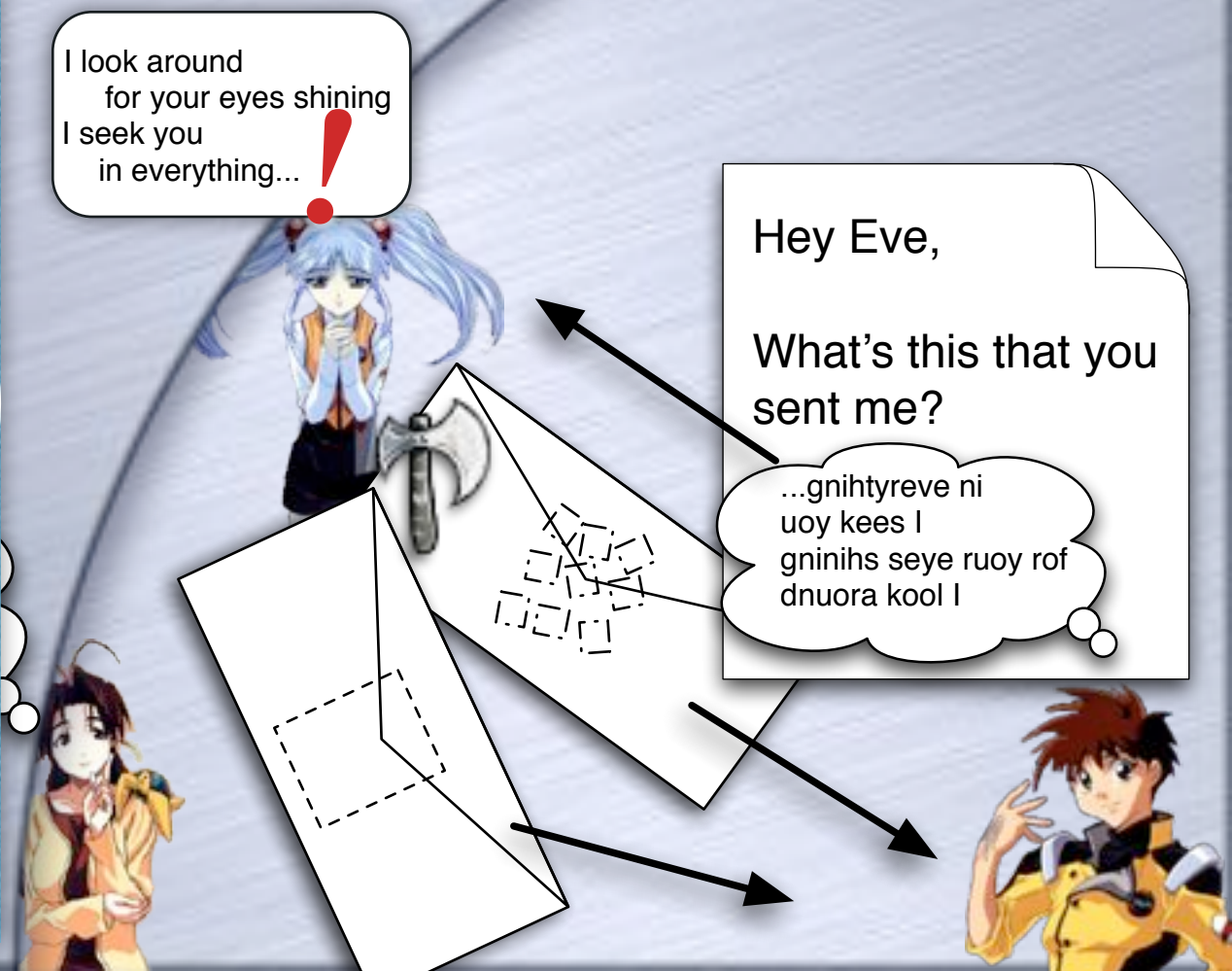




Chosen Ciphertext Attack

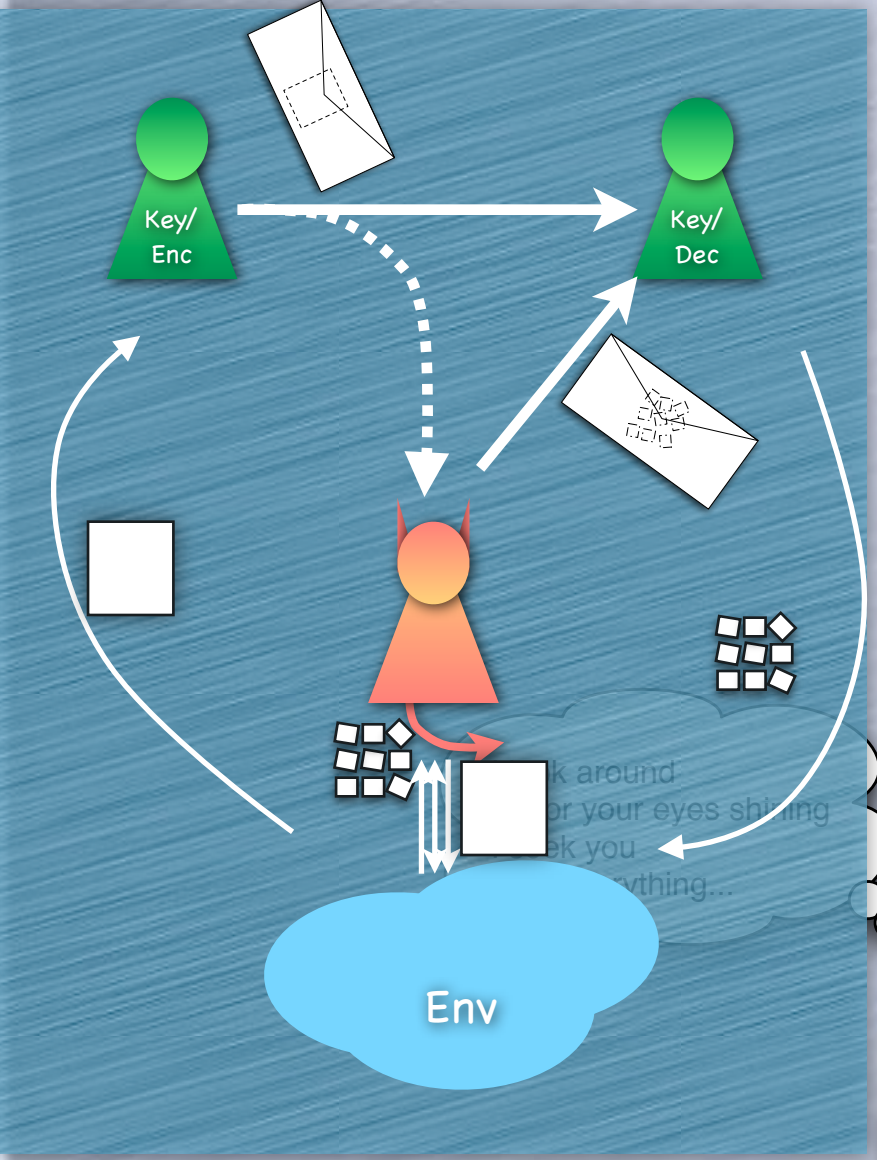
