UNIVERSITY AT ALBANY
State University of New York

Lecture 19: Cryptography

Dr. Chengjiang Long
Computer Vision Researcher at Kitware Inc.
Adjunct Professor at SUNY at Albany.
Email: **clong2@albany.edu**

Outline

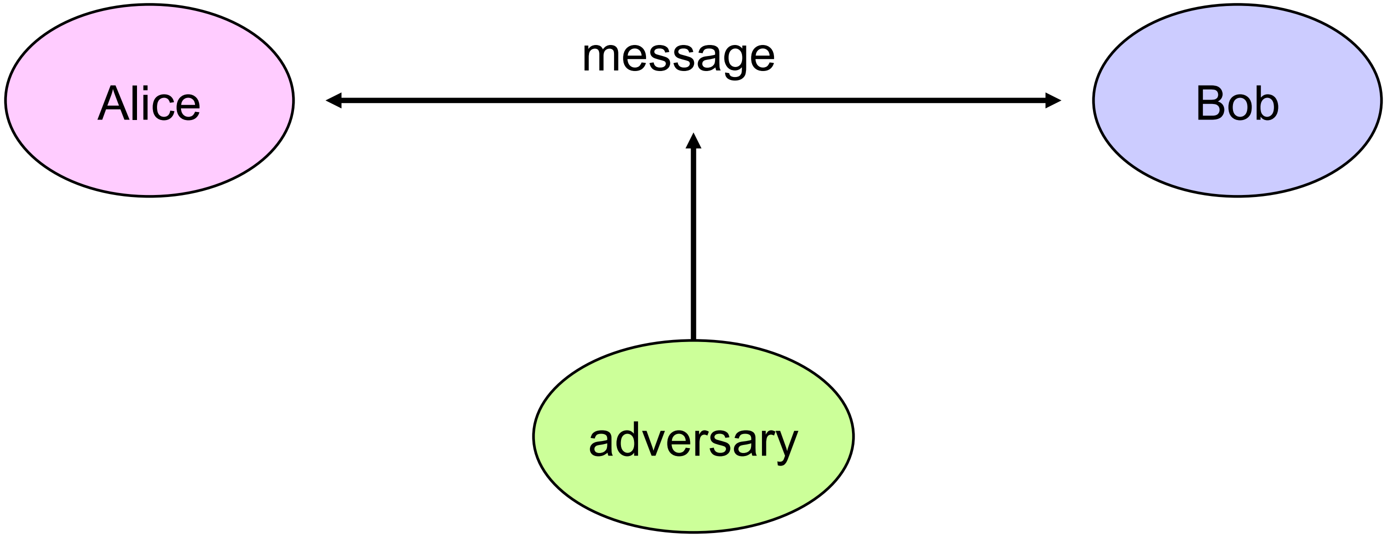
- Introduction to Cryptograph
- Turing Code
- Public Key Cryptography
- RSA Cryptosystem

Outline

- **Introduction to Cryptograph**
- Turing Code
- Public Key Cryptography
- RSA Cryptosystem

Cryptograph

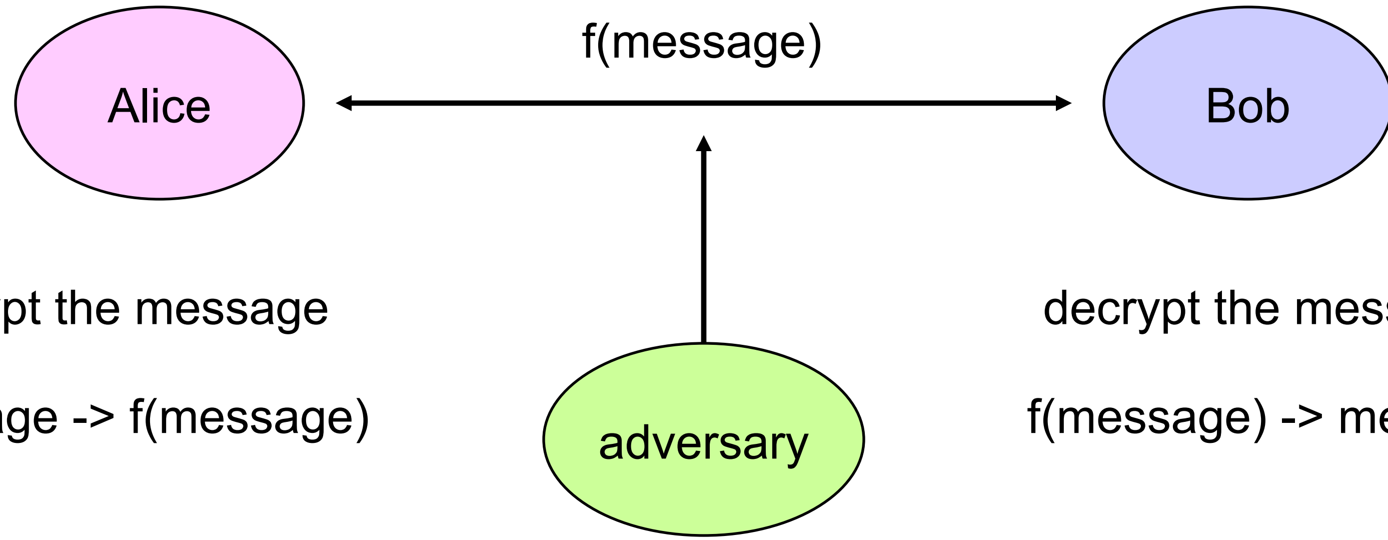
Cryptography is the study of methods for sending and receiving secret messages.



Goal: Even though an adversary can listen to your conversation, the adversary can not learn what the message was.

Cryptograph

Goal: Even though an adversary can listen to your conversation, the adversary can not learn what the message was.



encrypt the message

message -> f(message)

