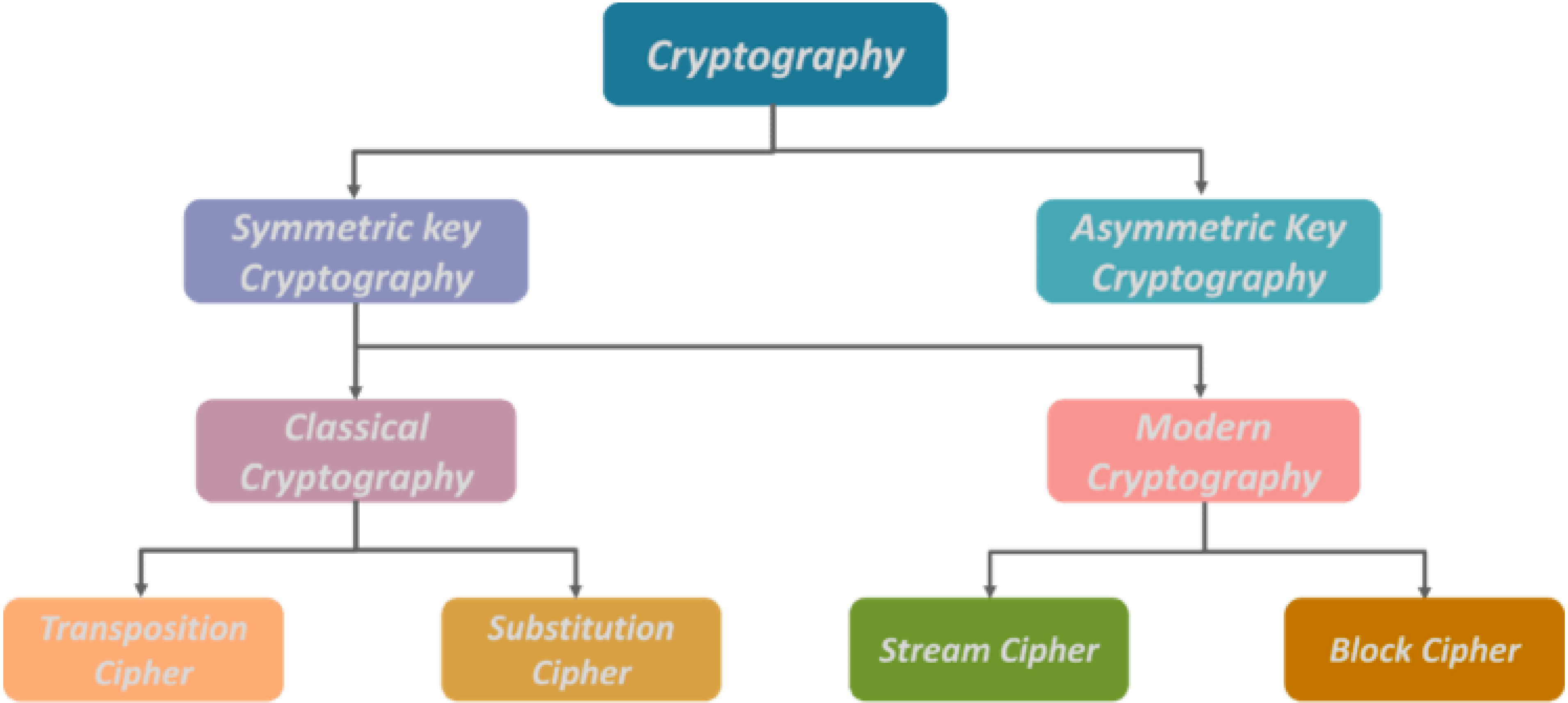
Practical Cryptography

Handout 5 – Public Key Cryptography

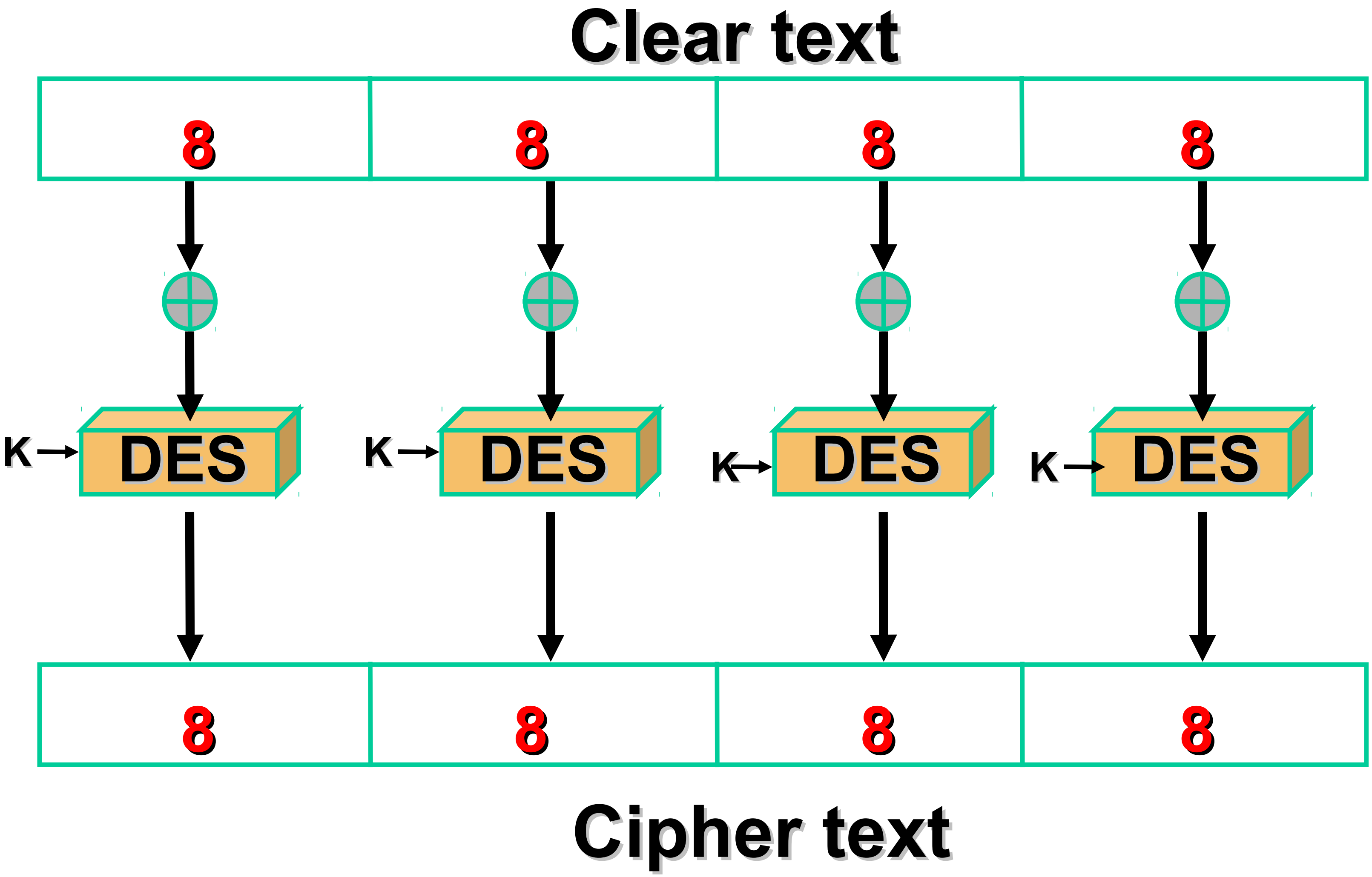
Kasun de Zoysa
kasun@ucsc.cmb.ac.lk

UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING

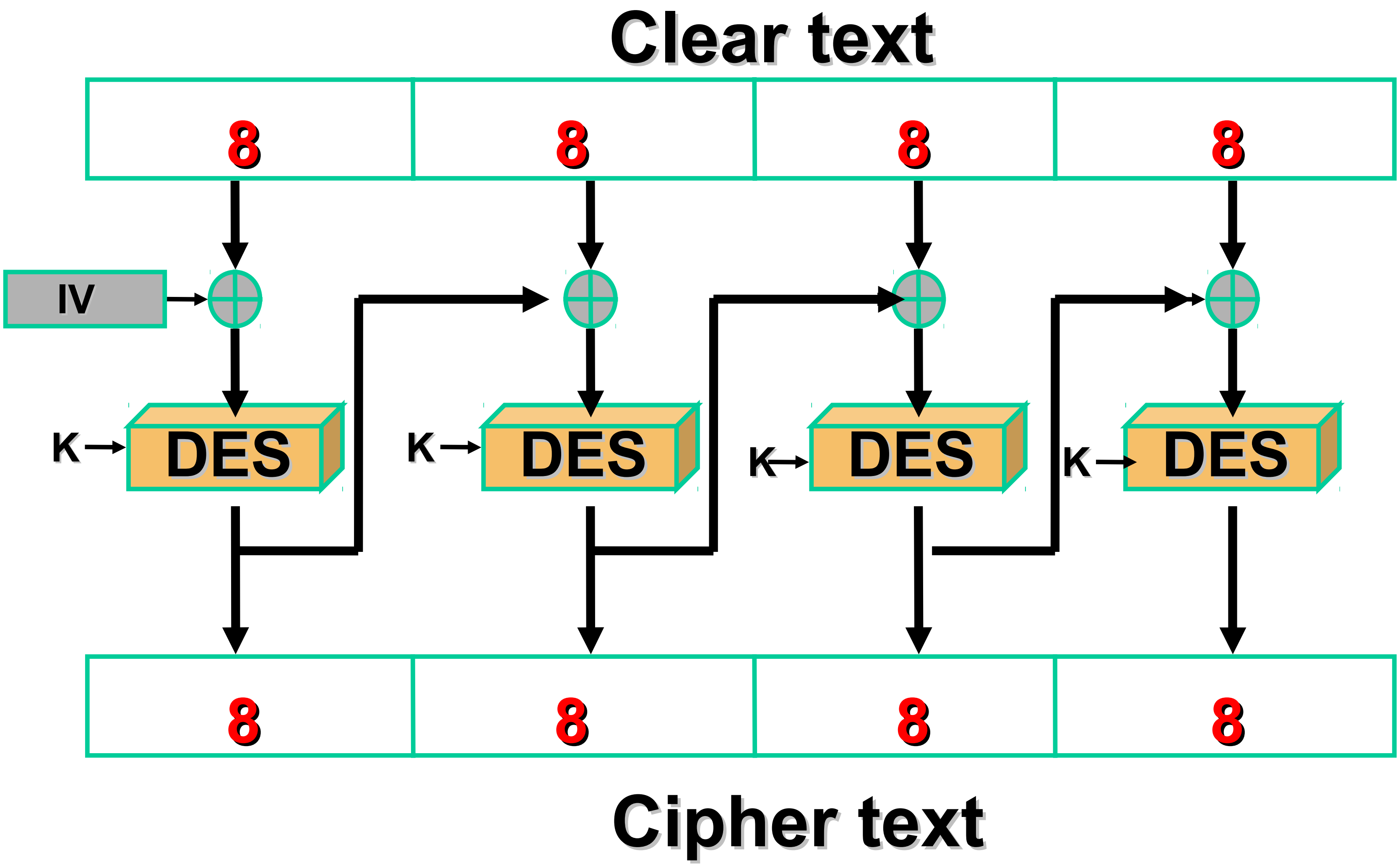
Cryptography



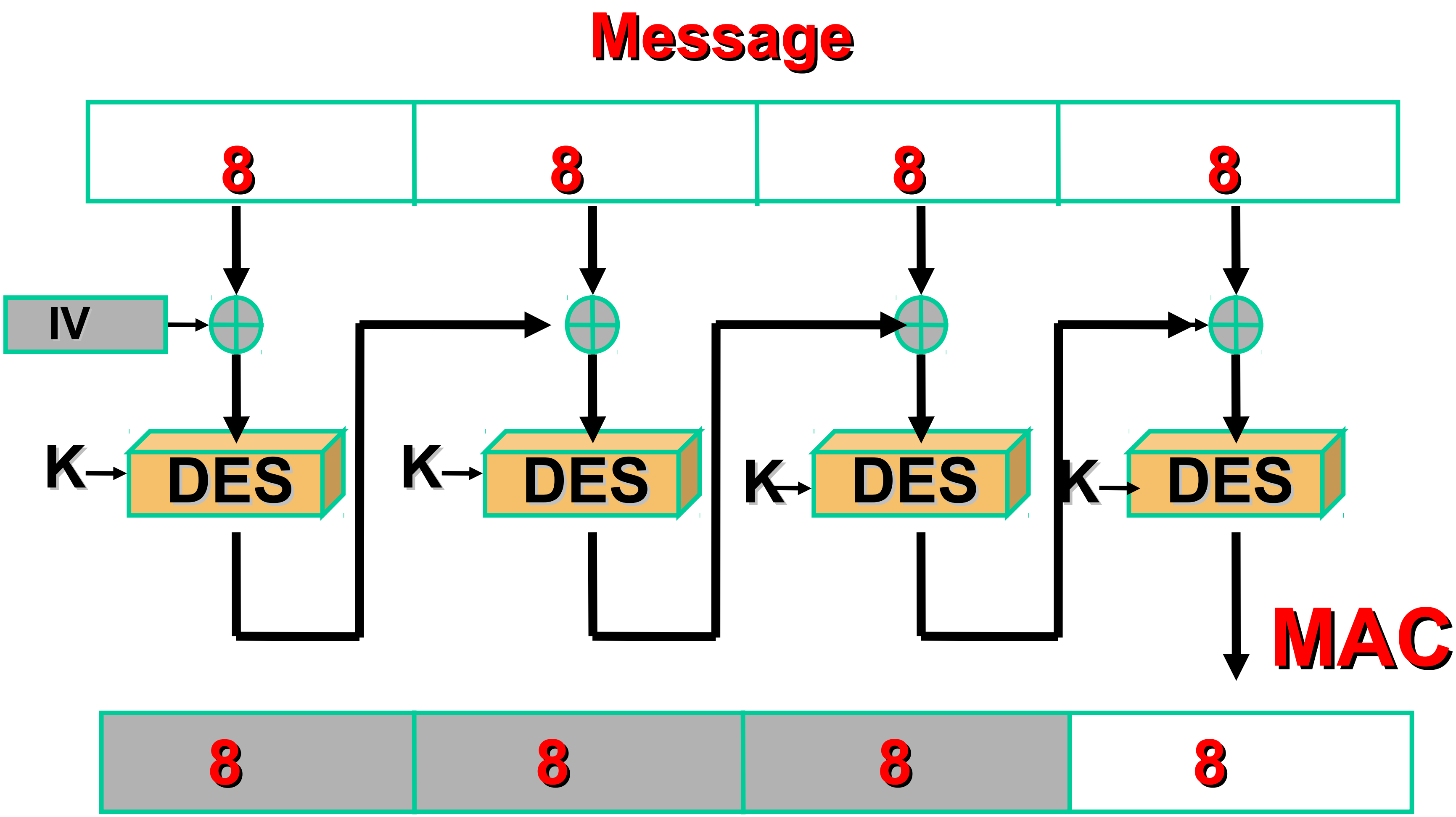
Electronic Code Book Mode (ECB)