- SIM-CCA: does capture this attack

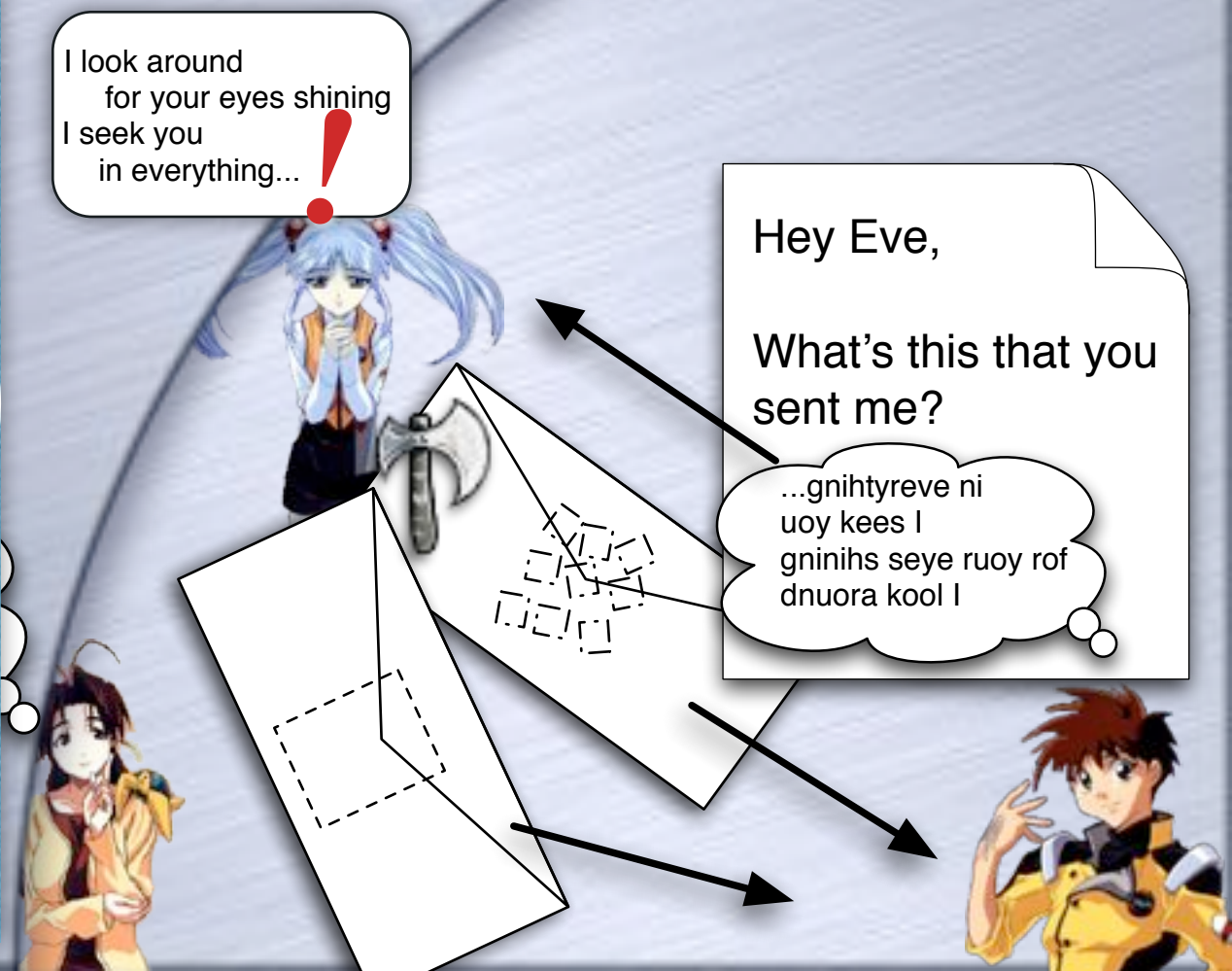




Chosen Ciphertext Attack

