ASYMMETRIC CIPHERS

Contents

1) Principles Of Public-Key Cryptosystems

2) RSA Algorithm

1. Principles Of Public-Key Cryptosystems

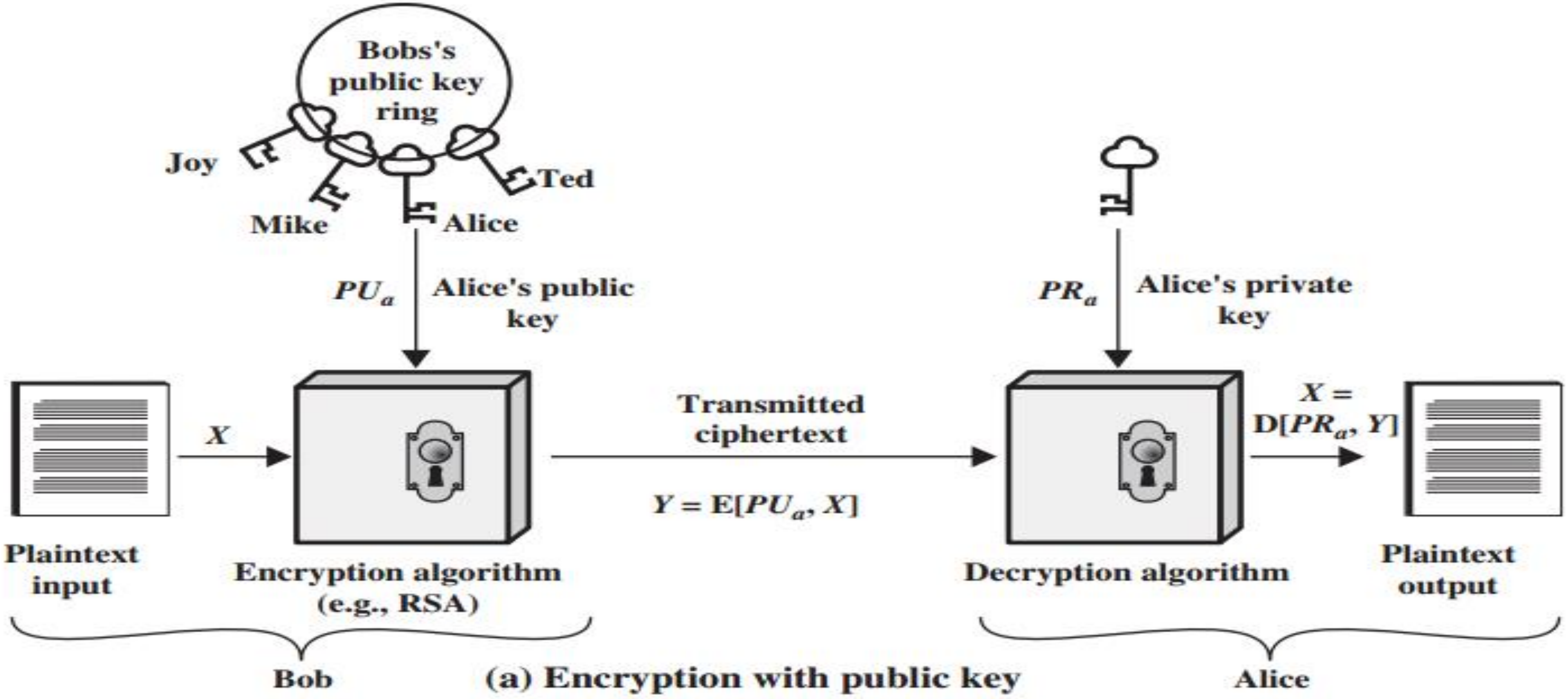
1. Principles Of Public-Key Cryptosystems

- Commonly know as public key cryptography
- Invented by Whitfield Diffie and Martin Hellman in 1976
- Uses a pair of key
  - A private key that is kept secret
  - A public key that can be sent to anyone