Cipher Block Chaining Mode (CBC)



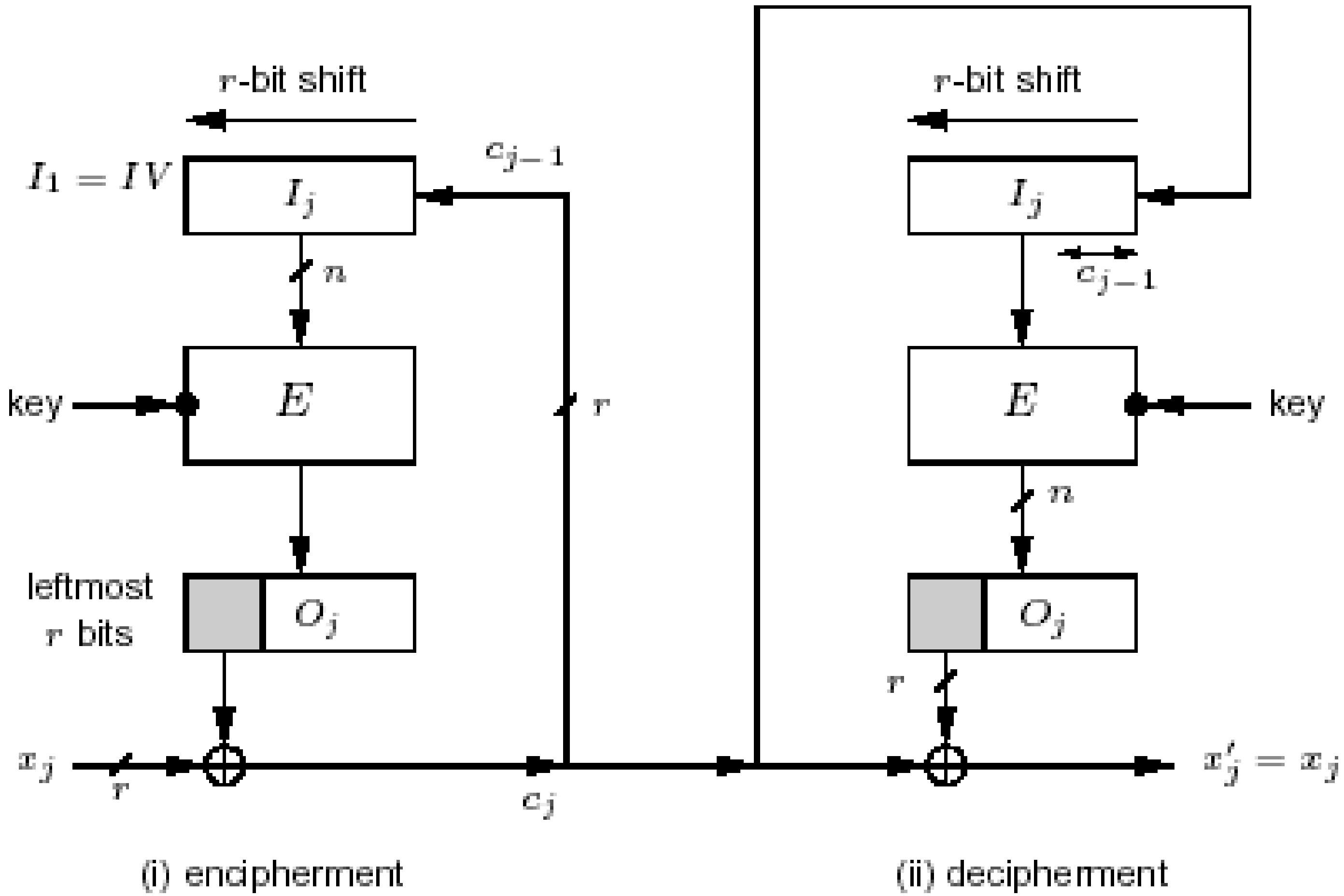
MAC based on CBC



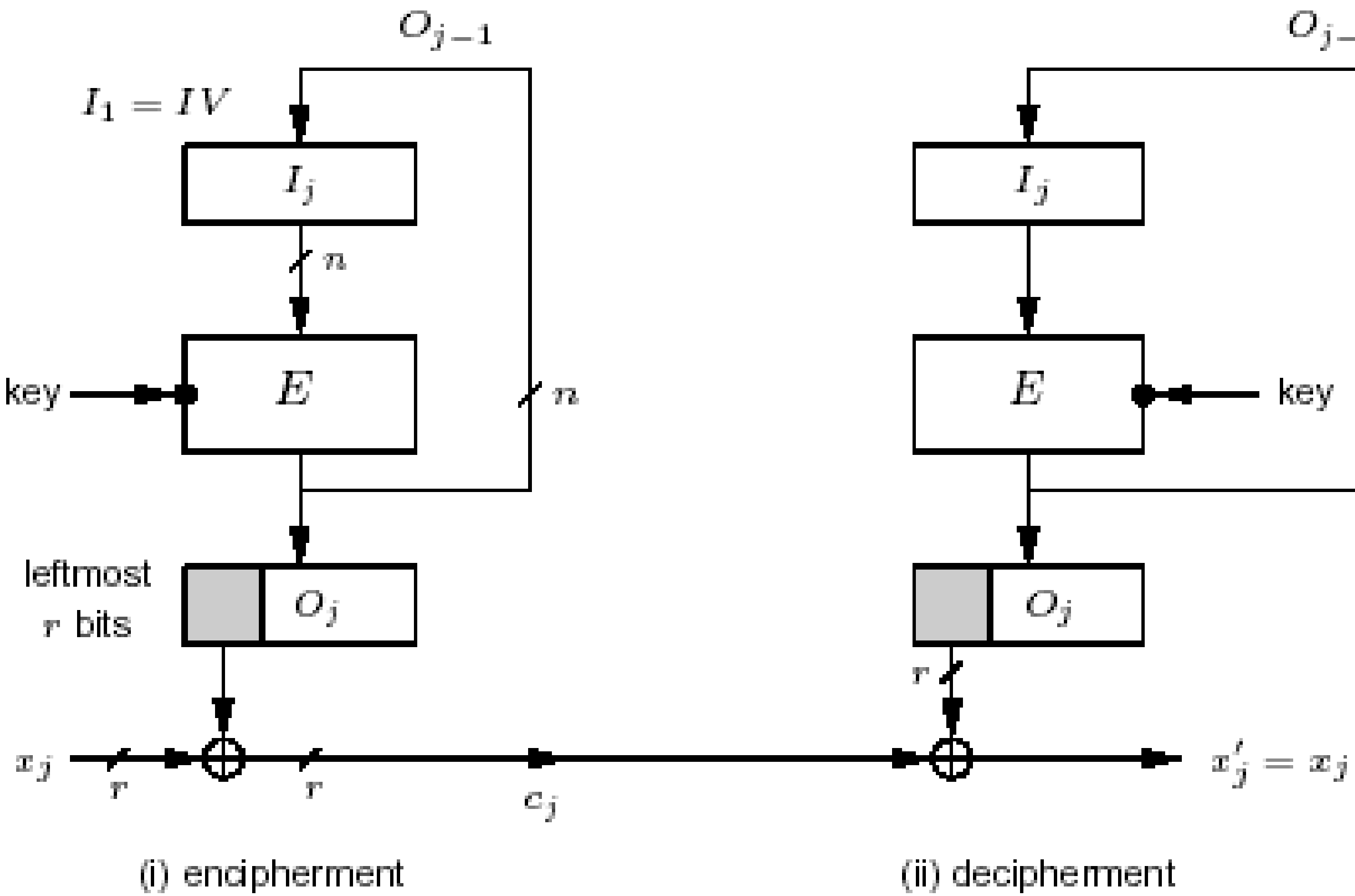
CBC-MAC vs CBC-Enc

- **Different security properties**
  - CBC-Enc is CPA secure encryption
  - CBC-MAC is secure MAC
- **Initialization**
  - CBC-Enc uses random IV