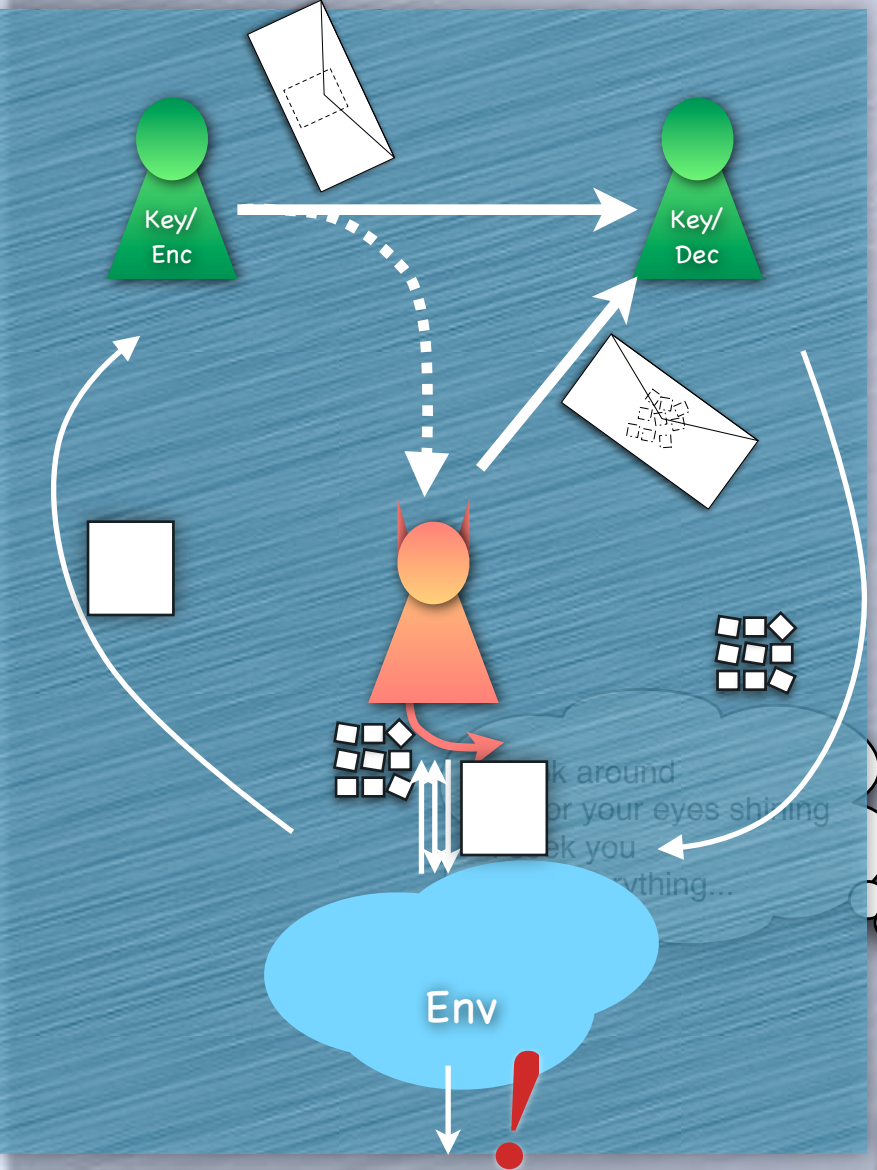
- SIM-CCA: does capture this attack

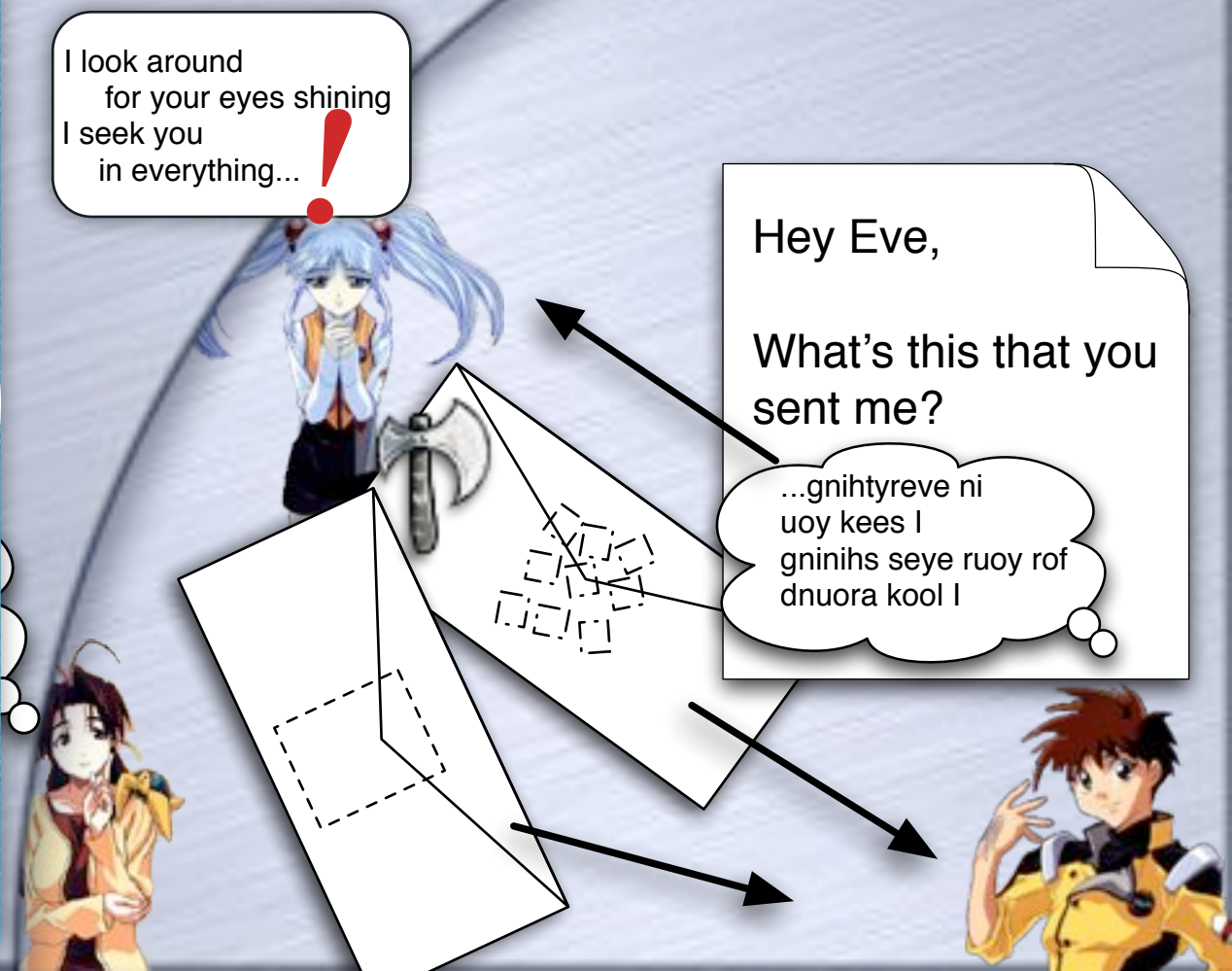




