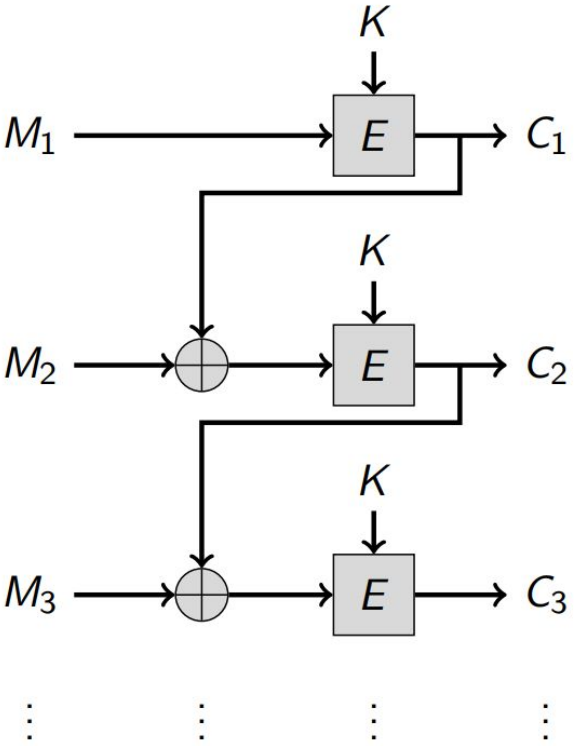
CS489/689
Privacy, Cryptography, Network and Data Security

Winter 2023, Tuesday/Thursday 8:30-9:50am

Assignment One

- Available on Learn
- Due **February 2nd 2023, 4pm**
- Written and programming
- Can currently do W1,2, and W3,4 after today
- Can do P1 now, P2 and P3, el gamal, Covered Jan 23

Clarification from Last Lecture



Q: Spot the difference?

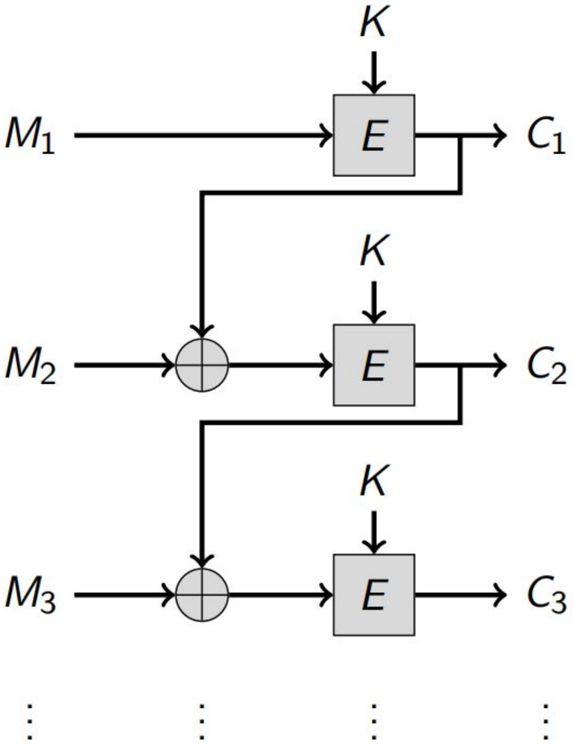
Q: Is it fixed this time?

Q: Does this avoid repeating patterns among blocks?

Q: What would happen if we encrypt the **message twice** with the **same key**?

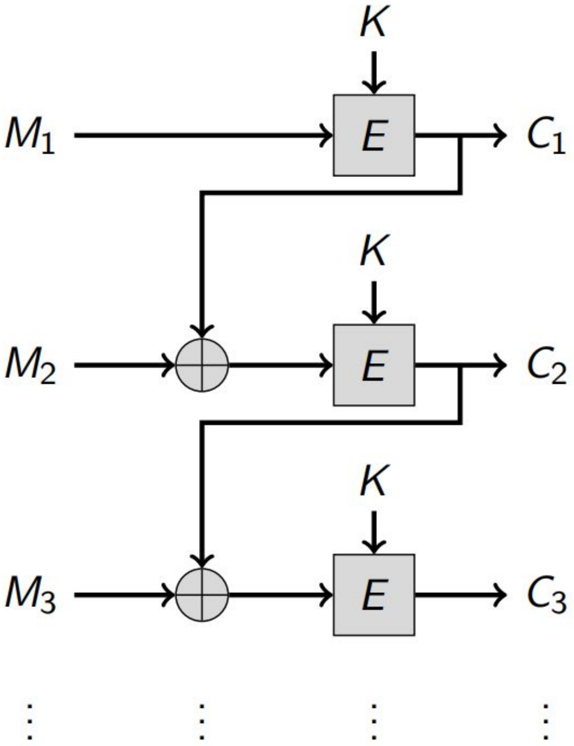
A: $C_1 = E_K(M), C_2 = E_K(M) \Rightarrow C_1 = C_2$

Clarification from Last Lecture



- Let M and N be our messages
- Then we have m_1 m_2 and n_1 n_2
- m_1 m_2 produce ciphertext c_1 c_2
- n_1 n_2 produce ciphertext d_1 d_2
- If m=n, then c=d