decrypt the message

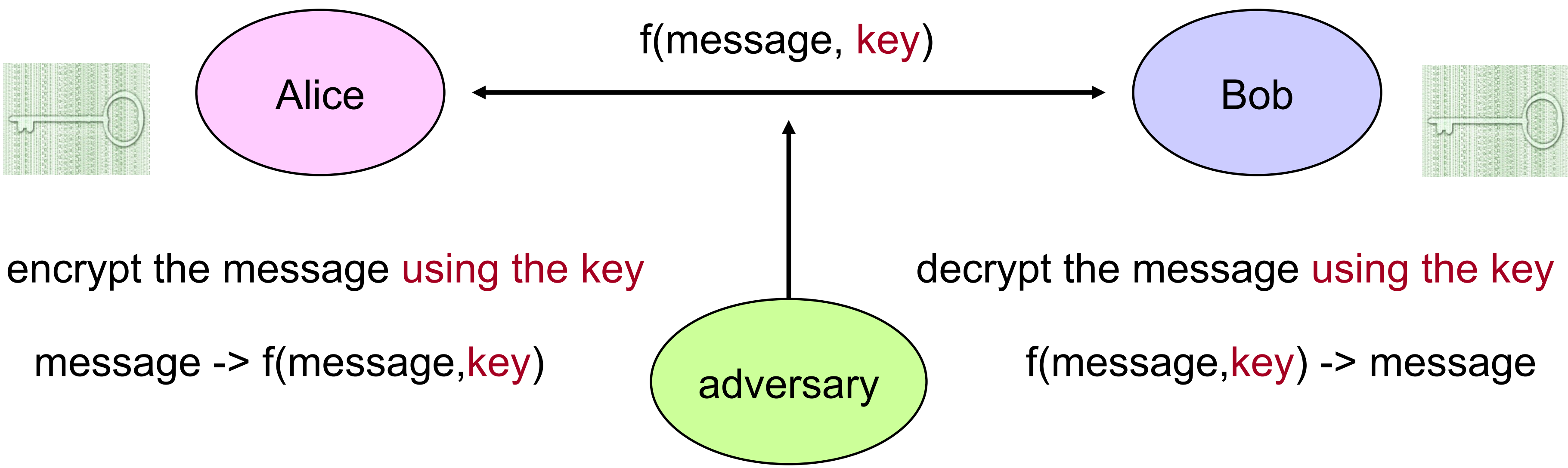
f(message) -> message

But the adversary has no clue how to obtain message from f(message)

A difficult goal!

Cryptograph: Key

Goal: Even though an adversary can listen to your conversation, the adversary can not learn what the message was.



But the adversary can not decrypt f(message,key) without the key

Use number theory!

Outline

- Introduction to Cryptograph
- **Turing Code**
- Public Key Cryptography
- RSA Cryptosystem

Turing's Code (Version 1.0)

The first step is to translate a message into a number

"v i c t o r y"
-> 22 09 03 20 15 18 25

Beforehand The sender and receiver agree on a secret key, which is a large number k.

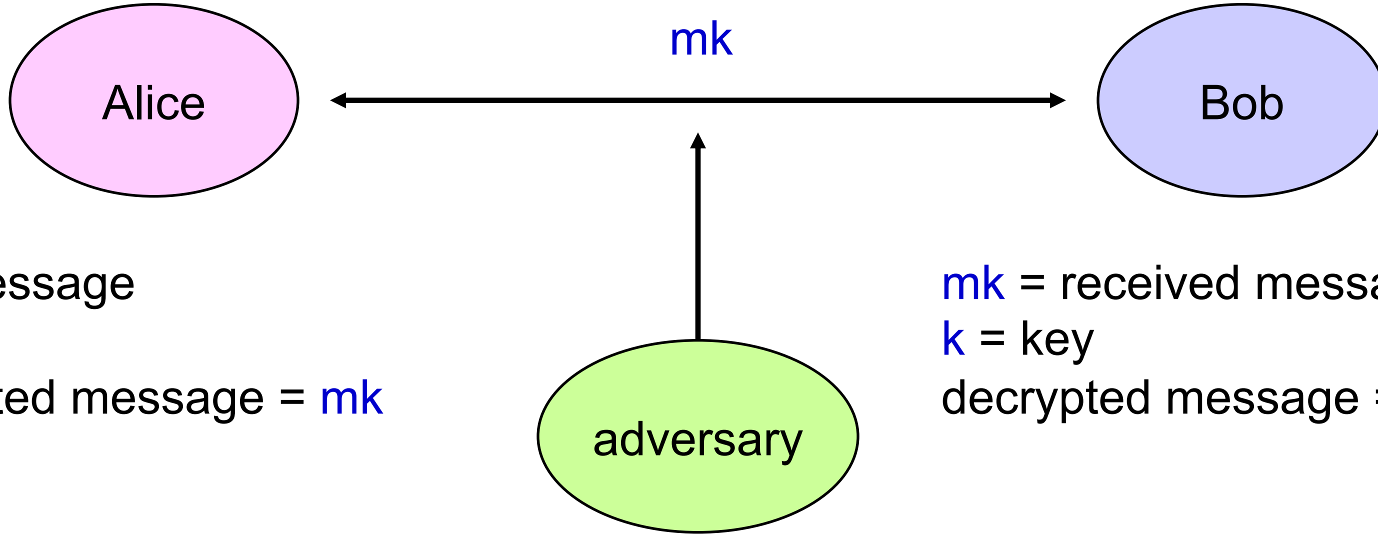
Encryption The sender encrypts the message m by computing:

$$m^* = m \cdot k$$

Decryption The receiver decrypts m by computing:

$$m^*/k = m \cdot k/k = m$$

Turing's Code (Version 1.0)



m = message
k = key
encrypted message = mk

mk = received message
k = key
decrypted message = mk/k=m

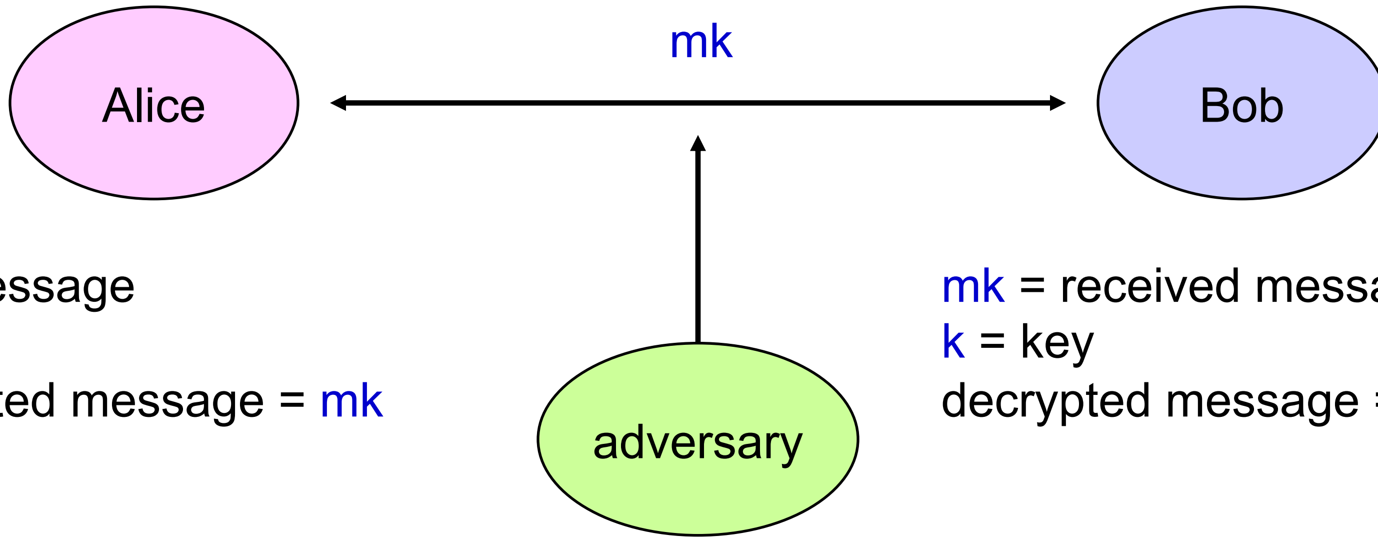
Why the adversary cannot figure out m?

