



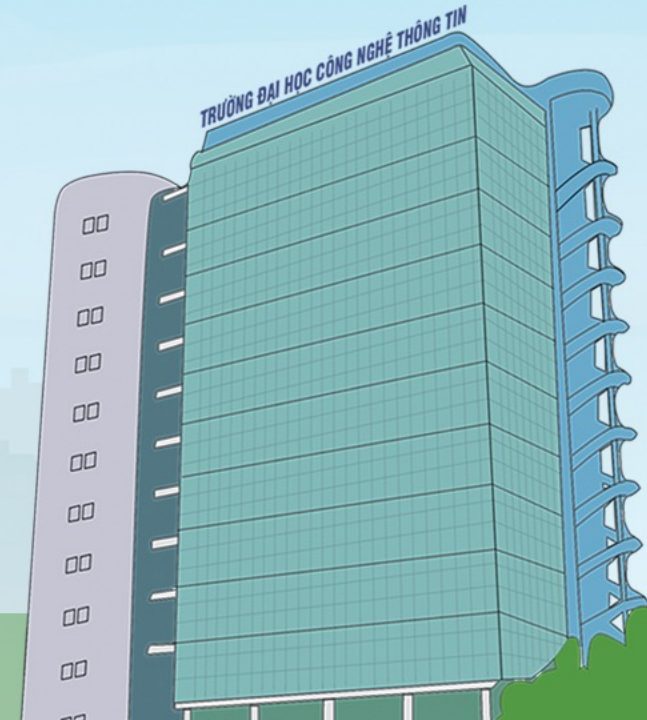
UNIVERSITY OF INFORMATION TECHNOLOGY – VNU-HCM
Faculty of Computer Networks and Communications

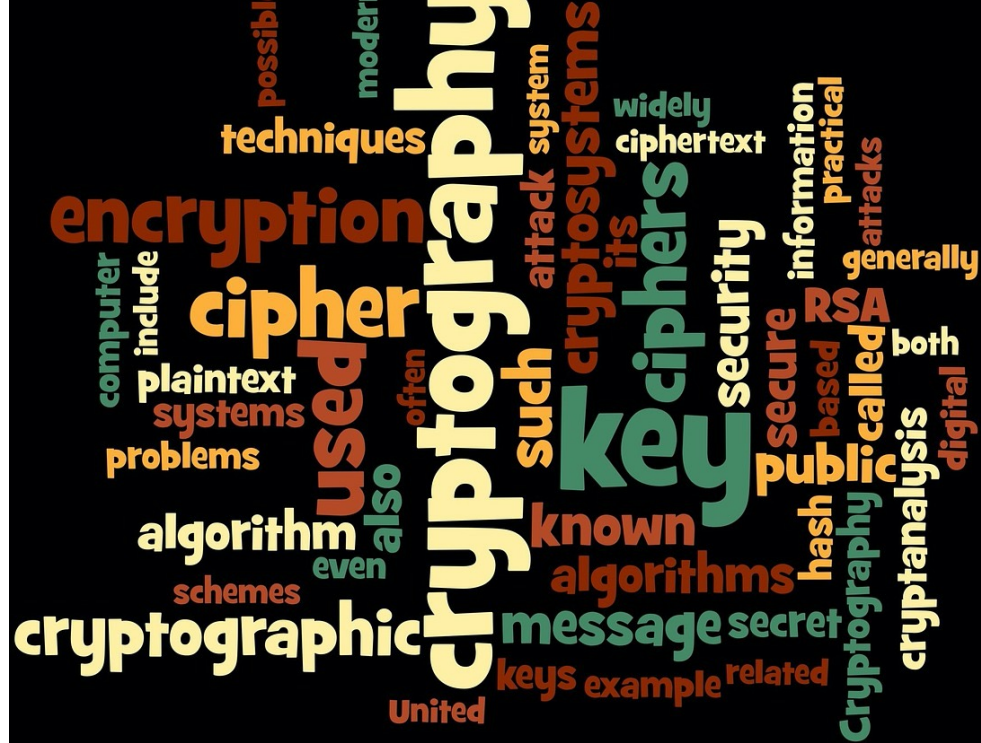
Public-key Cryptography and RSA

NT219 - CRYPTOGRAPHY

Lecturer: MSc. Hoa Nguyen-Thanh

hoant@uit.edu.vn





Today:
Public-key Cryptography and RSA

Reading:
• CS Chapter 9

Where we are today...