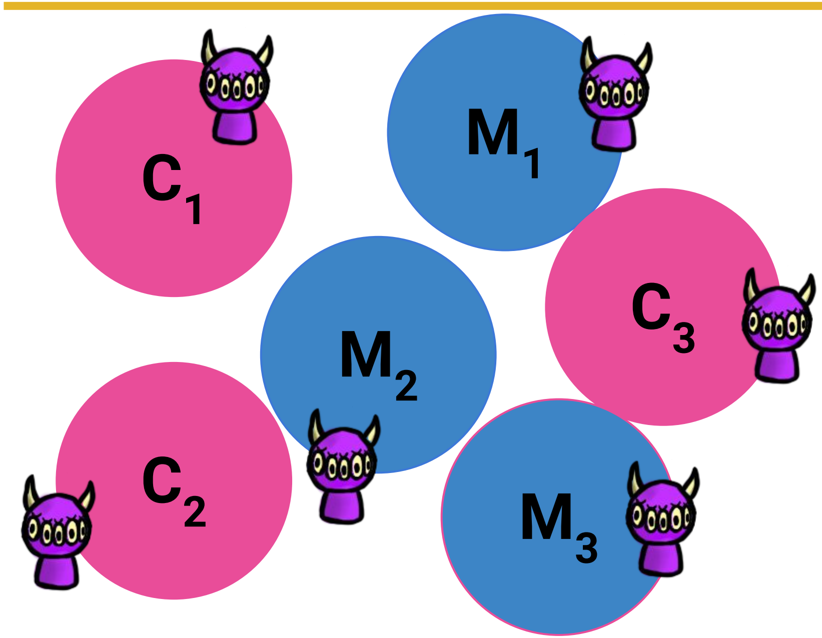
Clarification from Last Lecture



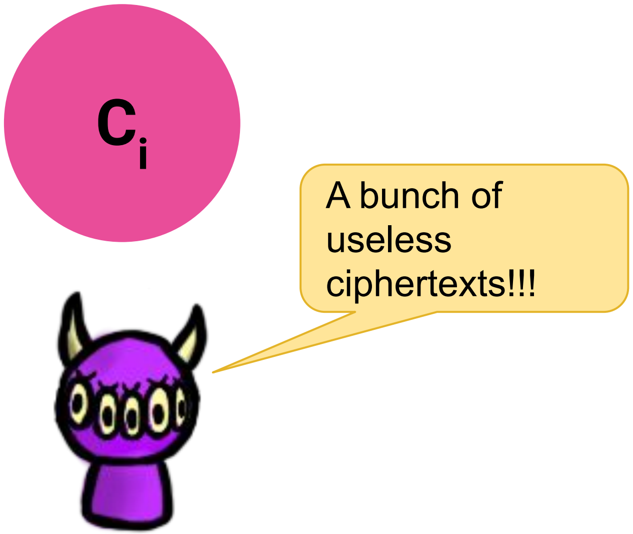
- Let M and N be our messages
- Then we have m_1 m_2 and n_1 n_2
- m_1 m_2 produce ciphertext c_1 c_2
- n_1 n_2 produce ciphertext d_1 d_2
- If m=n, then c=d

Helps facilitate ciphertext attacks as well as the general rule of patterns reveal information.

Cipher Security, IND-CCA2



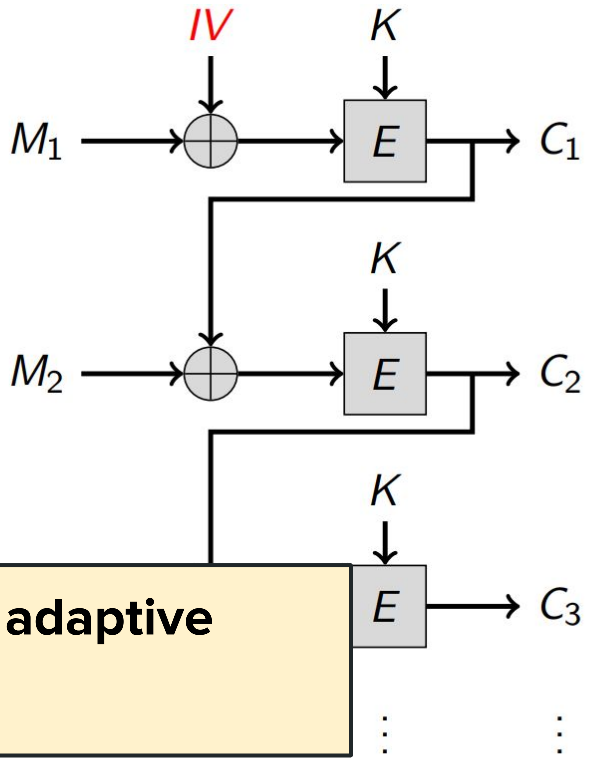
Adaptive chosen ciphertext attack



Eve cannot distinguish whether C_i is from M_1 or M_2

Recall CBC Mode for Block Ciphers:

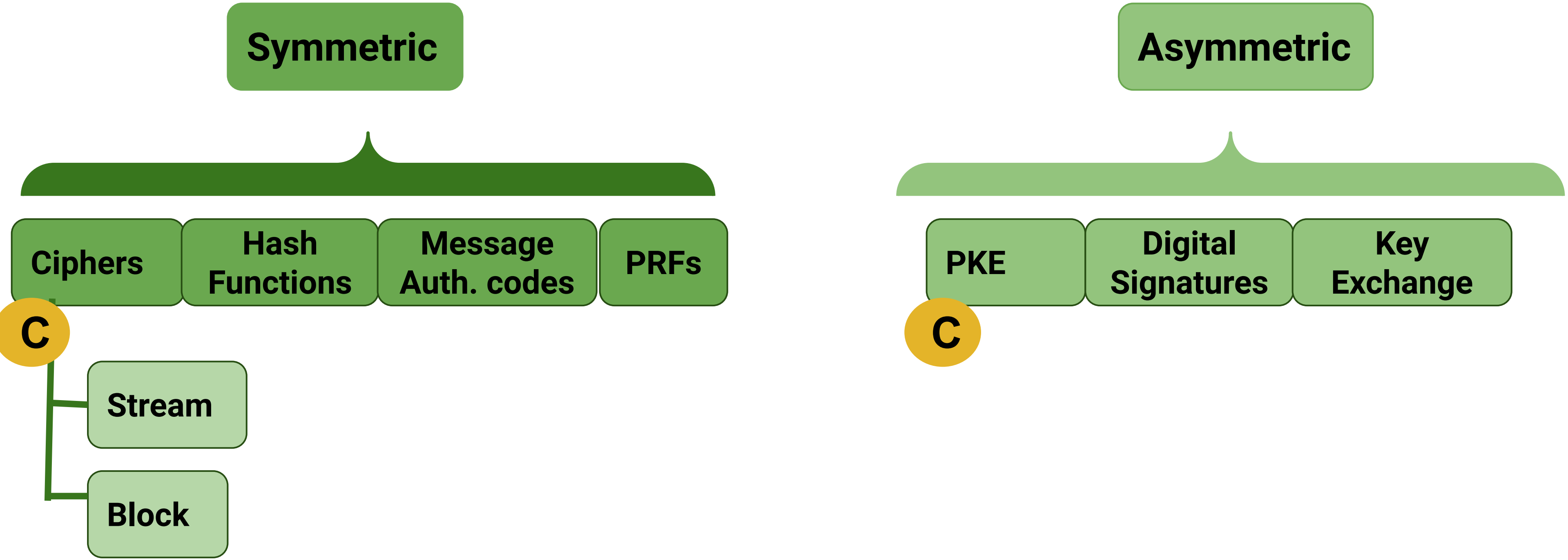
1. Generate a secret key k
2. Encrypt m using k and a generated IV
3. Decrypt c using k and the IV to get m



Security Goal: indistinguishability under adaptive chosen ciphertext attack (IND-CCA2)

