DTS203TC
Design and Analysis of Algorithms

Lecture 19: Number Theory

Dr. Qi Chen
School of AI and Advanced Computing

Learning Outcome

- Powers of an Element
- RSA public-key cryptography

Powers of an Element

- Consider the sequence of powers of a, modulo n where $a \in Z_n^*$.

For example,

i	0	1	2	3	4	5	6	7	8	9
3^i mod 7	1	3	2	6	4	5	1	3	2	6

i	0	1	2	3	4	5	6	7	8	9
2^i mod 7	1	2	4	1	2	4	1	2	4	1

Now, $< 2 > \;=\; \{1, 2, 4\}$ in $\mathbf{Z}_7^*$

$< 3 > \;=\; \{1, 3, 2, 6, 4, 5\}$ in $\mathbf{Z}_7^*$

Here, $\mathrm{ord}_7\,(2) = 3$ & $\mathrm{ord}_7\,(3) = 6$

Euler's theorem

For any integer n > 1,

$$a^{\phi(n)} \equiv 1\ (mod\ n) \text{ for all } a \in Z_n^*$$

Exercise:

Use Euler's theorem to calculate 7^{133} mod 26.

$$\phi(26) = 26 * \left(1 - \frac{1}{2}\right) * \left(1 - \frac{1}{13}\right) = 12$$

So $\quad 7^{12} \equiv 1\ (mod\ 26)$

Thus $\quad 7^{133} \equiv (7^{12})^{11} \times 7 \equiv 7\ (mod\ 26)$

Fermat's Theorem

For p is a prime, then

$$a^{p-1} \equiv 1\ (mod\ p) \text{ for all } a \in Z_p^*$$

Note that if p is a prime, then $\phi(p) = p - 1$

Modular Exponentiation

- Modular exponentiation:
  - A frequently occurring operation in number-theoretic computations is raising one number to a power modulo another number.

- **Repeated squaring** solves $a^b \bmod n$ efficiently using the binary representation of b.

Repeated Squaring for Modular Exponentiation

```
Modular-Exponentiation(a,b,n)
c = 0
d = 0
Let <b_k, b_{k-1} ,…, b_0 > be the binary representation of b
for i = k down to 0
    c = 2c
    d = (d·d) mod n
    if b_i == 1
        c = c+1
        d = (d·a) mod n
return d
```

Repeated Squaring for Modular Exponentiation

- Let's find the value of:

  $7^{560} \bmod 561$

```
Modular-Exponentiation(a,b,n)
c = 0
d = 1
Let <b_k, b_k-1 ,..., b_0 > be the binary representation of b
for i = k down to 0
    c = 2c
    d = (d·d) mod n
    if b_i == 1
        c = c+1
        d = (d·a) mod n
return d
```

i	9	8	7	6	5	4	3	2	1	0
b_i										
c										
d										

Repeated Squaring for Modular Exponentiation

- Let's find the value of:

  $7^{560} \bmod 561$

Binary Representation

$a = 7$

$b = 560 = \langle 1000110000 \rangle$

$n = 561$

```
Modular-Exponentiation(a,b,n)
c = 0
d = 1
Let <b_k, b_k-1 ,..., b_0 > be the binary representation of b
for i = k down to 0
    c = 2c
    d = (d·d) mod n
    if b_i == 1
        c = c+1
        d = (d·a) mod n
return d
```

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c										
d										

Repeated Squaring for Modular Exponentiation

- Let's find the value of:

  $7^{560} \bmod 561$

```
Modular-Exponentiation(a,b,n)
c = 0
d = 1
Let <b_k, b_{k-1} ,…, b_0 > be the binary representation of b
for i = k down to 0
    c = 2c
    d = (d·d) mod n
    if b_i == 1
        c = c+1
        d = (d·a) mod n
return d
```

i = 9

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1									
d	7									

Repeated Squaring for Modular Exponentiation

- Let's find the value of:

  $7^{560} \bmod 561$

$i = 8$

```
Modular-Exponentiation(a,b,n)
c = 0
d = 1
Let <b_k, b_{k-1} ,..., b_0 > be the binary representation of b
for i = k down to 0
    c = 2c
    d = (d·d) mod n
    if b_i == 1
        c = c+1
        d = (d·a) mod n
return d
```

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2								
d	7	49								

Repeated Squaring for Modular Exponentiation

- Let's find the value of:

  $7^{560} \bmod 561$

```
Modular-Exponentiation(a,b,n)
c = 0
d = 1
Let <b_k, b_k-1 ,..., b_0 > be the binary representation of b
for i = k down to 0
    c = 2c
    d = (d·d) mod n
    if b_i == 1
        c = c+1
        d = (d·a) mod n
return d
```

i = 7

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4							
d	7	49	157							

Repeated Squaring for Modular Exponentiation

- Let's find the value of:

