Public-Key Cryptography

Public-Key Cryptography

Lecture 7

Public-Key Encryption

Public-Key Cryptography

Lecture 7

Public-Key Encryption

CCA Security

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 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

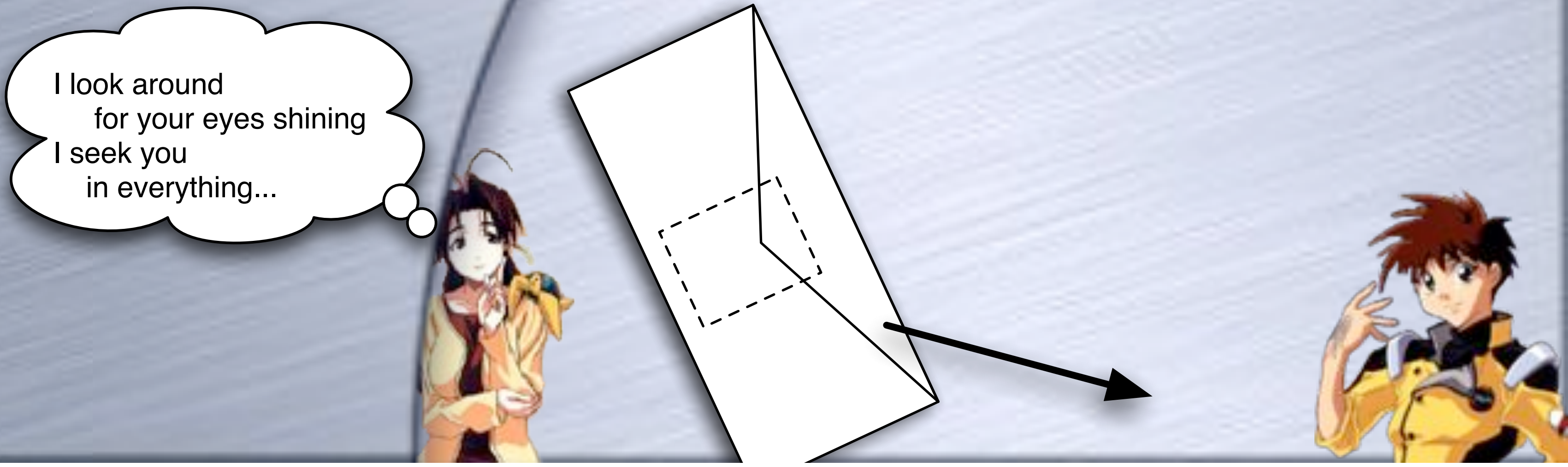
A subtle
e-mail attack

Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure

A subtle
e-mail attack

Alice → Bob: Enc(m)

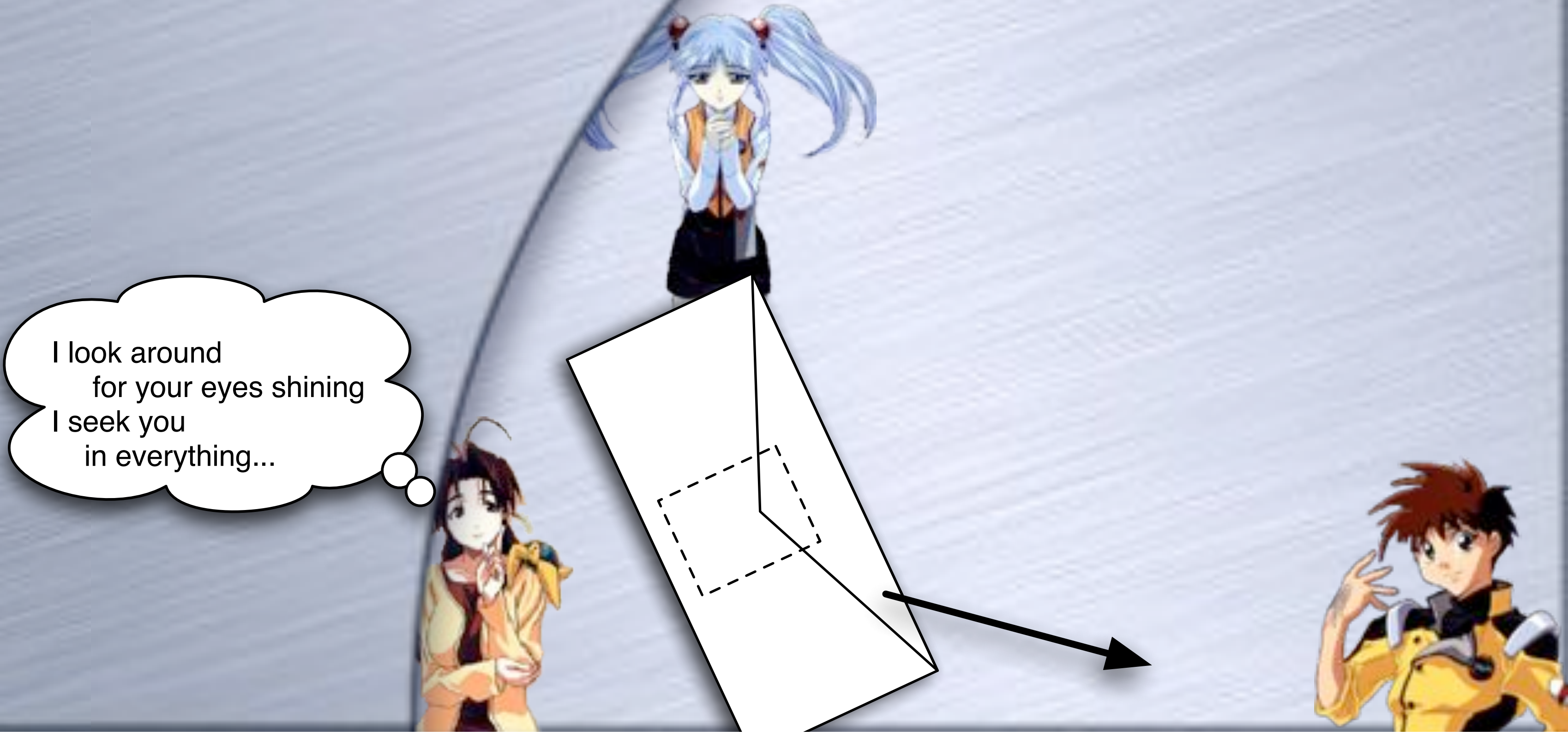


Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure

A subtle
e-mail attack

Alice → Bob: Enc(m)

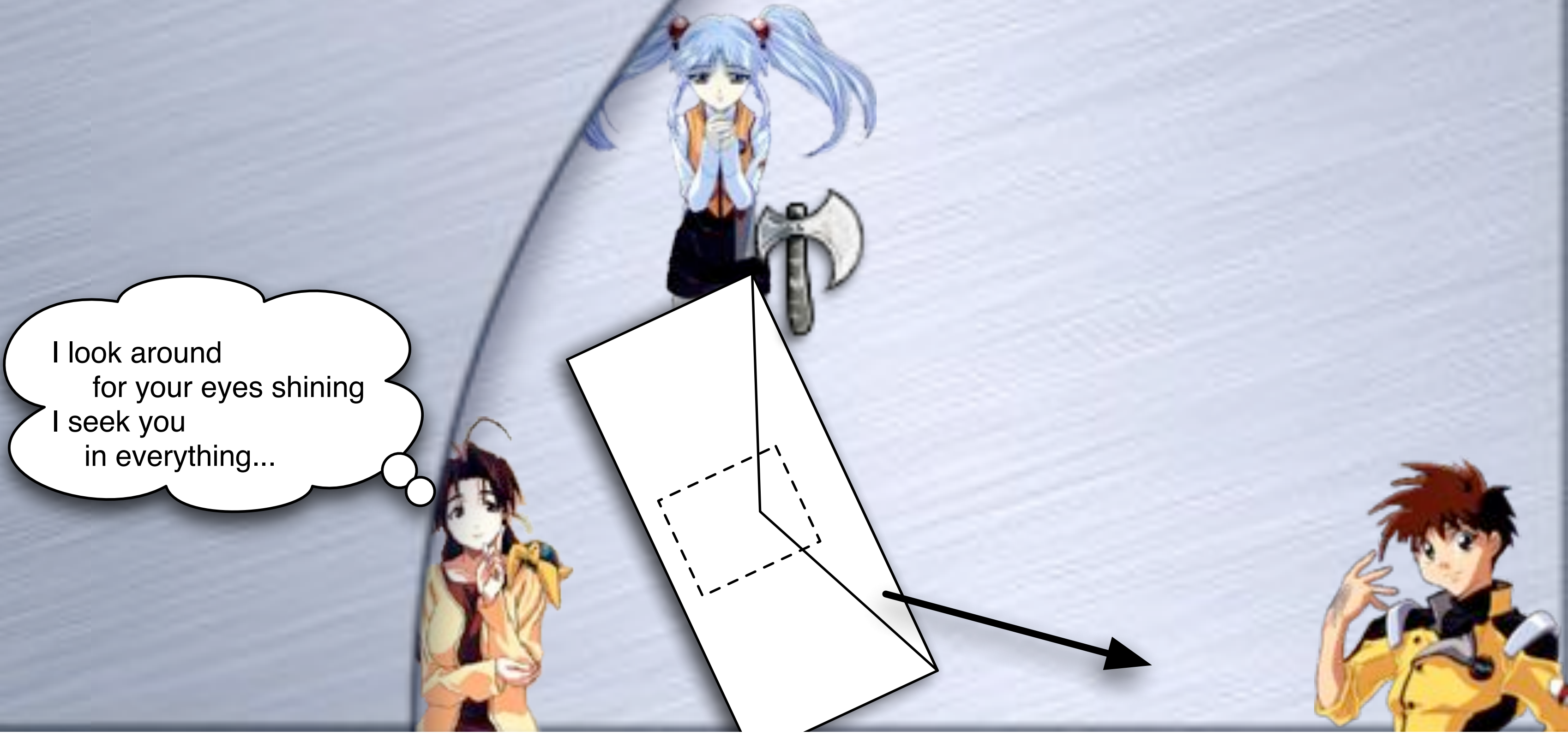


Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure

A subtle
e-mail attack

Alice → Bob: Enc(m)

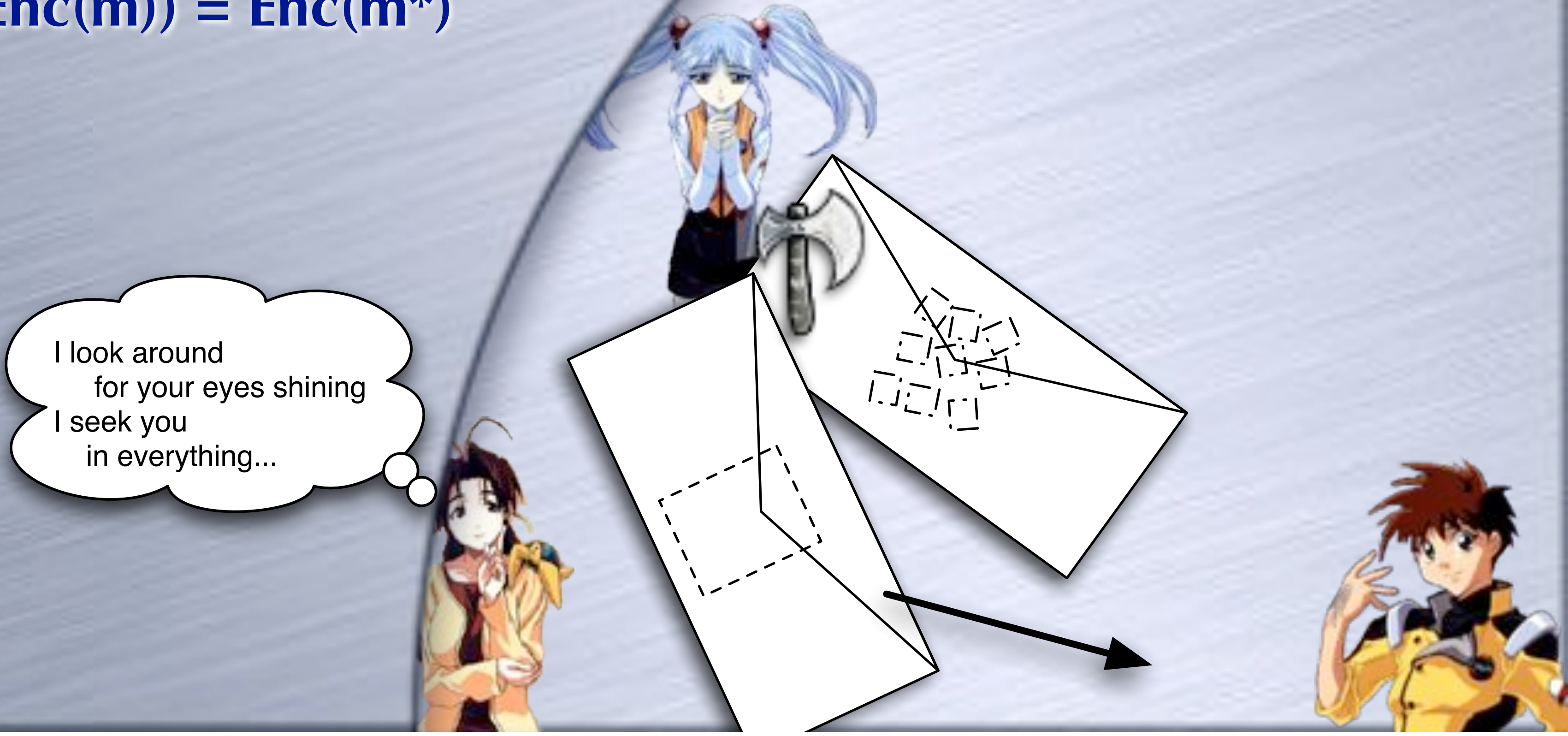


Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure

A subtle
e-mail attack

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

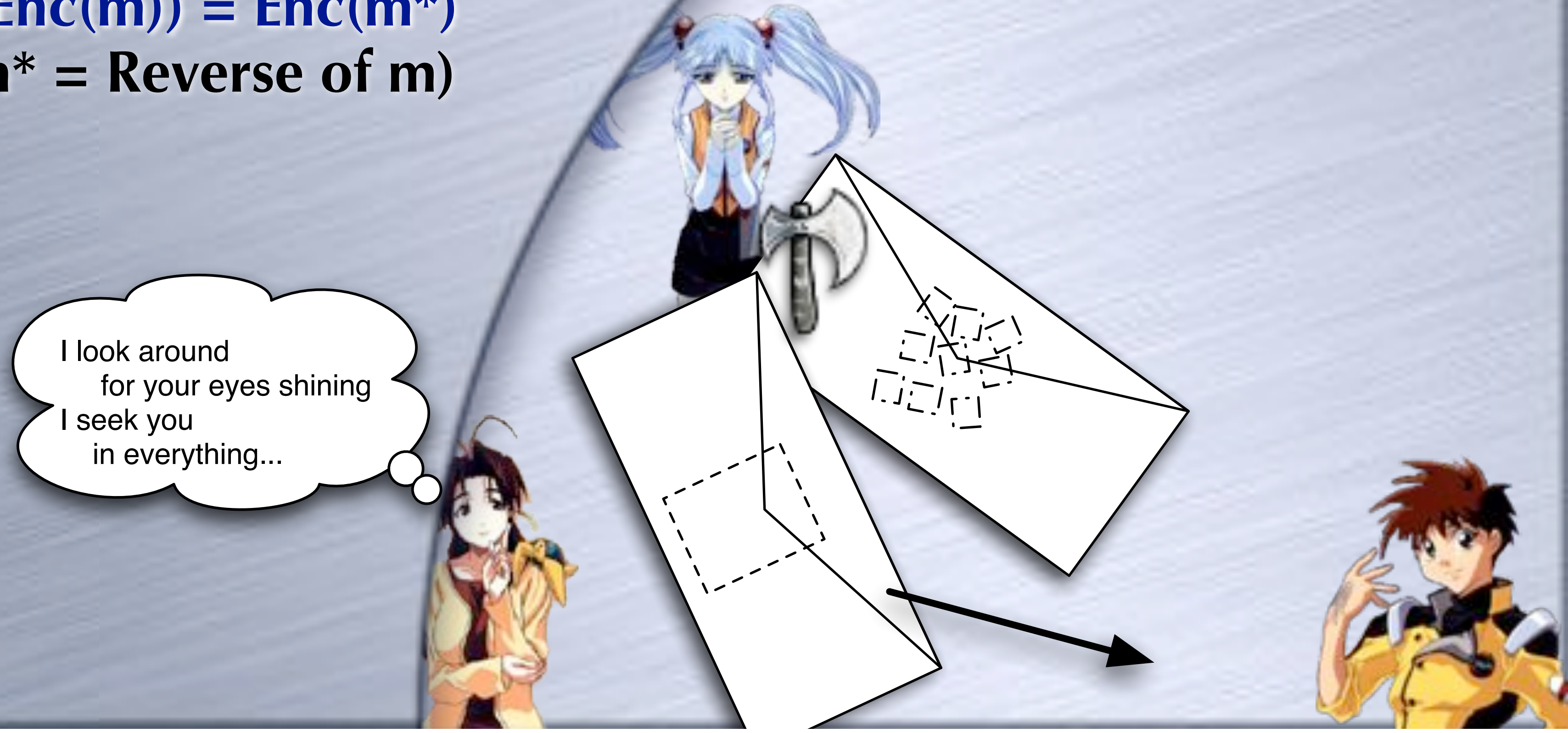


Chosen Ciphertext Attack

- 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)