Acknowledgement:

Slides are adapted from

CSC 580 course: Cryptography and Security in Computing

Professor Stephen R. Tate, The University of North Carolina at Greensboro, US

For educational purposes only.

Learning objectives

Overview

We will focus on:

- **Public-key Cryptography: Concepts**
 - **Some supporting math**

RSA



Recall...

Types of ciphers

- **Time:** Classical ciphers – Modern ciphers
- **Data unit:** Block ciphers – Stream ciphers
- **Key:** Symmetric (Secret-key) ciphers – Asymmetric (Public-key) ciphers
- **Goal:** Confidentiality protection ciphers - Integrity Protection ciphers

In this course,

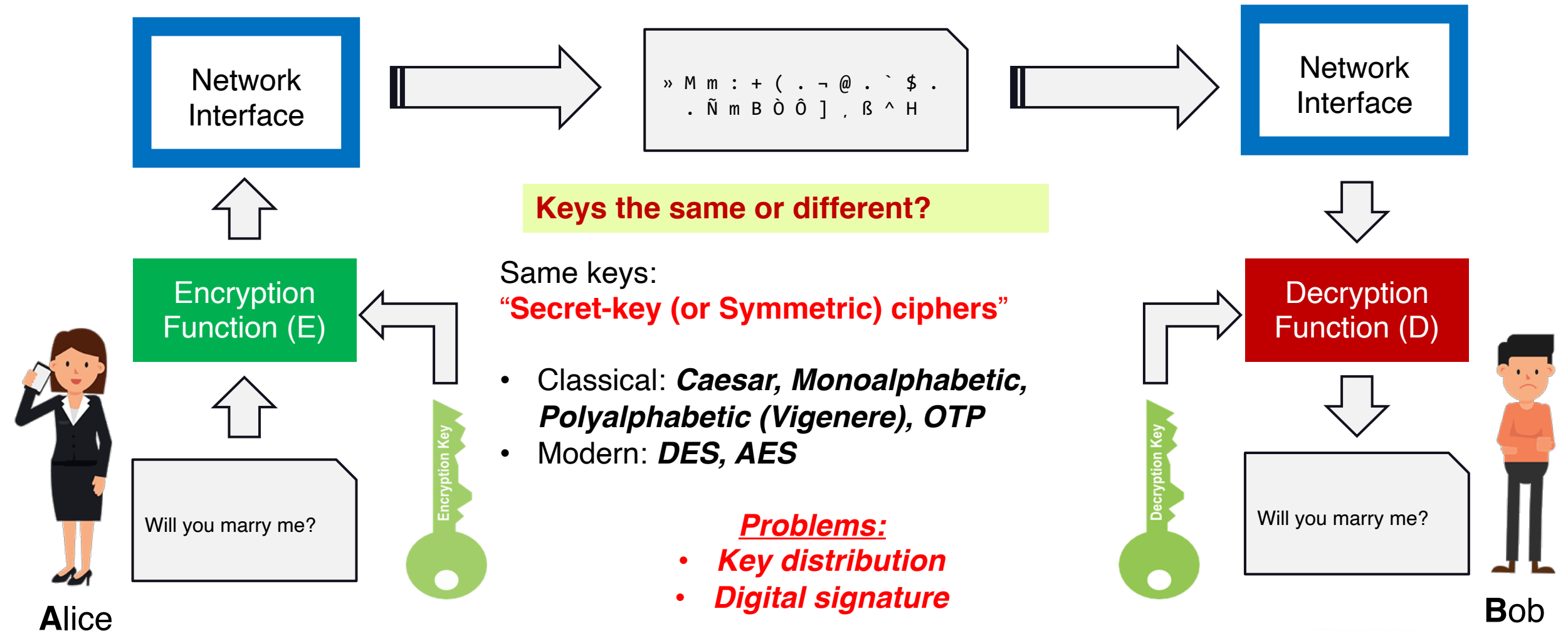
Cryptographic algorithms and protocols

- Symmetric encryption
- **Asymmetric encryption**
- **Data integrity algorithms**
- **Authentication protocols**