  7^{560} mod 561

i = 6

```
Modular-Exponentiation(a,b,n)
c = 0
d = 1
Let <b_k, b_{k-1} ,..., b_0 > be the binary representation of b
for i = k down to 0
    c = 2c
    d = (d·d) mod n
    if b_i == 1
        c = c+1
        d = (d·a) mod n
return d
```

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8						
d	7	49	157	526						

Repeated Squaring for Modular Exponentiation

- Let's find the value of:

  $7^{560} \bmod 561$

```
Modular-Exponentiation(a,b,n)
c = 0
d = 1
Let <b_k, b_k-1 ,…, b_0 > be the binary representation of b
for i = k down to 0
    c = 2c
    d = (d·d) mod n
    if b_i == 1
        c = c+1
        d = (d·a) mod n
return d
```

i = 5

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17					
d	7	49	157	526	160					

Repeated Squaring for Modular Exponentiation

- Let's find the value of:

  $7^{560} \bmod 561$

$i = 4$

```
Modular-Exponentiation(a,b,n)
c = 0
d = 1
Let <b_k, b_k-1 ,..., b_0 > be the binary representation of b
for i = k down to 0
    c = 2c
    d = (d·d) mod n
    if b_i == 1
        c = c+1
        d = (d·a) mod n
return d
```

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35				
d	7	49	157	526	160	241				

Repeated Squaring for Modular Exponentiation

- Let's find the value of:

  7^{560} mod 561

i = 3

```
Modular-Exponentiation(a,b,n)
c = 0
d = 1
Let <b_k, b_{k-1} ,…, b_0 > be the binary representation of b
for i = k down to 0
    c = 2c
    d = (d·d) mod n
    if b_i == 1
        c = c+1
        d = (d·a) mod n
return d
```

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70			
d	7	49	157	526	160	241	298			

Repeated Squaring for Modular Exponentiation

- Let's find the value of:

  $7^{560} \bmod 561$

```
Modular-Exponentiation(a,b,n)
c = 0
d = 1
Let <b_k, b_{k-1} ,…, b_0 > be the binary representation of b
for i = k down to 0
    c = 2c
    d = (d·d) mod n
    if b_i == 1
        c = c+1
        d = (d·a) mod n
return d
```

$i = 2$

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140		
d	7	49	157	526	160	241	298	166		

Repeated Squaring for Modular Exponentiation

- Let's find the value of:

  7^{560} mod 561

i = 1

```
Modular-Exponentiation(a,b,n)
c = 0
d = 1
Let <b_k, b_{k-1} ,..., b_0 > be the binary representation of b
for i = k down to 0
    c = 2c
    d = (d·d) mod n
    if b_i == 1
        c = c+1
        d = (d·a) mod n
return d
```

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	
d	7	49	157	526	160	241	298	166	67	

Repeated Squaring for Modular Exponentiation

- Let's find the value of:

  7^{560} mod 561

i = 0

```
Modular-Exponentiation(a,b,n)
c = 0
d = 1
Let <b_k, b_{k-1} ,…, b_0 > be the binary representation of b
for i = k down to 0
    c = 2c
    d = (d·d) mod n
    if b_i == 1
        c = c+1
        d = (d·a) mod n
return d
```

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
d	7	49	157	526	160	241	298	166	67	1

Repeated Squaring for Modular Exponentiation

- Let's find the value of:

  7^{560} mod 561

The final result is 1.