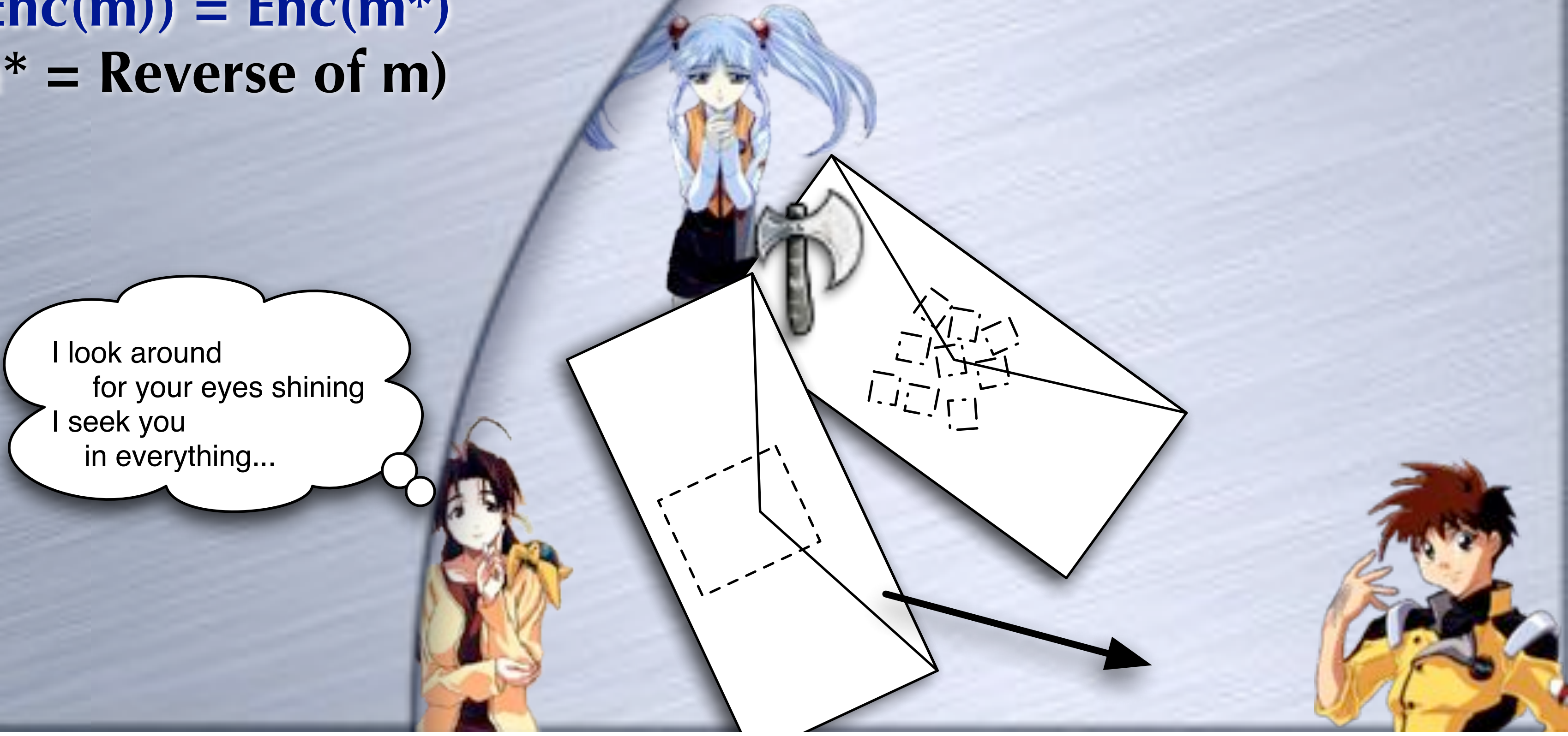


Chosen Ciphertext Attack

- 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)

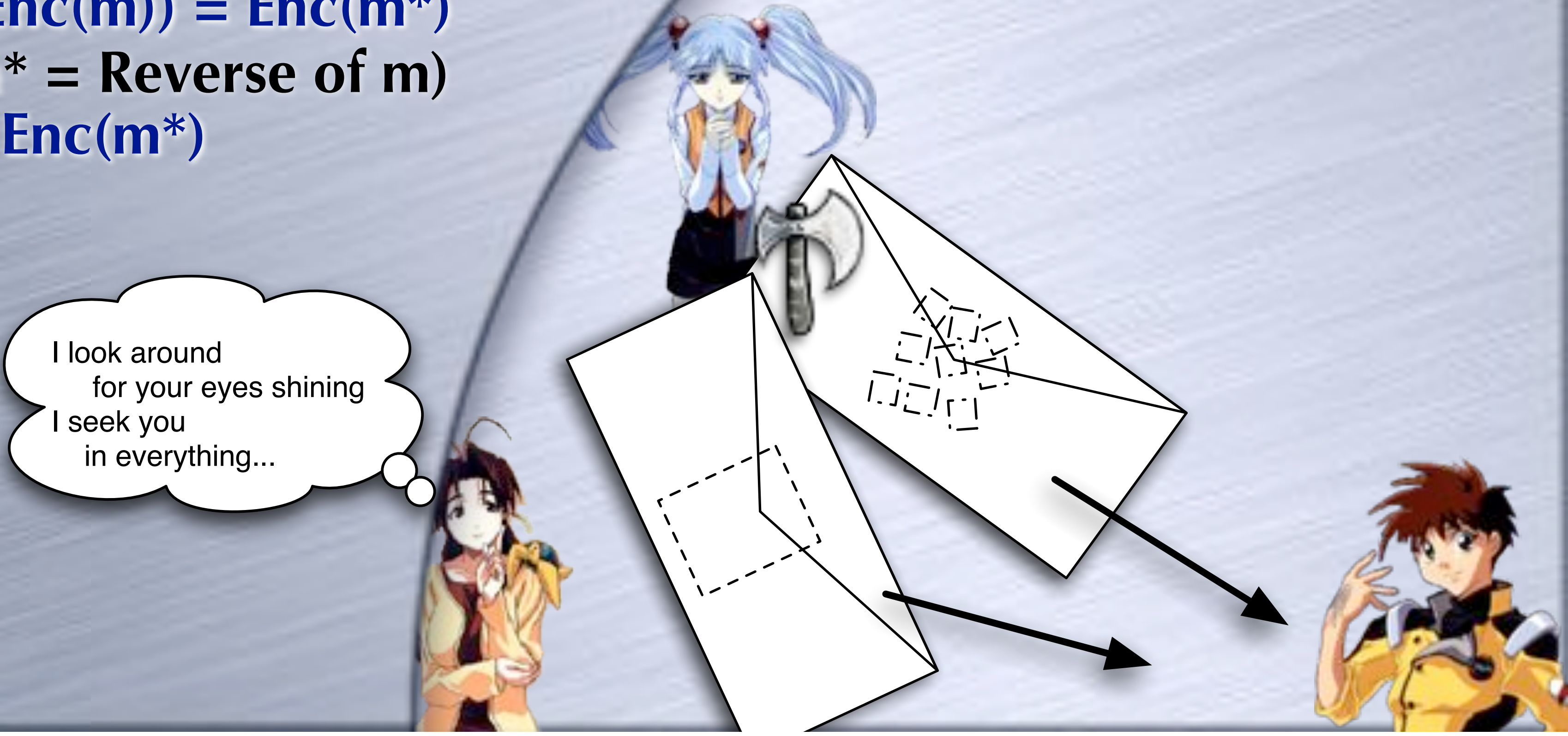


Chosen Ciphertext Attack

- 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*)

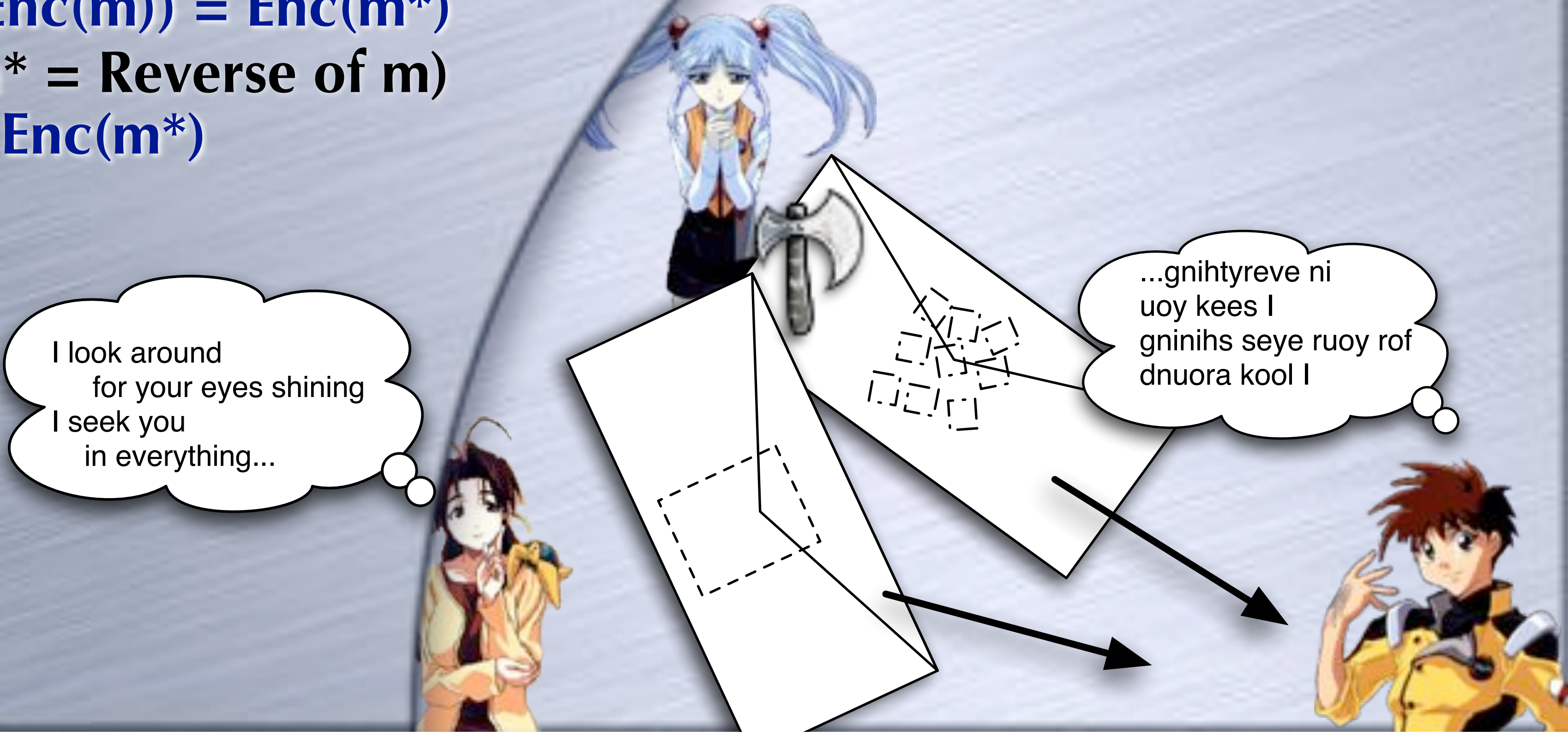


Chosen Ciphertext Attack

- 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*)

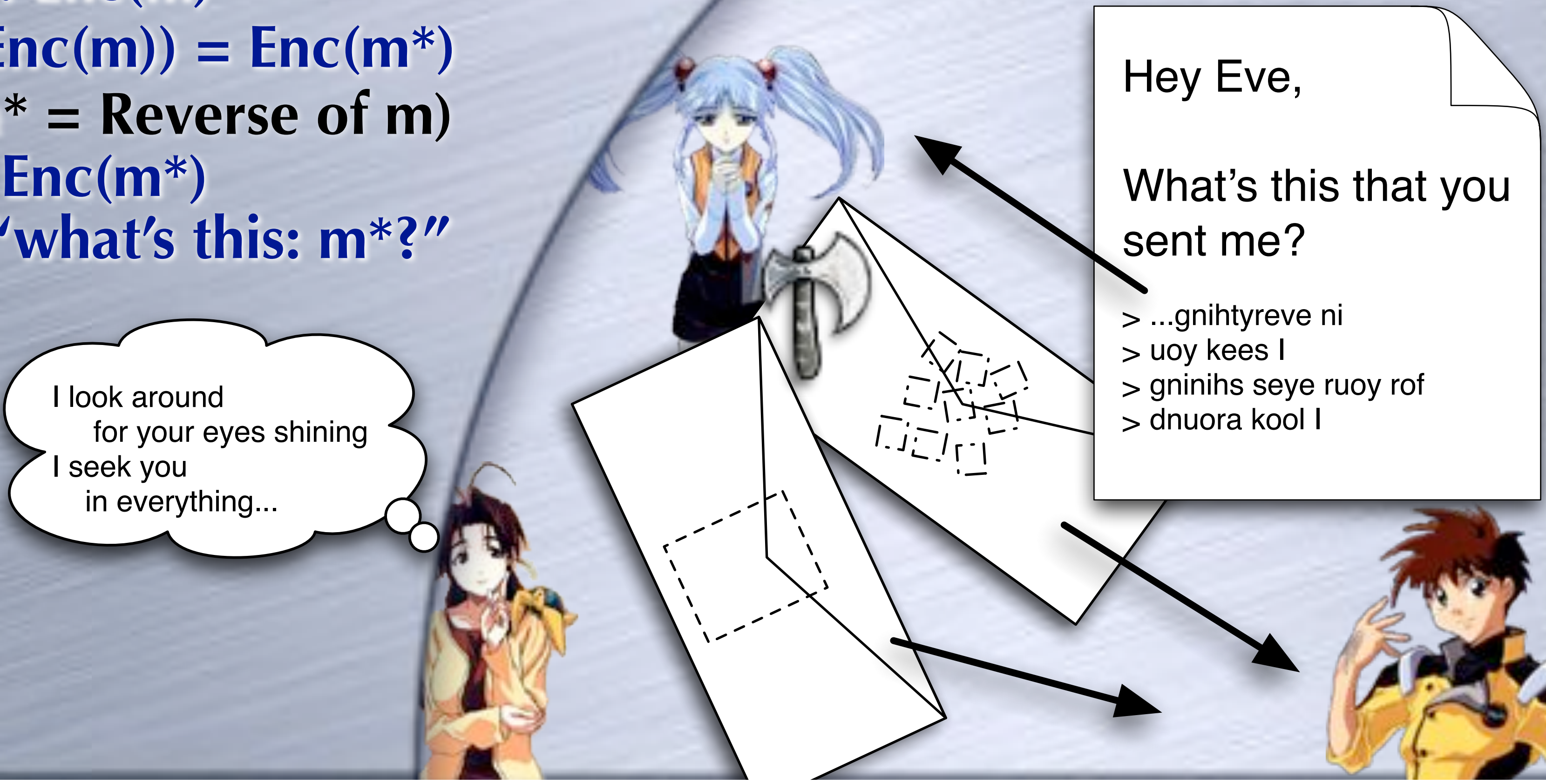


Chosen Ciphertext Attack

- 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*?"