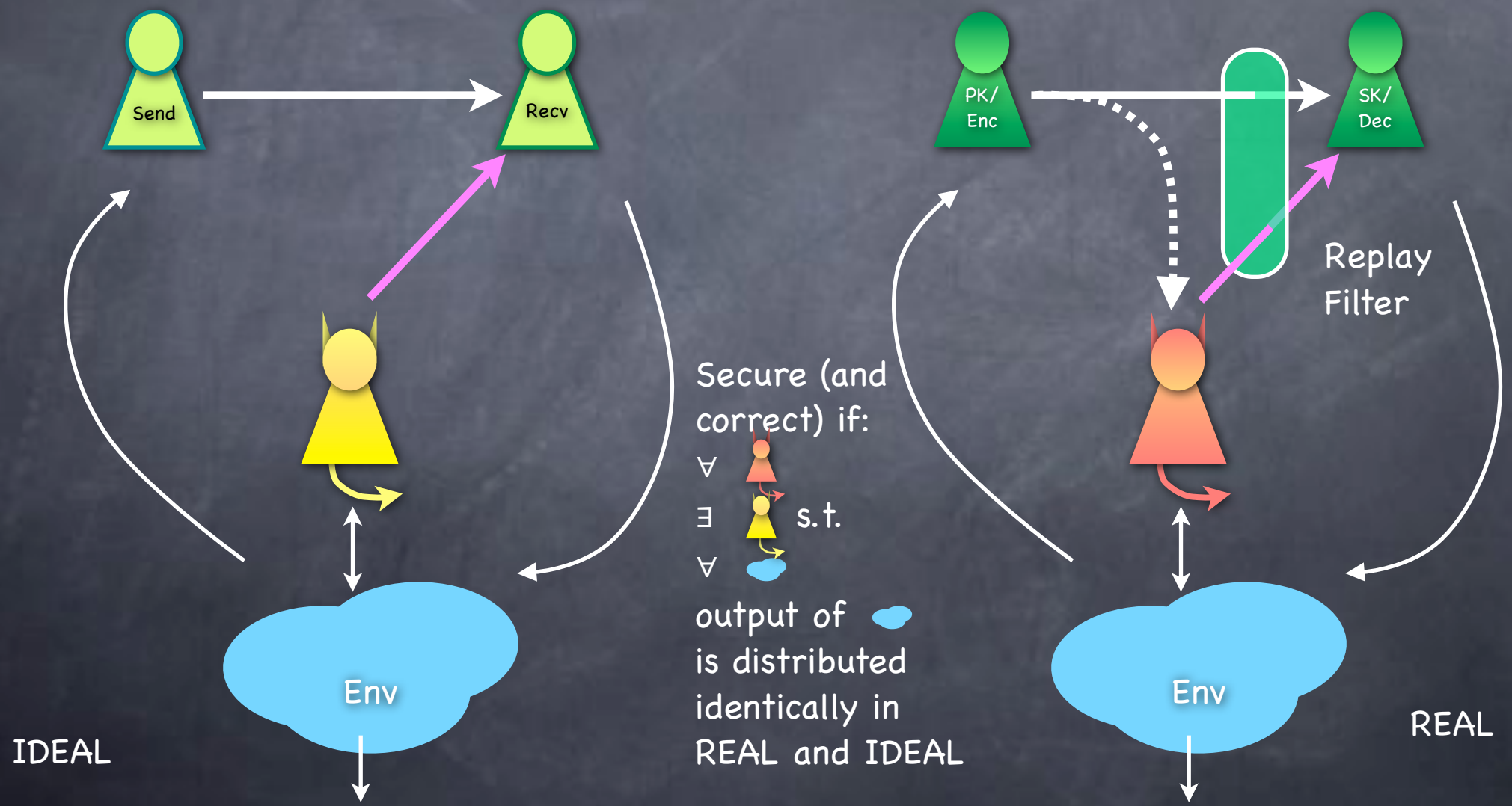
Chosen Ciphertext Attack

- SIM-CCA: does capture this attack

