Security of Information System

Secure Encryption Systems

Nandika Kasun

Department of Communication and Media Technologies
University of Colombo School of Computing
University of Colombo
Sri Lanka

kasun@cmb.ac.lk

Objectives:

- Handle properties of arithmetic, which are the fundamental of encryption systems
- Recognize the concept of symmetric and asymmetric key cryptography
- Describe the different symmetric and asymmetric key and hash algorithms

Secure Encryption Systems

2.1 Properties of Arithmetic Operations

- Inverses
- Primes
- Greatest Common Divisor
- Euclidean Algorithm
- Modular Arithmetic
- Properties of Modular Arithmetic
- Computing the inverse
- Fermat's Theorem
- Algorithm for Computing Inverses
- Random number generation

kasun@cmb.ac.lk

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
```


kasun@cmb.ac.lk

Prime Factorization

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

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

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

- E.g. $91 = 7 \times 13$, $3600 = 2^4 \times 3^2 \times 5^2$

UCSC
kasun@cmb.ac.lk

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

kasun@cmb.ac.lk

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) = $1/2^k$

UCSC

kasun@cmb.ac.lk

Greatest Common Divisor

Greatest Common Divisor - gcd(a,b)

- *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

UCSC
kasun@cmb.ac.lk

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
So that x divides (is a factor of) a-(k*b)

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

gcd(a,b)=gcd(b,r) a>b>r>=0

UCSC

kasun@cmb.ac.lk

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)
```


kasun@cmb.ac.lk

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 $GCD(18,300)=2^1\times3^1\times5^0=6$

kasun@cmb.ac.lk

Finding Inverse

- If *gcd(m,b) = 1* then *b* has a multiplicative inverse *mod m*. That is,

$$(\forall b : 0 < b < m : (\exists b^{-1} :: b.b^{-1} = 1 \bmod m))$$

- Note: this is clearly true when m is a prime.
- We can extend Euclid's algorithm to find this multiplicative inverse.

Finding Inverses

```
EXTENDED EUCLID(m, b)
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 = b^-1 mod m
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
```


kasun@cmb.ac.lk

Modular Arithmetic

Property	Example
Associatively	a+(b+c) mod n=(a+b) + c mod n a*(b*c) mod n=(a*b) * c mod n
Commutativity	a+b mod n= b+a mod n a*b mod n= b*a mod n
Distributivity	a*(b+c) mod n=((a*b) +(a* c)) mod n
Reducibility	(a+b) mod n=((a mod n) +(b mod n)) mod n (a*b) mod n=((a mod n) *(b mod n)) mod n

kasun@cmb.ac.lk

Fermat's Theorem

States that for any prime p and any element a<p

$a^p \bmod p = a$ ← 1

or

$a^{p-1} \bmod p = 1$ ← 2

For a prime p and an element a<p,
the inverse of a is the element x such that

$ax \bmod p = 1$

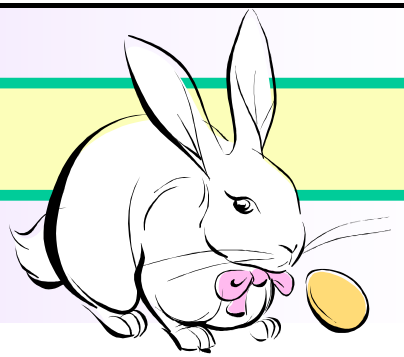
Combine with equation 2,

$ax \bmod p = 1 = a^{p-1} \bmod p$

$x = a^{p-2} \bmod p$

UCSC
kasun@cmb.ac.lk

Discrete Logarithms



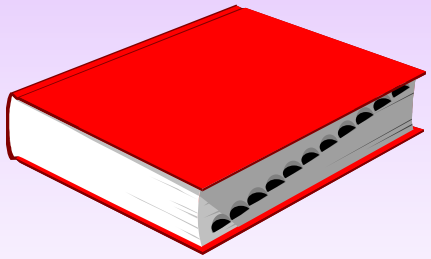
- The inverse problem to exponentiation is to find the **discrete logarithm** of a number modulo p
- That is, find x where ($a^x = b \bmod p$).
- This is also written as ($x = \log_a b \bmod p$). If a is a primitive root then the discrete logarithm always exists, otherwise it may not
  - $x = \log_3 4 \bmod 13$ (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

kasun@cmb.ac.lk

Random Number Generation

1. Truly Random numbers

- Books
- CD



2. Pseudo Random numbers

- Linear congruential random number generation

$$R_{i+1} = (a * R_i + b) \bmod n$$

$$R_1 = (a * R_0 + b) \bmod n$$
$$R_2 = (a * R_1 + b) \bmod n$$
$$R_3 = (a * R_2 + b) \bmod n$$

kasun@cmb.ac.lk

2.2 Public Key (Asymmetric key) Encryption Systems

- Concept and Characteristics of Public key Encryption System
- Introduction to Merkle-Hellman Knapsacks
- Rivest-Shamir-Adelman (RSA) Encryption in Detail
- Introduction to Digital Signature Algorithms
- The Digital Signature Standard (DSA)
- Introduction to Elliptic Curve (EC) Cryptography