The adversary doesn't have the key k,

and so can only factor mk to figure out m,

but factoring is a difficult task to do.

Turing's Code (Version 1.0)



m = message
k = key
encrypted message = mk

mk = received message
k = key
decrypted message = mk/k=m

So why don't we use this Turing's code today?

Major flaw: if you use the same key to send two messages m and m',

then from mk and m'k,

we can use gcd(mk,m'k) to figure out k,

and then decrypt every message.

Turing's Code (Version 2.0)

Beforehand The sender and receiver agree on a large prime p, which may be made public. (This will be the modulus for all our arithmetic.) They also agree on a secret key k in $\{1, 2, . . . , p - 1\}$.

Encryption The message m can be any integer in the set $\{0, 1, 2, . . . , p - 1\}$. The sender encrypts the message m to produce m^* by computing:

$$m^* = mk \bmod p$$

Decryption Let k' be the multiplicative inverse of k under modulo p.

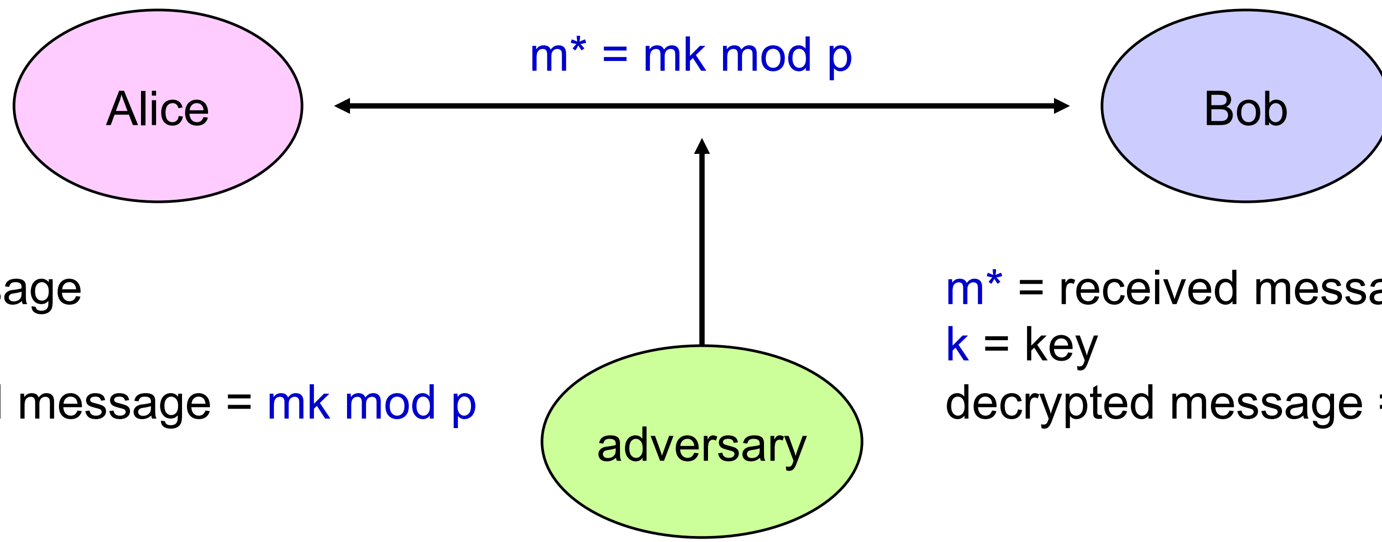
$$m^* \equiv mk \pmod p$$

$$m^*k' \equiv m \pmod p$$

$$m^*k' \equiv m$$

Turing's Code (Version 2.0)

Public information p



m = message
k = key
encrypted message = mk mod p

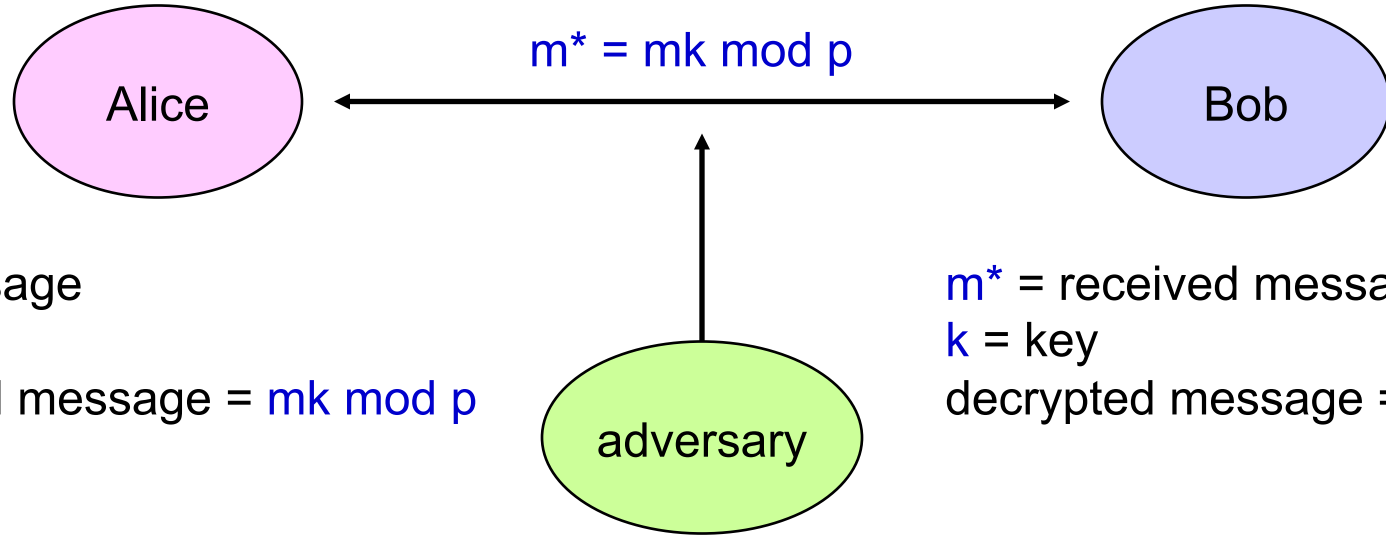
m* = received message
k = key
decrypted message = m*k' =m

Why the adversary cannot figure out m?

Many m and k can produce m* as output,
just impossible to determine m without k.

Turing's Code (Version 2.0)

Public information p



m = message
k = key
encrypted message = mk mod p

m* = received message
k = key
decrypted message = m*k' =m

So why don't we use this Turing's code today?