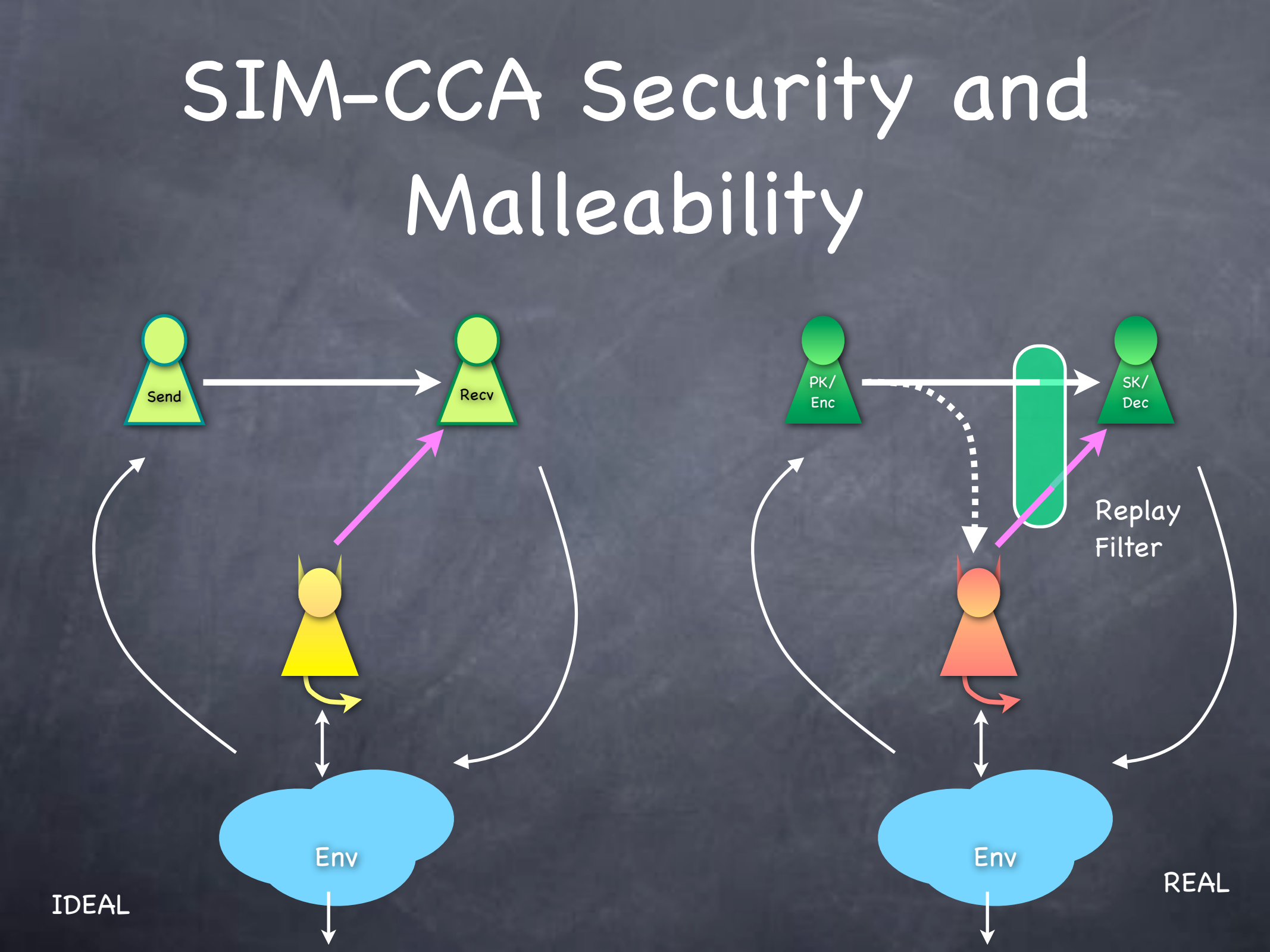




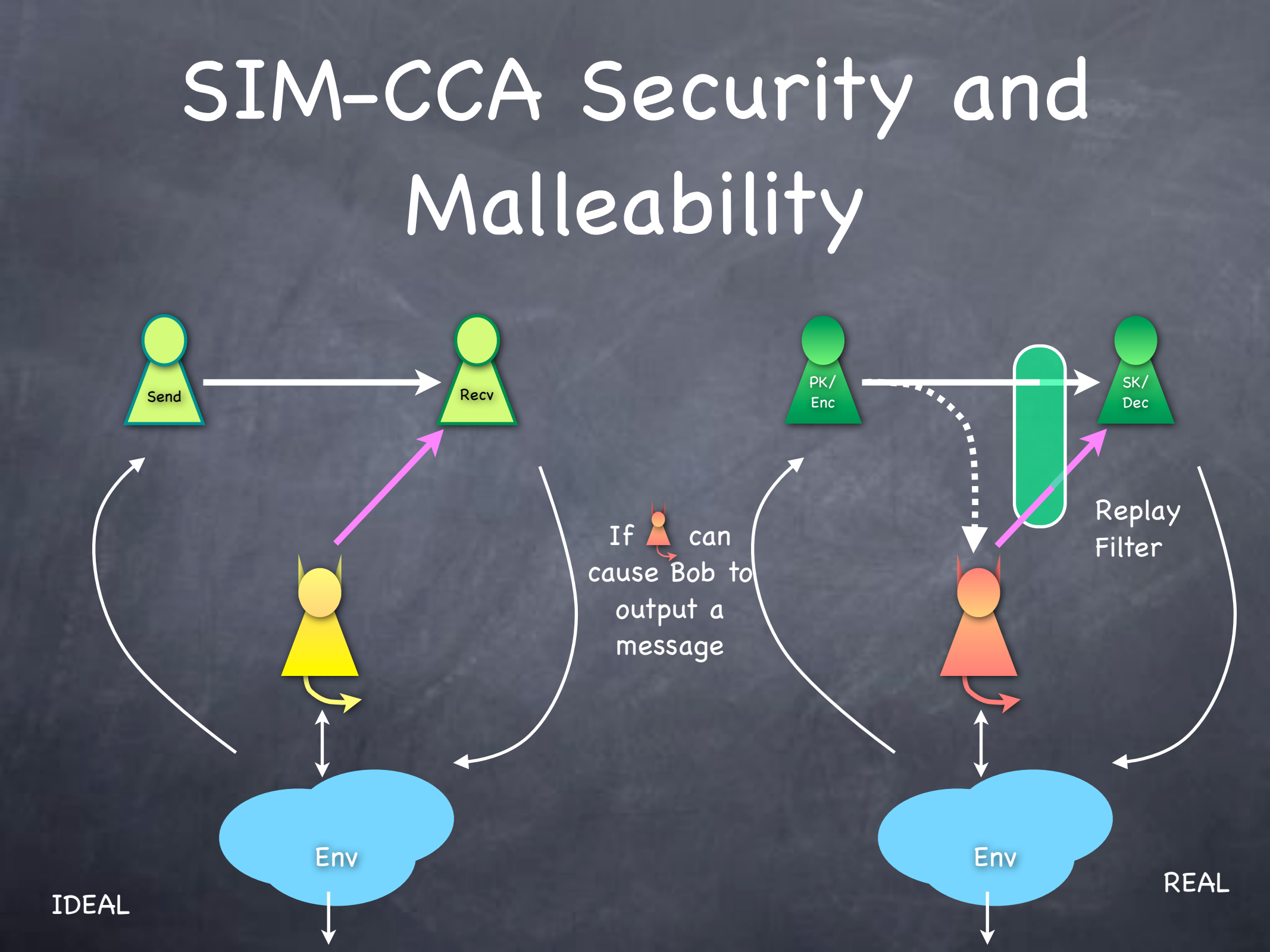
SIM-CCA Security (PKE)