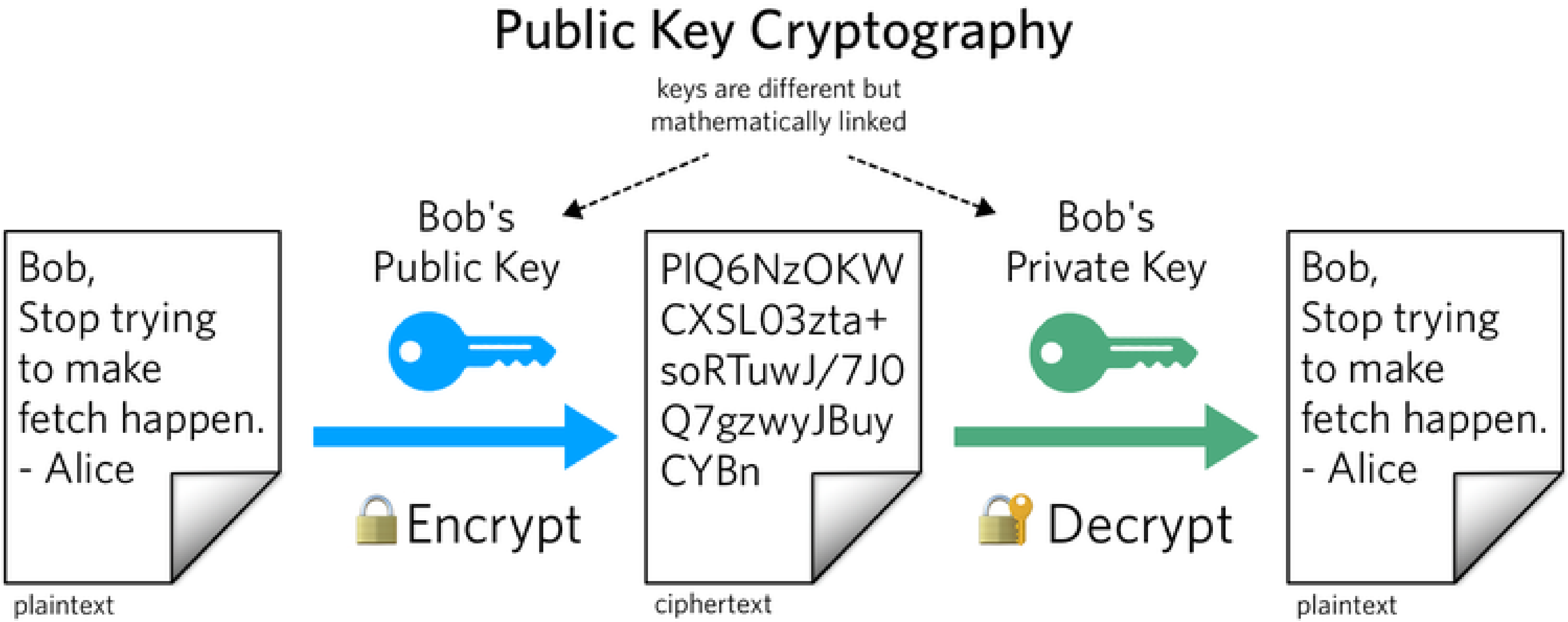
```
Modular-Exponentiation(a,b,n)
c = 0
d = 1
Let <b_k, b_k-1 ,…, b_0 > be the binary representation of b
for i = k down to 0
    c = 2c
    d = (d·d) mod n
    if b_i == 1
        c = c+1
        d = (d·a) mod n
return d
```

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
d	7	49	157	526	160	241	298	166	67	1

RSA public-key cryptography

Cryptography

Public-key cryptography



Public-key cryptography

- In public-key cryptography, each entity has two keys:
  - Public Key — to be shared
  - Private Key — to be kept secret

- If Bob want to send a message M to Alice. Here, we denote the public key and private key as P_A, S_A for Alice. Then we should have:
  - $M = S_A(P_A(M))$
  - At the same time, we should have $M = P_A(S_A(M))$

Public-key cryptography

- The scenario for sending the message goes as follows.
  - Bob obtains Alice's public key P_A (from a public directory or directly from Alice).
  - Bob computes the ciphertext $C=P_A(M)$ corresponding to the message M and sends C to Alice.
  - When Alice receives the ciphertext C, she applies her private key (secret key) S_A to retrieve the original message: $S_A(C)=S_A(P_A(M))=M$
- S_A and P_A are inverse functions.

RSA cryptosystem

- In the RSA public-key cryptosystem, the public and private keys are generated as follows :
  1. Select at random two large prime numbers p and q such that $p \neq q$.
  2. Compute n = pq.
  3. Select a small odd integer e that is relatively prime to $\phi(n)$.
  4. Compute d as the multiplicative inverse of e modulo $\phi(n)$.
  5. Publish the pair P = (e, n) as the participant's RSA public key.
  6. Keep secret the pair S = (d, n) as the participant's RSA private key.

Here, P(M) = ENCRYPT(M) = M^e mod n

S(C) = DECRYPT(C) = C^d mod n

RSA example

- Step 1: Select at random two large prime numbers p and q such that $p \neq q$.

- Let's choose $p = 7$, $q = 19$

RSA example

- Step 2: Compute n = pq

- Let's choose p = 7, q = 19
- n = 7*19 = 133

RSA example

- Step 3: Select a small odd integer e that is relatively prime to $\phi(n)$.

- p = 7, q = 19, n = 7*19 = 133
- $\phi(133) = (p-1)(q-1) = (7-1)(19-1) = 108$

- For example, we choose e=29

RSA example

- Step 4: Compute d as the multiplicative inverse of e modulo $\phi(n)$.