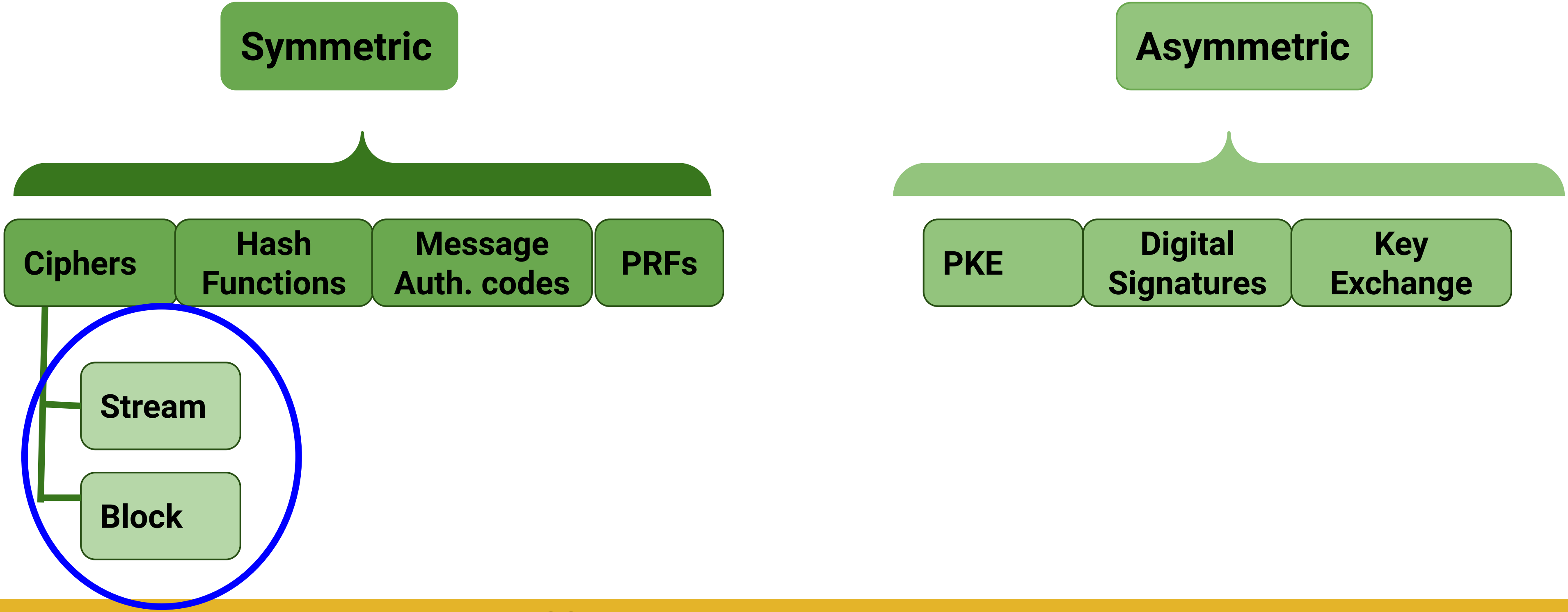
Today: More Cryptography

Cryptography Organization

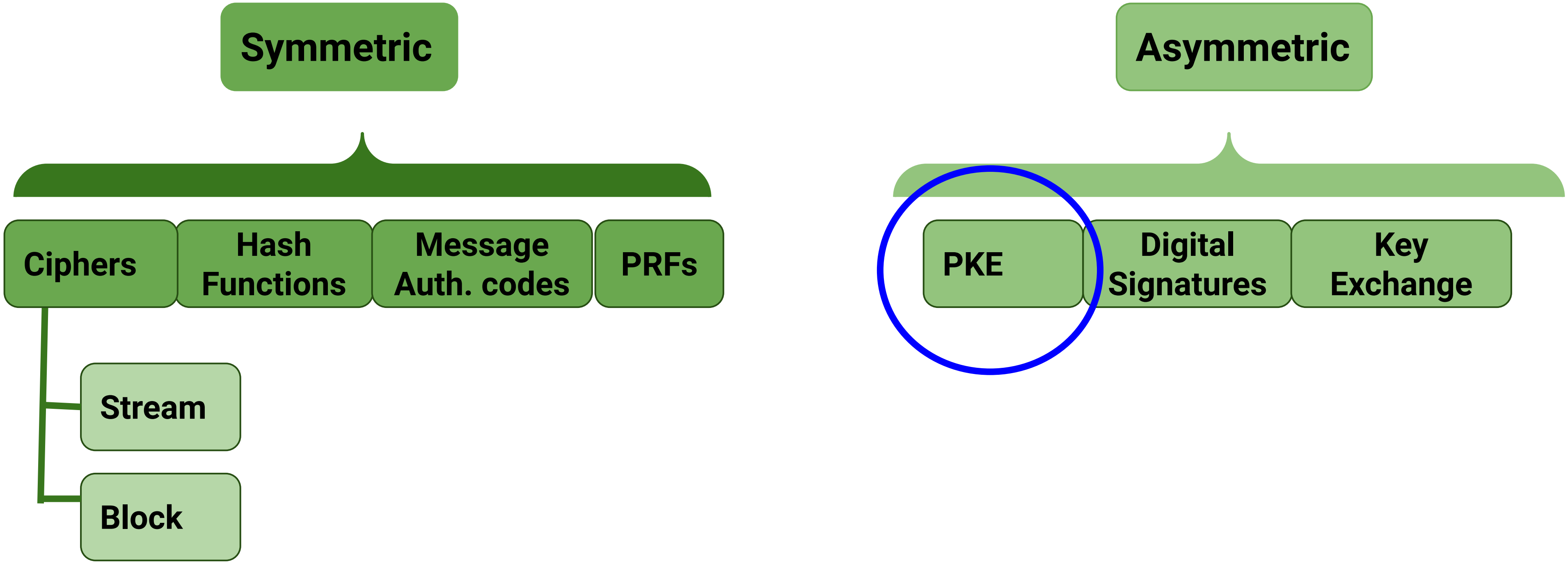


Organization display source: Doug Stebila

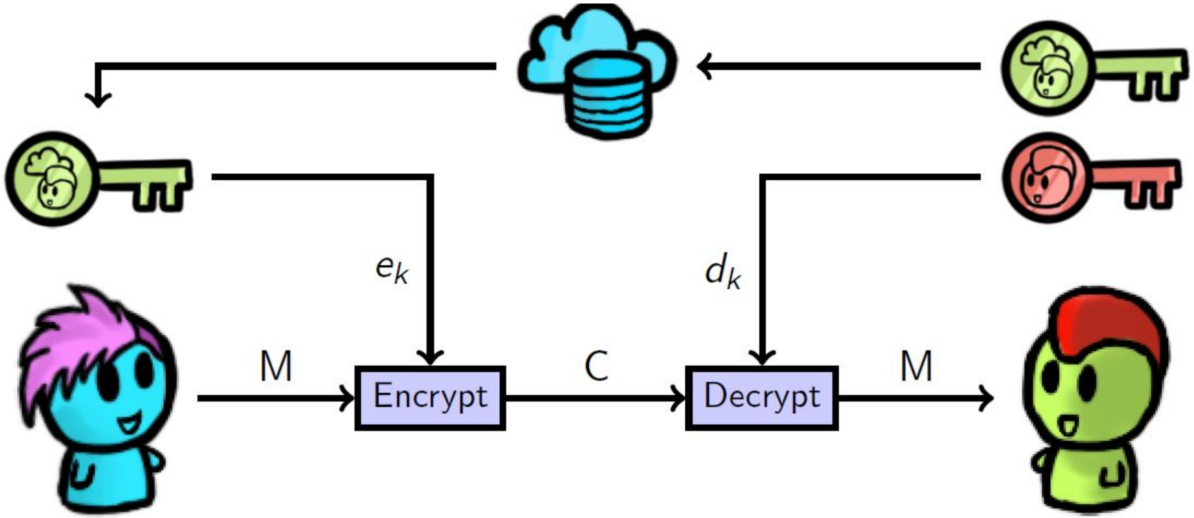
Cryptography Organization



Cryptography Organization



Public Key Cryptography, “1970s”

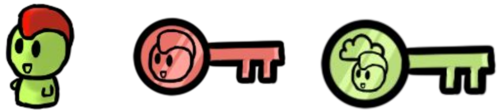


Examples:

- RSA, ElGamal, ECC, NTRU

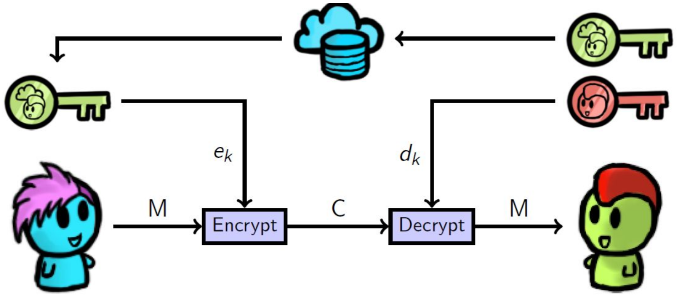
Steps for Public Key Cryptography?

1. Bob generates pair



2. Bob gives everyone the public key

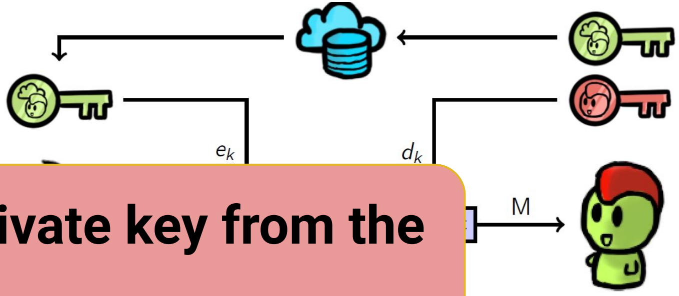
3. Alice encrypts m and sends it
4. Bob decrypts using private key



5. Eve and Alice can't decrypt, only have encryption key

Steps for Public Key Cryptography?