Public-Key Cryptosystems

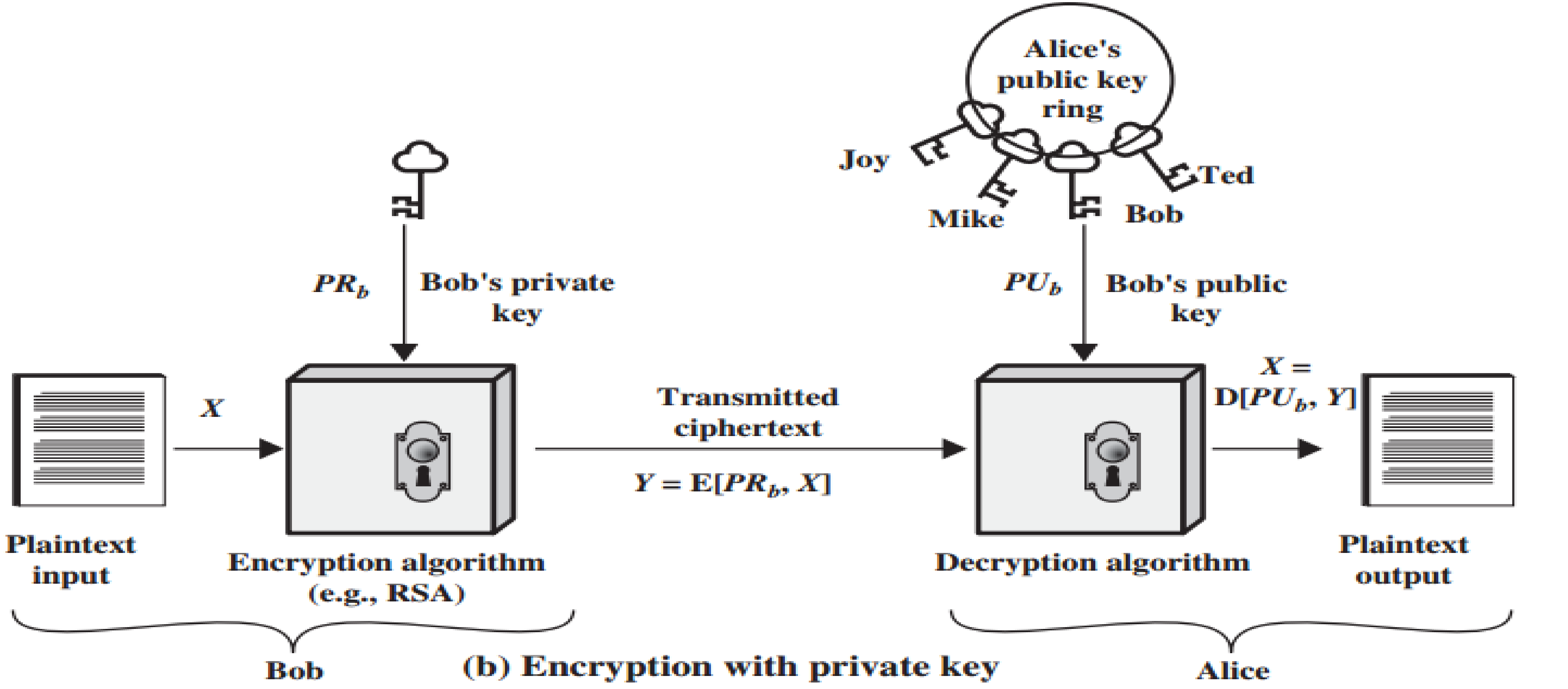
- Asymmetric algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristic.
  - It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.
  - Either of the two related keys can be used for encryption, with the other used for decryption.