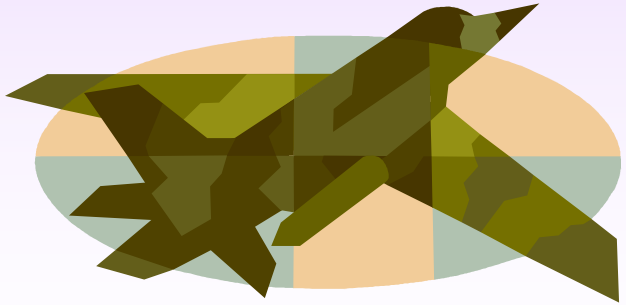
kasun@cmb.ac.lk

Why Public-Key Cryptography?

- Developed to address two issues:
  - **key distribution** – how to have secure communications in general without having to trust a Key Distribution Center (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

- Very significant advance in the history of cryptography
- Uses **two** keys – a public and a private key
- **Asymmetric** since parties are **not** equal
- Uses clever application of number theoretic concepts to function
- Complements **rather than** replaces symmetric key cryptography



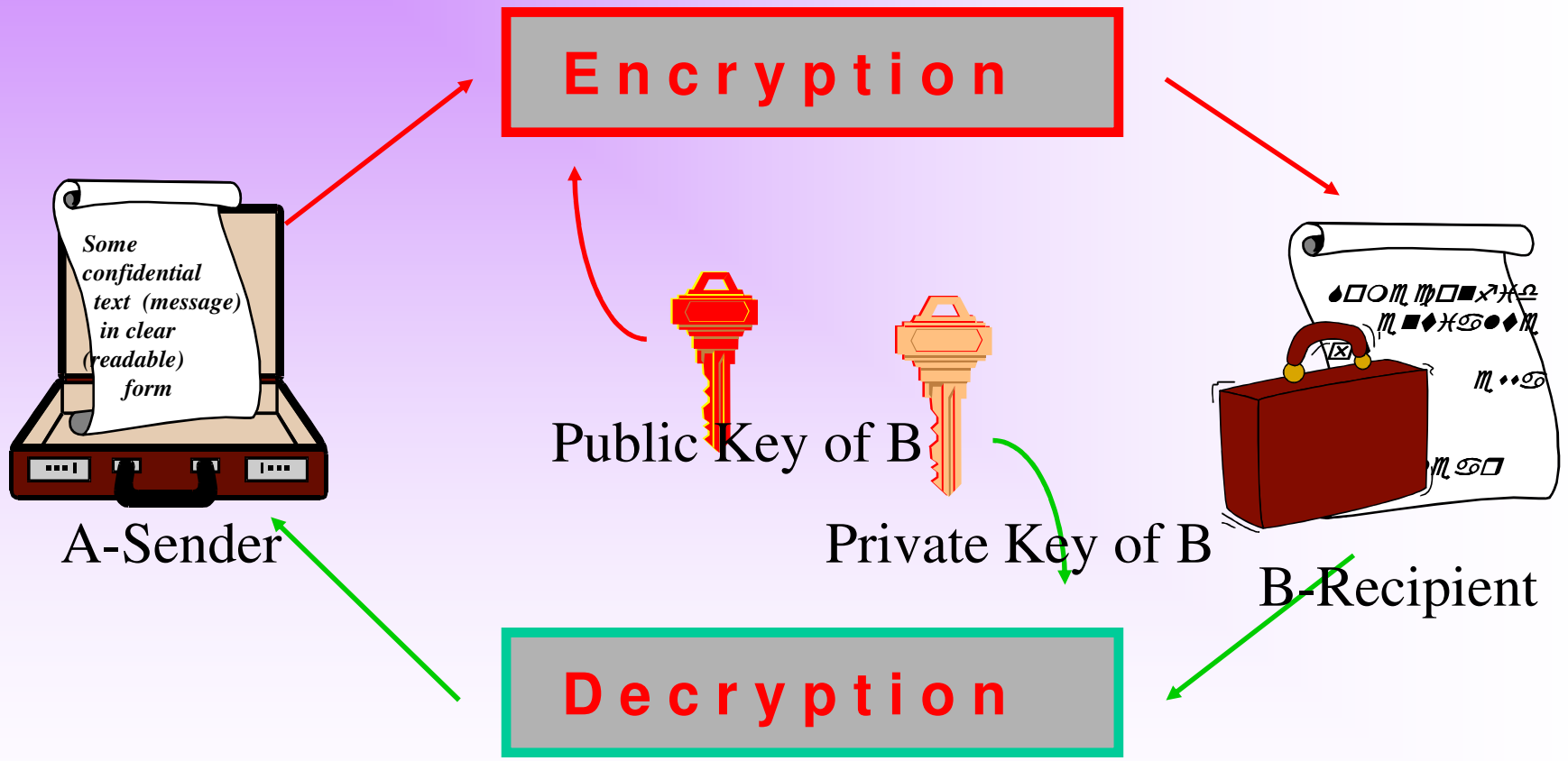
kasun@cmb.ac.lk

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

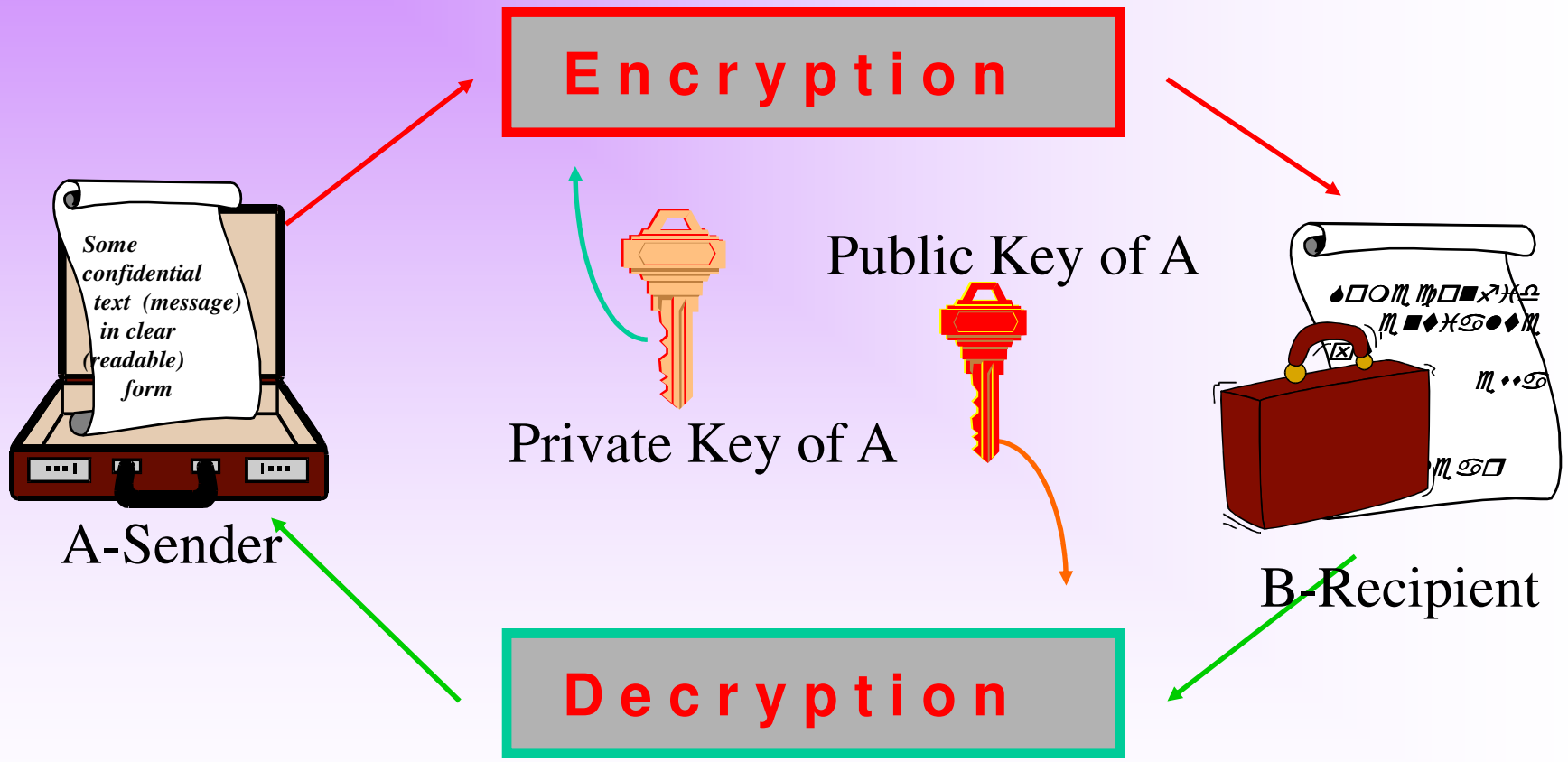
kasun@cmb.ac.lk

Public Key Encryption



UCSC
kasun@cmb.ac.lk

Signing



kasun@cmb.ac.lk

Public-Key Cryptography

- **Public-key/two-key/asymmetric** cryptography involves the use of **two** keys:
  - a **public-key**, which may be known by anybody, and can be used to **encrypt messages**, and **verify signatures**