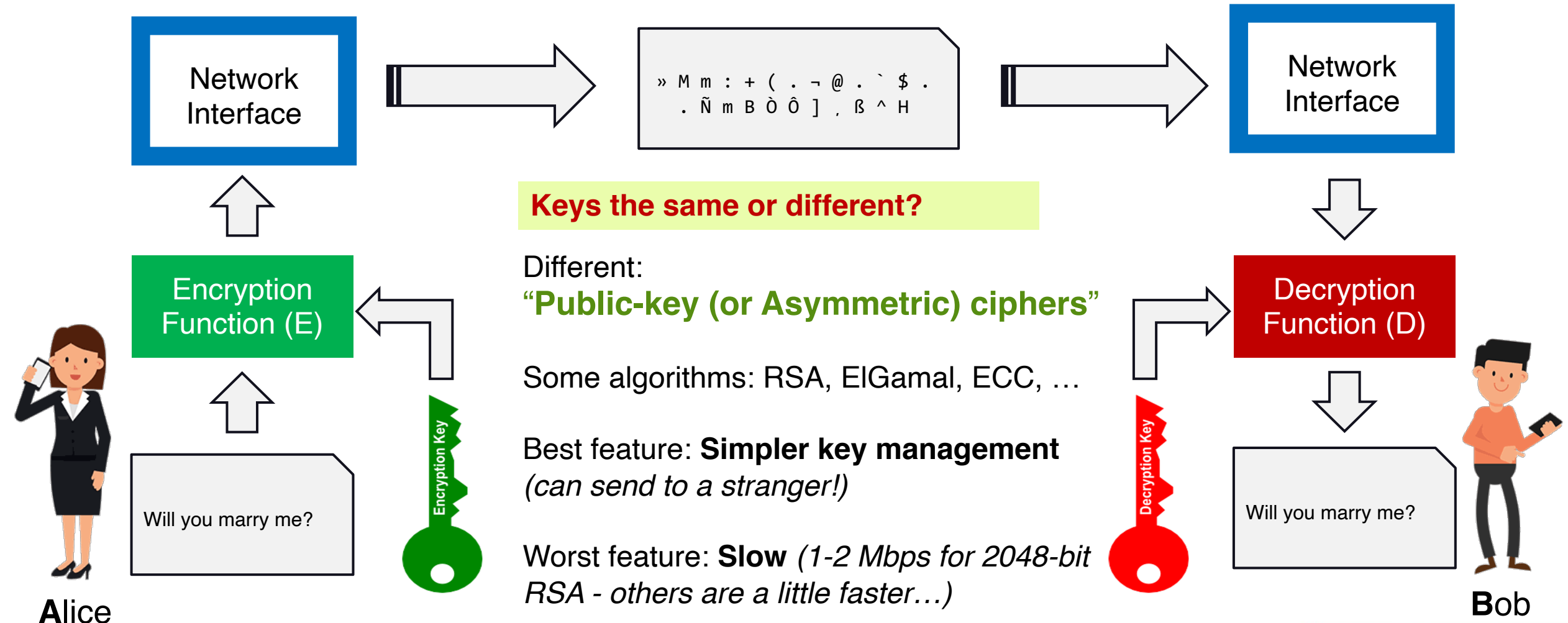
Recall...