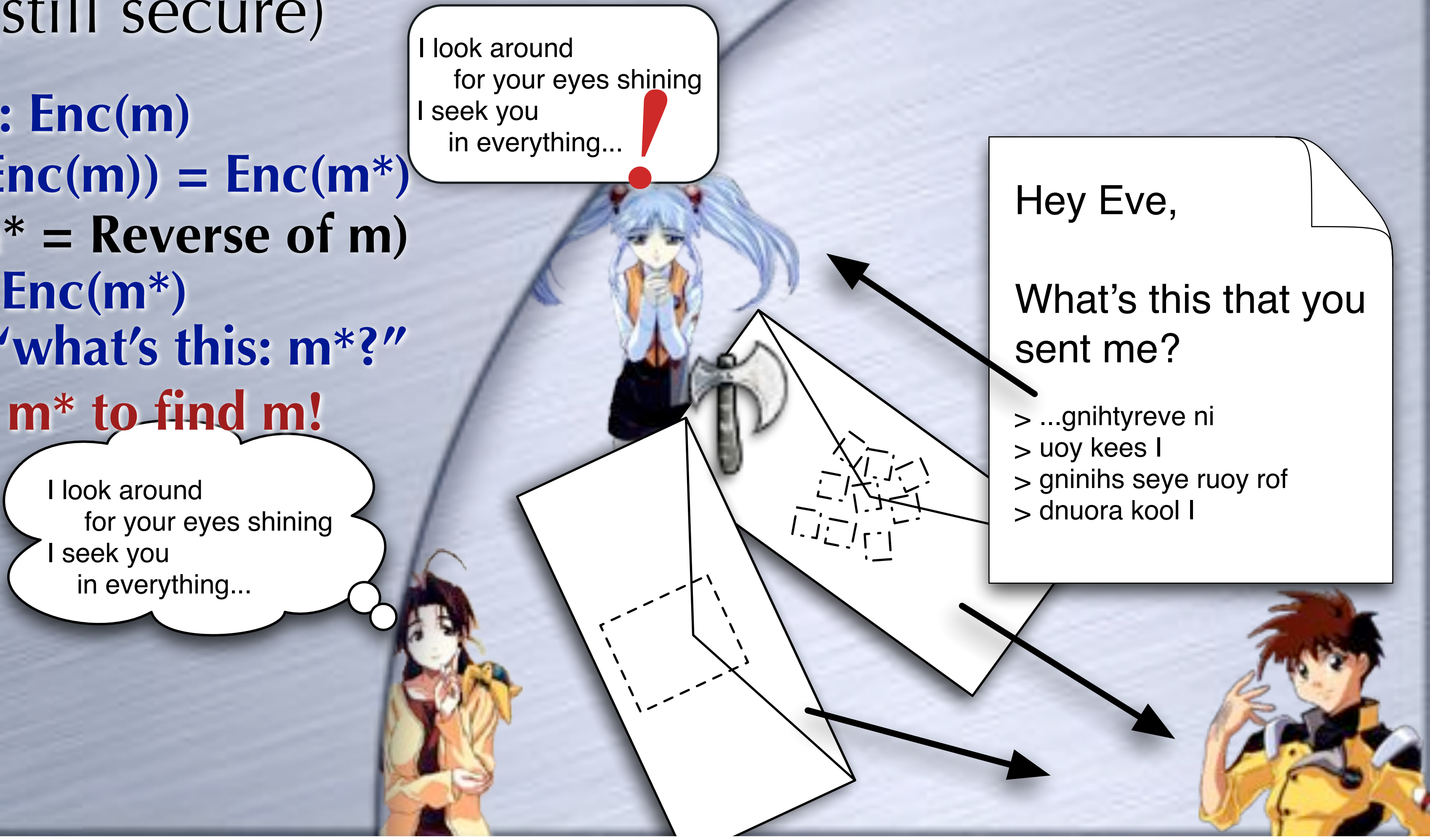


Chosen Ciphertext Attack

- 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!



Hey Eve,

What's this that you sent me?

> ...gnihtyreve ni
> uoy kees I
> gninihs seye ruoy rof
> dnuora kool I

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 CCA possible

Malleability

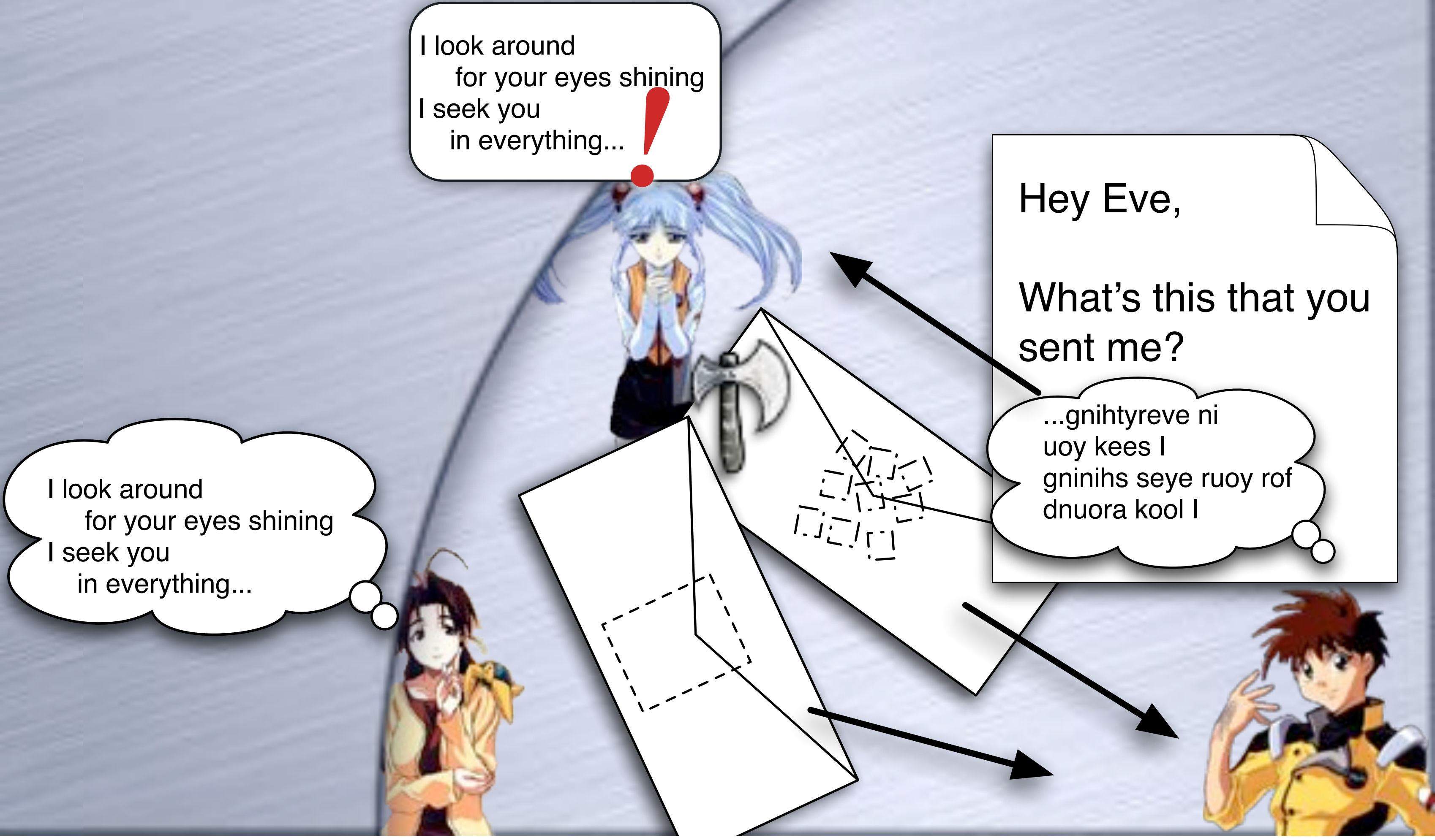
- Malleability: Eve can "malleate" a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a "related" message
- E.g.: Malleability of El Gamal
  - Recall: $\mathrm{Enc}_{(G,g,Y)}(m) = (g^x, M.Y^x)$
  - Given (X,C) change it to (X,TC): will decrypt to TM
  - Or change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