If the adversary somehow knows m,
then first compute m' := multiplicative inverse of m

$m^* \equiv mk \pmod p$

$m^*m' \equiv k \pmod p$

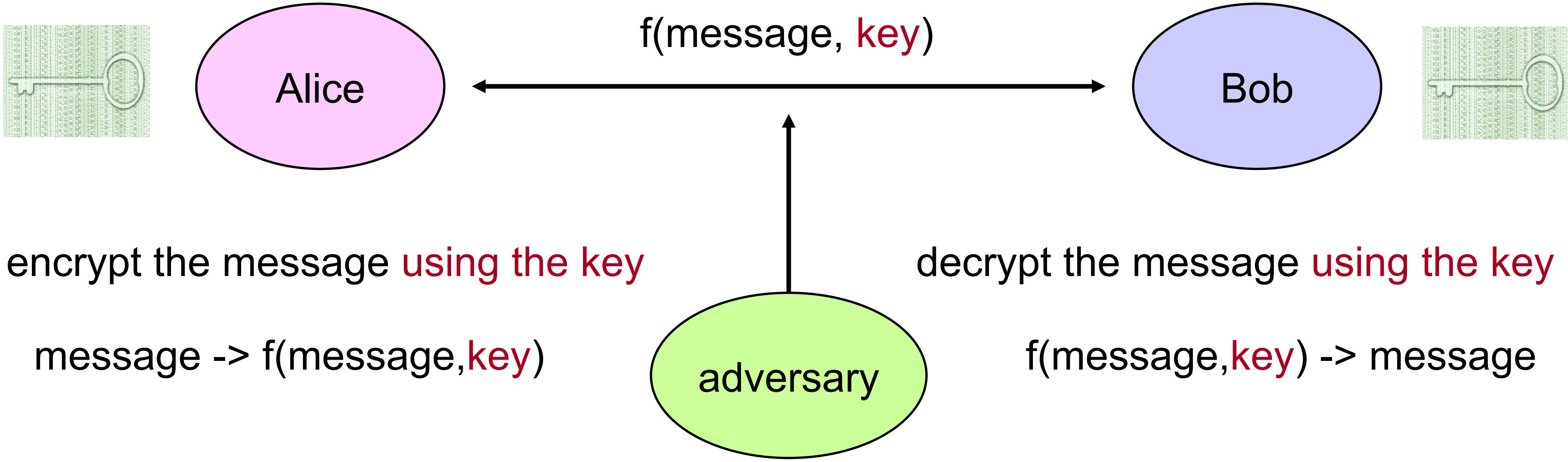
So the adversary can figure out k.

plain-text attack

Outline

- Introduction to Cryptograph
- Turing Code
- **Public Key Cryptography**
- RSA Cryptosystem

Private Key Cryptosystem



But the adversary can not decrypt f(message,key) without the key

Two parties have to agree on a **secret key**, which may be difficult in practice.

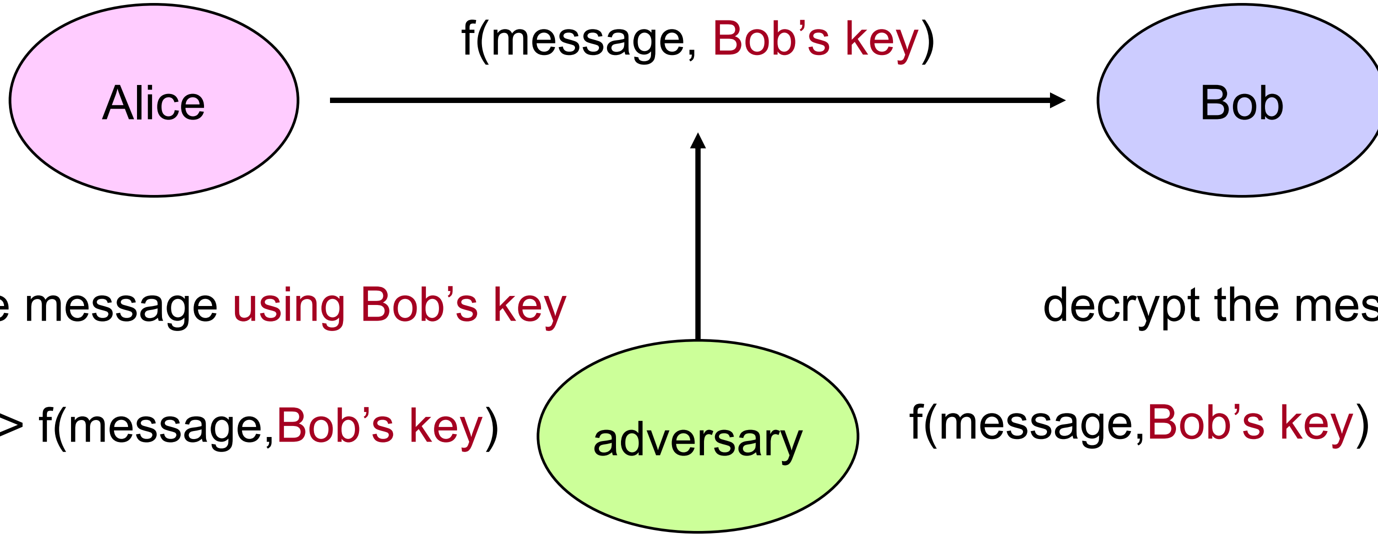
If we buy books from Amazon, we don't need to exchange a secret code.

Why is it secure?

Private Key Cryptosystem

Public information: Key for Alice

Public information: Key for Bob



encrypt the message using Bob's key

message -> f(message,Bob's key)

decrypt the message

f(message,Bob's key) -> message

But the adversary can not decrypt f(message, Bob's key)!

Only Bob can decrypt the message sent to him!

There is no need to have a secret key between Alice and Bob.

How is it possible???

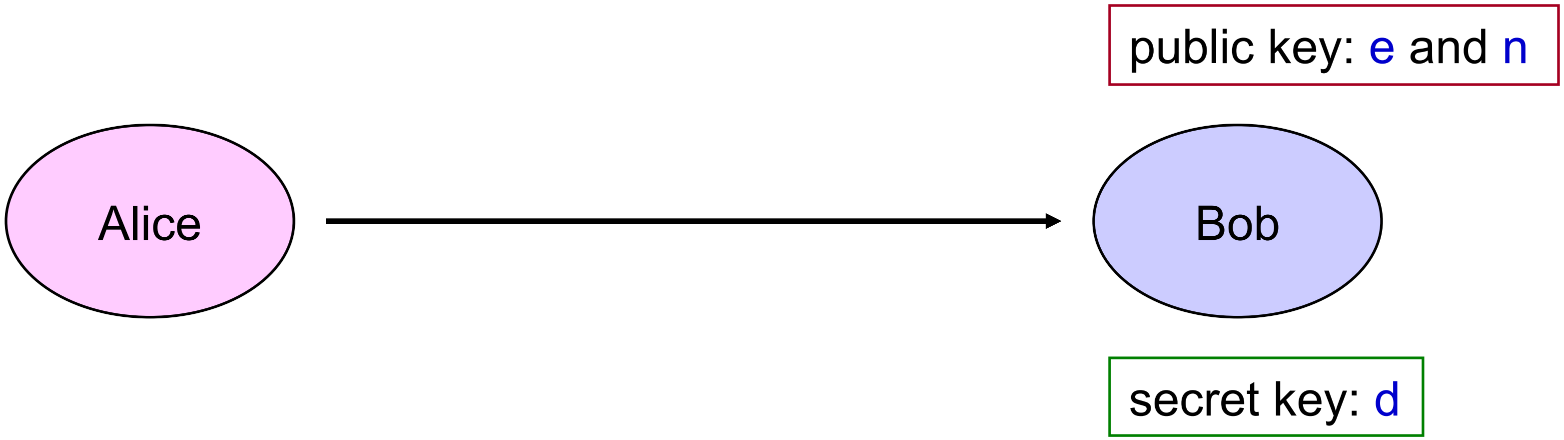
Outline

- Introduction to Cryptograph
- Turing Code
- Public Key Cryptography
- **RSA Cryptosystem**

RSA Cryptosystem

RSA are the initials of three Computer Scientists, Ron Rivest, Adi Shamir and Len Adleman, who discovered their algorithm when they were working together at MIT in 1977.