Encryption with public key



(a) Encryption with public key

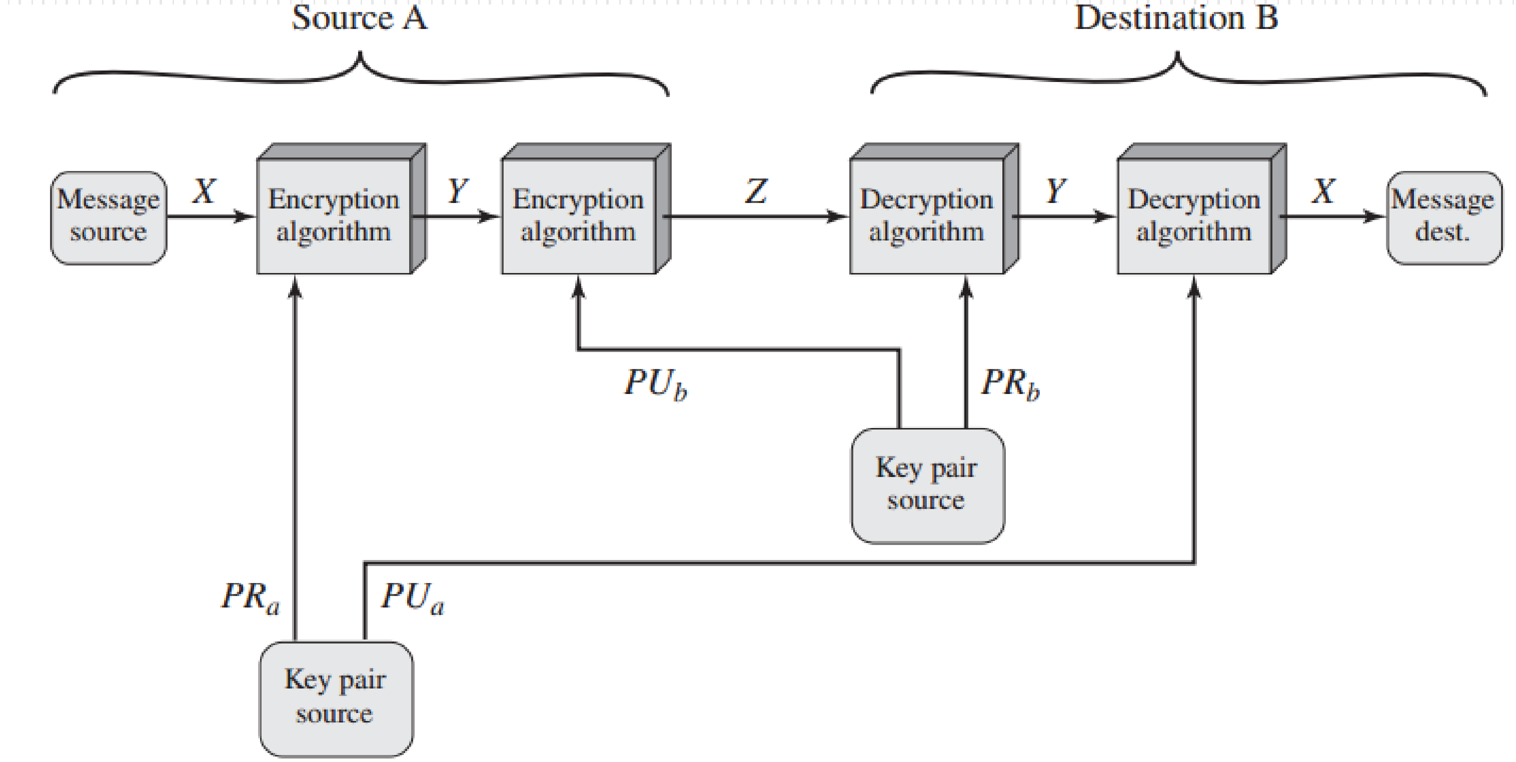
Encryption with private key



(b) Encryption with private key

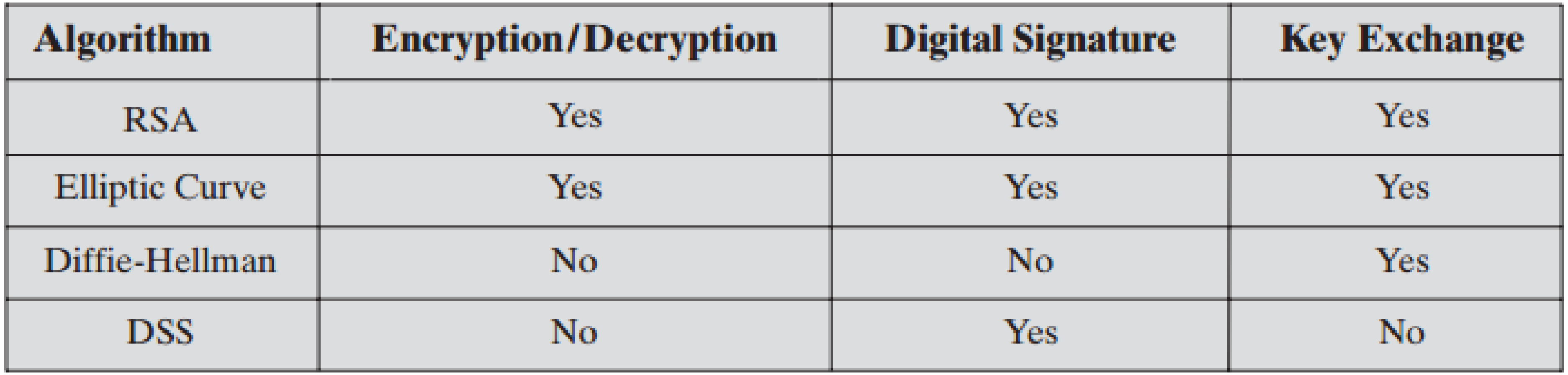
Authentication and confidentiality

- possible to provide both the authentication function and confidentiality by a double use of the public-key.
- $Z=E(PU_b,E(PR_a,X))$
- $X=D(PU_a,D(PR_b,Z))$



Applications for Public-Key Cryptosystems

- **Encryption/decryption:** The sender encrypts a message with the recipient's public key.
- **Digital signature:** The sender "signs" a message with its private key.
- **Key exchange:** Two sides cooperate to exchange a session key.



Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Requirements for Public-Key Cryptography

- It is computationally easy for a party B to generate a pair.
- It is computationally easy for a sender A, knowing the public key and the message to be encrypted,M, to generate the corresponding ciphertext.

$$C=E(PUb,M)$$

- It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:

Requirements for Public-Key Cryptography

- It is computationally infeasible for an adversary, knowing the public key,PU_b,to determine the private key,PR_b.
- It is computationally infeasible for an adversary, knowing the public key, PU_b, and a ciphertext, C, to recover the original message, M.

2. RSA ALGORITHM

RSA Algorithm

- Developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman.
- The RSA scheme is a block cipher in which 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. That is, n is less than 2^{1024}
- Based on exponentiation in a finite field over intergers modulo a prime

Description of the Algorithm

- Select two large prime numbers: p and q
- Calculate: $n = pq$
- Calculate: $m=(p-1)(q-1)$
- Choose a small number e, co prime to m, with GCD(m,e)=1; $1<e<m$
- Find d, such that $e.d \equiv 1 \bmod m$
- PU = (n, e), PR = (n, d)

Description of the Algorithm

- Encryption:

$$C = M^e \bmod n \text{ (với } M < n)$$

- Decryption:

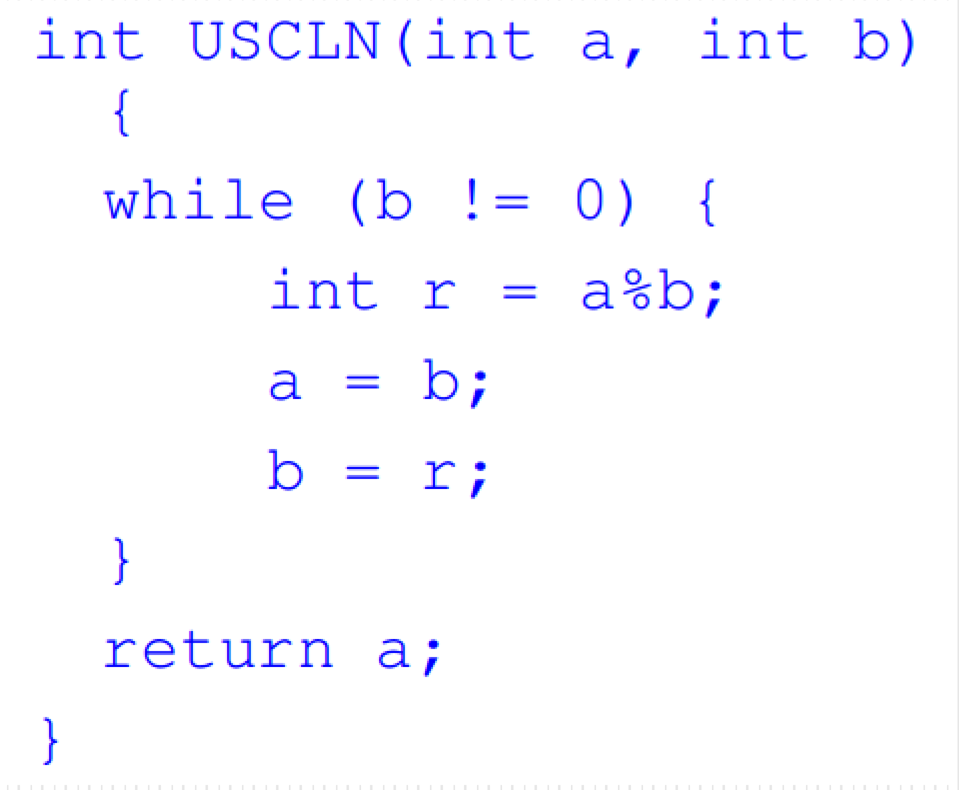
$$M = C^d \bmod N$$

Euclid’s algorithm

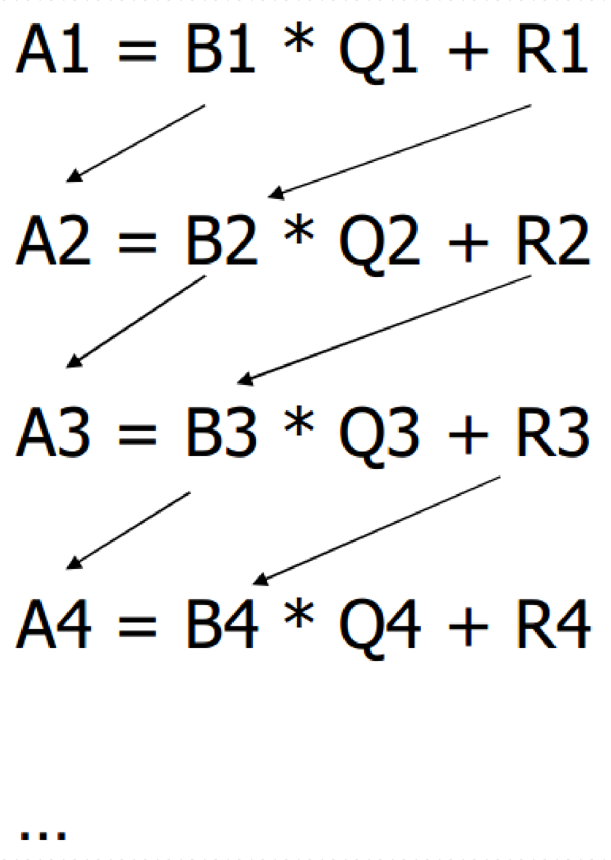
- Computing the greatest common divisor (GCD) of two numbers,

gcd(a,b) = gcd(b, a mod b)

```
1. A ← a; B ← b
2. if B = 0 return A = gcd(a, b)
3. R = A mod B
4. A ← B
5. B ← R
6. goto 2
```



```
int USCLN(int a, int b)
  {
  while (b != 0) {
        int r = a%b;
        a = b;
        b = r;
  }
  return a;
}
```