  - CBC-MAC uses first block fixed at 0
  - CBC-MAC with random IV is insecure!
- **Output**
  - CBC-Enc outputs all intermediate blocks (to decrypt)
  - CBC-MAC outputs only last block

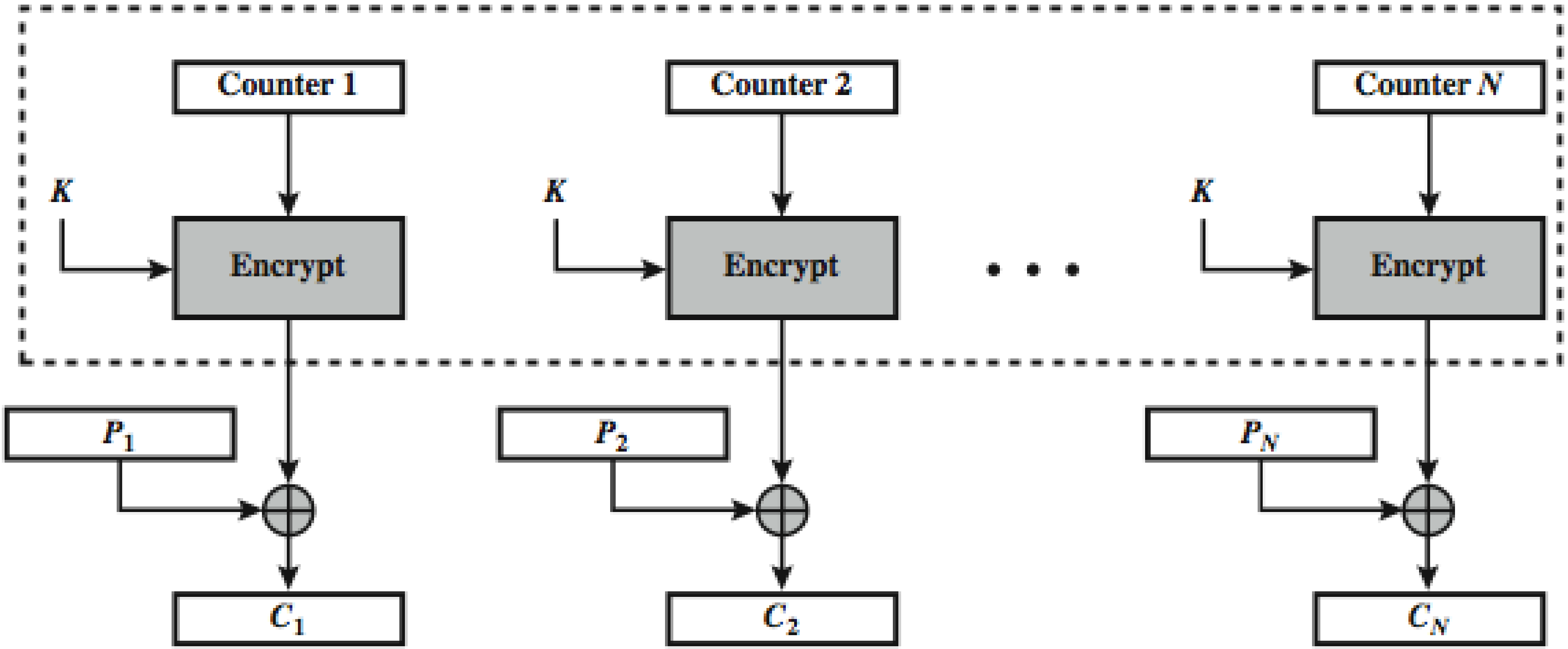
Cipher Feedback Mode (CFB)



Output Feedback Mode (OFB)