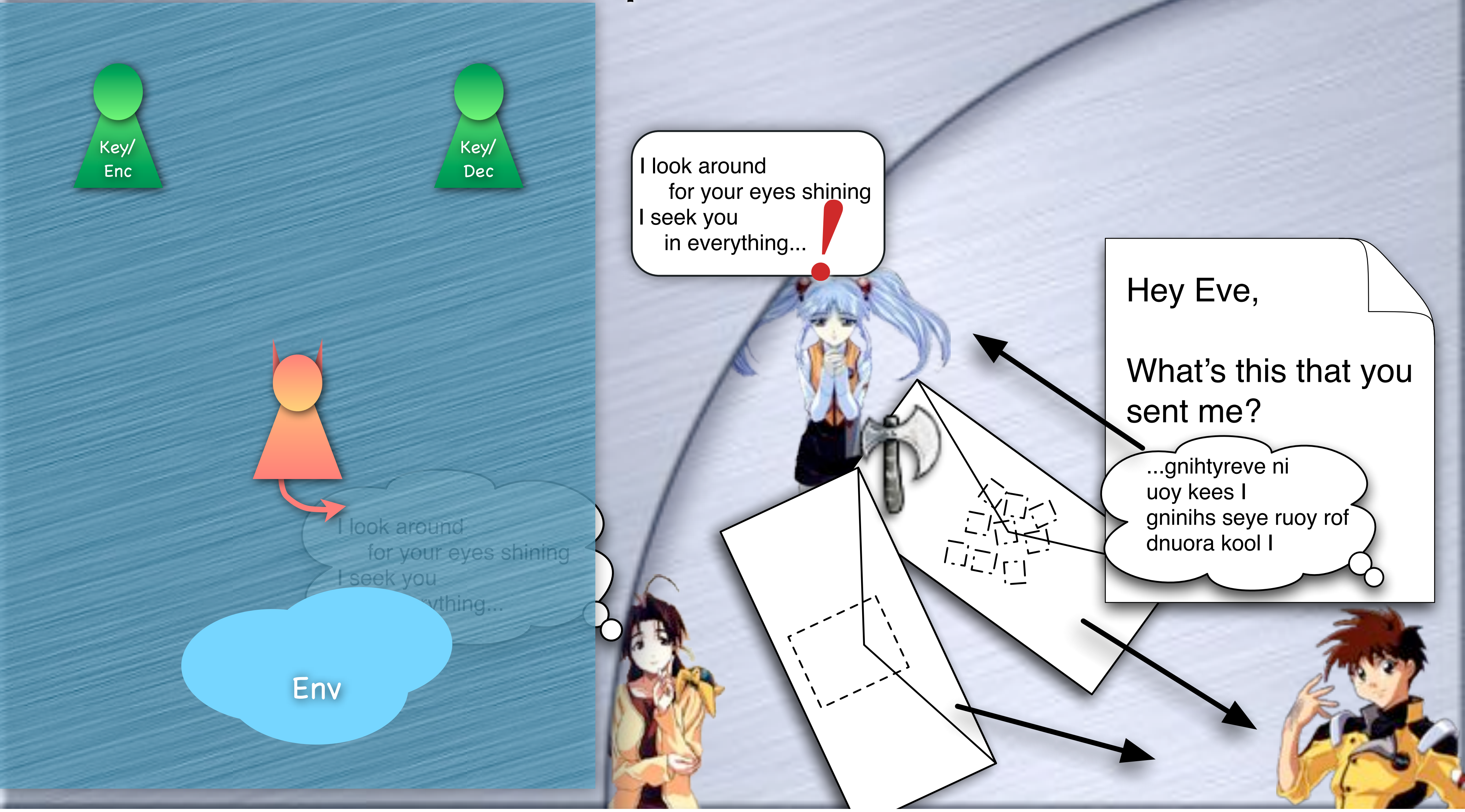
Chosen Ciphertext Attack

- SIM-CCA: does capture this attack



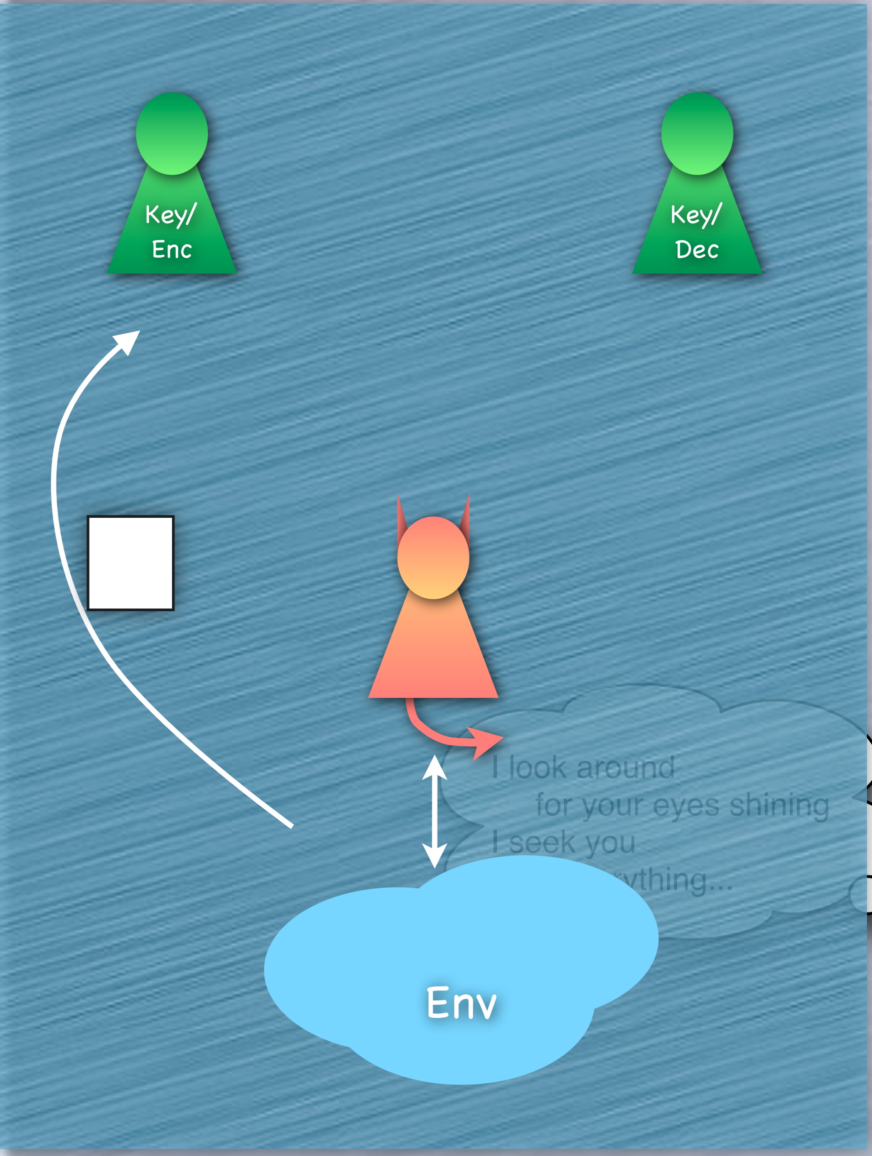
Chosen Ciphertext Attack

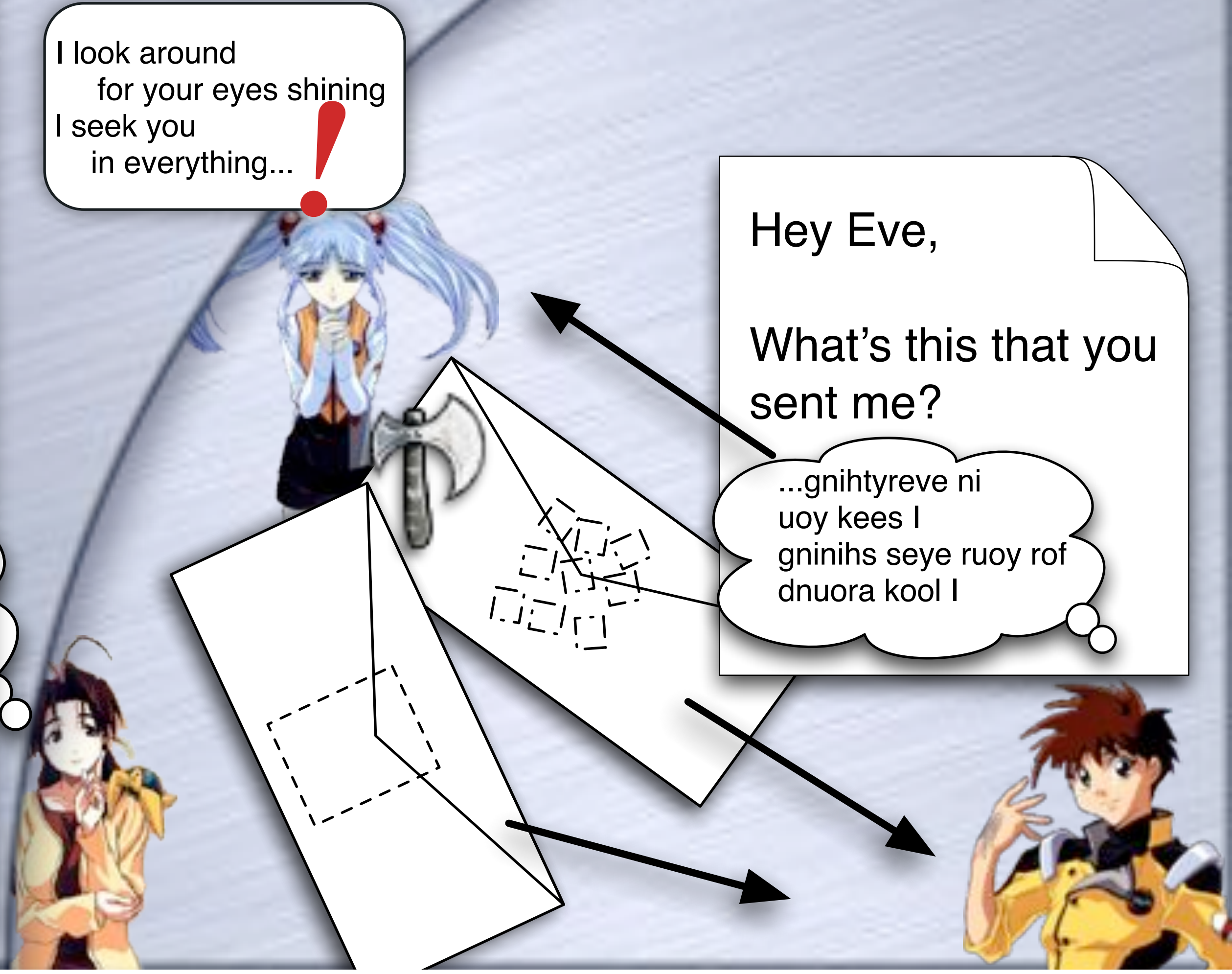
- SIM-CCA: does capture this attack



Chosen Ciphertext Attack

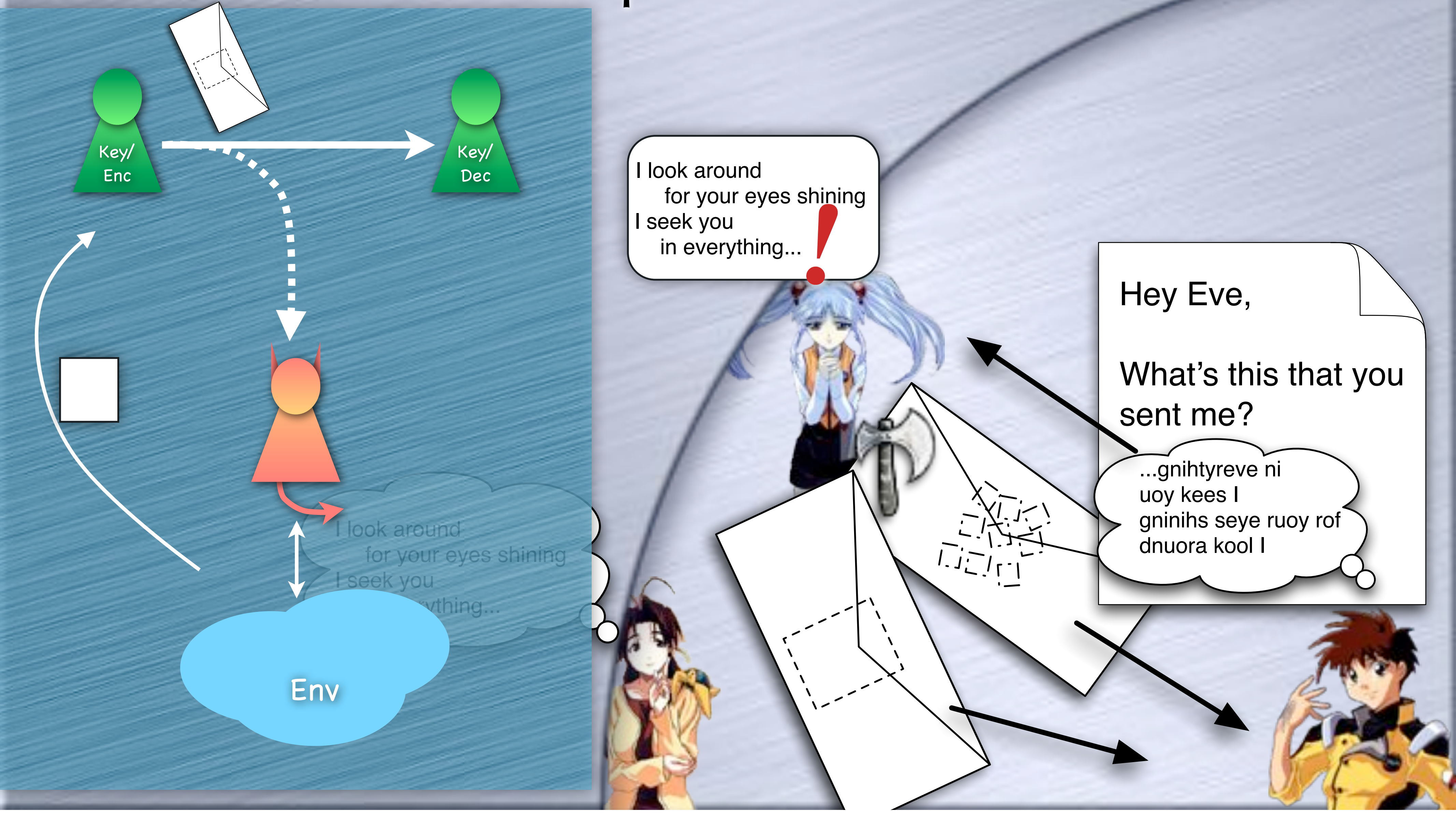
- SIM-CCA: does capture this attack





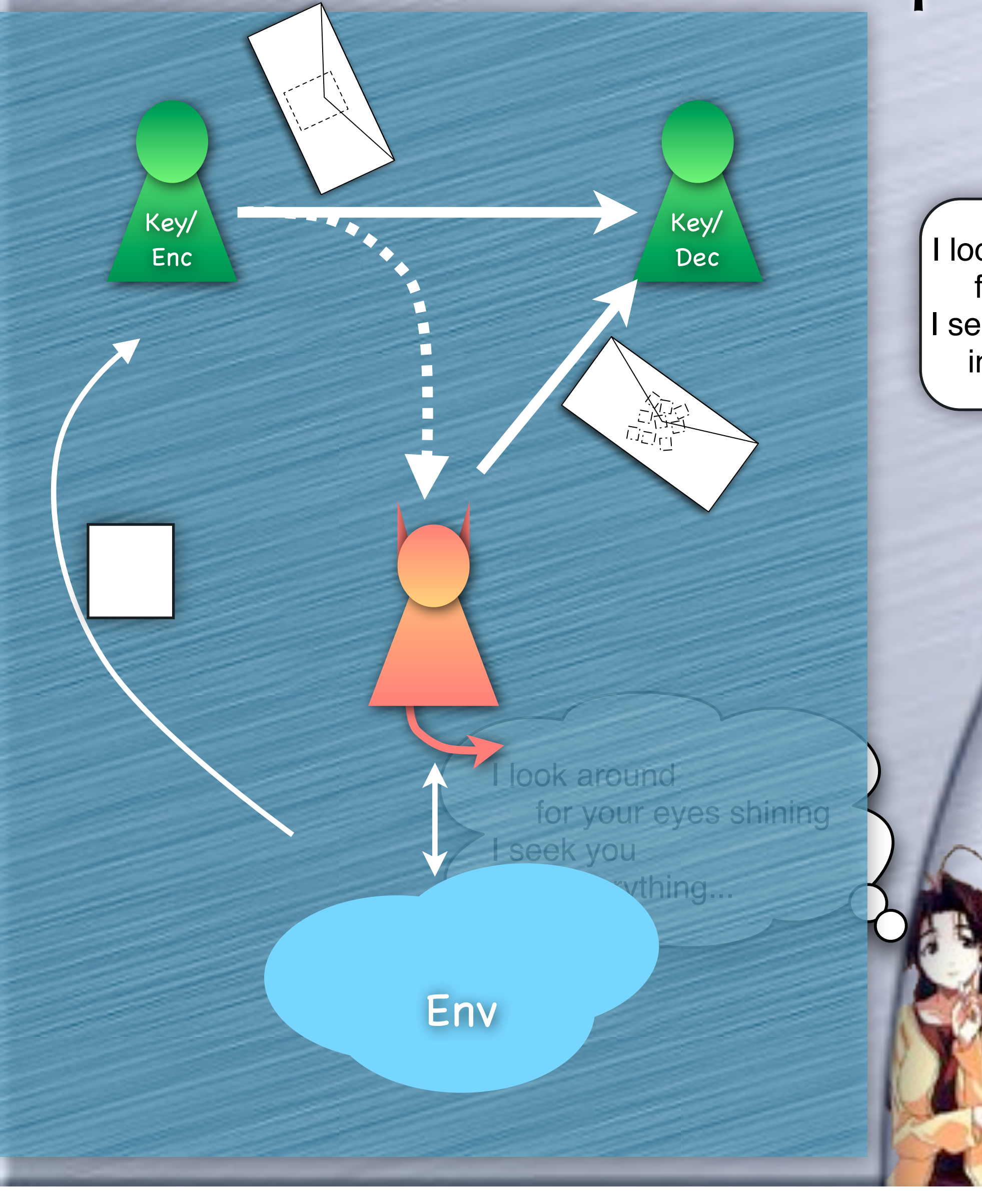
Chosen Ciphertext Attack

- SIM-CCA: does capture this attack

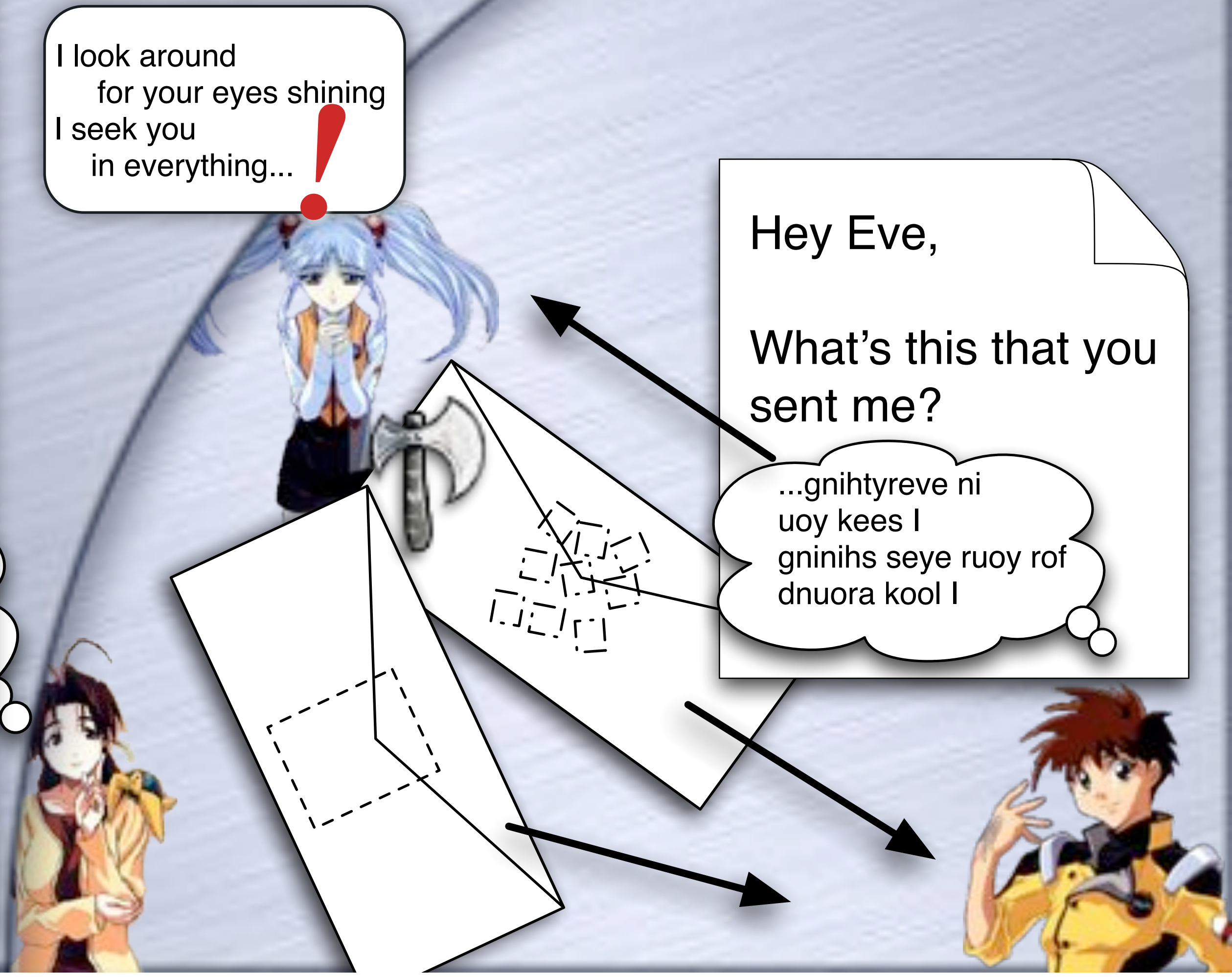

Key/ Enc
Key/ Dec
Env
I look around for your eyes shining I seek you in everything...
I look around for your eyes shining I seek you in everything...
Hey Eve,
What's this that you sent me?
...gnihtyreve ni uoy kees I gninihs seye ruoy rof dnuora kool I