1. Bob generates pair 
2. Bob gives everyone the public key 
3. Alice encrypts m and sends it



4. Bob d

It must be hard to derive the private key from the public key

5. Eve and Alice can’t decrypt, only have encryption key

Requirements for PKE

- The encryption function? Must be easy to compute 
- The inverse, decryption? Must be hard for anyone without the key  vs. 

Thus, we require so called “one-way” functions for this.

Requirements for PKE

- The encryption function? Must be easy to compute 

- The inverse, decryption? Must be hard for anyone without the key  vs. 

Thus, we require so called “one-way” functions for this.

Because of encryption, also injective

Requirements for PKE

- The encryption function? Must be easy to compute 

- The inverse, decryption? Must be hard for anyone without the key  vs. 

Thus, we require so called “one-way” functions for this.

Because of encryption, also injective

Because of decryption, we need a “trapdoor”

Time for Textbook RSA

- Computational difficulty of the **factoring problem**
  - Given two large primes n = p*q, it is very hard to factor n.
- Modular arithmetic: integer numbers that “wrap around”
- Overview:

$$(m^e)^d \equiv m \pmod{n}$$

Easy for me to pick e, d, and n that satisfy that equation

Ugh. I know e and n (even m) and can’t find d!!!

Fun (?) Facts::

- RSA first popular public-key encryption method, published in 1977

Textbook RSA (Simplified Overview)

1. Choose two “**large primes**” *p* and *q* (secretly)
2. Compute n = p*q
3. “Choose” value e and find d such that $(m^e)^d \equiv m \pmod{n}$
4. **Public key**: (e, n)
5. **Private key**: d (other numbers tossed)
6. Encryption: $c \equiv m^e \pmod{n}$
7. Decryption: $c^d \pmod{n}$

Textbook RSA (Simplified Overview)

1. Choose two “**large primes**” *p* and *q* (secretly)
2. Compute n = p*q
3. “Choose” value e and find d such that $(m^e)^d \equiv m \pmod{n}$
4. **Public key**: (e, n)
5. **Private key**: d (other numbers tossed)
6. Encryp
7. Decryp

Decryption works, but factoring n breaks this!

Note:

- RSA? Rivest, Shamir, and Adleman

A Closer Look at RSA: Rec… Para…eters

1. Choose two **“large primes”**
2. Compute n = p*q
3. “Choose” valu…
4. **Public k…**
5. **Privat…** …)
6. Encryptio…
7. Decryp…n: c^d (r… n)

An “obvious” attack is for Eve
to attempt to factor *n*.
Then, compute c and e as
Bob would…

Note:
- RSA? Rivest, Shamir, and Adleman

A Closer Look at RSA: Rec Para eters

1. Choose two "**large pri**
2. Compute n = p*q
3. "Choose" valu
4. **Public ke**
5. **Privat**
6. Encryptio
7. Decryp : c^d (r d n)

An "obvious" a
to attempt
Then, compute
Bob woul

s for Eve

WARNING: Factoring is
hard...but

Note:

- RSA? Rivest, Shamir, and Adleman

Factoring and RSA

- You want to factor the public modulus? 
- Good news, abundant literature on factoring algorithms
- Bad news, “appropriate” primes will not be defeated

Factoring and RSA

- You want to factor the public modulus? 
- Good news, abundant literature on factoring algorithms
- Bad news, “appropriate” primes will not be defeated

Bad primes: easily factored

Reference: Pollard p-1 Factoring Algorithm

Inputs: Odd integer n and a "bound" b

1. a = 2
2. for j =2 to B
   a. Do a = a^j mod n
3. d = gcd(a-1,n)
4. if 1 < d < n
   a. Then return (d)
   b. Else return ("failure")

Note:

- This algorithm dates from 1974

Reference: Pollard p-1 Factoring Algorithm

Inputs: Odd integer n and a “bound” b

Suppose p is a prime divisor of n and q <= B

1. $a = 2$
2. for $j = 2$ to B
    a. Do $a = a^j \bmod n$
3. $d = \gcd(a-1, n)$
4. if $1 < d < n$
    a. Then return (d)
    b. Else return (“failure”)

Note:

- This algorithm dates from 1974

Reference: Pollard p-1 Factoring Algorithm

Inputs: Odd integer n and a “bound” b

Suppose p is a prime divisor of n and q <= B

Then for every q|(p-1), (p-1)|B!

1. a = 2
2. for j =2 to B
   a. a … o a = a^j mod n

…cd(a-1,n)

…if 1 < d < n

   a. Then return (d)
   b. Else return (“failure”)

Note:

- This algorithm dates from 1974

Reference: Pollard p-1 Factoring Algorithm

Inputs: Odd integer n and a "bound" b

1. a = 2
2. for j =2 to B
   a. Do $a = a^j \bmod n$
3. d = gcd(a-1,n)
4. if 1 < d < n
   a. Then return (d)
   b. Else return ("failure")

By the end...
$a \equiv 2^{B!} \pmod n$

Note:

- This algorithm dates from 1974

Reference: Pollard p-1 Factoring Algorithm

- Well, p|n, so

By the end...
$a \equiv 2^{B!} \pmod{n}$

1. a = 2
2. for j =2 to B
    a. Do $a = a^j \bmod n$
3. d = gcd(a-1,n)
4. if 1 < d < n
    a. Then return (d)
    b. Else return (“failure”)