CTR



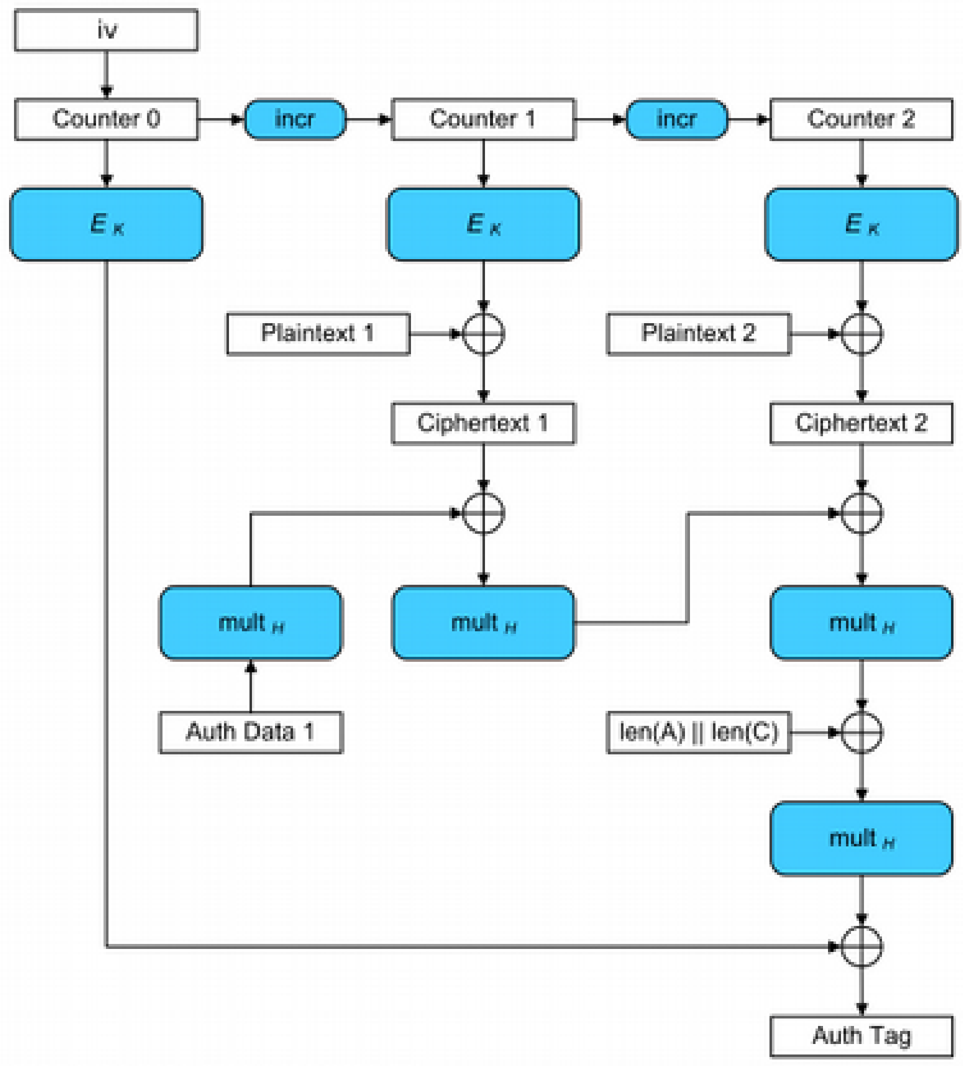
(a) Encryption

GCM (Galois/Counter) Block Mode

The GCM mode uses a counter, which is increased for each block and calculated a message authentication tag (MAC code) after each processed block.

The final authentication tag is calculated from the last block. Like all counter modes, GCM works as a stream cipher, and so it is essential that a different IV is used at the start for each stream that is encrypted.

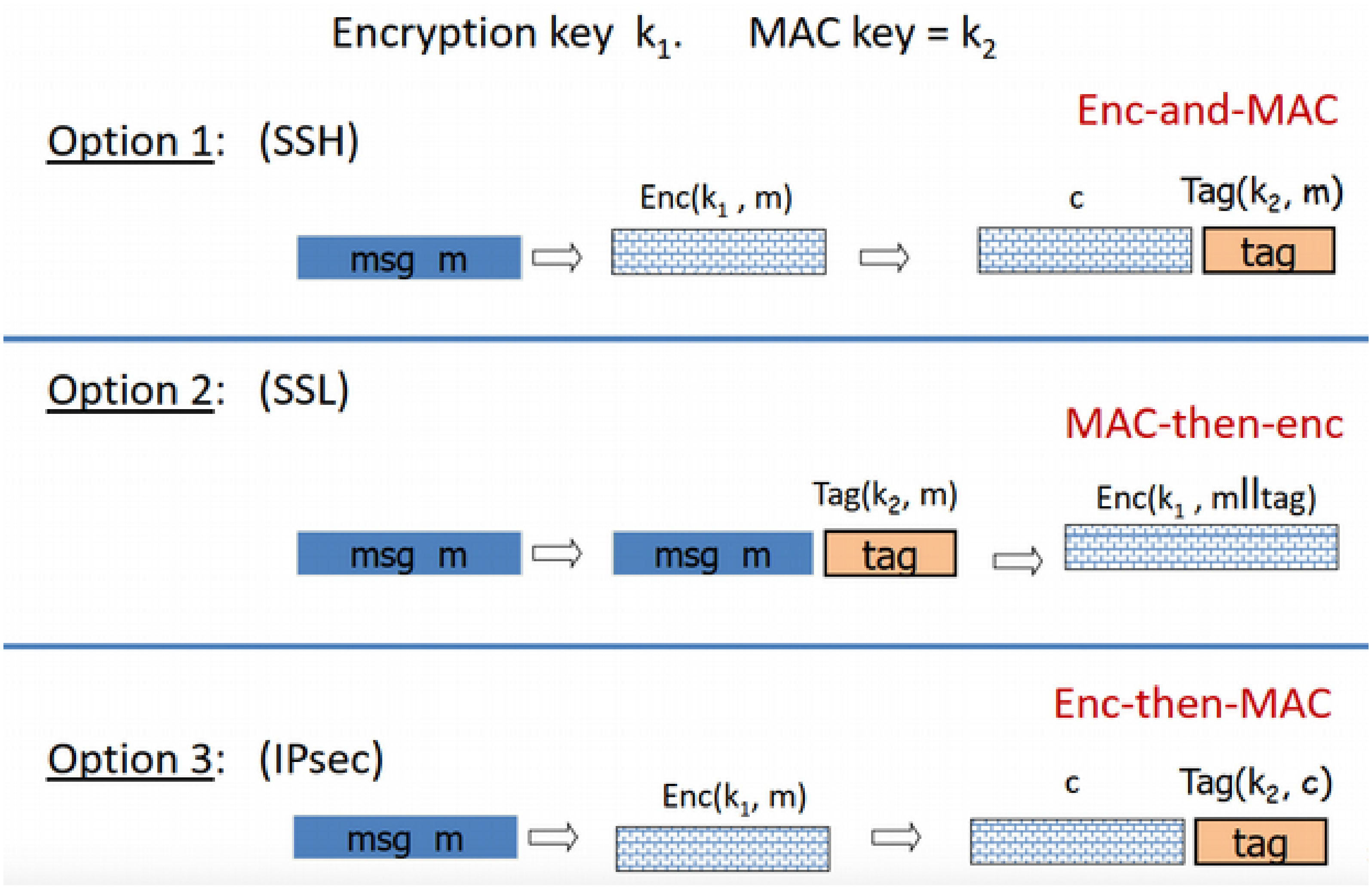
The key-feature is the ease of parallel-computation of the Galois field multiplication used for authentication.