  - a **private-key**, known only to the recipient, used to **decrypt messages**, and **sign** (create) **signatures**
- is **asymmetric** because
  - those who encrypt messages or verify signatures **cannot** decrypt messages or create signatures

kasun@cmb.ac.lk

Requirements for Public-Key Cryptography

1. Computationally easy for a party B to generate a pair (public key KU_b, private key KR_b)
2. Easy for sender to generate ciphertext:

$$C = E_{KUb}(M)$$

1. Easy for the receiver to decrypt ciphertect using private key:

$$M = D_{KRb}(C) = D_{KRb}[E_{KUb}(M)]$$

kasun@cmb.ac.lk

Requirements for Public-Key Cryptography

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

$$M = D_{KRb}[E_{KUb}(M)] = D_{KUb}[E_{KRb}(M)]$$

kasun@cmb.ac.lk

Security of Public Key Schemes

- Like symmetric key schemes brute force **exhaustive search** attacks are always theoretically possible
- But keys used are much larger (> 512 bits)
- Security relies on computationally infeasibility of the cryptanalysis problem

kasun@cmb.ac.lk

Public-Key Cryptographic Algorithms

- Knapsack based encryption
- RSA - Ron Rives, Adi Shamir and Len Adleman at MIT, in 1977.
  - RSA is a block cipher
  - The most widely implemented
- Diffie-Hellman
  - Echange a secret key securely
  - Compute discrete logarithms

kasun@cmb.ac.lk

The Knapsack Problem

Given $\{a_1, a_2, \ldots, a_n\}$, and $x \in \{0,1\}^n$, computing $y=f(x)=\sum_i a_i x_i$ is easy, yet finding a subset of $\{a_i\}_i$ that sums up to a given y is NP-complete.

Problems:

1. f cannot be degenerate.
2. f cannot be super-increasing.

Is f hard on average?

…Probably not.

Knapsack based encryption – given `77 [Merkle, Hellman], broken `82 [Shamir] and later others.

UCSC
kasun@cmb.ac.lk

Revest-Shamir-Adelman (RSA)

By Rivest, Shamir and Adelman in 1978

1. Find 2 large prime numbers p and q (100 digits=512bits)

2. Calculate the product n=p*q (n is around 200 digits)

3. Select large integer e relatively prime to (p-1)(q-1)

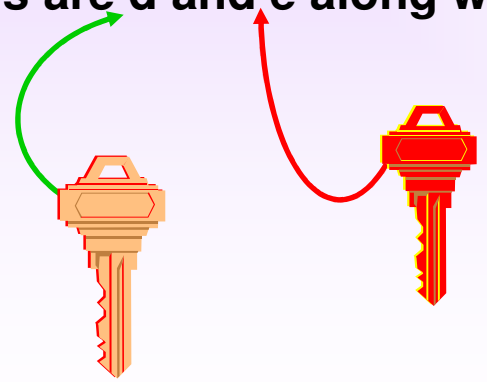
Relatively prime means e has no factors in common with (p-1)(q-1).
Easy way is select another prime that is larger than both(p-1) and (q-1).