Chosen Ciphertext Attack

- SIM-CCA: does capture this attack



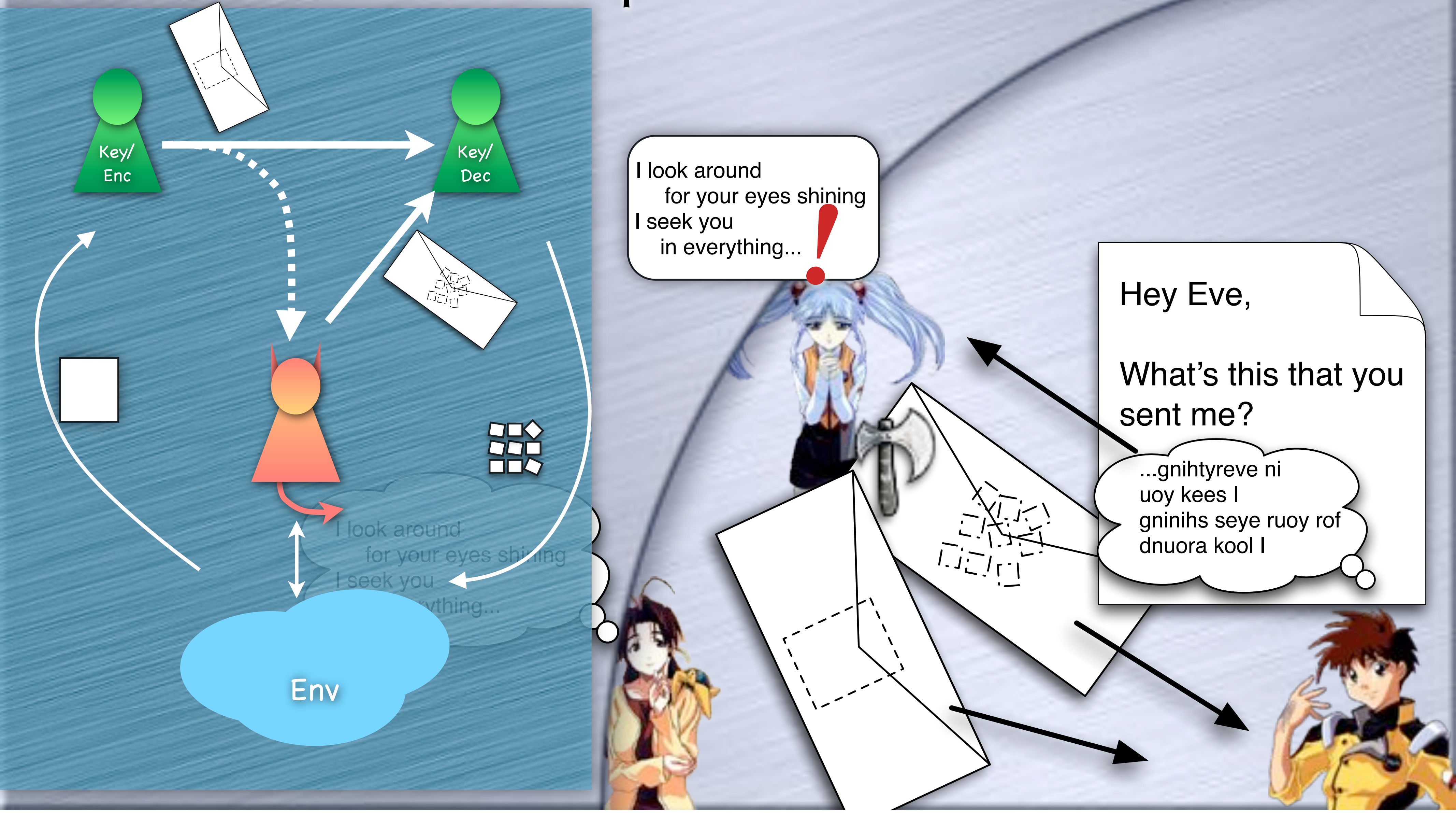
Key/ Enc
Key/ Dec
Env
I look around for your eyes shining I seek you in everything...



I look around for your eyes shining I seek you in everything...
Hey Eve,
What's this that you sent me?
...gnihtyreve ni uoy kees I gninihs seye ruoy rof dnuora kool I

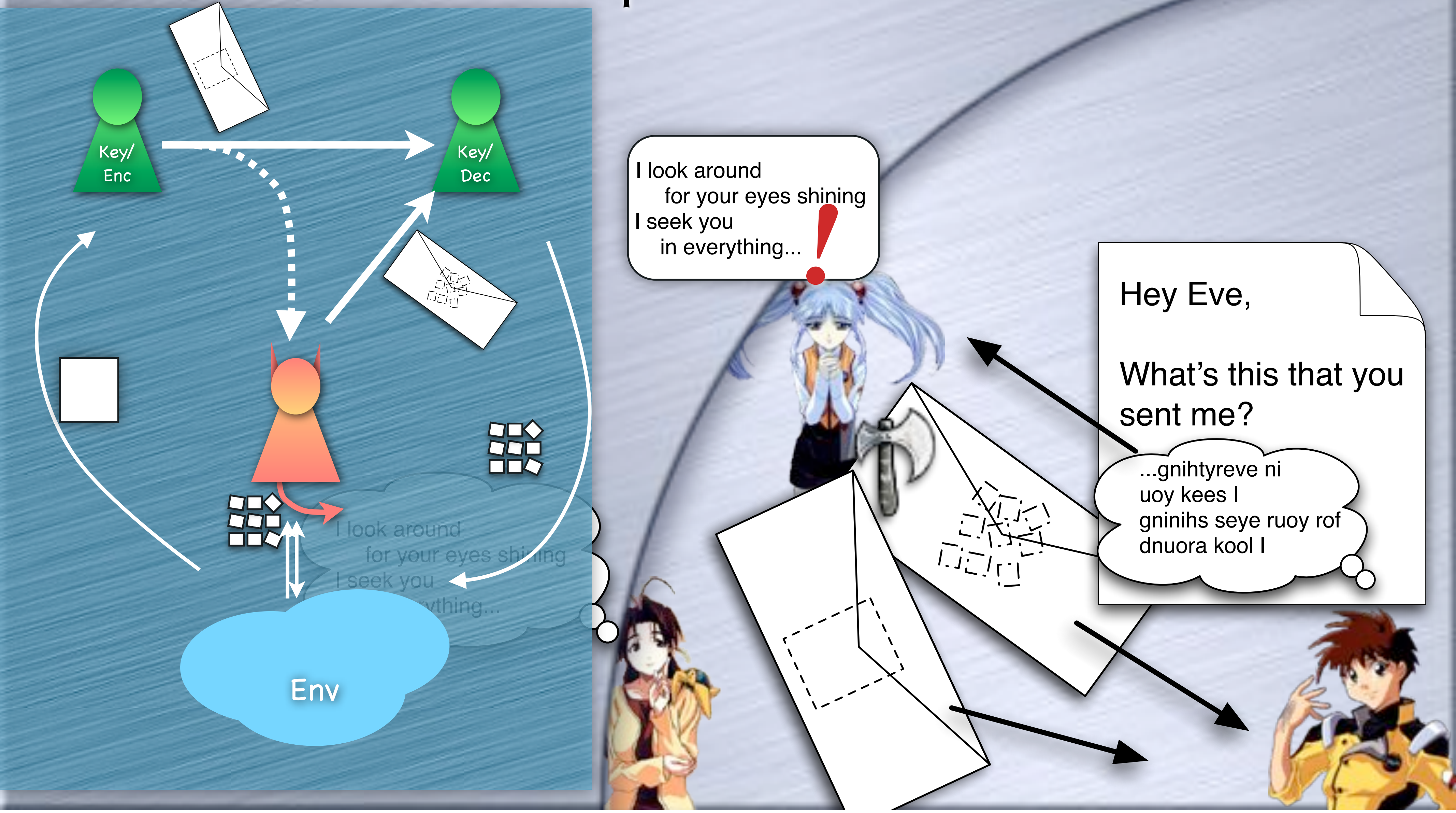
Chosen Ciphertext Attack

- SIM-CCA: does capture this attack


Key/
Enc
Key/
Dec
Env
I look around for your eyes shining I seek you in everything...
I look around for your eyes shining I seek you in everything...
!
Hey Eve,
What's this that you sent me?
...gnihtyreve ni
uoy kees I
gninihs seye ruoy rof
dnuora kool I

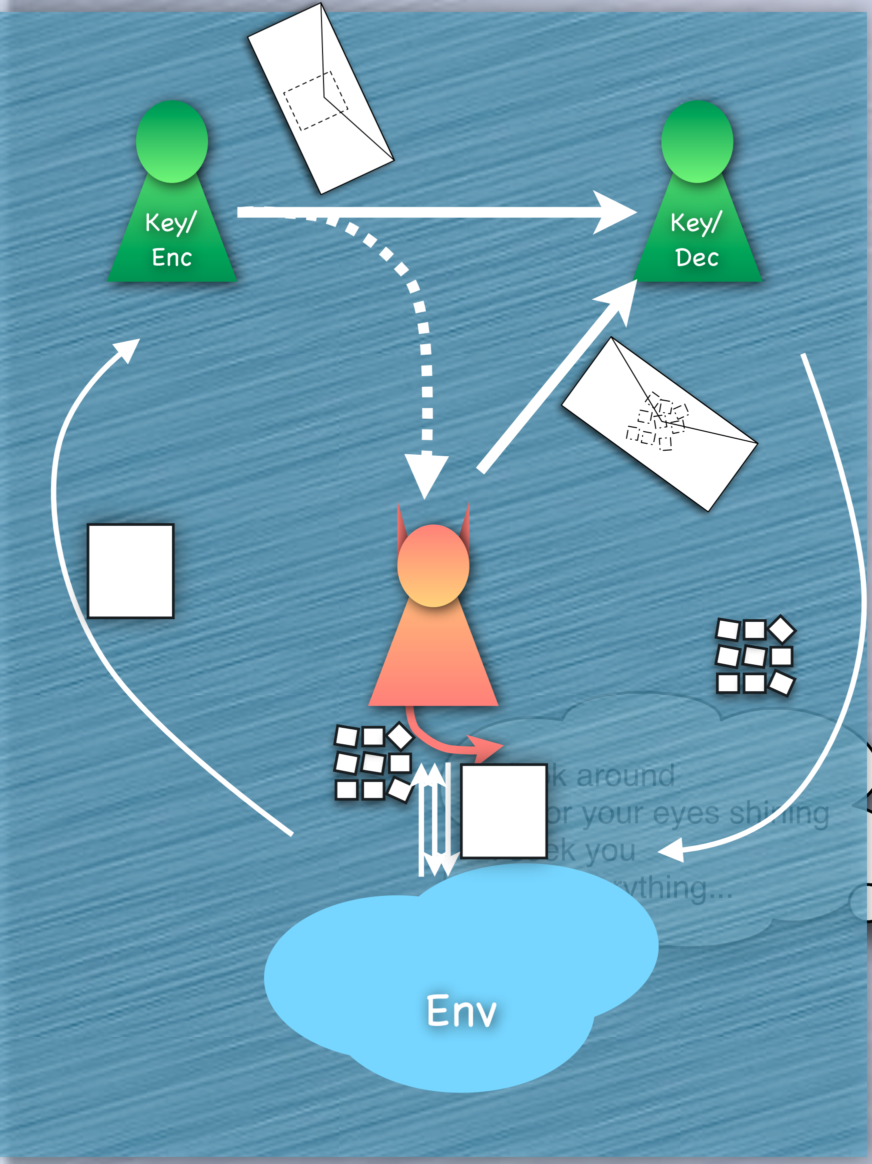
Chosen Ciphertext Attack

- SIM-CCA: does capture this attack


Key/ Enc
Key/ Dec
Env
I look around for your eyes shining I seek you in everything...
I look around for your eyes shining I seek you in everything...
Hey Eve,
What's this that you sent me?
...gnihtyreve ni uoy kees I gninihs seye ruoy rof dnuora kool I