Generating Public Key



How Bob creates his public keys?

Secret key only known to Bob.

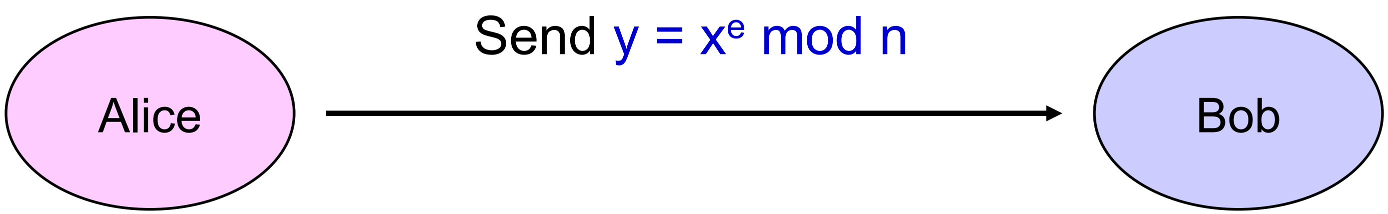
- Choose 2 large prime numbers p and q.
- Set $n = pq$ and $T = (p-1)(q-1)$
- Choose $e \neq 1$ so that $\gcd(e,T)=1$
- Calculate d so that $de \equiv 1 \pmod{T}$
- Publish e and n as public keys
- Keep d as secret key

> 150 digits

Encrypting Message

message x

public key: e and n



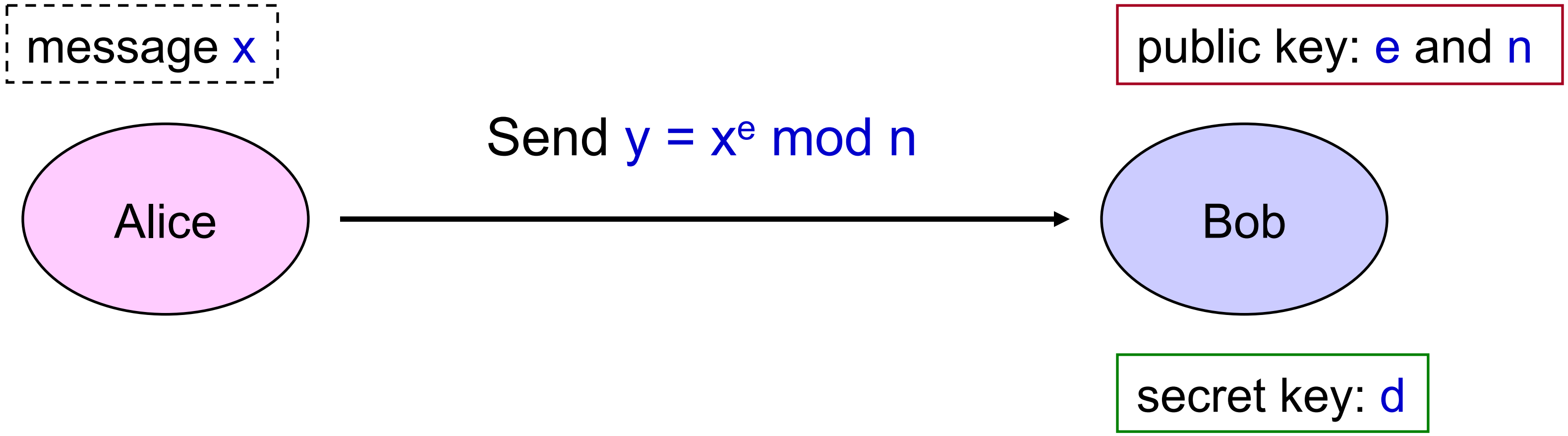
secret key: d

How Alice sends a message to Bob?

- Look at Bob's homepage for e and n.
- Send $y = x^e \bmod n$

Alice does not need to know Bob's secret key to send the message.

Decrypting Message

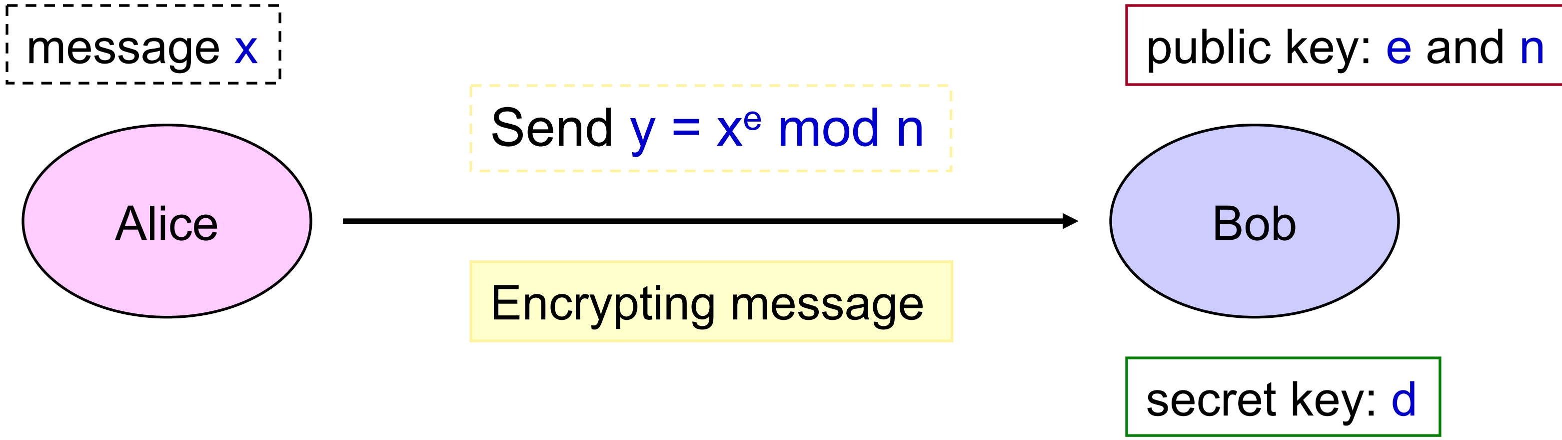


How Bob recover Alice's message?

- Receive $y = x^e \bmod n$
- Compute $z = y^d \bmod n$

Bob uses z is the original message that Alice sent.