4. Select d such that e*d mod (p-1)*(q-1)=1

Two keys are d and e along with n

Encryption
$C=P^e \bmod n$

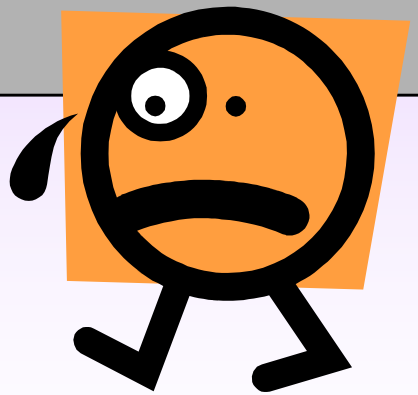
Decryption
$P=C^d \bmod n$



kasun@cmb.ac.lk

The RSA Algorithm - Encryption

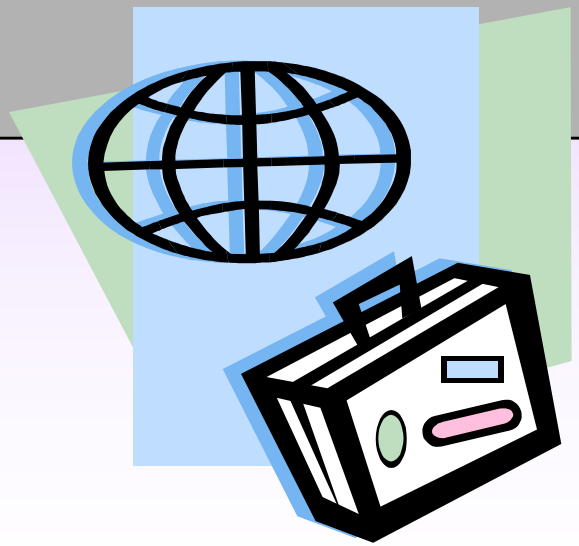
- **Plaintext:** $M<n$
- **Ciphertext:** $C = M^e \pmod n$



kasun@cmb.ac.lk

The RSA Algorithm - Decryption

- **Ciphertext:** C
- **Plaintext:** $M = C^d \pmod{n}$



kasun@cmb.ac.lk

Example of RSA Algorithm

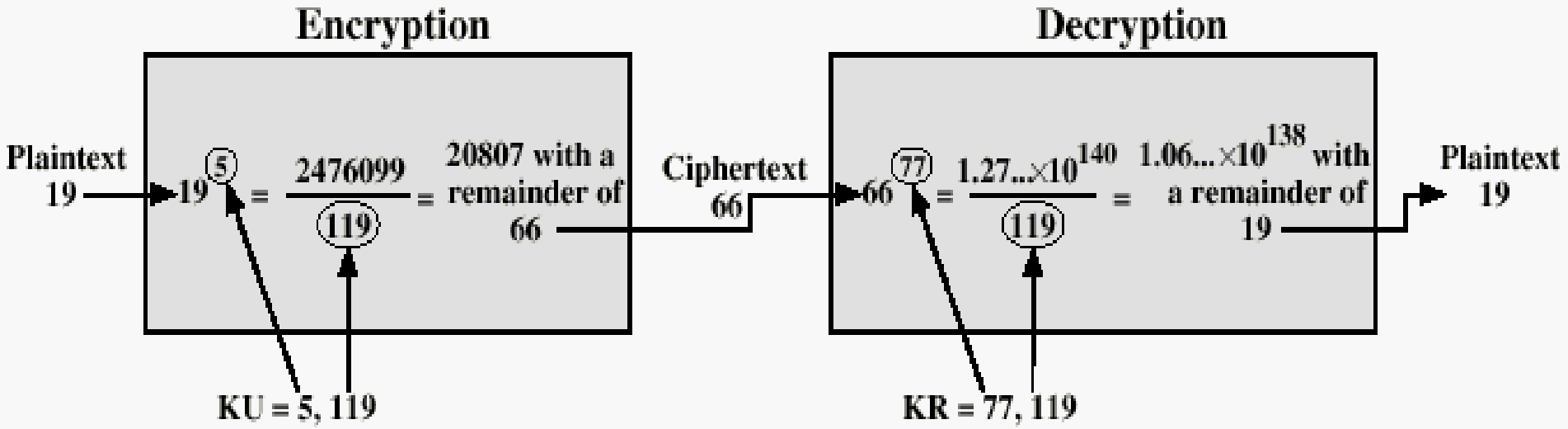


Figure 3.9 Example of RSA Algorithm

RSA - Simple Example

1. Find 2 prime numbers p and q

Let p=11 and q=13

2. Calculate the product n=p*q

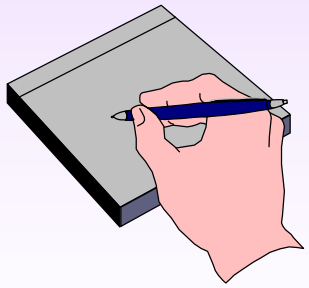
*n = 11*13=143*

3. Select large integer e relatively prime to (p-1)(q-1)

*E=11 ; 11 IS Relatively prime to (p-1)(q-1) =10*12=120*

4. Select d such that e*d mod (p-1)*(q-1)=1

*d=11 because, 11*11 mod 10*12=121 mod 120 =1*



Encryption

$C=P^e \bmod n$

Let p=7 so that $C=7^{11} \bmod 143$; C=106

Decryption

$P=C^d \bmod n$

$p=106^{11} \bmod 143$; P=7

kasun@cmb.ac.lk

El Gamal Algorithm

- In 1984 by El Gamal
- This algorithm relies on the difficulty of computing discrete logarithm over finite field