Authenticated Encryption

- Combine confidentiality and integrity
- Security properties
  - Confidentiality: CCA security
  - Integrity: attacker cannot create new ciphertexts that decrypt properly
- Decryption returns either
  - Valid messages
  - Or invalid symbol (when ciphertext is not valid)

Combining MAC and ENC



WPA2 - CCM

- Counter mode (CTR) is used for encryption
- Cipher Block Chaining Message Authentication Code (CBCMAC) is used for integrity
- CCM = CTR + CBC-MAC for confidentiality and integrity

Advantages & Disadvantages

Advantages

Algorithms are fast

- ***Encryption & decryption are handled by same key***
- ***As long as the key remains secret, the system also provide authentication***

Disadvantages

Key is revealed, the interceptors can decrypt all encrypted information

- ***Key distribution problem***
- ***Number of keys increases with the square of the number of people exchanging secret information***

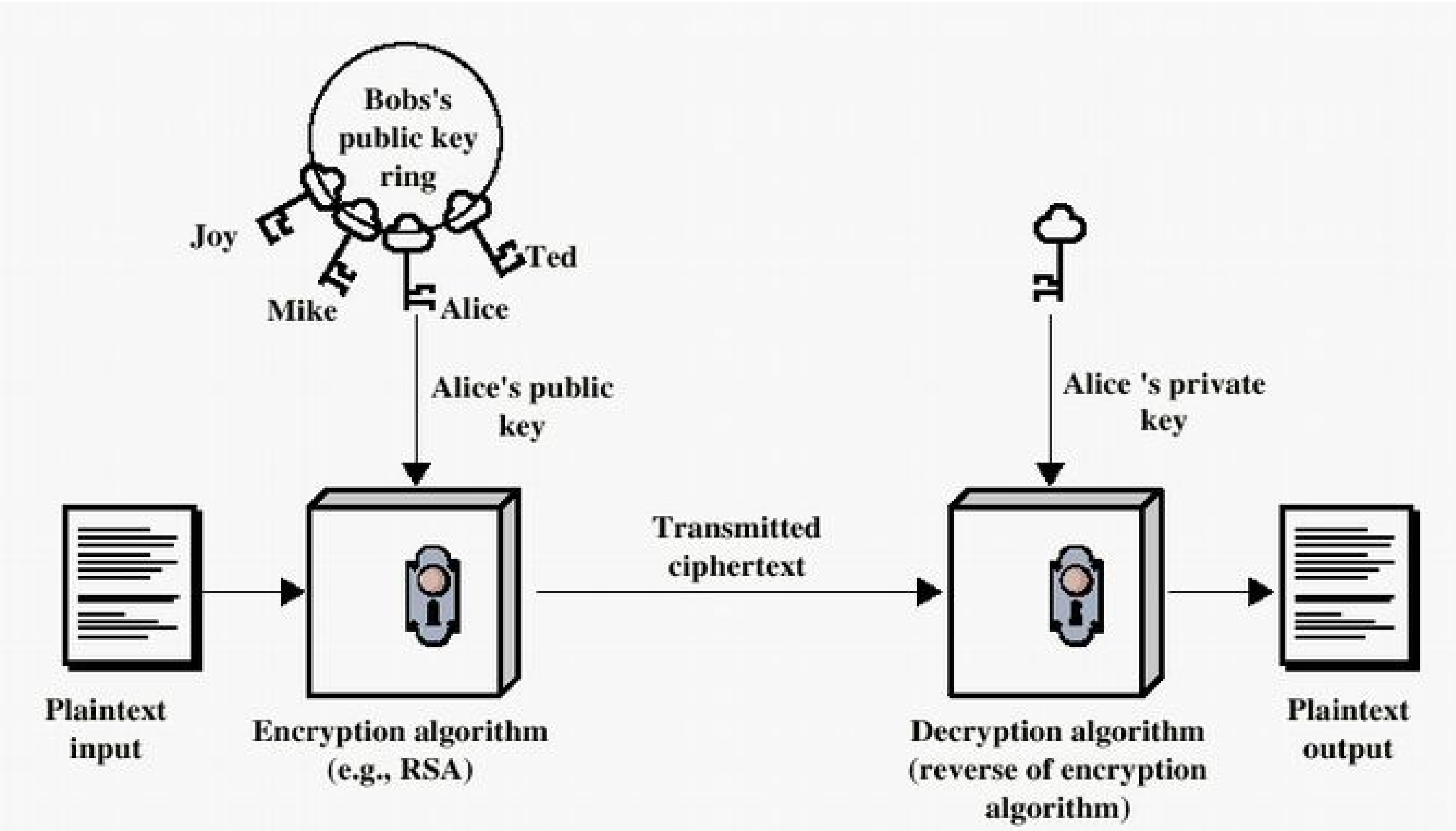
Public-Key Cryptography

- Developed to address two issues:
  - **key distribution** – how to have secure communications in general without having to trust a KDC with your key
- **digital signatures** – how to verify a message comes intact from the claimed sender
- Whitfield Diffie and Martin Hellman in 1976 known earlier in classified community