RSA Cryptosystem



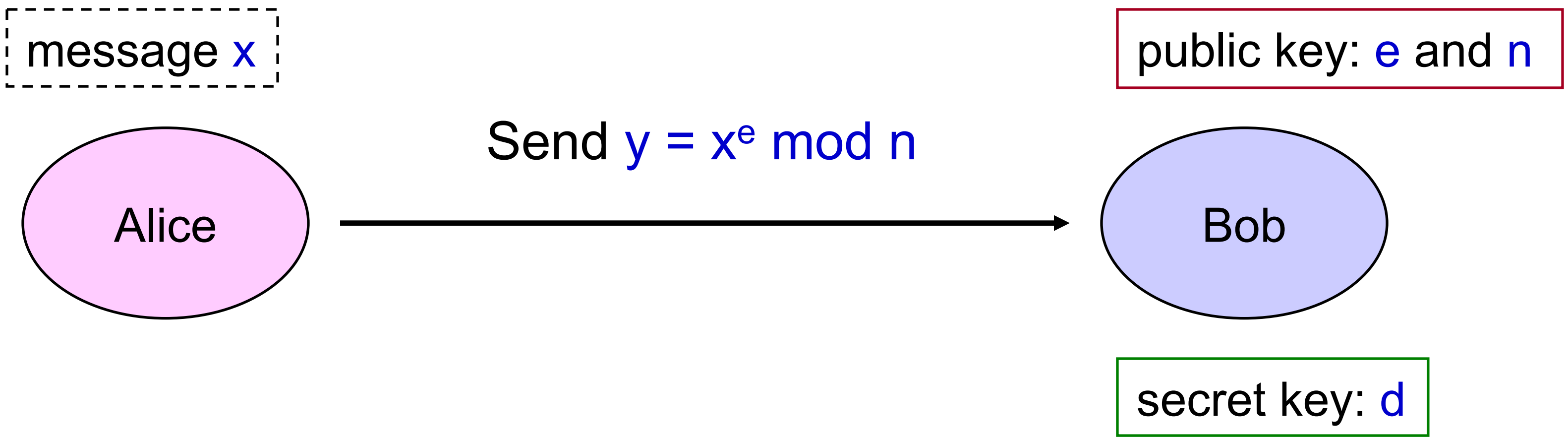
Key generation

- Choose 2 large prime numbers p and q.
- Set $n = pq$ and $T = (p-1)(q-1)$
- Choose $e \neq 1$ so that $\gcd(e,T)=1$
- Calculate d so that $de \equiv 1 \pmod{T}$
- Publish e and n as public keys
- Keep d as secret key

Decrypting message

Compute $z = y^d \bmod n$

RSA Cryptosystem



For the RSA cryptosytem to work,

we need to show:

1) $z = x$

2) Without the secret key d,

we can not compute the original message

before the sun burns out.

Compute $z = y^d \bmod n$

with additional assumptions…

Correctness

message x

Alice — Send $y = x^e \bmod n$ → Bob

public key: e and n

secret key: d

Compute $z = y^d \bmod n$

1) $z = x$

Note that $z = y^d \bmod n = x^{ed} \bmod n$.

Therefore we need to prove $x = x^{ed} \bmod n$.

(a) $x \bmod p = x^{ed} \bmod p$

(b) $x \bmod q = x^{ed} \bmod q$

(c) $x \bmod n = x^{ed} \bmod n$

p, q prime

$n = pq$

$T = (p-1)(q-1)$

e s.t. $\gcd(e,T)=1$

$de \equiv 1 \pmod T$

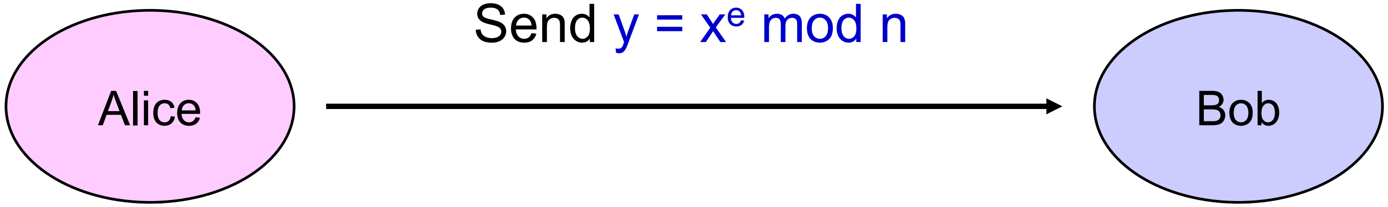
Therefore, if Alice sends $x < n$, then Bob can recover correctly.