Basic Encryption scheme



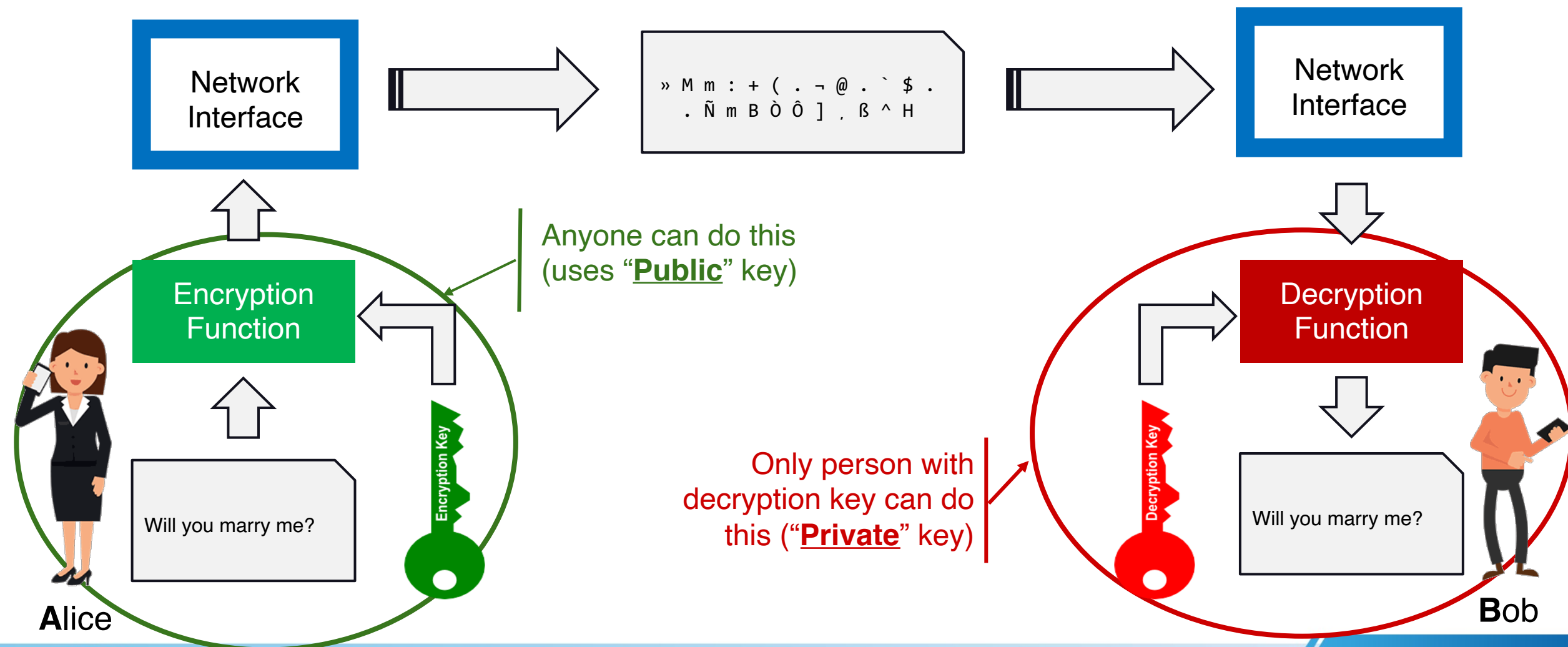
Recall...