Reference: Pollard p-1 Factoring Algorithm

- Well, p|n, so $a \equiv 2^{B!} \pmod{p}$

1. a = 2
2. for j =2 to B
    a. Do $a = a^j \bmod n$
3. d = gcd(a-1,n)
4. if 1 < d < n
    a. Then return (d)
    b. Else return ("failure")

Reference: Pollard p-1 Factoring Algorithm

- Well, $p|n$, so $a \equiv 2^{B!} \pmod{p}$
- $2^{p-1} \equiv 1 \pmod{p}$

Fermat's little theorem

1. $a = 2$
2. for $j = 2$ to B
   a. Do $a = a^j \bmod n$
3. $d = \gcd(a-1, n)$
4. if $1 < d < n$
   a. Then return (d)
   b. Else return ("failure")

Reference: Pollard p-1 Factoring Algorithm

- Well, p|n, so $a \equiv 2^{B!} \pmod{p}$
- $2^{p-1} \equiv 1 \pmod{p}$
- Since (p-1) | B!, it follows that …
- $a \equiv 1 \pmod{p}$ meaning p | (a-1)

1. a = 2
2. for j =2 to B
   a. Do $a = a^j \bmod n$
3. d = gcd(a-1,n)
4. if 1 < d < n
   a. Then return (d)
   b. Else return (“failure”)

Reference: Pollard p-1 Factoring Algorithm

- Well, $p|n$, so $a \equiv 2^{B!} \pmod{p}$
- $2^{p-1} \equiv 1 \pmod{p}$
- Since $(p-1) \mid B!$, it follows that …
- $a \equiv 1 \pmod{p}$ meaning $p \mid (a-1)$

1. $a = 2$
2. for $j = 2$ to B
   a. Do $a = a^j \bmod n$
3. $d = \gcd(a-1, n)$
4. if $1 < d < n$
   a. Then return (d)
   b. Else return (“failure”)

Since $p|n$ and $p|d$ with $d = \gcd(a-1, n)$, then d is a non-trivial divisor of n

Note:

- Assumes a<>1

Bad Primes: Pollard p-1 Factoring Example

- N = 15770708441, apply algorithm with B=180
- Found a = 11620221425 and computed d = 135979

$$15770708441 = 135979 * 115979$$

- The algorithm works because of **135979**

What are the factors of 135978?

Bad Primes: Pollard p-1 Factoring Example

- N = 15770708441, apply algorithm with B=180
- Found a = 11620221425 and computed d = 135979

$$15770708441 = 135979 * 115979$$

- The algorithm works because of **135979**

What are the factors of 135978 (the p-1)?

A: 135978 = 2 * 3*131 * 173

Bad Primes: Pollard p-1 Factoring Example

- N = 15770708441, apply algorithm with B=180
- Found a = 11620221425 and computed d = 135979

15770708441= 135979 * 115979

- The algorithm works because of **135979**

What are the factors of 135978 (the p-1)?

A: 135978 = 2 * 3*131 * 173

- By taking B>=173, 135978|B! as desired!

Textbook RSA (Simplified Overview)

1. Choose two "**large primes**" *p* and *q* ... ecretly)
2. Compute n = p*q
3. "Choose" value e and f... (mod *n*)
4. **Public key**: (e, n)
5. **Priv...**
6. Encryp...
7. Decryp...

...**toring n breaks this!**

WARNING: this was textbook RSA, do **not** use!!!

Why not "Textbook RSA"? Start with an Example

Example: (Tiny RSA), p=53, q=101, e=139, d=1459

Encryption: $c \equiv m^e \pmod{n}$, **Decryption:** c (n)

- Compute n
- Compute $C_1 = E_e(1011)$. Verify the decryption works
- Compute $C_2 = E_e(4)$. Verify the decryption works
- Compute $D_d(C_1 * C_2)$. What is happening…and why?

Note::

- The * here indicates multiplication/compute a product

Why not "Textbook RSA"? Start with an Example

Example: (Tiny RSA), $p=53$, $q=101$, $e=139$, $d=1459$

Encryption: $c \equiv m^e \pmod{n}$, **Decryption:** $c^d \pmod{n}$

- Compute n
- Compute $C_1 = E_e(1011)$. Verify the decryption works
- Compute $C_2 = E_e(4)$. Verify the decryption works
- Compute $D_d(C_1 * C_2)$. What is happening…and why?

Note:

- The * he

A: The decryption is the product of the original plaintexts!!!

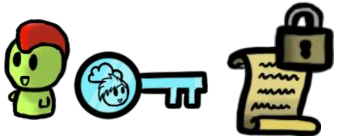
Malleability

$$\mathbf{A:}\ (m_1)^e * (m_2)^e = (m_1 * m_2)^e$$

It is possible to transform a ciphertext into another ciphertext that decrypts to a related plaintext

Undesirable (most of the time)

RSA and a Chosen Ciphertext Attack

- Alice is using RSA, public key (e, n) 
- Bob sends $c = E_e(m)$ 
- We are Eve! We snag c. 
- Alice…is confident about textbook RSA, will decrypt any ciphertext except c for us

Goal: Ask Alice to decrypt something (other than c) that helps us learn m

Executing CCA on Textbook RSA

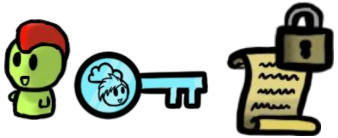
- Alice is using RSA, public key (e, n)  
- Bob sends $c = E_e(m)$   
- We-Eve ask Alice to decrypt $c_2 = 2^e * c_1$ 

I am so clever mwahaha

Q: Decrypts to?

Q: Decrypts to?