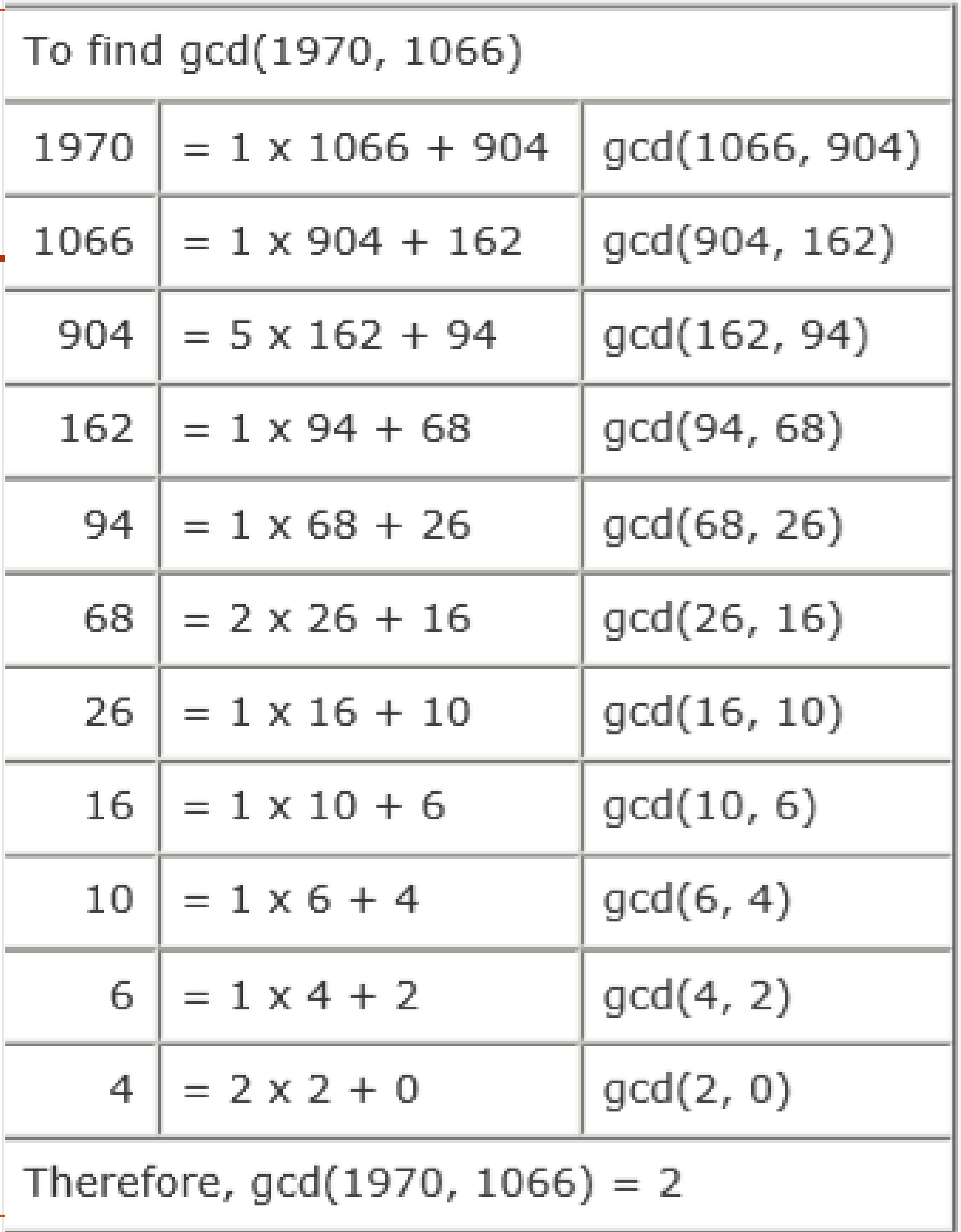
A1 = B1 * Q1 + R1

A2 = B2 * Q2 + R2

A3 = B3 * Q3 + R3

A4 = B4 * Q4 + R4

...



To find gcd(1970, 1066)

1970	= 1 x 1066 + 904	gcd(1066, 904)
1066	= 1 x 904 + 162	gcd(904, 162)
904	= 5 x 162 + 94	gcd(162, 94)
162	= 1 x 94 + 68	gcd(94, 68)
94	= 1 x 68 + 26	gcd(68, 26)
68	= 2 x 26 + 16	gcd(26, 16)
26	= 1 x 16 + 10	gcd(16, 10)
16	= 1 x 10 + 6	gcd(10, 6)
10	= 1 x 6 + 4	gcd(6, 4)
6	= 1 x 4 + 2	gcd(4, 2)
4	= 2 x 2 + 0	gcd(2, 0)