Chosen Ciphertext Attack

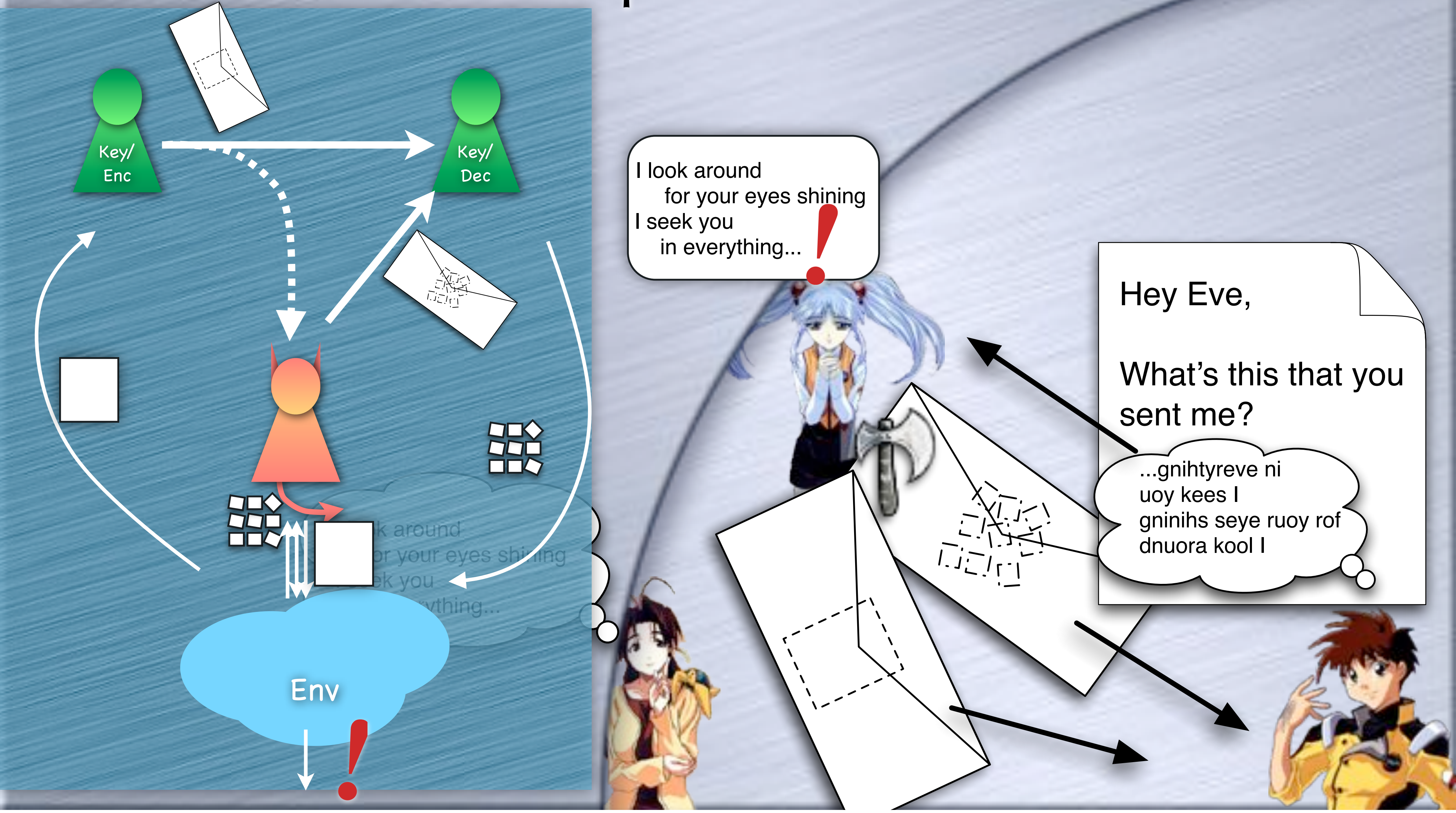
- SIM-CCA: does capture this attack



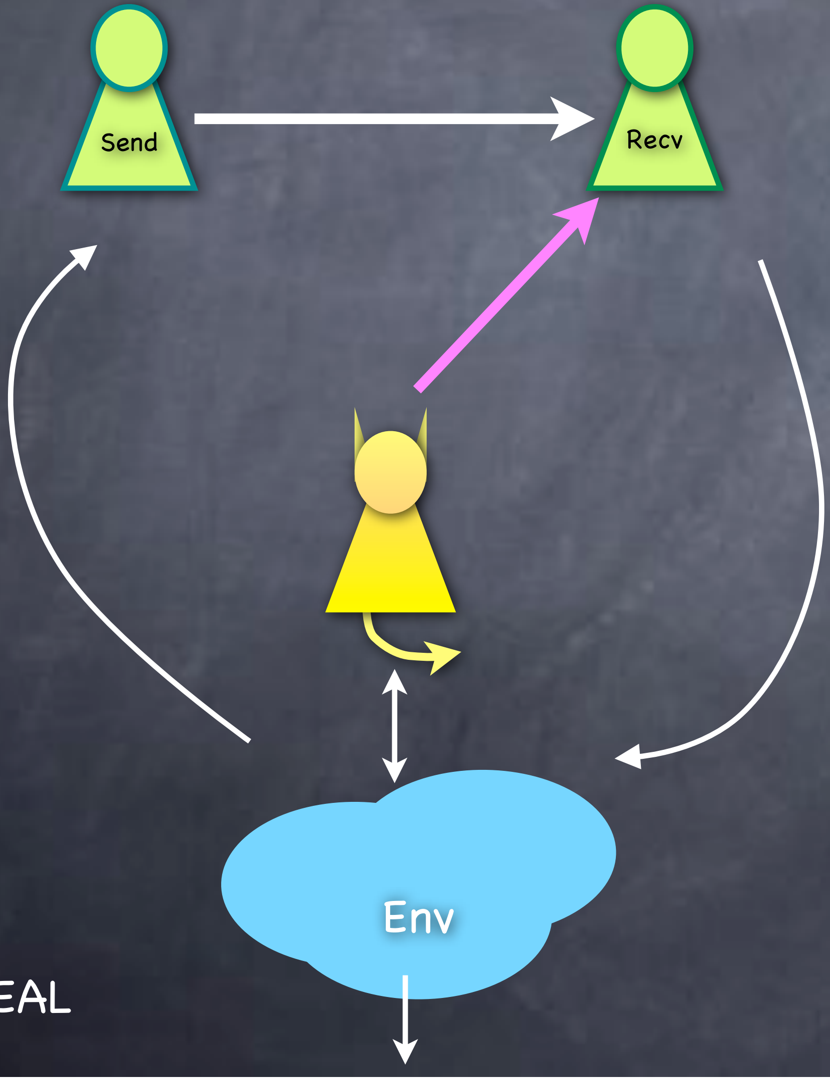
I look around for your eyes shining I seek you in everything...
!
Hey Eve,
What's this that you sent me?
...gnihtyreve ni uoy kees I gninihs seye ruoy rof dnuora kool I

Chosen Ciphertext Attack

- SIM-CCA: does capture this attack


Key/ Enc
Key/ Dec
Env
!
I look around for your eyes shining I seek you in everything...
!
Hey Eve, What's this that you sent me?
...gnihtyreve ni uoy kees I gninihs seye ruoy rof dnuora kool I

SIM-CCA Security (PKE)



Secure (and correct) if:

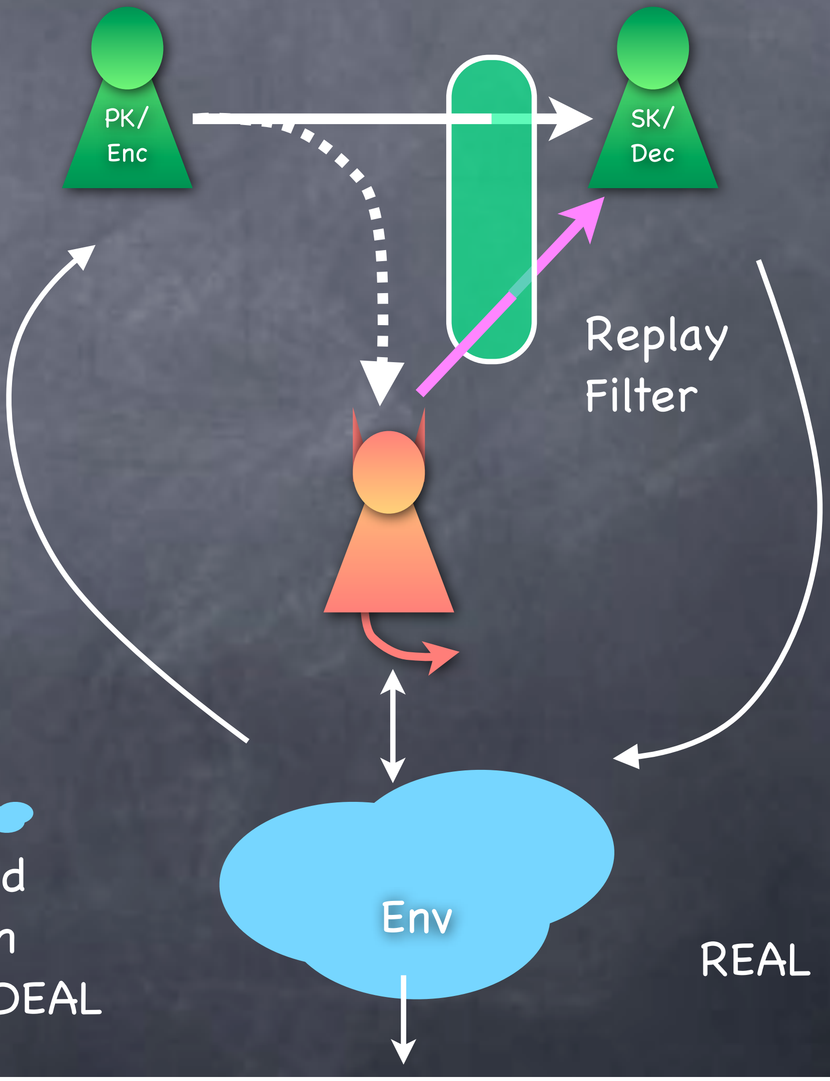
∀

∃ s.t.