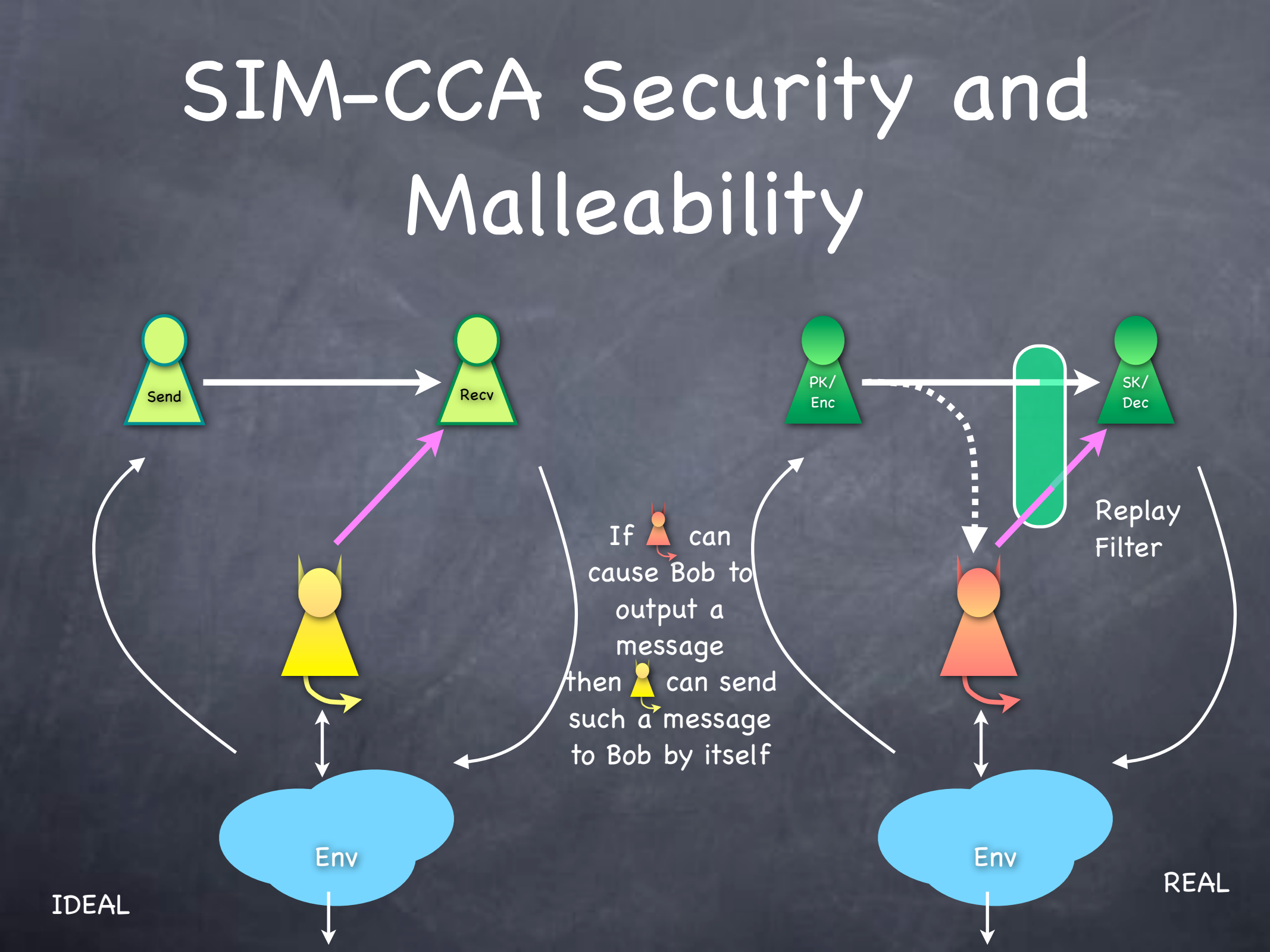
SIM-CCA Security and Malleability



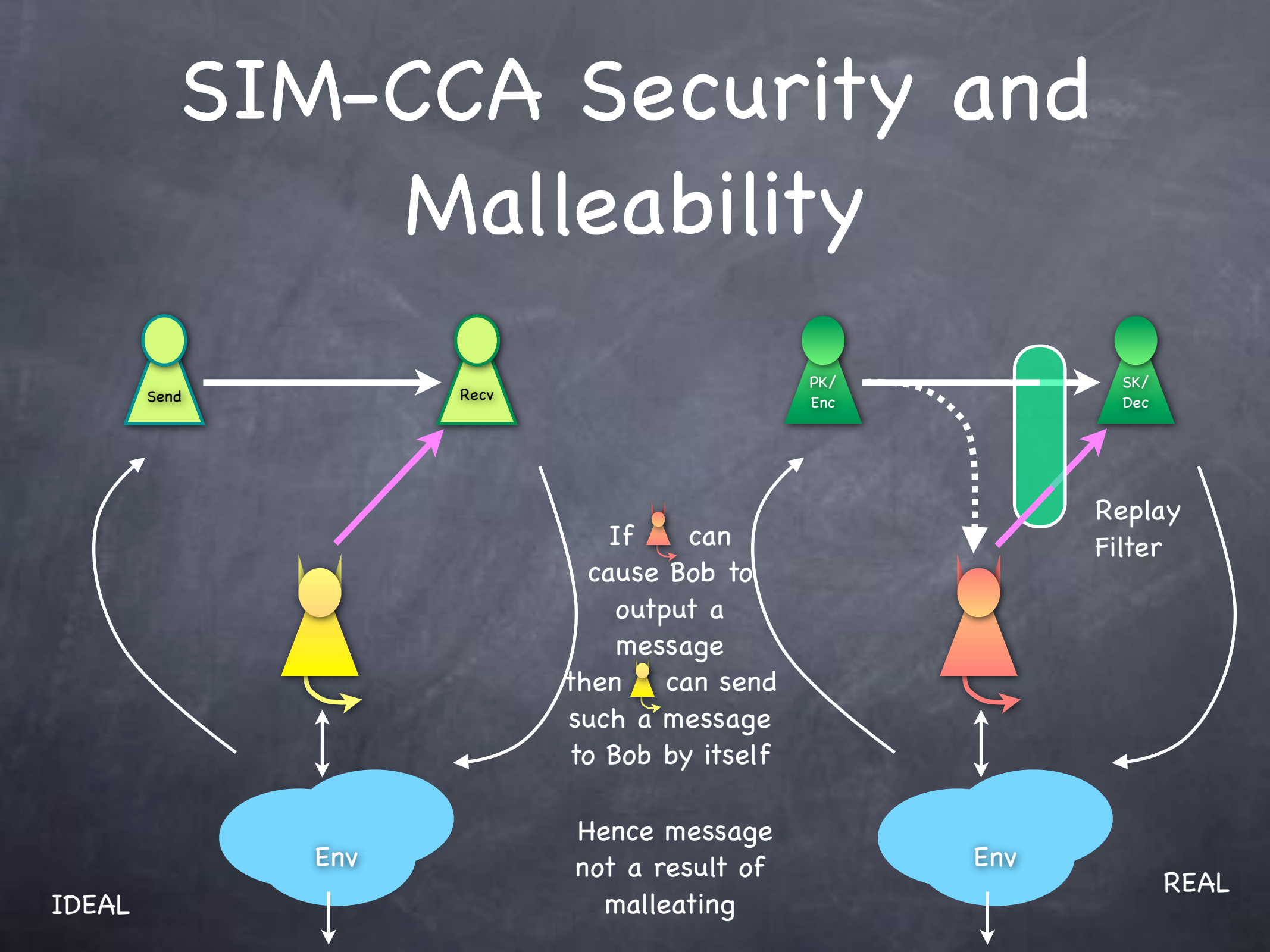
SIM-CCA Security and Malleability



SIM-CCA Security and Malleability



SIM-CCA Security and Malleability



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"

CCA Secure PKE: Cramer-Shoup

CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption

CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime p)

CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime p)
- Uses a collision-resistant hash function inside an "integrity tag"

CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime p)
- Uses a collision-resistant hash function inside an "integrity tag"
  - Enc(M) = (C,S)

CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime p)
- Uses a collision-resistant hash function inside an "integrity tag"
  - Enc(M) = (C,S)
    - $C = (g_1^x, g_2^x, MY^x)$ and $S = (WZ^{H(C)})^x$

CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime p)
- Uses a collision-resistant hash function inside an "integrity tag"
  - Enc(M) = (C,S)
    - $C = (g_1^x, g_2^x, MY^x)$ and $S = (WZ^{H(C)})^x$
    - g_1, g_2, Y, W, Z are part of PK

CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime p)