Correctness

message x

public key: e and n

Send $y = x^e \bmod n$



secret key: d

1) $z = x$

(a) $x \bmod p = x^{ed} \bmod p$

Compute $z = y^d \bmod n$

Note that $de = 1 + kT \quad = 1 + k(p-1)(q-1)$

Hence, $x^{ed} \bmod p = x^{1+k(p-1)(q-1)} \bmod p$

$= x \cdot x^{k(p-1)(q-1)} \bmod p$

$= x \cdot (x^{k(q-1)})^{(p-1)} \bmod p$

p, q prime

$n = pq$

$T = (p-1)(q-1)$

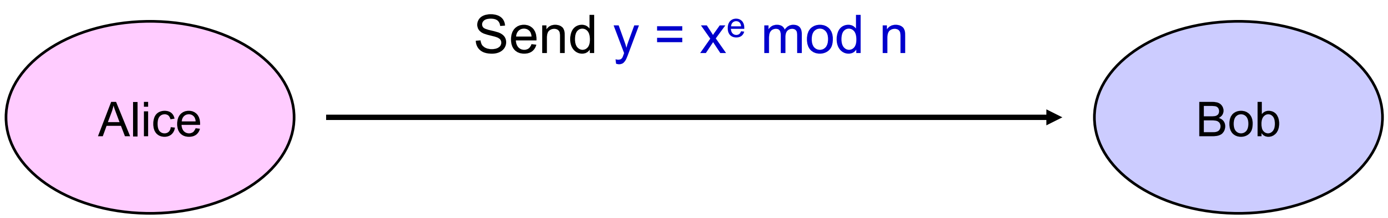
e s.t. $\gcd(e,T)=1$

$de \equiv 1 \pmod{T}$

Correctness

message x

public key: e and n



secret key: d

1) $z = x$

(a) $x \bmod p = x^{ed} \bmod p$

Compute $z = y^d \bmod n$

Fermat's little theorem: If $p \nmid a$, then $a^{p-1} \equiv 1 \bmod p$

Hence, $x^{ed} \bmod p = x^{1+k(p-1)(q-1)} \bmod p$

$= x \cdot x^{k(p-1)(q-1)} \bmod p$

$= x \cdot (\underbrace{x^{k(q-1)}}_{a})^{(p-1)} \bmod p$

$= x \bmod p$

p, q prime

$n = pq$

$T = (p-1)(q-1)$

e s.t. $\gcd(e,T)=1$

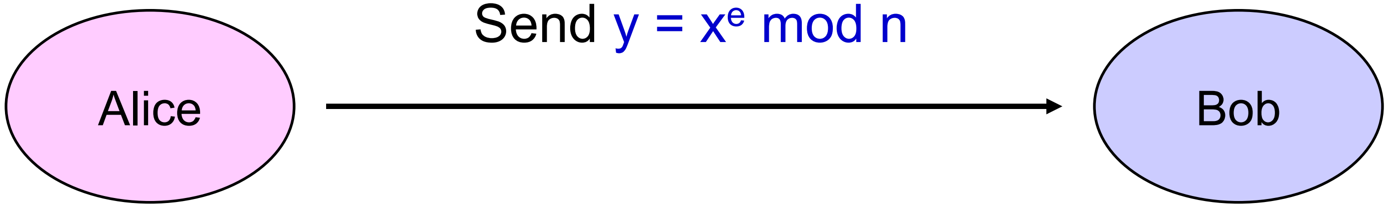
$de \equiv 1 \pmod T$

Correctness

message x

public key: e and n

Send $y = x^e \bmod n$



secret key: d

1) $z = x$

(a) $x \bmod p = x^{ed} \bmod p$

Compute $z = y^d \bmod n$

This means $p \mid x^{k(q-1)}$, implying $p \mid x$, since p is prime

What if $p \mid a$?

Hence, $x^{ed} \bmod p = x^{1+k(p-1)(q-1)} \bmod p$

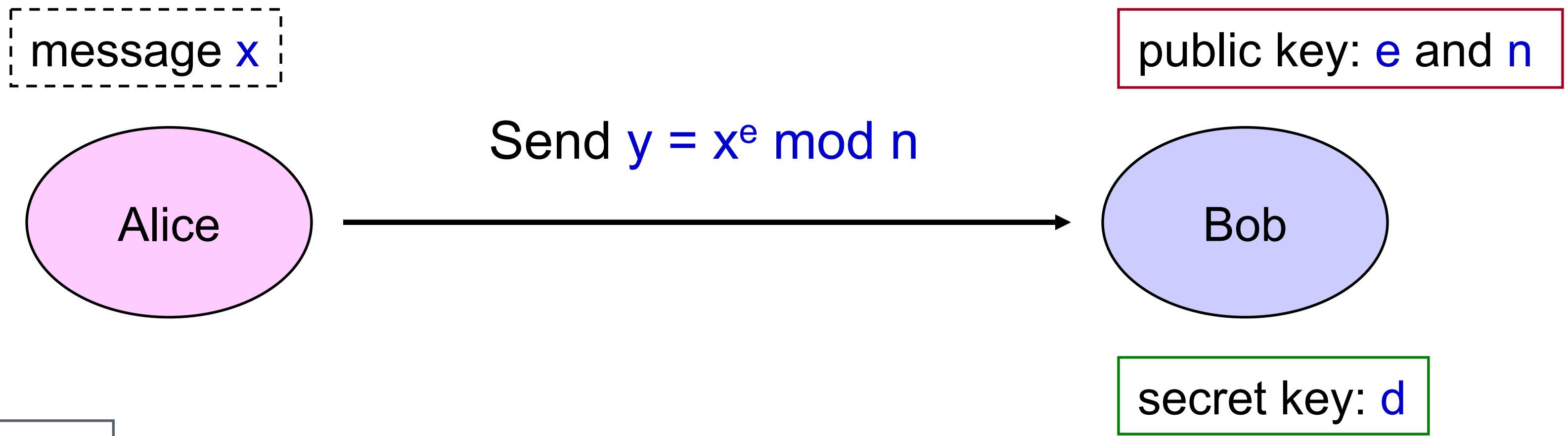
$= x \cdot x^{k(p-1)(q-1)} \bmod p$

$= x \cdot (x^{k(q-1)})^{(p-1)} \bmod p$, where $a = x^{k(q-1)}$

- p, q prime
- $n = pq$
- $T = (p-1)(q-1)$
- e s.t. $\gcd(e,T)=1$
- $de \equiv 1 \pmod{T}$

Since $p \mid x$, we have $x^{ed} \bmod p = x \bmod p = 0$.

Correctness



1) $z = x$

Compute $z = y^d \bmod n$

Note that $z = y^d \bmod n = x^{ed} \bmod n$.

Therefore we need to prove $x = x^{ed} \bmod n$.

(a) $x \bmod p = x^{ed} \bmod p$ ✓

(b) $x \bmod q = x^{ed} \bmod q$ ✓ ← The same proof.

(c) $x \bmod n = x^{ed} \bmod n$ ✓

p, q prime

$n = pq$

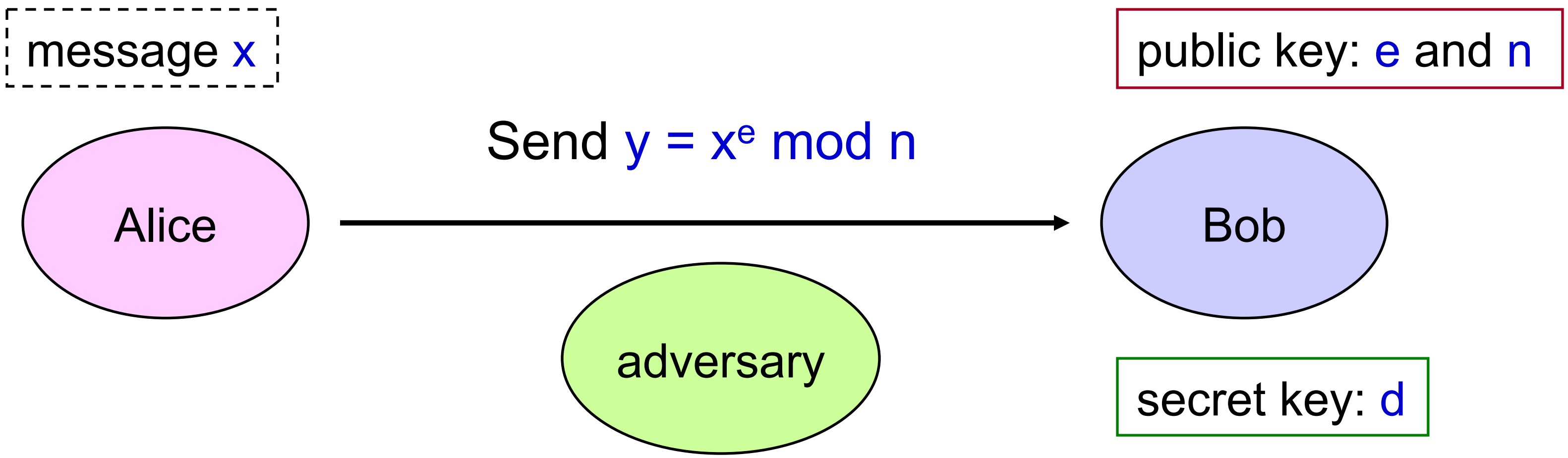
$T = (p-1)(q-1)$

e s.t. $\gcd(e,T)=1$

$de \equiv 1 \pmod T$

(c) can be proved directly, also follows from Chinese Remainder theorem.

Why is This Secure?