Modern crypto – public-key cryptography



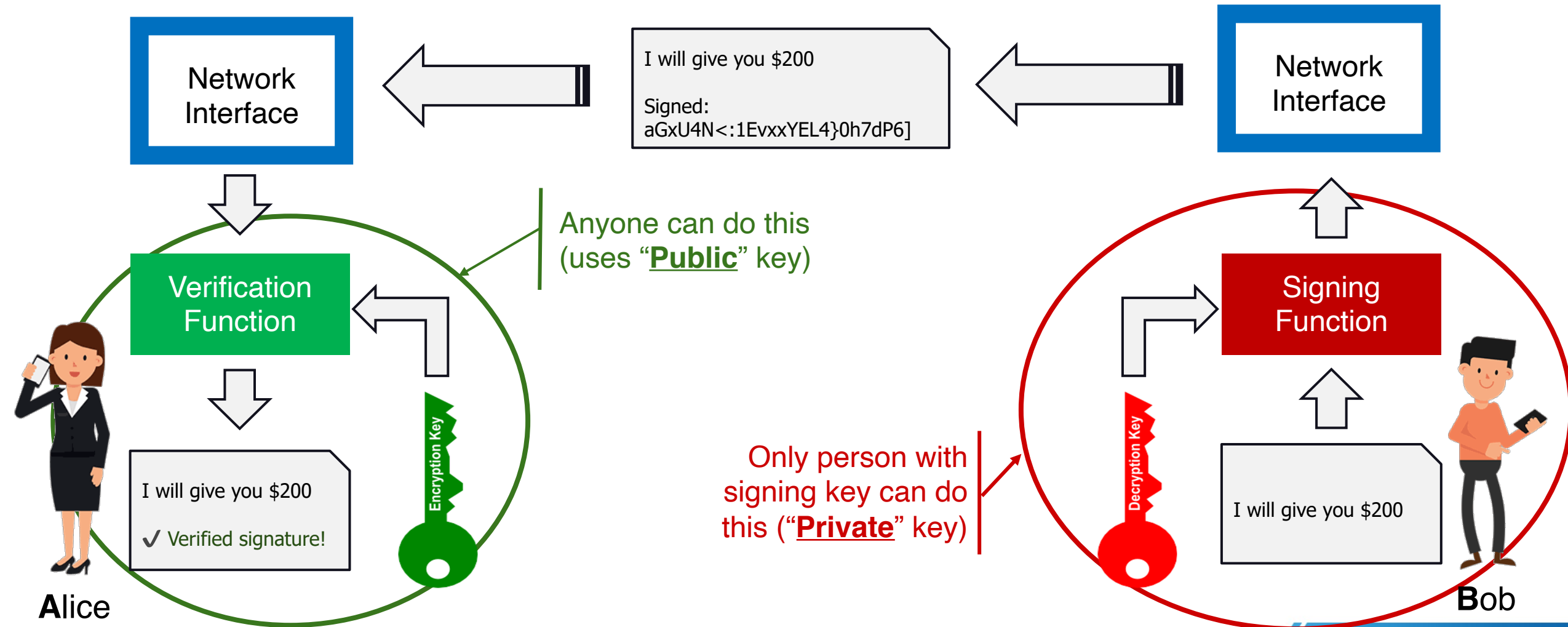
Public-key crypto for Confidentiality

Do we need a shared secret? ...model for confidentiality



Public-key crypto for Integrity Protection

Asymmetric (public-key) model for integrity



Where do the keys come from?

Public Key Crypto

Symmetric Ciphers

Randomness (R)



Secret Key (SK)

Public Key Crypto

Randomness (R)



KeyPair Generator (KPG)



PubKey (PU)



PrivKey (PR)

- Mathematical/Computational Properties

- $KPG(R) \rightarrow (PU, PR)$ is efficiently computable (polynomial time)
- For all messages M , $D(PR, E(PU, M)) = M$ (decryption works)
- Computing PR from PU is computationally infeasible (we hope!)

- Generally: PR has some “additional information” that makes some function of PU easy to compute (which is hard without that info) - this is the “**trapdoor secret**”

How can this be possible?

“trapdoor secret”

- To get a sense of how trapdoor secrets help:

Problem: How many numbers $x \in \{1, N-1\}$ have $\gcd(x, N) > 1$ for $N=32,501,477$?
(or: how many have a non-trivial common factor with N ?)

- How could you figure this out?

How long would it take to compute?

What if N were 600 digits instead of 8 digits?

What if I told you the **prime factorization** of N is $5,407 * 6,011$?

- 5,406 multiples of 6,011 share the factor 6,011 with N
 - 6,010 multiples of 5,407 share the factor 5,407 with N
 - No numbers in common between these two sets (prime numbers!)
- So $5,406 + 6,010 = 11,416$ numbers share a factor with 32,501,477

The factorization of N is a “trapdoor” that allows you to compute some functions of N faster

A Step Toward Public-Key Crypto

Overview

- So, when solving the problem: **Given a number N , how many positive integers share a non-trivial factor with N ?**
 - If you know the prime factorization of N , this is easy.
 - If you don't know the factorization, **don't know efficient solution**
- How does this fit into the public key crypto model?
 - Pick two large (e.g., 1024-bit) prime numbers p and q
 - Compute the product $N = p * q$
 - Public key is N (hard to find p and q !), private is the pair (p,q)
- Questions:
 - ***How do we pick (or detect) large prime numbers?***
 - ***How do we use this trapdoor knowledge to encrypt?***



Prime Numbers

Math review

- A prime number is a number p for which its only positive divisors are 1 and p
- *Why prime numbers are important?*

E.g: Primes Under 2000

Read more:

<https://www.mathsisfun.com/prime-factorization.html>

2	101	211	307	401	503	601	701	809	907	1009	1103	1201	1301	1409	1511	1601	1709	1801	1901
3	103	223	311	409	509	607	709	811	911	1013	1109	1213	1303	1423	1523	1607	1721	1811	1907
5	107	227	313	419	521	613	719	821	919	1019	1117	1217	1307	1427	1531	1609	1723	1823	1913
7	109	229	317	421	523	617	727	823	929	1021	1123	1223	1319	1429	1543	1613	1733	1831	1931
11	113	233	331	431	541	619	733	827	937	1031	1129	1229	1321	1433	1549	1619	1741	1847	1933
13	127	239	337	433	547	631	739	829	941	1033	1151	1231	1327	1439	1553	1621	1747	1861	1949
17	131	241	347	439	557	641	743	839	947	1039	1153	1237	1361	1447	1559	1627	1753	1867	1951
19	137	251	349	443	563	643	751	853	953	1049	1163	1249	1367	1451	1567	1637	1759	1871	1973
23	139	257	353	449	569	647	757	857	967	1051	1171	1259	1373	1453	1571	1657	1777	1873	1979
29	149	263	359	457	571	653	761	859	971	1061	1181	1277	1381	1459	1579	1663	1783	1877	1987
31	151	269	367	461	577	659	769	863	977	1063	1187	1279	1399	1471	1583	1667	1787	1879	1993
37	157	271	373	463	587	661	773	877	983	1069	1193	1283		1481	1597	1669	1789	1889	1997
41	163	277	379	467	593	673	787	881	991	1087		1289		1483		1693			1999
43	167	281	383	479	599	677	797	883	997	1091		1291		1487		1697			
47	173	283	389	487		683		887		1093		1297		1489		1699			
53	179	293	397	491		691				1097				1493					
59	181			499										1499					
61	191																		
67	193																		
71	197																		
73	199																		
79																			
83																			
89																			
97																			