- Uses a collision-resistant hash function inside an "integrity tag"
  - Enc(M) = (C,S)
    - $C = (g_1^x, g_2^x, MY^x)$ and $S = (WZ^{H(C)})^x$
    - g_1, g_2, Y, W, Z are part of PK
      - $Y = g_1^{y1} g_2^{y2}$, $W = g_1^{w1} g_2^{w2}$, $Z = g_1^{z1} g_2^{z2}$.
        SK contains $(y_1,y_2,w_1,w_2,z_1,z_2)$

CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime p)
- Uses a collision-resistant hash function inside an "integrity tag"
  - Enc(M) = (C,S)
    - $C = (g_1^x, g_2^x, MY^x)$ and $S = (WZ^{H(C)})^x$
    - g_1, g_2, Y, W, Z are part of PK
      - $Y = g_1^{y1} g_2^{y2}$, $W = g_1^{w1} g_2^{w2}$, $Z = g_1^{z1} g_2^{z2}$.
        SK contains $(y_1,y_2,w_1,w_2,z_1,z_2)$

Multiple SKs can explain the same PK (unlike El Gamal)

CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime p)
- Uses a collision-resistant hash function inside an "integrity tag"
  - Enc(M) = (C,S)
    - $C = (g_1^x, g_2^x, MY^x)$ and $S = (WZ^{H(C)})^x$
    - g_1, g_2, Y, W, Z are part of PK
      - $Y = g_1^{y1} g_2^{y2}$, $W = g_1^{w1} g_2^{w2}$, $Z = g_1^{z1} g_2^{z2}$.
        SK contains $(y_1,y_2,w_1,w_2,z_1,z_2)$

Multiple SKs can explain the same PK (unlike El Gamal)