- p = 7, q = 19, n =133, e=29, $\phi(133)$=108
- Compute 29^{-1} mod 108

Exercise

```
EXTENDED-EUCLID(a, b)
If b = = 0
   return (a,1,0)
else
   (d', x', y')  =  EXTENDED-EUCLID(b, a mod b)
   (d, x, y) = (d', y', x' - ⌊a/b⌋y')
   return   (d, x, y)
```

gcd(29, 108)=gcd(108, 29)

$29^{-1} \equiv 41 \pmod{108}$

a	b	$\lfloor a/b \rfloor$	d	x	y
108	29	3	1	-11	41
29	21	1	1	8	-11
21	8	2	1	-3	8
8	5	1	1	2	-3
5	3	1	1	-1	2
3	2	1	1	1	-1
2	1	2	1	0	1
1	0	-	1	1	0

RSA example

- Step 5: Publish the pair $P = (e, n)$ as the participant's RSA public key

- $p = 7$, $q = 19$, $n = 133$, $e = 29$, $\phi(133) = 108$, $d = 41$

- Publish $P = (29, 133)$

RSA example

- Step 5: Keep secret the pair S = (d, n) as the participant's RSA private key.

- p = 7, q = 19, n =133, e=29, $\phi(133)$=108, d = 41

- Keep S = (41, 133)

RSA example

- If Bob want to send 99 to Alice, what is the cipher text?

$$P(M) = \text{ENCRYPT}(M) = M^e \bmod n$$

$$S(C) = \text{DECRYPT}(C) = C^d \bmod n$$

- p = 7, q = 19, n =133, e=29, $\phi(133)$=108, d = 41

- Compute: $99^{29} \bmod 133$

RSA example

- Compute: $99^{29} \bmod 133$

$a = 99$, $b = 29 = \langle 11101 \rangle$

$n = 133$

- Final Result: 92

Modular-Exponentiation(a,b,n)
c = 0
d = 1
Let $\langle b_k, b_{k-1}, \ldots, b_0 \rangle$ be the binary representation of b
for i = k down to 0
    c = 2c
    d = (d·d) mod n
    if b_i == 1
        c = c+1
        d = (d·a) mod n
return d

i	4	3	2	1	0
b_i	1	1	1	0	1
c	1	3	7	14	29
d	99	64	120	36	92

RSA example

- If Bob sent 92 to Alice, what is the original message?

$$P(M) = \text{ENCRYPT}(M) = M^e \bmod n$$

$$S(C) = \text{DECRYPT}(C) = C^d \bmod n$$

- p = 7, q = 19, n =133, e=29, $\phi(133)$=108, d = 41
- Compute: $92^{41} \bmod 133$