1. First choose a prime p and two integers, a and x, such that a<p and x<p
2. The prime p should be chosen so that (p-1) has a large prime factor q
3. Calculate $y=a^x \bmod p$
4. The private key is x and public key is y, along with parameter p and a
5. To sign a message m, choose a random integer k , $0<k<p-1$, which has not used before, and which is relatively prime to (p-1)
6. Compute $r = a^k \bmod p$
   $s = k^{-1} (m-xr) \bmod (p-1)$

Signature is r and s

7. To verify the signature compute $y^r r^x \bmod p$ and determine that it is equivalent to $a^m \bmod p$

kasun@cmb.ac.lk

Other Public-Key Cryptographic Algorithms

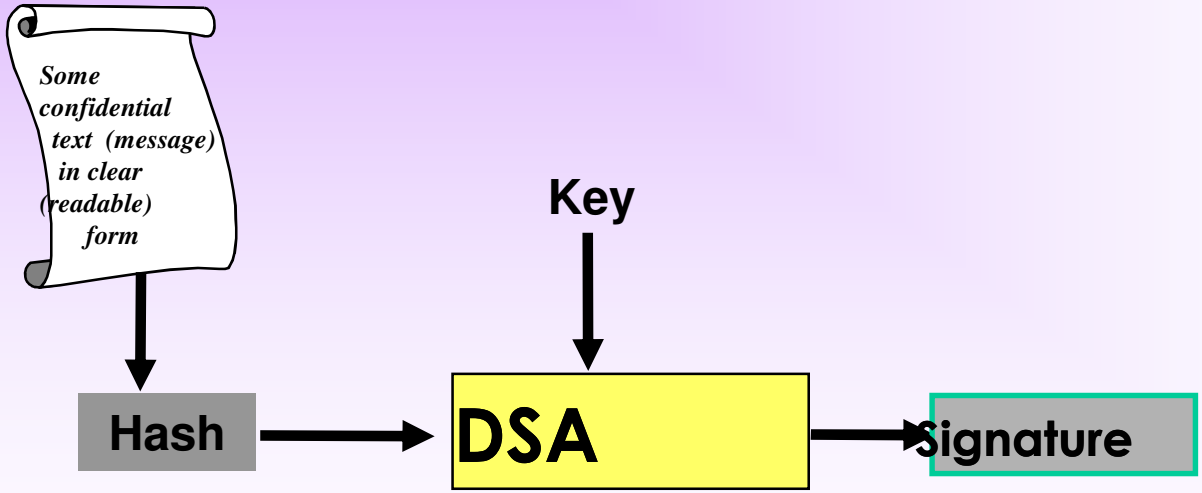
- **Digital Signature Standard (DSS)**
  - Makes use of the SHA-1
  - Not for encryption or key echange
- **Elliptic-Curve Cryptography (ECC)**
  - Good for smaller bit size
  - Low confidence level, compared with RSA
  - Very complex

kasun@cmb.ac.lk

Digital Signature Algorithm (DSA)

Use El Gamal algorithm with few restriction;

- **Fist, The size of p is specially fixed at $2^{511} < p < 2^{512}$**
- **Second, q, the large prime factor of (p-1) is chosen so that $2^{159} < q < 2^{160}$**



UCSC
kasun@cmb.ac.lk

Elliptic curve cryptography (ECC)

Why ECC?

There are other public key cryptographic systems. However, we choose to study ECC because

- The sub-exponential algorithm of breaking ECC has not been found, that is : ECC is not less secure than RSA or some other public key crypto algorithms.
- ECC with smaller key size can achieve the same security as RSA or some other crypto algorithms. Hence ECC is more efficient for secure wireless applications.
- High scalability.
- More potential due to EC theory (rich theory with many alternatives).