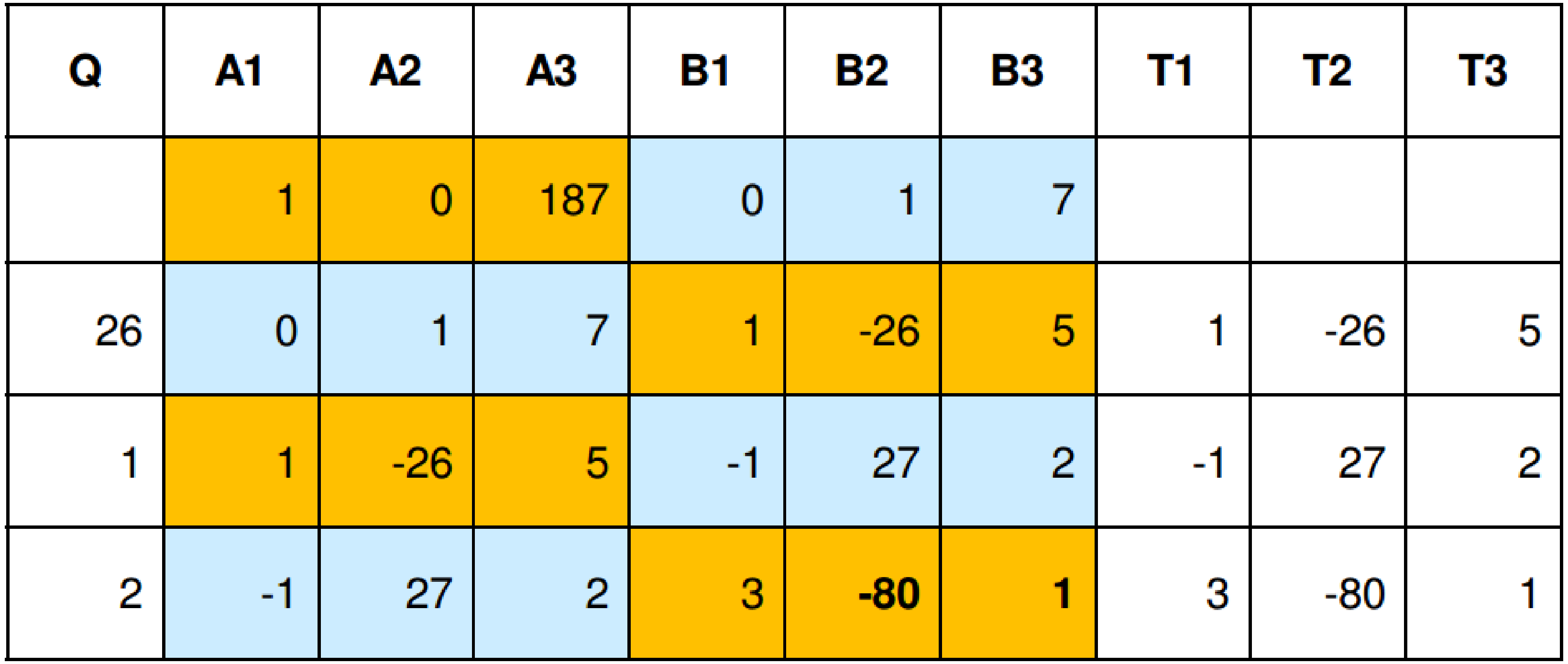
Therefore, gcd(1970, 1066) = 2

Extended Euclid’s algorithm

```
1.  (A1, A2, A3) ← (1, 0, m); (B1, B2, B3) ← (0, 1, b)
2.  if B3 = 0 return A3 = gcd(m, b); no inverse
3.  if B3 = 1 return B3 = gcd(m, b); B2
4.  Q = A3 div B3
5.  (T1, T2, T3) ←(A1 – Q*B1, A2 – Q*B2, A3 – Q*B3)
6.  (A1, A2, A3) ← (B1, B2, B3)
7.  (B1, B2, B3) ← (T1, T2, T3)
8.  goto 2
```

Extended Euclid’s algorithm - example

- Finding inverse of 7 in modulo 187



Q	A1	A2	A3	B1	B2	B3	T1	T2	T3
	1	0	187	0	1	7			
26	0	1	7	1	-26	5	1	-26	5
1	1	-26	5	-1	27	2	-1	27	2
2	-1	27	2	3	**-80**	**1**	3	-80	1

=>Result: 80

RSA Example

- p = 11, q = 3 => n = pq=33
- m= (p-1)(q-1) = (11 – 1)(3 – 1) = 20
- Gcd(m,e)=1
- e corprime to m, means that the largest numbet that can be exactly divide both e and m (their greatest common divisor, or gcd) is 1. Euclid's algorithm is used to find the GCD of two numbers

RSA Example

- e=2 => GCD(20,e) = 2 (no)
- e=3 => GCD(20,e)=1 (yes!)
- Find d: using Extended Euclid's algorithm ? d=7
- PU (33, 3), PR = (33, 7)

Plaintext: M = 15:

Encryption: C = 15^3mod 33 = 9

Deencryption: c=9

M = 9^7mod 33 = 15

RSA Security

- Brute-force attack
- Mathematical attack
- Timing attack
- Chosen ciphertext attack

Thanks