Prime Numbers cont'd

Math review

- Question: **How common are prime numbers?**
 - The Prime Number Theorem states that there are approximately $n / \ln n$ prime numbers less than n .
 - Picking a random b -bit number, probability that it is prime is approximately $1/\ln(2^b) = (1/\ln 2) * (1/b) \approx 1.44 * (1/b)$
 - For 1024-bit numbers this is about 1/710
 - “Pick random 1024-bit numbers until one is prime” takes on average 710 trials (“pick random odd 1024-number” finds primes faster!)
 - This is efficient - if we can tell when a number is prime!

Primality Testing

Math review

- Problem: **Given a number n , is it prime?**

Basic algorithm: Try dividing all numbers $2, \dots, \sqrt{n}$ into n

Question: How long does this take if n is 1024 bits?

Fermat's Little Theorem

Prime number properties

- To do better, we need to understand some properties of prime numbers, such as...
- **Fermat's Little Theorem:**
If p is prime and a is a positive integer not divisible by p (*a coprime to p*), then:

$$a^{p-1} \equiv 1 \pmod{p}$$

Proof is on page 46 of the textbook (not difficult!).

Fermat's Little Theorem - cont'd

Prime number properties

- Explore this formula for different values of n and random a :

a	$a^{n-1} \bmod n$ ($n = 221$)	$a^{n-1} \bmod n$ ($n = 331$)	$a^{n-1} \bmod n$ ($n = 441$)	$a^{n-1} \bmod n$ ($n = 541$)
64	1	1	379	1
189	152	1	0	1
82	191	1	46	1
147	217	1	0	1
113	217	1	232	1
198	81	1	270	1

Question 1: What conclusion can be drawn about the primality of 221?

Question 2: What conclusion can be drawn about the primality of 331?

Primality Testing - First Attempt

Prime number properties

- Tempting (but incorrect) primality testing algorithm for n :

Pick random $a \in \{2, \dots, n-2\}$
if $a^{n-1} \bmod n \neq 1$ then return “not prime”
else return “probably prime”

- Why doesn't this work?

Carmichael numbers...

Example: 2465 is obviously not prime, but

Note: Not just for these a 's, but $a^{n-1} \bmod n = 1$ for **all** a 's that are relatively prime to n

a	$a^{n-1} \bmod n$ ($n = 2465$)
64	1
189	1
82	1
147	1
113	1
198	1

Read more: https://en.wikipedia.org/wiki/Carmichael_number

Miller-Rabin Alogrithm

Some Supporting Math

- The previous idea is good, with some modifications

MILLER-RABIN-TEST(n) // Assume n is odd

Find $k > 0$ and q odd such that $n-1 = 2^k q$

Pick random $a \in \{2, \dots, n-2\}$

$x = a^q \bmod n$

if $x = 1$ or $x = n-1$ then return “**possible prime**”

for $j = 1$ to $k-1$ do

$x = x^2 \bmod n$

 if $x = n-1$ then return “**possible prime**”

return “composite”

- If n is prime, always returns “**possible prime**”
If n is composite, says “**possible prime**” (incorrect) with probability $< 1/4$
- *Idea*: Run 50 times, and accept as prime iff all say “possible prime”

Euler's Totient Function and Theorem

Some Supporting Math

- *Euler's totient function*: $\phi(n)$ = number of integers from 1 .. $n-1$ that are **relatively prime** to n .
- If $s(n)$ is count of 1.. $n-1$ that share a factor with n , $\phi(n) = n - 1 - s(n)$
 - $s(n)$ was our “**trapdoor function**” example
 - $\phi(n)$ easy to compute if **factorization** of n known
 - Don't know how to efficiently compute otherwise
- If n is product of two primes, $n=p^*q$, then $s(n)=(p-1) + (q-1) = p + q - 2$
 - So $\phi(p^*q) = p^*q - 1 - (p + q - 2) = p^*q - p - q + 1 = (p-1)^*(q-1)$
- Euler generalized Fermat's Little Theorem to **composite moduli**:
 - *Euler's Theorem*: For every a and n that are relatively prime (i.e., $\gcd(a,n)=1$),