2.3 Hash Algorithms

- Hash Concept
- Description of Hash Algorithms
- Message Digest Algorithms

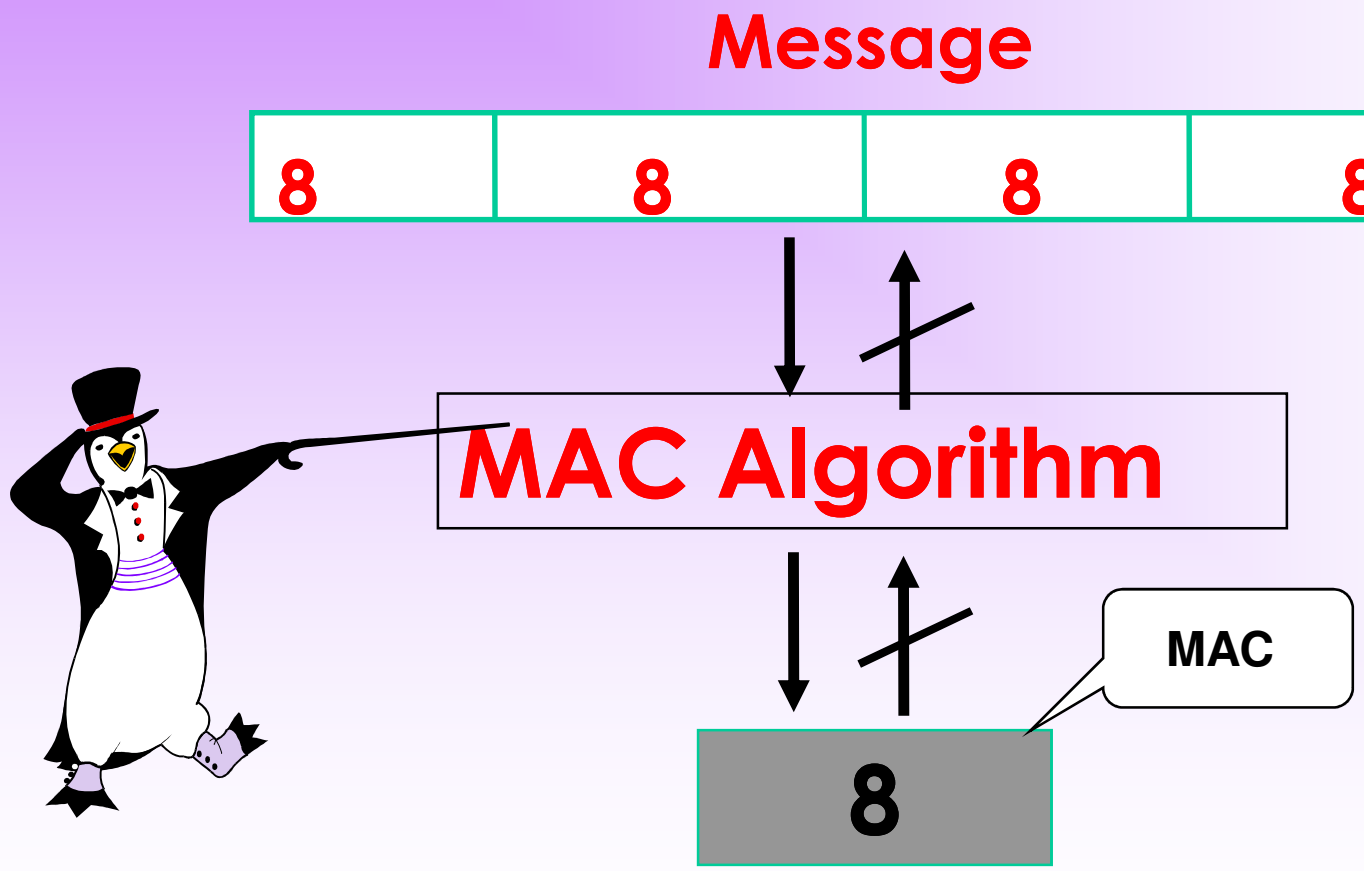
kasun@cmb.ac.lk

Message Authentication Code (MAC)

- Generated by an algorithm that creates a small fixed-sized block depending on both message and some key
- need not be reversible
- Receiver performs same computation on message and checks if it matches the MAC
- Provides assurance that message is unaltered and comes from sender

UCSC
kasun@cmb.ac.lk

Message Authentication Code (MAC)



kasun@cmb.ac.lk

MAC Properties

- A MAC is a cryptographic checksum

MAC = CK(M)

- condenses a variable-length message M
- using a secret key K to a fixed-sized authenticator
- It is a many-to-one function
- potentially many messages have same MAC but finding these needs to be very difficult