  - Trapdoor: Using SK, and (g_1^x,g_2^x) can find Y^x, W^x, Z^x

CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime p)
- Uses a collision-resistant hash function inside an "integrity tag"
  - Enc(M) = (C,S)
    - $C = (g_1^x, g_2^x, MY^x)$ and $S = (WZ^{H(C)})^x$
    - g_1, g_2, Y, W, Z are part of PK
      - $Y = g_1^{y1} g_2^{y2}$, $W = g_1^{w1} g_2^{w2}$, $Z = g_1^{z1} g_2^{z2}$.
        SK contains $(y_1,y_2,w_1,w_2,z_1,z_2)$

        Multiple SKs can explain the same PK (unlike El Gamal)
  - Trapdoor: Using SK, and (g_1^x, g_2^x) can find Y^x, W^x, Z^x
    - If (g_1^{x1}, g_2^{x2}), $x_1 \neq x_2$, then "Y^x, W^x, Z^x" vary with different SKs

CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime p)
- Uses a collision-resistant hash function inside an "integrity tag"
  - Enc(M) = (C,S)
    - $C = (g_1^x, g_2^x, MY^x)$ and $S = (WZ^{H(C)})^x$
    - g_1, g_2, Y, W, Z are part of PK
      - $Y = g_1^{y1} g_2^{y2}$, $W = g_1^{w1} g_2^{w2}$, $Z = g_1^{z1} g_2^{z2}$.
        SK contains $(y_1,y_2,w_1,w_2,z_1,z_2)$

        Multiple SKs can explain the same PK (unlike El Gamal)
  - Trapdoor: Using SK, and (g_1^x,g_2^x) can find Y^x, W^x, Z^x
    - If (g_1^{x1},g_2^{x2}), $x_1 \neq x_2$, then "Y^x, W^x, Z^x" vary with different SKs
  - Decryption: Check S (assuming $x_1=x_2$) and extract M

Security of CS Scheme: Proof Sketch

Security of CS Scheme: Proof Sketch

- An "invalid encryption" can be used for challenge such that

Security of CS Scheme: Proof Sketch

- An "invalid encryption" can be used for challenge such that
  - It contains no information about the message (given just PK)

Security of CS Scheme: Proof Sketch

- An "invalid encryption" can be used for challenge such that
  - It contains no information about the message (given just PK)
  - Is indistinguishable from valid encryption, under DDH assumption

Security of CS Scheme: Proof Sketch

$(g_1, g_1^{x1}, g_2, g_2^{x2})$ is of the form (g, g^x, g^y, g^{xy}) iff $x_1 = x_2$

- An "invalid encryption" can be used for challenge such that
  - It contains no information about the message (given just PK)
  - Is indistinguishable from valid encryption, under DDH assumption

Security of CS Scheme: Proof Sketch

$(g_1, g_1^{x1}, g_2, g_2^{x2})$ is of the form (g, g^x, g^y, g^{xy}) iff $x_1 = x_2$

- An "invalid encryption" can be used for challenge such that
  - It contains no information about the message (given just PK)
  - Is indistinguishable from valid encryption, under DDH assumption
- But adversary could get information about the specific SK from decryption queries?

Security of CS Scheme: Proof Sketch

$(g_1, g_1^{x1}, g_2, g_2^{x2})$ is of the form (g, g^x, g^y, g^{xy}) iff $x_1 = x_2$

- An "invalid encryption" can be used for challenge such that
  - It contains no information about the message (given just PK)
  - Is indistinguishable from valid encryption, under DDH assumption
- But adversary could get information about the specific SK from decryption queries?
  - By querying decryption with only valid ciphertexts, adversary gets no information about SK (beyond given by PK)

Security of CS Scheme: Proof Sketch

$(g_1, g_1^{x1}, g_2, g_2^{x2})$ is of the form (g, g^x, g^y, g^{xy}) iff $x_1 = x_2$

- An "invalid encryption" can be used for challenge such that
  - It contains no information about the message (given just PK)
  - Is indistinguishable from valid encryption, under DDH assumption
- But adversary could get information about the specific SK from decryption queries?
  - By querying decryption with only valid ciphertexts, adversary gets no information about SK (beyond given by PK)
  - Adversary can't create new "invalid ciphertexts" that get past the integrity check (except with negligible probability)

Security of CS Scheme: Proof Sketch

$(g_1, g_1^{x1}, g_2, g_2^{x2})$ is of the form (g, g^x, g^y, g^{xy}) iff $x_1 = x_2$

- An "invalid encryption" can be used for challenge such that
  - It contains no information about the message (given just PK)
  - Is indistinguishable from valid encryption, under DDH assumption
- But adversary could get information about the specific SK from decryption queries?
  - By querying decryption with only valid ciphertexts, adversary gets no information about SK (beyond given by PK)
  - Adversary can't create new "invalid ciphertexts" that get past the integrity check (except with negligible probability)
    - Any new invalid ciphertext can fool at most a negligible fraction of the possible SKs: so the probability of adversary fooling the specific one used is negligible

Security of CS Scheme: Proof Sketch

$(g_1, g_1^{x1}, g_2, g_2^{x2})$ is of the form (g, g^x, g^y, g^{xy}) iff $x_1 = x_2$

- An "invalid encryption" can be used for challenge such that
  - It contains no information about the message (given just PK)
  - Is indistinguishable from valid encryption, under DDH assumption
- But adversary could get information about the specific SK from decryption queries?
  - By querying decryption with only valid ciphertexts, adversary gets no information about SK (beyond given by PK)
  - Adversary can't create new "invalid ciphertexts" that get past the integrity check (except with negligible probability)
    - Any new invalid ciphertext can fool at most a negligible fraction of the possible SKs: so the probability of adversary fooling the specific one used is negligible
- Formally using "hybrid argument" (0 advantage in last hybrid)