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

Read more: <https://crypto.stanford.edu/pbc/notes/numbertheory/order.html>

RSA cipher

History



Ron **R**ivest, Adi **S**hamir and Leonard **A**dleman

- RSA is one of the first public-key cryptosystems and is widely used for secure communication.
- This cipher was developed in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman at MIT and first published in 1978.
- In RSA scheme, the plaintext and ciphertext are integers between 0 and $n-1$ for some n . A typical size for n is **1024 bits**, or **309** decimal digits.

RSA Algorithm

RSA Public-key encryption

- Key Generation:
 1. Pick two large primes p and q
 2. Calculate $n = p \cdot q$ and $\phi(n) = (p-1) \cdot (q-1)$
 3. Pick a random e is **relatively prime** to $\phi(n)$ or $\gcd(\phi(n), e) = 1$ and $1 < e < \phi(n)$
 4. Determine d such that $de \equiv 1 \pmod{\phi(n)}$ or $d = e^{-1} \pmod{\phi(n)}$ (Modulo inverse)
[Use extended Euclid's algorithm!]
 5. Public key is $PU = (e, n)$; Private key is $PR = (d, n)$
- Encryption of message $M \in \{0, \dots, n-1\}$: $E(PU, M) = M^e \bmod n$
- Decryption of ciphertext $C \in \{0, \dots, n-1\}$: $D(PR, C) = C^d \bmod n$

RSA Example

RSA Public-key encryption

- Simple example:

$$p = 17, q = 11$$

$$n = p * q = 17 * 11 = 187$$

$$\phi(n) = (p-1) * (q-1) = 16 * 10 = 160$$

$$e = 7$$

$$\rightarrow d = ?$$

Recall: Euclid's algorithm

A very important algorithm!

- Numbers a and b are **relatively prime** if $\gcd(a,b) = 1$
- How to compute gcd fast?



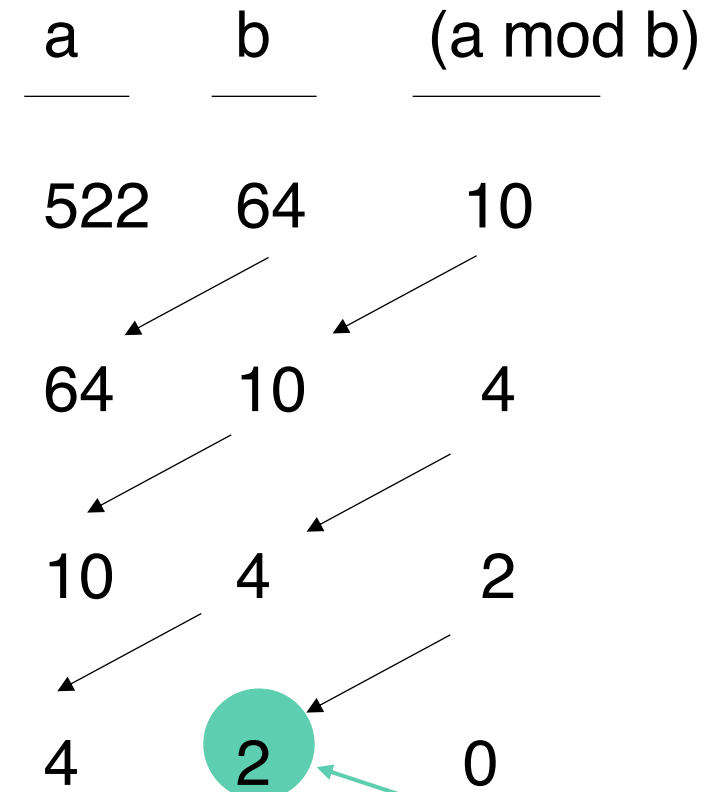
Euclid's Algorithm

Assuming $a > b$:

$\gcd(a,b)$:
if $(b \mid a)$ then return b
else return $\gcd(b, (a \bmod b))$

Running time: **$O(\log b)$**

Example: $\gcd(522, 64)$



$a \bmod b = 0$ means $b \mid a$, so done
Final answer $\gcd(522, 64) = 2$

Extended Euclid's algorithm

Supporting math

- Compute integers x and y such that $ax + by = \gcd(a,b)$ given a and b
- Extended Euclid's algorithm

```
euclid_extended(a,b):
```

```
  x0 = 1, x1 = 0, y0 = 0, y1 = 1
```

```
  while (b > 0)
```

```
    r = a mod b; if r=0 then Break
```

```
    q= a / b
```

```
    x= x0-x1*q
```

```
    y= y0-y1*q
```

```
    a=b ; b=r ; x0=x1 ; x1=x ; y0=y1 ; y1=y
```

```
  return x,y
```

Extended Euclid's algorithm

Compute modulo inverse

- Compute d such that $de \equiv 1 \pmod{\phi(n)}$ or $d = e^{-1} \pmod{\phi(n)}$
- Using Extended Euclid's algorithm

```
modulo_inverse(e,  $\phi(n)$ ):
```

```
  y0 = 0, y1 = 1
```

```
  while (e > 0)
```

```
    r =  $\phi(n)$  mod e ; if r=0 then Break
```

```
    q =  $\phi(n)$  / e
```

```
    d = y0 - y1*q
```

```
     $\phi(n)$ =e ; e=r ; y0=y1 ; y1=d
```

```
  return d
```

Note: If $d < 0$, then $d = d + \phi(n)$

RSA Example

RSA Public-key encryption

- Simple example:

$$p = 17, q = 11$$

$$n = p * q = 17 * 11 = 187$$

$$\phi(n) = (p-1) * (q-1) = 16 * 10 = 160$$

$$e = 7$$

$$\rightarrow d = 23 \text{ [Note: } 23 * 7 = 161 = 160 + 1 \text{]}$$

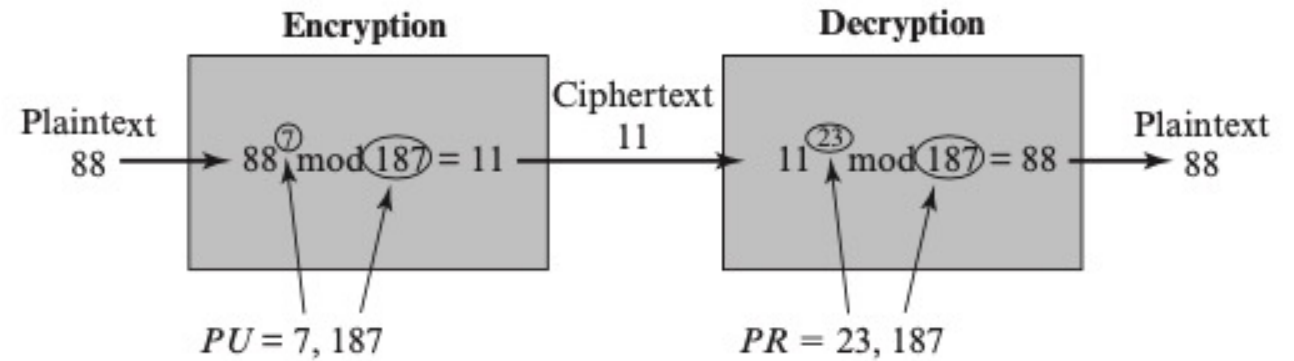
$$\rightarrow \text{PU} = (e, n) = (7, 187); \text{PR} = (d, n) = (23, 187)$$

- Encrypting message **M= 88**:

$$88^7 \bmod 187 = 11$$

- Decrypting **C=11**:

$$11^{23} \bmod 187 = 88$$



Your turn!

RSA practices

Perform encryption and decryption (for Confidentiality) using the RSA algorithm:


- Find PU, PR
- Encrypt M to C, then decrypt C
 - **a.** $p=11$; $q=3$, $e=3$; $M=4$
 - **b.** $p=7$; $q=17$, $e=5$; $M=32$
 - **c.** $p=23$; $q=41$; $e = 7$; $M = 35$
 - **d.** $p=61$; $q= 53$; $e= 17$; $M = 65$

For next class...

Looking ahead

- Today: **Public-key Cryptography and RSA**
 - Lab 03
- Ready for next class: **Diffie-Hellman Key Exchange**
 - Reading: **Textbook CS**
 - **Chapter 2:** (2.8 - Discrete logarithms)
 - **Chapter 10** (10.1 - Diffie–Hellman key exchange)





Today end,
Congrats!

👤 Hoa Nguyen-Thanh

✉ hoant@uit.edu.vn

🌐 NT219 - Cryptography