Executing CCA on Textbook RSA

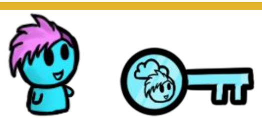
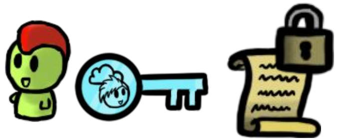
- Alice is using RSA, public key (e, n) 
- Bob sends $c = E_e(m)$ 
- We-Eve ask Alice to decrypt $c_2 = 2^e{*}c_1$ 

Q: Decrypts to?

Q: Decrypts to?

A: decryption gives $(2^e * c_1)^d \equiv 2m$

Executing CCA on Textbook RSA

- Alice is using RSA, public key (e, n) 
- Bob sends $c = E_e(m)$ 
- We-Eve ask Alice to decrypt $c_2 = 2^e * c_1$ 

I am so clever mwahaha

Q: Decrypts to?

A: decryption gives $(2^e * c_1)^d \equiv 2m$

Textbook RSA: vulnerable to CCA
Note: Can be addressed with padding techniques

Show Naive RSA Encryption is not IND-CPA Secure

1. Eve produces two plaintexts, m_0 and m_1 

Show Naive RSA Encryption is not IND-CPA Secure

1. Eve produces two plaintexts, m_0 and m_1 

2. “Challenger” encrypts an m as $c^* \leftarrow m_b^e \pmod N$, secret b 

Show Naive RSA Encryption is not IND-CPA Secure

1. Eve produces two plaintexts, m_0 and m_1 

2. "Challenger" encrypts an m as $c^* \leftarrow m_b^e \pmod N$, secret b 

3. Eve's goal? Determine $b \in \{0,1\}$

Show Naive RSA Encryption is not IND-CPA Secure

1. Eve produces two plaintexts, m_0 and m_1 

2. "Challenger" encrypts an m as $c^* \leftarrow m_b^{\,e} \pmod N$, secret b 

3. Eve's goal? Determine $b \in \{0,1\}$

4. Sooo, Eve computes $c \leftarrow m_1^{\,e} \pmod N$

   If $c^* = c$ then Eve knows $m_b = m_1$
   If $c^* <> c$ then Eve knows $m_b = m_0$

Show Naive RSA Encryption is not IND-CPA Secure

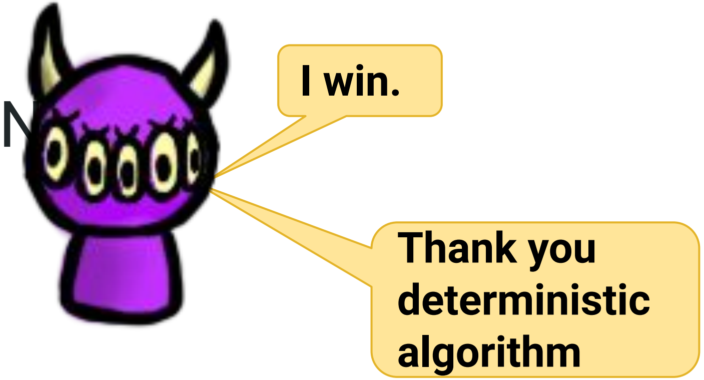
1. Eve produces two plaintexts, m_0 and m_1 

2. “Challenger” encrypts an m as $c^* \leftarrow m_b^{\,e}$ (mod N), secret b 

3. Eve’s goal? Determine $b \in \{0,1\}$

4. Sooo, Eve computes $c \leftarrow m_1^{\,e}$ (mod N

   If $c^* = c$ then Eve knows $m_b = m_1$
   If $c^* <> c$ then Eve knows $m_b = m_0$



Implications of CCA versus CPA Security

Act.

Consider, when selecting an appropriate cryptosystem, what are the trade-offs (in security and practice) of a system that is:

- IND-CCA secure
- IND-CPA secure
- IND-CCA and IND-CPA secure

Identify: at least one implication for each of the above. Submit to Learn

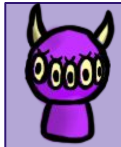
Adversaries and their Goals

You've assumed my goal is the secret/private key…

Adversaries and their Goals

Adversaries and their Goals

Goal 1: Total Break

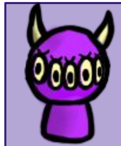


- Win the secret key k or
- Win Bob's private key k_b
- Can decrypt any c_i *for:*

$$c_i = E_k(m) \text{ or } c_i = E_{kb}(m)$$

- All messages using compromised k revealed
- Unless **detected** game over

Goal 2: Partial Break



- Decrypt **a ciphertext** *c* (without the key)
- Learn **some** specific information about a message *m* from *c*

**Need to occur with non-negligible probability.

- **Some (or a)** message revealed

Goal 3: Distinguishable Ciphertexts

- $P\{learn\ b \in \{0,1\}\}$ exceeds ½
- Distinguish between $E(m_1)$ and $E(m_2)$ or between E(m) and E(random string)

- The ciphertexts are leaking small/some information…

Semantic Security of RSA

- We saw CCA against Naive RSA

- We showed IND-CPA on Naive RSA

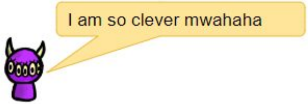
Show Naive RSA Encryption is not IND-CPA Secure

1. Eve produces two plaintexts, m_0 and m_1
2. “Challenger” encrypts an m as $c^* \leftarrow m_b^{\ e}$ (mod N), secret b
3. Eve’s goal? Determine $b \in \{0,1\}$
4. Sooo, Eve computes $c^* \leftarrow m_1^{\ e}$ (mod N)

   If $c^* = c$ then Eve knows $m_b = m_1$
   If $c^* = c$ then Eve knows $m_b = m_0$

Executing CCA on Textbook RSA

- Alice is using RSA, public key (e, n)
- Bob sends $c = E_e(m)$
- We-Eve ask Alice to decrypt $c_2 = 2^e * c_1$



Q: Decrypts to?