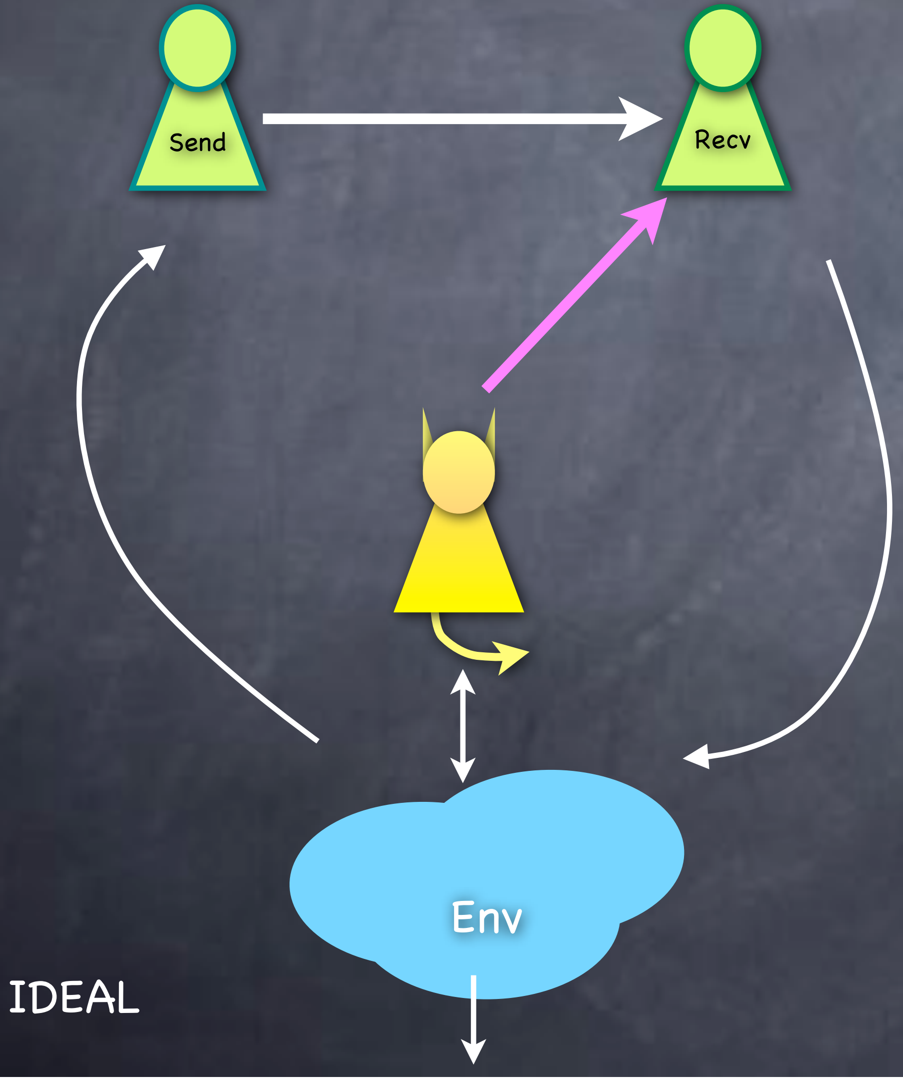
∀

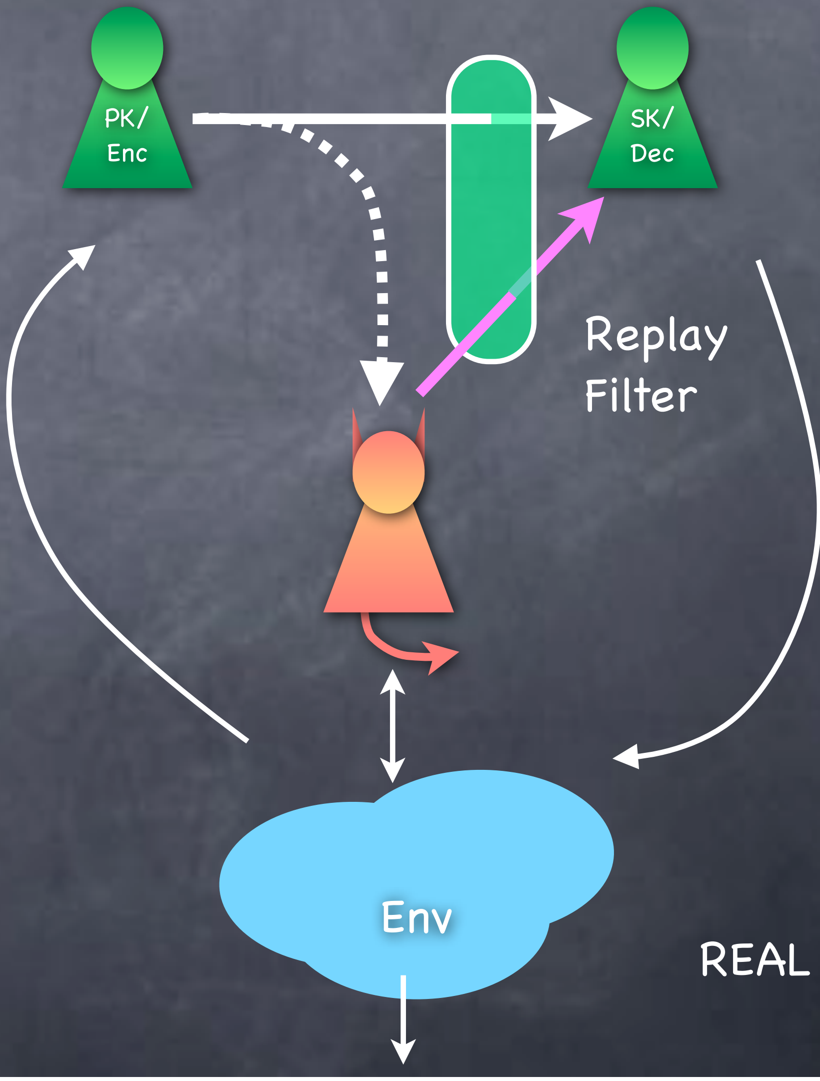
output of

is distributed identically in REAL and IDEAL

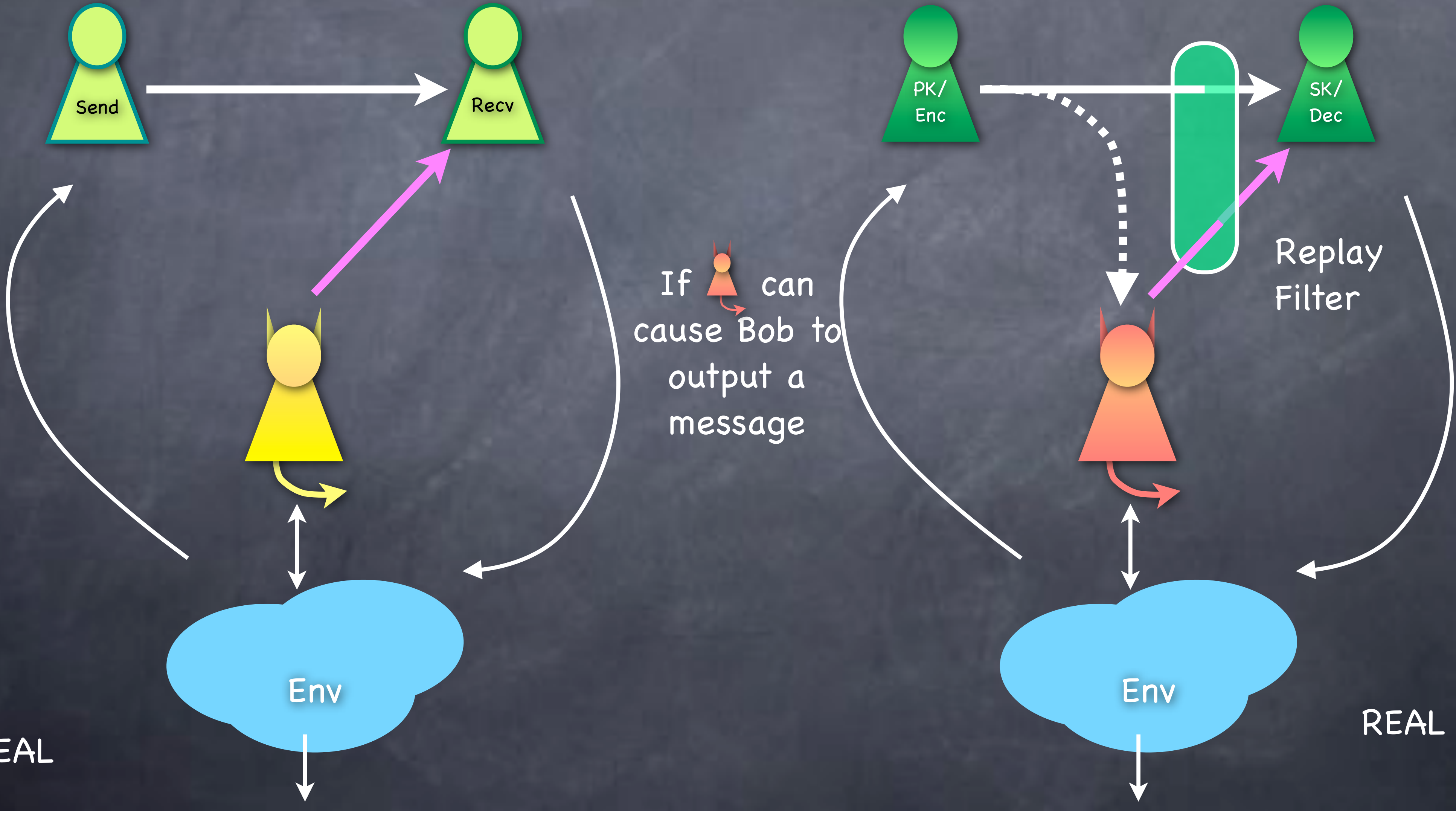


SIM-CCA Security and Malleability

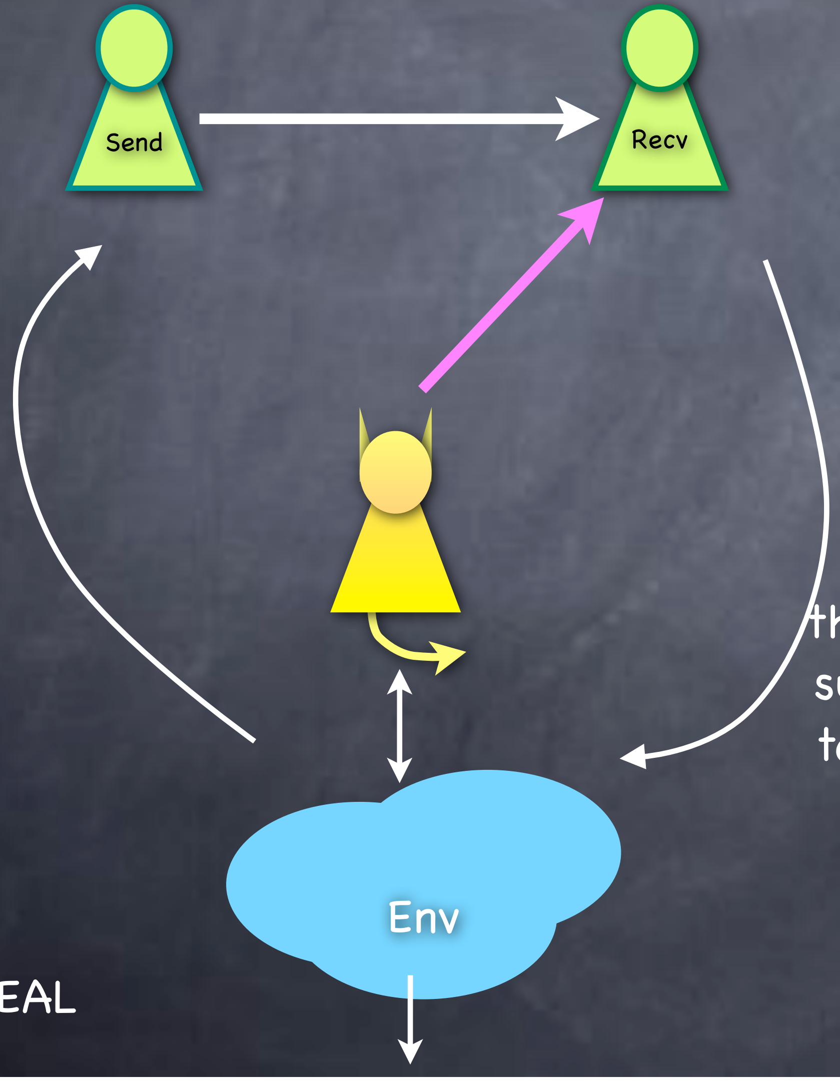




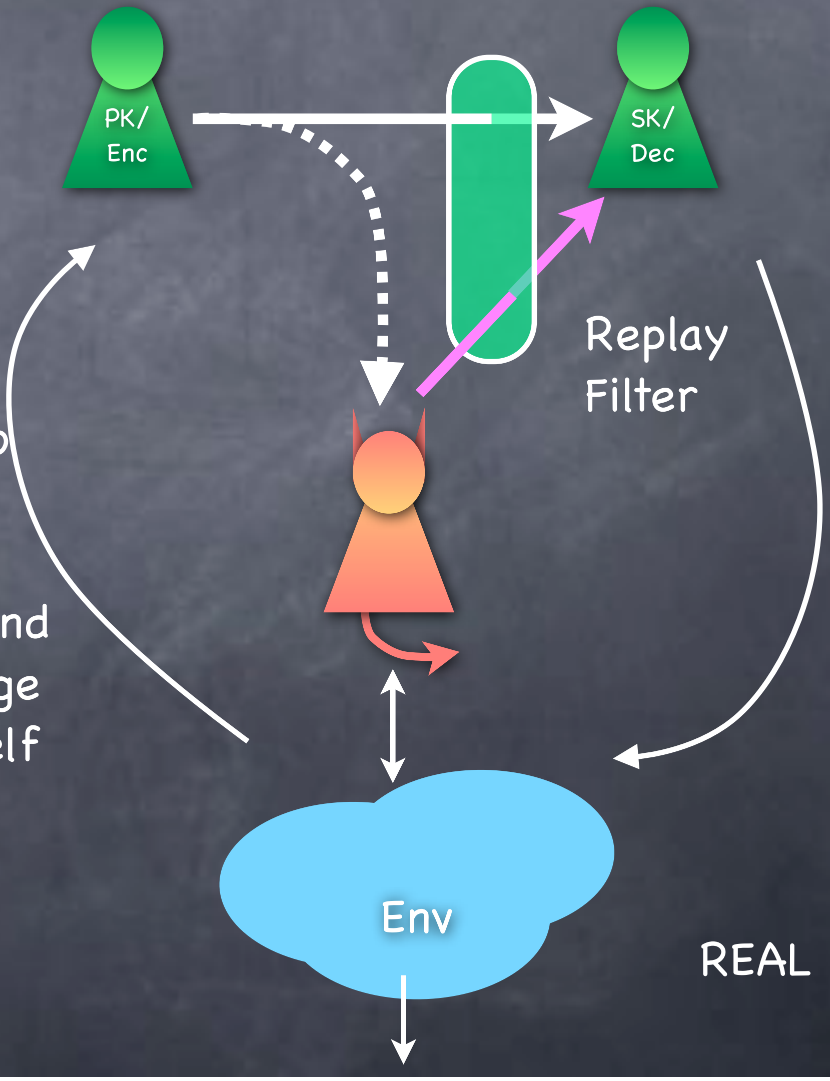
SIM-CCA Security and Malleability



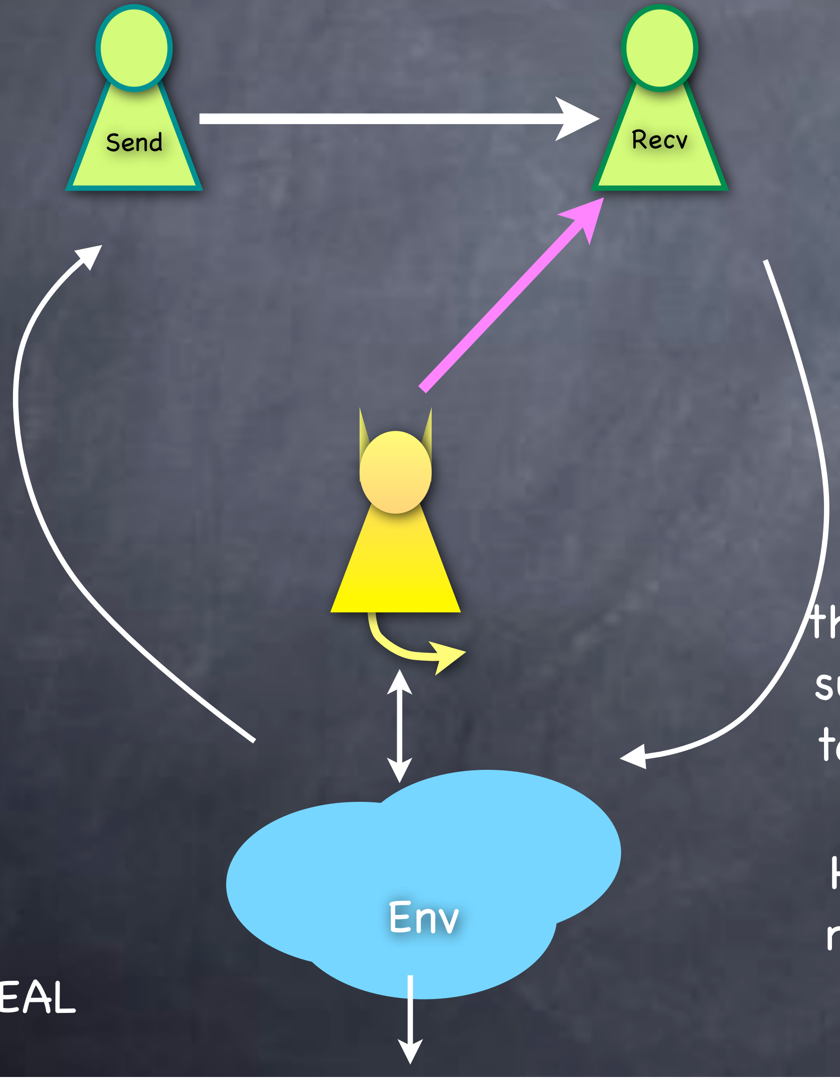
SIM-CCA Security and Malleability



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

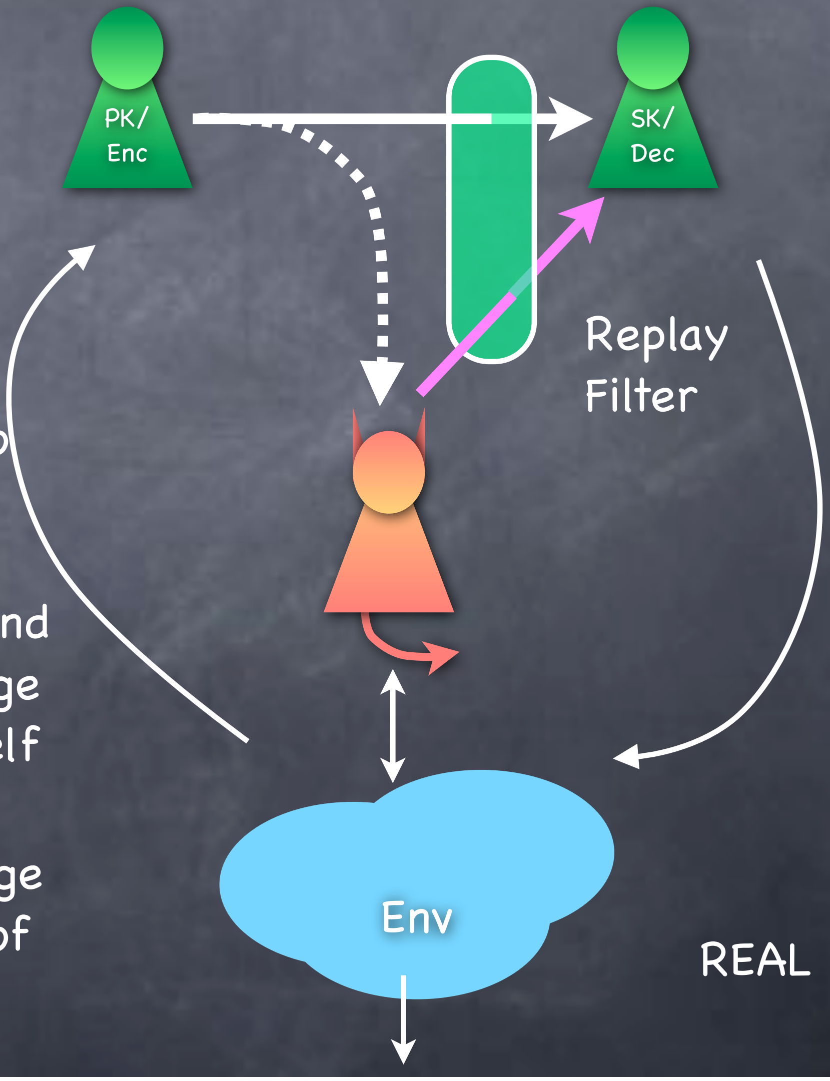


SIM-CCA Security and Malleability



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"

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 collision-resistant hash function inside an "integrity tag"

CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- 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 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 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 part of PK

CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- 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 part of PK
      - $Y = g_1^{y_1} g_2^{y_2}$, $W = g_1^{w_1} g_2^{w_2}$, $Z = g_1^{z_1} g_2^{z_2}$. 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 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 part of PK
      - $Y = g_1^{y_1} g_2^{y_2}$, $W = g_1^{w_1} g_2^{w_2}$, $Z = g_1^{z_1} g_2^{z_2}$. 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 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 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 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 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 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 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

$(g_1, g_1^{x_1}, g_2, g_2^{x_2})$ 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

$(g_1,g_1^{x_1},g_2, g_2^{x_2})$ 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

$(g_1, g_1^{x_1}, g_2, g_2^{x_2})$ 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

$(g_1,g_1^{x_1},g_2,g_2^{x_2})$ 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

$(g_1,g_1^{x_1},g_2, g_2^{x_2})$ 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 <u>no information</u> about the message (given just PK)
  - Is <u>indistinguishable</u> 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 <u>information</u> 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

$(g_1, g_1^{x_1}, g_2, g_2^{x_2})$ 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)

Another CCA Secure PKE: RSA-OAEP

Another CCA Secure PKE: RSA-OAEP

- RSA-OAEP

Another CCA Secure PKE: RSA-OAEP

- RSA-OAEP
  - "Text-book RSA encryption" (i.e., f_{RSA}, the T-OWP candidate) applied to an "encoding" of the message

Another CCA Secure PKE: RSA-OAEP

- RSA-OAEP
  - "Text-book RSA encryption" (i.e., f_{RSA}, the T-OWP candidate) applied to an "encoding" of the message
    - Encoding is randomized