2) Without the secret key d,
we can not compute the original message
before the sun burns out.

Compute $z = y^d \bmod n$

p, q prime
$n = pq$
$T = (p-1)(q-1)$
e s.t. $\gcd(e,T)=1$
$de \equiv 1 \pmod{T}$

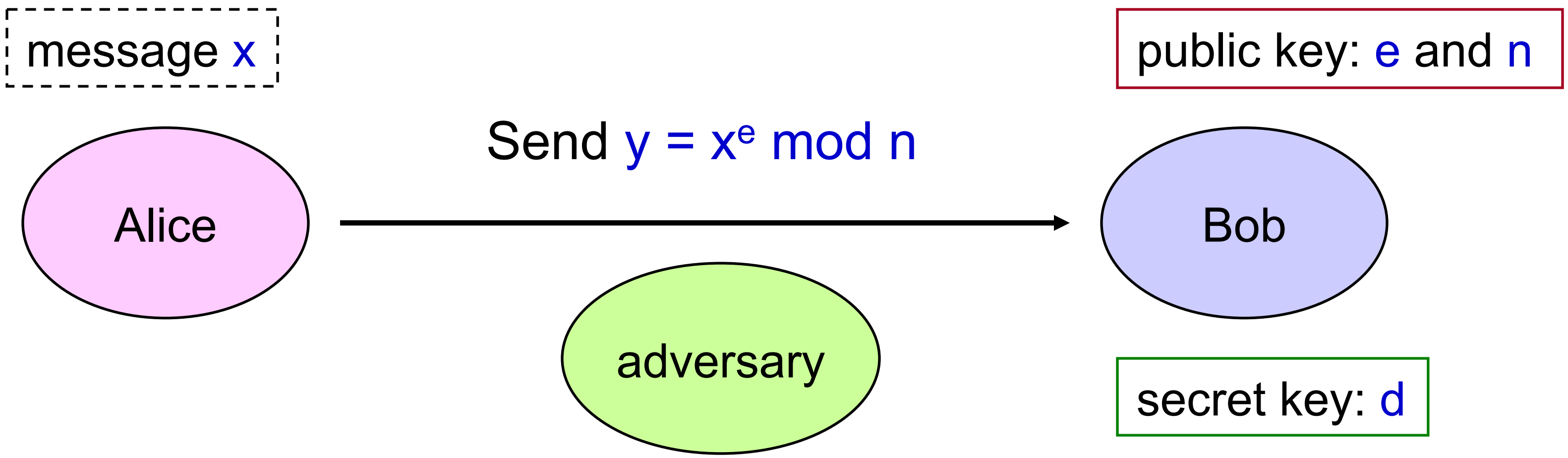
Method 1:

From $y=x^e \bmod n$, don't know how to compute x.

Thus not possible to work backward.

It is an example of an "one-way" function.

Why is This Secure?



2) Without the secret key d,
we can not compute the original message
before the sun burns out.

Compute $z = y^d \bmod n$

p, q prime
$n = pq$
$T = (p-1)(q-1)$
e s.t. $\gcd(e,T)=1$
$de \equiv 1 \pmod{T}$

Method 2:

Factor $n = pq$. Compute secrete key d.

Then decrypt everything!

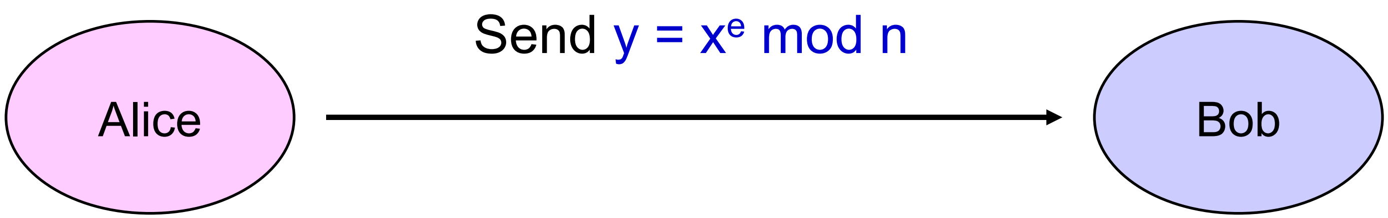
No one knows an efficient way to do factoring.

The security is based on assumptions that some computational problems are hard.

RSA Example

message x

public key: e and n



secret key: d

Then Alice sends the encrypted message.

Compute $z = y^d \bmod n$

$x=33$ $y = 33^{23} \bmod 55$

$y = 84298649517881922539738734663399137 \bmod 55$

How to compute it efficiently?

$p=5$ $q=11$
$n = 55$
$T = 40$
$e = 7$
$d = 23$

→

p, q prime
$n = pq$
$T = (p-1)(q-1)$
e s.t. $\gcd(e,T)=1$
$de \equiv 1 \pmod T$

First Bob generated his keys.

Exponentiation

To compute exponentiation mod n

144^4 mod 713

= 144 * 144 * 144 * 144 mod 713

= 20736 * 144 * 144 mod 713

= 59 * 144 * 144 mod 713

= 8496 * 144 mod 713

= 653 * 144 mod 713

= 94032 mod 713

= 629 mod 713

This still takes too long when the exponent is large.

→

20736 * 20736 mod 713

= 59 * 59 mod 713

= 3481 mod 713

= 629 mod 713

This is much more efficient.

Repeated Squaring

Note that 50 = 32 + 16 + 2

$144^{50} \bmod 713$

$= 144^{32}\, 144^{16}\, 144^{2} \bmod 713$

$= 648 \cdot 485 \cdot 59 \bmod 713$

$= 242$

$144^{2} \bmod 713 = 59$

$144^{4} \bmod 713$
$= 144^{2} \cdot 144^{2} \bmod 713$
$= 59 \cdot 59 \bmod 713$
$= 629$

$144^{8} \bmod 713$
$= 144^{4} \cdot 144^{4} \bmod 713$
$= 629 \cdot 629 \bmod 713$
$= 639$

$144^{16} \bmod 713$
$= 144^{8} \cdot 144^{8} \bmod 713$
$= 639 \cdot 639 \bmod 713$
$= 485$

$144^{32} \bmod 713$
$= 144^{16} \cdot 144^{16} \bmod 713$
$= 485 \cdot 485 \bmod 713$
$= 648$

Remarks

- We have derived everything from basic principle.
- RSA cryptosystem is one of the most important achievements in compute science.
  (The researchers won the Turing award for their contribution.)
- Number theory is also very useful in coding theory (e.g. compression).
- Mathematics is very important in computer science.

Remarks

Theorem: if n is composite, for more than half of $a < n$,
the strong primality test will say n is composite!

The proof uses Chinese Remainder theorem and some elementary number theory. (Introduction to Algorithms, MIT press)

Conjecture: It is enough to try a to up to roughly $\log(n)$.

Theroem (Primes is in P, 2004)
There is an efficient and deterministic primality test.

Major Open Problem:
Is there an efficient algorithm to compute the prime factorization?

Next class

- Topic: Proof by Mathematical Induction
- Pre-class reading: Chap 5.1