Q: Decrypts to?

A: decryption gives $(2^e * c_1)^d \equiv 2m$

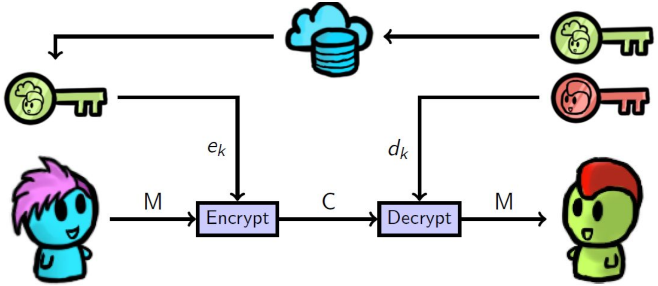
Fix it? Ciphertext Distinguishability

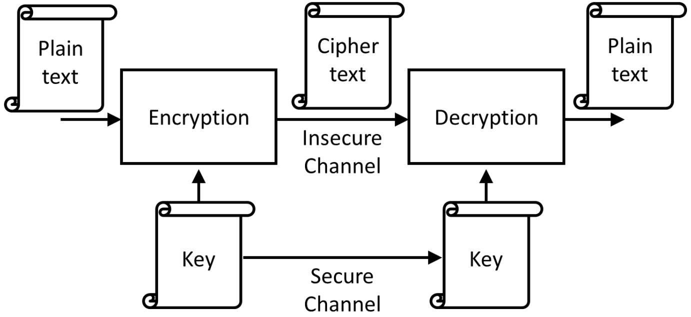
Goal: prove (given comp. assumptions) no information regarding the *m* is revealed in polynomial time by examining $c = E(m)$

- If E( ) is deterministic, fail
- Thus, require some randomization

RSA-OAEP: Optimal Asymmetric Encryption Padding

Practicality of Public-Key versus Private-Key

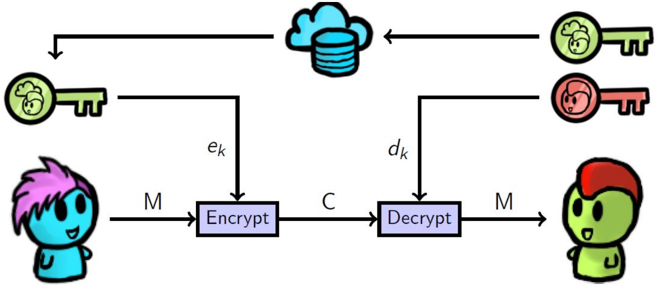




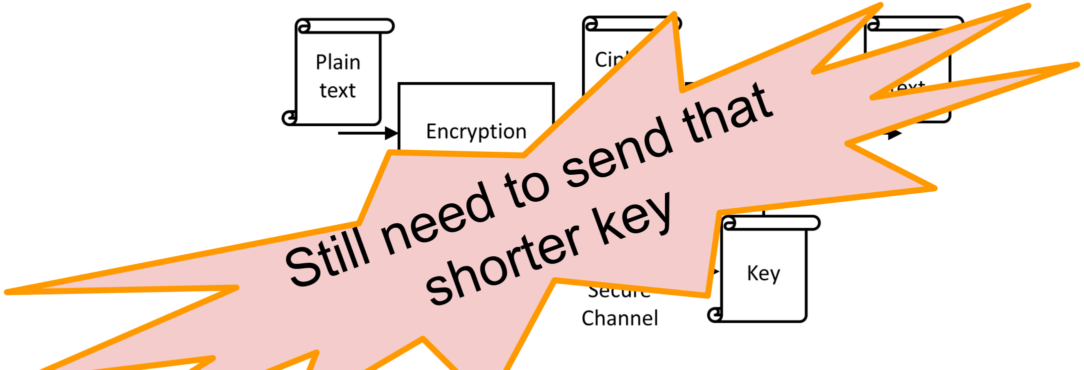
1. Longer keys
2. Slower
3. Different keys for E(m) and D(c)

1. Shorter keys
2. Faster
3. Same key for E(m) and D(c)

Practicality of Public-Key versus Private-Key



1. Longer keys
2. Slower
3. Different keys for E(m) and D(c)



1. Shorter keys
2. Faster
3. Same key for E(m) and D(c)

Hybrid Cryptography

- Combine the two!!!!!!!
- Pick a random “128-bit” key K for a secret-key system
- Encrypt the large message with the key K (e.g., using AES)

And then…

- Encrypt the key K using a public-key system!
- Send the encrypted message and encrypted key to Bob

Hybrid Cryptography

- Combine the two!!!!!!!
- Pick a random “128-bit” key K for a secret-key system
- Encrypt the large message with the key K (e.g., using AES)

And then…

- Encrypt the key K using a public-key system!
- Send the encrypted message and encrypted key to Bob

Hybrid cryptography is used in (many) applications on the internet

Just Checking...

Public: (e_A, d_A)

Secret: K

Public: (e_B, d_B)

Secret: ?

- Enc/Dec functions: $E_{key}(*)$, $D_{key}(*)$
- Alice wants to send a **large** message *m* to Bob,

Q: How should Alice build the message efficiently? How does Bob recover m?

Just Checking…

Public: (e_A, d_A)

Secret: K

Public: (e_B, d_B)

Secret: ?

- Enc/Dec functions: $E_{key}(*)$, $D_{key}(*)$
- Alice wants to send a **large** message *m* to Bob,

Q: How should Alice build the message efficiently? How does Bob recover m?

A: Alice computes $c_1 = E_{eb}(K)$, $c_2 = E_K(m)$ and sends $<c_1||c_2>$
Bob recovers $K = D_{db}(c_1)$ and then $m = D_K(c_2)$

Thursday: More Cryptography…