Another CCA Secure PKE: RSA-OAEP

- RSA-OAEP
  - "Text-book RSA encryption" (i.e., f_{RSA}, the T-OWP candidate) applied to an "encoding" of the message
    - Encoding is randomized
    - Encoding uses a hash function modeled as a "Random Oracle"

Another CCA Secure PKE: RSA-OAEP

- RSA-OAEP
  - "Text-book RSA encryption" (i.e., f_{RSA}, the T-OWP candidate) applied to an "encoding" of the message
    - Encoding is randomized
    - Encoding uses a hash function modeled as a "Random Oracle"
    - Security in the RO Model, assuming f_{RSA} a OWP

Another CCA Secure PKE: RSA-OAEP

- RSA-OAEP
  - "Text-book RSA encryption" (i.e., f_{RSA}, the T-OWP candidate) applied to an "encoding" of the message
    - Encoding is randomized
    - Encoding uses a hash function modeled as a "Random Oracle"
    - Security in the RO Model, assuming f_{RSA} a OWP
  - Part of RSA Cryptography Standard (PKCS#1 Ver 2.1). Commonly used in SSL/TLS implementations

Random Oracle Model

Random Oracle Model

- Random Oracle: a mythical oracle that, when initialized, picks a random function $R:\{0,1\}^* \rightarrow \{0,1\}^{n(k)}$ and when queried with x, returns R(x)

Random Oracle Model

- Random Oracle: a mythical oracle that, when initialized, picks a random function $R:\{0,1\}^* \rightarrow \{0,1\}^{n(k)}$ and when queried with x, returns R(x)
    - All parties have access to the same RO

Random Oracle Model

- Random Oracle: a mythical oracle that, when initialized, picks a random function $R:\{0,1\}^* \to \{0,1\}^{n(k)}$ and when queried with x, returns R(x)
  - All parties have access to the same RO
- In ROM, evaluating some "hash function" H would be modeled as accessing an RO

Random Oracle Model

- Random Oracle: a mythical oracle that, when initialized, picks a random function $R:\{0,1\}^* \rightarrow \{0,1\}^{n(k)}$ and when queried with x, returns R(x)
  - All parties have access to the same RO
- In ROM, evaluating some "hash function" H would be modeled as accessing an RO
  - Especially when H has "no simple structure"

Random Oracle Model

- Random Oracle: a mythical oracle that, when initialized, picks a random function $R:\{0,1\}^* \rightarrow \{0,1\}^{n(k)}$ and when queried with x, returns R(x)
  - All parties have access to the same RO
- In ROM, evaluating some "hash function" H would be modeled as accessing an RO
  - Especially when H has "no simple structure"
- Sometimes security definitions need to be adapted for ROM

Random Oracle Model

- Random Oracle: a mythical oracle that, when initialized, picks a random function $R:\{0,1\}^* \rightarrow \{0,1\}^{n(k)}$ and when queried with x, returns R(x)
  - All parties have access to the same RO
- In ROM, evaluating some "hash function" H would be modeled as accessing an RO
  - Especially when H has "no simple structure"
- Sometimes security definitions need to be adapted for ROM
  - Regular proofs of security, once in the ROM

Random Oracle Model

Random Oracle Model

- There is no Pseudo-RO

Random Oracle Model

- **There is no Pseudo-RO**
  - Unlike PRF, RO must be locally evaluable for all parties. (think: giving out the seed of a PRF)

Random Oracle Model

- **There is no Pseudo-RO**
    - Unlike PRF, RO must be locally evaluable for all parties. (think: giving out the seed of a PRF)
- There are schemes secure in ROM, such that for any instantiation of the RO, the scheme is insecure!

Random Oracle Model

- **There is no Pseudo-RO**
  - Unlike PRF, RO must be locally evaluable for all parties. (think: giving out the seed of a PRF)
- There are schemes secure in ROM, such that for any instantiation of the RO, the scheme is insecure!
  - Also natural constructs/primitives which are realizable in ROM, but not in the standard model!

Random Oracle Model

- There is no Pseudo-RO
  - Unlike PRF, RO must be locally evaluable for all parties. (think: giving out the seed of a PRF)
- There are schemes secure in ROM, such that for any instantiation of the RO, the scheme is insecure!
  - Also natural constructs/primitives which are realizable in ROM, but not in the standard model!
- What does a proof in ROM tell us?

Random Oracle Model

- There is no Pseudo-RO
  - Unlike PRF, RO must be locally evaluable for all parties. (think: giving out the seed of a PRF)
- There are schemes secure in ROM, such that for any instantiation of the RO, the scheme is insecure!
  - Also natural constructs/primitives which are realizable in ROM, but not in the standard model!
- What does a proof in ROM tell us?
  - Secure against attacks that treat H as a blackbox (and for which H is pseudorandom)

Hybrid Encryption

Hybrid Encryption

- PKE is far less efficient compared to SKE (even in ROM)

Hybrid Encryption

- PKE is far less efficient compared to SKE (even in ROM)
    - SKE using Block Ciphers (e.g. AES) and MAC is very fast

Hybrid Encryption

- PKE is far less efficient compared to SKE (even in ROM)
  - SKE using Block Ciphers (e.g. AES) and MAC is very fast
  - RSA-OAEP uses exponentiations (Cramer-Shoup even more)

Hybrid Encryption

- PKE is far less efficient compared to SKE (even in ROM)
  - SKE using Block Ciphers (e.g. AES) and MAC is very fast
  - RSA-OAEP uses exponentiations (Cramer-Shoup 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 (even in ROM)
  - SKE using Block Ciphers (e.g. AES) and MAC is very fast
  - RSA-OAEP uses exponentiations (Cramer-Shoup 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 (even in ROM)
  - SKE using Block Ciphers (e.g. AES) and MAC is very fast
  - RSA-OAEP uses exponentiations (Cramer-Shoup 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 (even in ROM)
  - SKE using Block Ciphers (e.g. AES) and MAC is very fast
  - RSA-OAEP uses exponentiations (Cramer-Shoup 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

Or to generate a key

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

CCA Secure PKE: DHIES

CCA Secure PKE: DHIES

- Diffie-Hellman Integrated Encryption Scheme

CCA Secure PKE: DHIES

- Diffie-Hellman Integrated Encryption Scheme
    - Part of some standards

CCA Secure PKE: DHIES

- Diffie-Hellman Integrated Encryption Scheme
  - Part of some standards
- Essentially a hybrid scheme

CCA Secure PKE: DHIES

- Diffie-Hellman Integrated Encryption Scheme
  - Part of some standards
- Essentially a hybrid scheme
  - Data Encapsulation: CPA secure SKE and MAC

CCA Secure PKE: DHIES

- Diffie-Hellman Integrated Encryption Scheme
  - Part of some standards
- Essentially a hybrid scheme
  - Data Encapsulation: CPA secure SKE and MAC
  - Key Encapsulation: $X=g^x$. Let $K=Y^x$, where Y is the PK (as in El Gamal), and (K_{SKE},K_{MAC}) = Hash(K) (where $K=Y^x=X^y$)

CCA Secure PKE: DHIES

- Diffie-Hellman Integrated Encryption Scheme
  - Part of some standards
- Essentially a hybrid scheme
  - Data Encapsulation: CPA secure SKE and MAC
  - Key Encapsulation: $X=g^x$. Let $K=Y^x$, where Y is the PK (as in El Gamal), and (K_{SKE},K_{MAC}) = Hash(K) (where $K=Y^x=X^y$)
- CCA security based on a strong (non-standard) assumption involving Hash and the group: "Oracle Diffie-Hellman Assumption"

Another PKE Scheme: CCA Secure in RO Model

Another PKE Scheme: CCA Secure in RO Model

- Fujisaki-Okamoto Hybrid scheme

Another PKE Scheme: CCA Secure in RO Model

- Fujisaki-Okamoto Hybrid scheme
    - KEM encrypts x, using random coins derived as H(m,x) where m is the message and H a "random oracle"

Another PKE Scheme: CCA Secure in RO Model

- Fujisaki-Okamoto Hybrid scheme
  - KEM encrypts x, using random coins derived as H(m,x) where m is the message and H a "random oracle"
  - DEM encrypts with key K = G(x), where G is another "random oracle"

Another PKE Scheme: CCA Secure in RO Model

- Fujisaki-Okamoto Hybrid scheme
  - KEM encrypts x, using random coins derived as H(m,x) where m is the message and H a "random oracle"
  - DEM encrypts with key K = G(x), where G is another "random oracle"
  - Decryption decrypts x, then m, and then checks if KEM was correct

Another PKE Scheme: CCA Secure in RO Model

- Fujisaki-Okamoto Hybrid scheme
  - KEM encrypts x, using random coins derived as H(m,x) where m is the message and H a "random oracle"
  - DEM encrypts with key K = G(x), where G is another "random oracle"
  - Decryption decrypts x, then m, and then checks if KEM was correct
  - Very weak security sufficient for encryptions used in KEM and DEM (but only with H, G modeled as random oracles)

Identity-Based Encryption

Identity-Based Encryption

- In PKE, KeyGen produces a random (PK,SK) pair

Identity-Based Encryption

- In PKE, KeyGen produces a random (PK,SK) pair
- Can I have a “fancy public-key” (e.g., my name)?

Identity-Based Encryption

- In PKE, KeyGen produces a random (PK,SK) pair
- Can I have a "fancy public-key" (e.g., my name)?
  - But no one should be able to pick a PK and find an SK for it

Identity-Based Encryption

- In PKE, KeyGen produces a random (PK,SK) pair
- Can I have a "fancy public-key" (e.g., my name)?
  - But no one should be able to pick a PK and find an SK for it
- But suppose a trusted authority for key generation

Identity-Based Encryption

- In PKE, KeyGen produces a random (PK,SK) pair
- Can I have a “fancy public-key” (e.g., my name)?
  - But no one should be able to pick a PK and find an SK for it
- But suppose a trusted authority for key generation
  - Then: Can it generate a valid (PK,SK) pair for any PK?

Identity-Based Encryption

- In PKE, KeyGen produces a random (PK,SK) pair
- Can I have a "fancy public-key" (e.g., my name)?
  - But no one should be able to pick a PK and find an SK for it
- But suppose a trusted authority for key generation
  - Then: Can it generate a valid (PK,SK) pair for any PK?
  - Identity-Based Encryption: a key-server (with a master secret-key) that can generate such pairs

Identity-Based Encryption

- In PKE, KeyGen produces a random (PK,SK) pair
- Can I have a "fancy public-key" (e.g., my name)?
  - But no one should be able to pick a PK and find an SK for it
- But suppose a trusted authority for key generation
  - Then: Can it generate a valid (PK,SK) pair for any PK?
  - Identity-Based Encryption: a key-server (with a master secret-key) that can generate such pairs
    - Encryption will use the master public-key, and the receiver's "identity" (i.e., fancy public-key)

Identity-Based Encryption

- In PKE, KeyGen produces a random (PK,SK) pair
- Can I have a "fancy public-key" (e.g., my name)?
  - But no one should be able to pick a PK and find an SK for it
- But suppose a trusted authority for key generation
  - Then: Can it generate a valid (PK,SK) pair for any PK?
  - Identity-Based Encryption: a key-server (with a master secret-key) that can generate such pairs
    - Encryption will use the master public-key, and the receiver's "identity" (i.e., fancy public-key)
    - In PKE, sender has to retrieve PK for every party it wants to talk to (from a trusted public directory)

Identity-Based Encryption

- In PKE, KeyGen produces a random (PK,SK) pair
- Can I have a "fancy public-key" (e.g., my name)?
  - But no one should be able to pick a PK and find an SK for it
- But suppose a trusted authority for key generation
  - Then: Can it generate a valid (PK,SK) pair for any PK?
  - Identity-Based Encryption: a key-server (with a master secret-key) that can generate such pairs
    - Encryption will use the master public-key, and the receiver's "identity" (i.e., fancy public-key)
    - In PKE, sender has to retrieve PK for every party it wants to talk to (from a trusted public directory)
    - In IBE, receiver has to obtain its SK from the authority

Identity-Based Encryption

Identity-Based Encryption

- Security requirement for IBE (will skip formal statement):

Identity-Based Encryption

- Security requirement for IBE (will skip formal statement):
  - Environment/adversary decides the ID of the honest parties

Identity-Based Encryption

- Security requirement for IBE (will skip formal statement):
  - Environment/adversary decides the ID of the honest parties
  - Adversary can adaptively request SK for any number of IDs (which are not used for honest parties)

Identity-Based Encryption

- Security requirement for IBE (will skip formal statement):
  - Environment/adversary decides the ID of the honest parties
  - Adversary can adaptively request SK for any number of IDs (which are not used for honest parties)
  - "Semantic security" for encryption with the ID of honest parties (i.e., with no access to decryption: CPA security)

Identity-Based Encryption

- Security requirement for IBE (will skip formal statement):
  - Environment/adversary decides the ID of the honest parties
  - Adversary can adaptively request SK for any number of IDs (which are not used for honest parties)
  - "Semantic security" for encryption with the ID of honest parties (i.e., with no access to decryption: CPA security)
- IBE (even CPA-secure) can easily give CCA-secure PKE!

Identity-Based Encryption

- Security requirement for IBE (will skip formal statement):
  - Environment/adversary decides the ID of the honest parties
  - Adversary can adaptively request SK for any number of IDs (which are not used for honest parties)
  - "Semantic security" for encryption with the ID of honest parties (i.e., with no access to decryption: CPA security)
- IBE (even CPA-secure) can easily give CCA-secure PKE!
  - IBE: Can't malleate ciphertext for one ID into one for another

Identity-Based Encryption

- Security requirement for IBE (will skip formal statement):
  - Environment/adversary decides the ID of the honest parties
  - Adversary can adaptively request SK for any number of IDs (which are not used for honest parties)
  - "Semantic security" for encryption with the ID of honest parties (i.e., with no access to decryption: CPA security)
- IBE (even CPA-secure) can easily give CCA-secure PKE!
  - IBE: Can't malleate ciphertext for one ID into one for another
  - $\text{PKEnc}_{\text{MPK}}(m)$ = (verkey, $C=\text{IBEnc}_{\text{MPK}}(\text{id=verkey}; m)$, $\text{sign}_{\text{signkey}}(C)$)

Identity-Based Encryption

- Security requirement for IBE (will skip formal statement):
  - Environment/adversary decides the ID of the honest parties
  - Adversary can adaptively request SK for any number of IDs (which are not used for honest parties)
  - "Semantic security" for encryption with the ID of honest parties (i.e., with no access to decryption: CPA security)
- IBE (even CPA-secure) can easily give CCA-secure PKE!
  - IBE: Can't malleate ciphertext for one ID into one for another
  - $\text{PKEnc}_{\text{MPK}}(m) = (\text{verkey}, C{=}\text{IBEnc}_{\text{MPK}}(id{=}\text{verkey}; m), \text{sign}_{\text{signkey}}(C)\ )$

Digital Signature

Today

Today

- CCA secure PKE

Today

- CCA secure PKE
    - Cramer-Shoup

Today

- CCA secure PKE
  - Cramer-Shoup
  - Hybrid Encryption: KEM/DEM

Today

- CCA secure PKE
  - Cramer-Shoup
  - Hybrid Encryption: KEM/DEM
  - In Random Oracle Model: e.g. RSA-OAEP

Today

- CCA secure PKE
  - Cramer-Shoup
  - Hybrid Encryption: KEM/DEM
  - In Random Oracle Model: e.g. RSA-OAEP
  - Using Identity Based Encryption

Today

- CCA secure PKE
  - Cramer-Shoup
  - Hybrid Encryption: KEM/DEM
  - In Random Oracle Model: e.g. RSA-OAEP
  - Using Identity Based Encryption
- Next up: Digital Signatures