RSA example

- Compute: $92^{41} \bmod 133$

$a = 92$, $b = 41 = \langle 101001 \rangle$

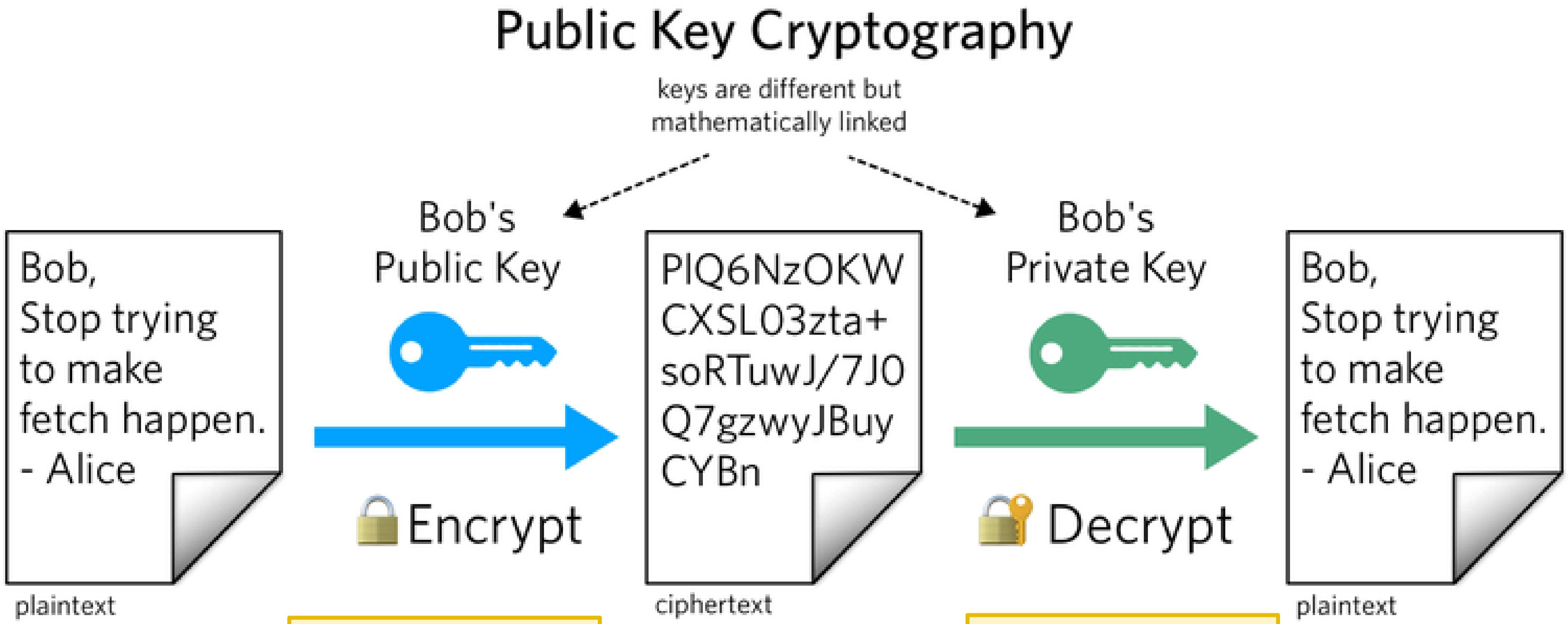
$n = 133$

- Final Result: 99

```
Modular-Exponentiation(a,b,n)
c = 0
d = 1
Let <b_k, b_{k-1} ,..., b_0 > be the binary representation of b
for i = k down to 0
    c = 2c
    d = (d·d) mod n
    if b_i == 1
        c = c+1
        d = (d·a) mod n
return d
```

i	5	4	3	2	1	0
b_i	1	0	1	0	0	1
c	1	2	5	10	20	41
d	92	85	99	92	85	99

RSA example



99

P=(29,133)
99^{29} mod 133

92

S=(41,133)
92^{41} mod 133

99

RSA correctness

- We want to show $M^{ed} \equiv M \ (mod\ n)$

- Since e and d are multiplicative inverses modulo $\phi(n)$,

  ed = 1 + k(p-1)(q-1) for some integer k

- Then, $M^{ed} \equiv M(M^{p-1})^{k(q-1)} \quad (mod\ p)$

  $\equiv M(M^{p-1}\ mod\ p)^{k(q-1)} \quad (mod\ p)$

  $\equiv M(1)^{k(q-1)} \quad (mod\ p)$ ← Fermat's Theorem

  $\equiv M \quad (mod\ p)$

RSA correctness

- We have $M^{ed} \equiv M \ (mod\ p)$
- Similarly, $M^{ed} \equiv M \ (mod\ q)$

- According to Chinese remainder theorem (Please refer to textbook chapter 31.5).

  We get $M^{ed} \equiv M \ (mod\ n)$

RSA

- Why it's hard to find private key given public key?

- The RSA public-key cryptosystem relies on the dramatic difference between
  - the ease of finding large prime numbers
  - the difficulty of factoring the product of two large prime numbers.
    - Even with today's supercomputers and the best algorithms to date, we cannot feasibly factor an arbitrary 1024-bit number.

Exercise

The RSA Encryption Scheme is often used to encrypt and decrypt electronic communications. Suppose Alice wants her friends to encrypt email messages before sending them to her. Alice chooses two prime numbers p=7 and q=11, and publishes her public key (e,n)=(13,77).

i. Use Euler's totient function φ to count the number of integers between 1 and 76 which are relatively prime to 77.

ii. Use the extended Euclidean algorithm to find Alice's private key (d,n).

iii. Alice received an encrypted number 25 from Bob. What is the original integer chosen by Bob? Use repeated squaring method to find the integer.

Exercise

The RSA Encryption Scheme is often used to encrypt and decrypt electronic communications. Suppose Alice wants her friends to encrypt email messages before sending them to her. Alice chooses two prime numbers p=7 and q=11, and publishes her public key (e,n)=(13,77).

i. Use Euler's totient function φ to count the number of integers between 1 and 76 which are relatively prime to 77.

60

ii. Use the extended Euclidean algorithm to find Alice's private key (d,n).

(37,77)

iii. Alice received an encrypted number 25 from Bob. What is the original integer chosen by Bob? Use repeated squaring method to find the integer.

53

Learning Outcome

- Powers of an Element
- RSA public-key cryptosystem