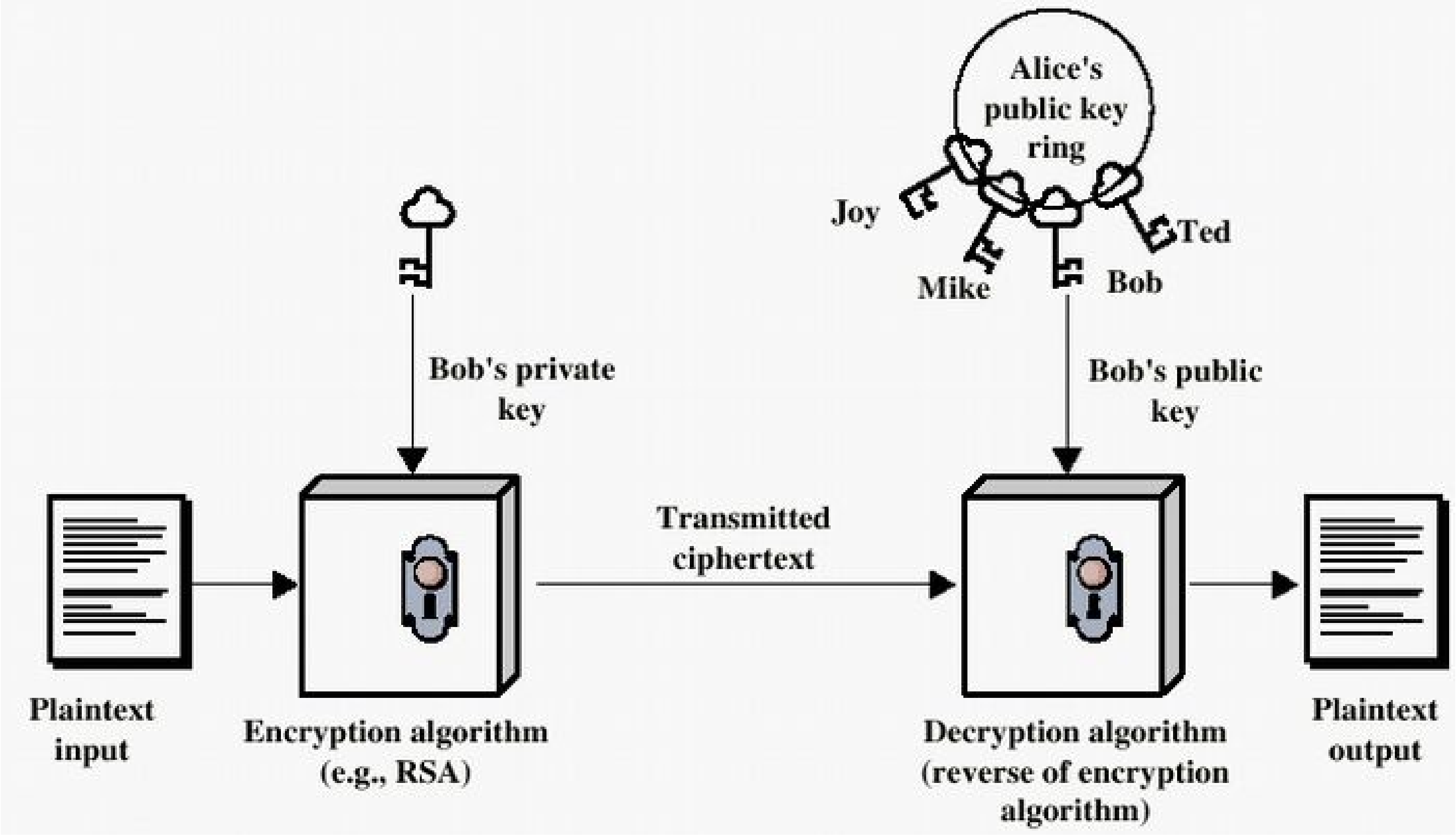
Public-Key Cryptography Principles

- The use of two keys has consequences in: key distribution, confidentiality and authentication.
- The scheme has six ingredients
  - Plaintext
  - Encryption algorithm
  - Public and private key
  - Ciphertext
  - Decryption algorithm

Encryption using Public-Key system



Authentication using Public-Key System



Applications for Public-Key Cryptosystems

- Three categories:
  - **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 two exchange a session key.

Requirements for Public-Key Cryptography

- Computationally easy for a party B to generate a pair (public key KU_b, private key KR_b)
- Easy for sender to generate ciphertext
- Easy for the receiver to decrypt ciphertext using private key

Requirements for Public-Key Cryptography

- Computationally infeasible to determine private key (KR_b) knowing public key (KU_b)
- Computationally infeasible to recover message M, knowing KU_b and ciphertext C
- Either of the two keys can be used for encryption, with the other used for decryption:

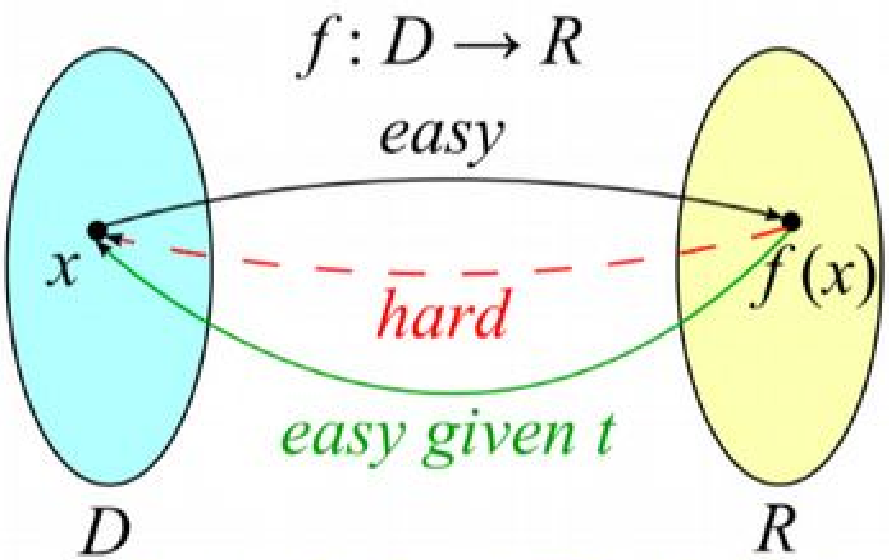
Public-Key Cryptographic Algorithms

- **Diffie-Hellman**
  - Exchange a secret key securely
  - Compute discrete logarithms
- **RSA** - Ron Rives, Adi Shamir and Len Adleman at MIT, in 1977.
  - The most widely implemented
- **Elliptic Curve Cryptography (ECC)**

Trapdoor Function

Trapdoor functions

- Easy to compute in one direction
- Difficult to compute in other direction (finding the inverse) but easy to compute, with some special information (trapdoor)



Source: https://en.wikipedia.org/wiki/Trapdoor_function

Discrete Logarithms

- The inverse problem to exponentiation is to find the **discrete logarithm** of a number modulo p
- That is, find x where ($a^x = b \bmod p$).
- This is also written as ($x = \log_a b \bmod p$). If a is a primitive root then the discrete logarithm always exists, otherwise it may not
  - $x = \log_3 4 \bmod 13$ (x st $3^x = 4 \bmod 13$) has no answer
  - $x = \log_2 3 \bmod 13 = 4$ by trying successive powers
- While exponentiation is relatively easy, finding discrete logarithms is generally a **hard** problem

Diffie-Hellman Key Agreement


- **Published in 1976**
- **Based on difficulty of calculating discrete logarithm in a finite field**

DH key pair generation

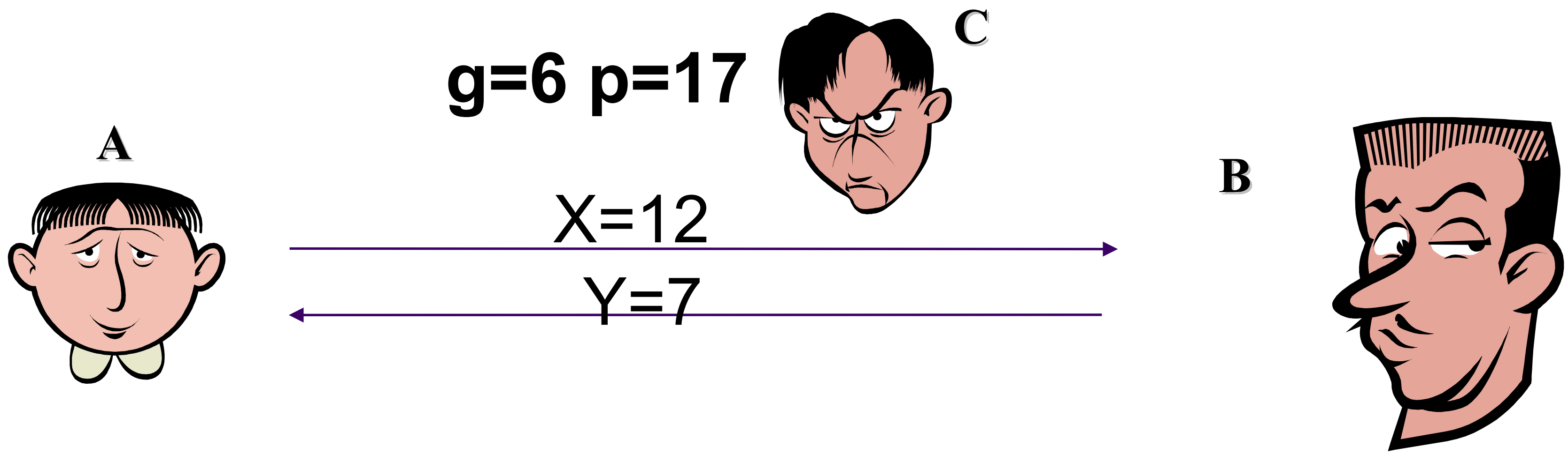
- G is finite group with generator g, p is a prime and q is a prime divisor of p-1.
- Randomly select x from [1, q-1]
- Compute $y=g^x$ (mod p)

The public key is *y*, and private key is *x*.

Observation: $x=\log_g y$(mod p), x is called the discrete logarithm of y to the base g.

Given g,x, and p, it is trivial to calculate y. However, given y, g, and p it is difficult to calculate x.

Diffie-Hellman Key Agreement



$X=g^x$ mod p
$k=Y^x$ mod p= g^{xy} mod p

Alice picks x=3
Alice's X = 6^3 mod 17
= 216 mod 17 = 12

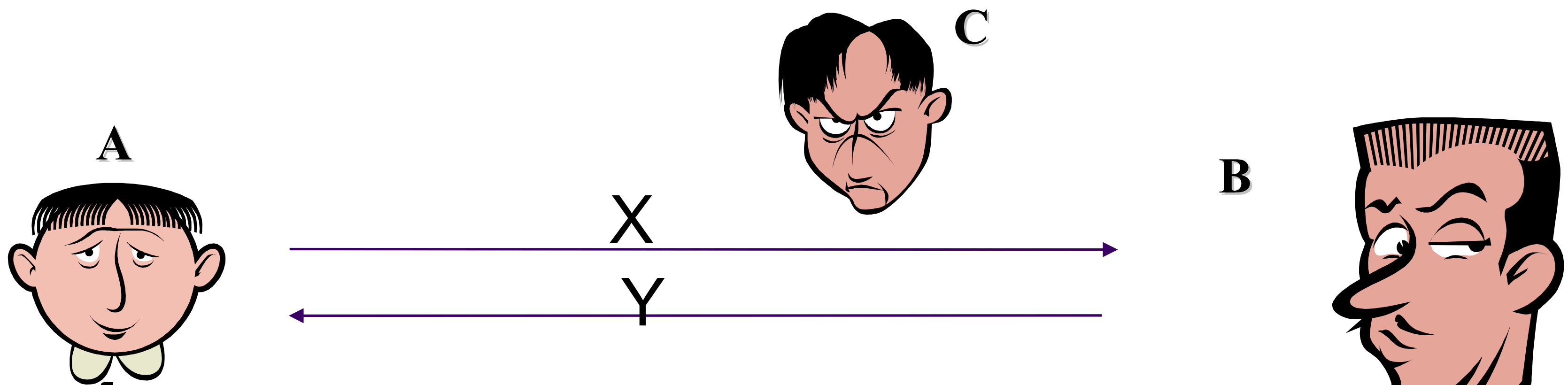
Alice's k = 7^3 mod 17
= 243 mod 17 = 3

$Y=g^y$ mod p
$k=X^y$ mod p= g^{xy} mod p

Bob picks y=5
Bob's Y = 6^5 mod 17
= 7776 mod 17 = 7

Bob's k = 12^5 mod 17
= 248832 mod 17 = 3

Attacks on Diffie-Hellman Key Agreement



$X = g^x \bmod n$

$k = Y^x \bmod n = g^{xy} \bmod n$

$Y = g^y \bmod n$

$k = X^y \bmod n = g^{xy} \bmod n$

Possible to do man in the middle attack

- You really don't know anything about who you have exchanged keys with
- Alice and Bob think they are talking directly to each other, but Caldera is actually performing two separate exchanges
- **You need to have an authenticated DH exchange**

RSA (Rivest, Shamir, Adelman)

- A dominant public key algorithm
  - The algorithm itself is conceptually simple
  - Why it is secure is very deep (number theory)
  - Use properties of exponentiation modulo a product of large primes

"A method for obtaining Digital Signatures and Public Key Cryptosystems“, Communications of the ACM, Feb., 1978 21(2) pages 120-126.

Prime Factorization

- An integer, *n* > *1*, can be factored in a unique way as:

$$n = p_1^{a_1} . p_2^{a_2} \ldots . p_t^{a_t}$$

where $p_1 < p_2 < \ldots < p_t$ and a_i is a positive integer

- E.g. `91=7×13,  3600=2⁴×3²×5²`

Relatively Prime Numbers & GCD

- Two numbers *a* and *b* are **relatively prime** if they have **no common divisors** apart from 1
  - eg. 8 & 15 are relatively prime since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and 1 is the only common factor
- Can determine the greatest common divisor by comparing their prime factorizations and using least powers
  - E.g. $300=2^1\times3^1\times5^2$, $18=2^1\times3^2$ hence $GCD(18,300)=2^1\times3^1\times5^0=6$

Prime Numbers

- Prime numbers only have divisors of 1 and self they cannot be written as a product of other numbers
- E.g. 2,3,5,7 are prime, 4,6,8,9,10 are not
- Prime numbers are central to number theory

List of prime number less than 200 is:

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191193 197 199

Greatest Common Divisor

<u>Greatest Common Divisor - gcd(a,b)</u>

- ***The largest integer that divides a set of numbers***
- ***If p is a prime, for any number q<p, gcd(p,q)=1***
- ***gcd(a,b)=gcd(b,a)***

Example : gcd(15,10)=5

Euclidean Algorithm

If x divides a and b, x also divides a-(k*b) for every k

Suppose x divides both a and b; then
a=x*a1; b=x*b1
a-(k*b)=x*a1 - (k*x*b1)
= x*(a1-k*b1)
= x*d
<u>So that x divides (is a factor of) a-(k*b)</u>

Suppose x=gcd(a,b), where a>b
a=m*b+r
a-(m*b)=r So that gcd(b,r)=x

<u>gcd(a,b)=gcd(b,r)</u> a>b>r>=0

Euclid's GCD Algorithm

An efficient way to find the GCD (a, b)
uses theorem that:
GCD (a, b) = GCD (b, a mod b)
Euclid's Algorithm to compute GCD (a, b):

```
A=a; B=b;
while (B>0) {
R = A mod B;
A = B;
B = R;
}
return A;
```

Example: 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)

Primality Testing

- In Cryptography, we often need to find large prime numbers
- Traditionally method using **trial division**
- i.e. divide by all numbers (primes) in turn less than the square root of the number
- only works for small numbers
- Alternatively can use statistical primality tests based on properties of primes
- for which all primes numbers satisfy property but some composite numbers, called pseudo-primes, also satisfy the property

How to find a large prime? (Solovay and Strassen)

1. If p is prime and r is any number less than p

gcd(p,r)=1 ; greatest common devisor

2. Jacobi function

$J(r,p) = 1$ *if* $r=1$

$J(r/2)*(-1)^{(p2-1)/8}$ *if r is even*

$J(p \bmod r, r)*(-1)^{(r-1)*(p-1)/4}$ *if r is odd and* $r = 1$

$J(r,p) \bmod p = r^{(p-1)/2}$

Test :

If test 1 and 2 passes probability(prime p) = 1/2.

Otherwise p should not be prime.

If test repeated k time probability(prime p) = $1/2^k$

Prime Numbers

```
for N in $(seq 1 200); do openssl prime $N
| awk '/is prime/ {print "ibase=16;"$1}' |
bc; done
```

```
openssl prime 2123131931239123991233
```

Discussion