kasun@cmb.ac.lk

Requirements for MACs

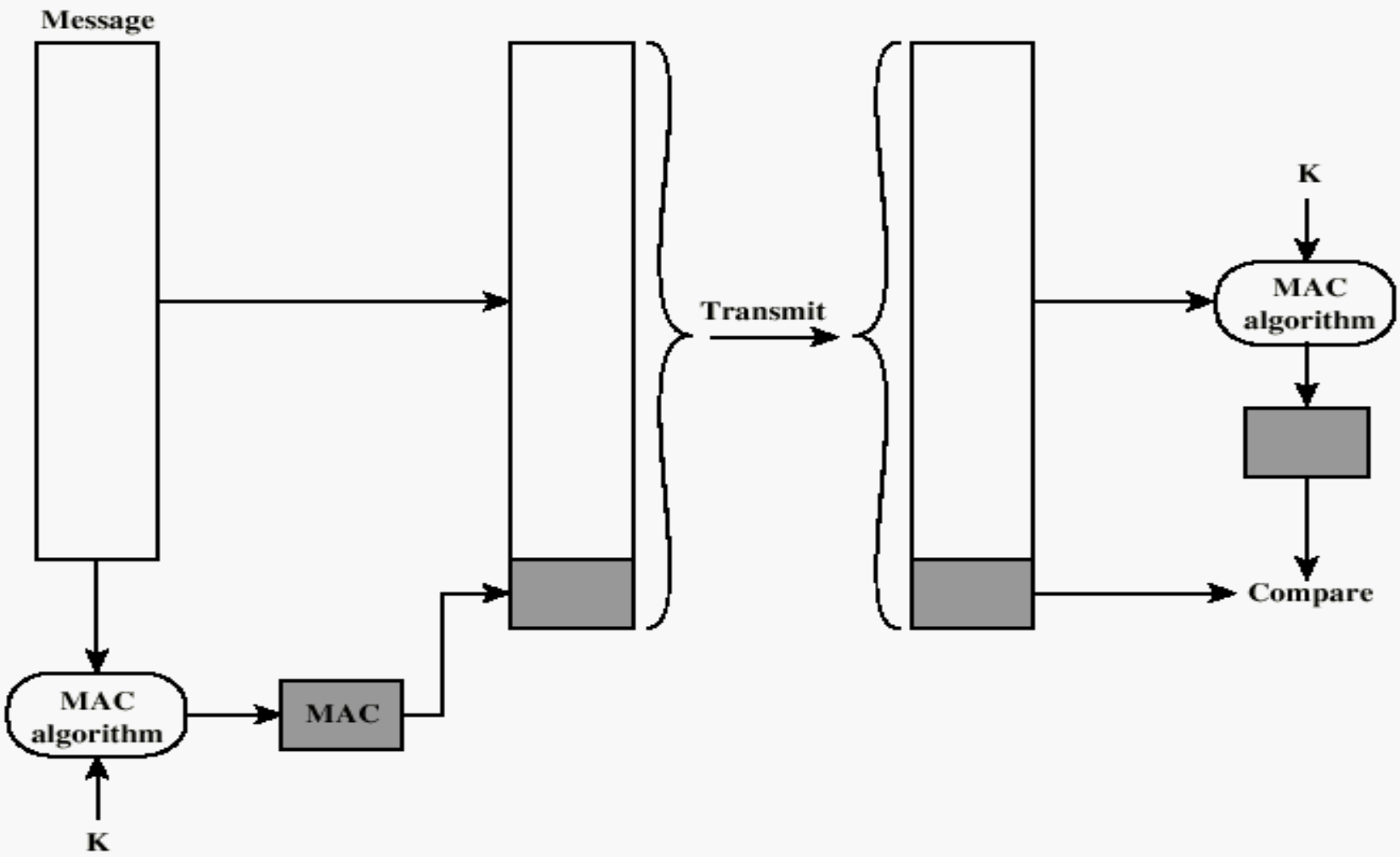
- Given a message and a MAC, it should be infeasible to find another message with same MAC
- MACs should be uniformly distributed
- MAC should depend equally on all bits of the message

kasun@cmb.ac.lk

Approaches to Message Authentication

- Authentication Using Conventional Encryption
  - Only the sender and receiver should share a key
- Message Authentication without Message Encryption
  - An authentication tag is generated and appended to each message
- Message Authentication Code
  - Calculate the MAC as a function of the message and the key. MAC = F(K, M)

kasun@cmb.ac.lk



Message Authentication Using a Message Authentication Code (MAC)

Hash Functions

- Condenses arbitrary message to fixed size
- Usually assume that the hash function is public and not keyed
  - MAC which is keyed
- Hash used to detect changes to message
- Can use in various ways with message
  - most often to create a digital signature

kasun@cmb.ac.lk

Hash Function Properties

- A Hash Function produces a fingerprint of some file/message/data

  h = H(M)

- condenses a variable-length message M to a fixed-sized fingerprint
- Assumed to be public

Requirements for Hash Functions

- Can be applied to any sized message M
- Produces fixed-length output h
- Easy to compute h = H(M) for any message M
- Given h, it is infeasible to find x s.t. H(x) = h

one-way property

- Given x, it is infeasible to find y s.t. H(y) = H(x)

weak collision resistance

- It is infeasible to find any x,y s.t. H(y) = H(x)

strong collision resistance

kasun@cmb.ac.lk

Simple Hash Functions

- There are several proposals for simple functions
- Some are based on XOR of message blocks
- Not secure since one can manipulate any message and either not change hash or manipulate the hash as well
- need a stronger cryptographic function

kasun@cmb.ac.lk

MD5

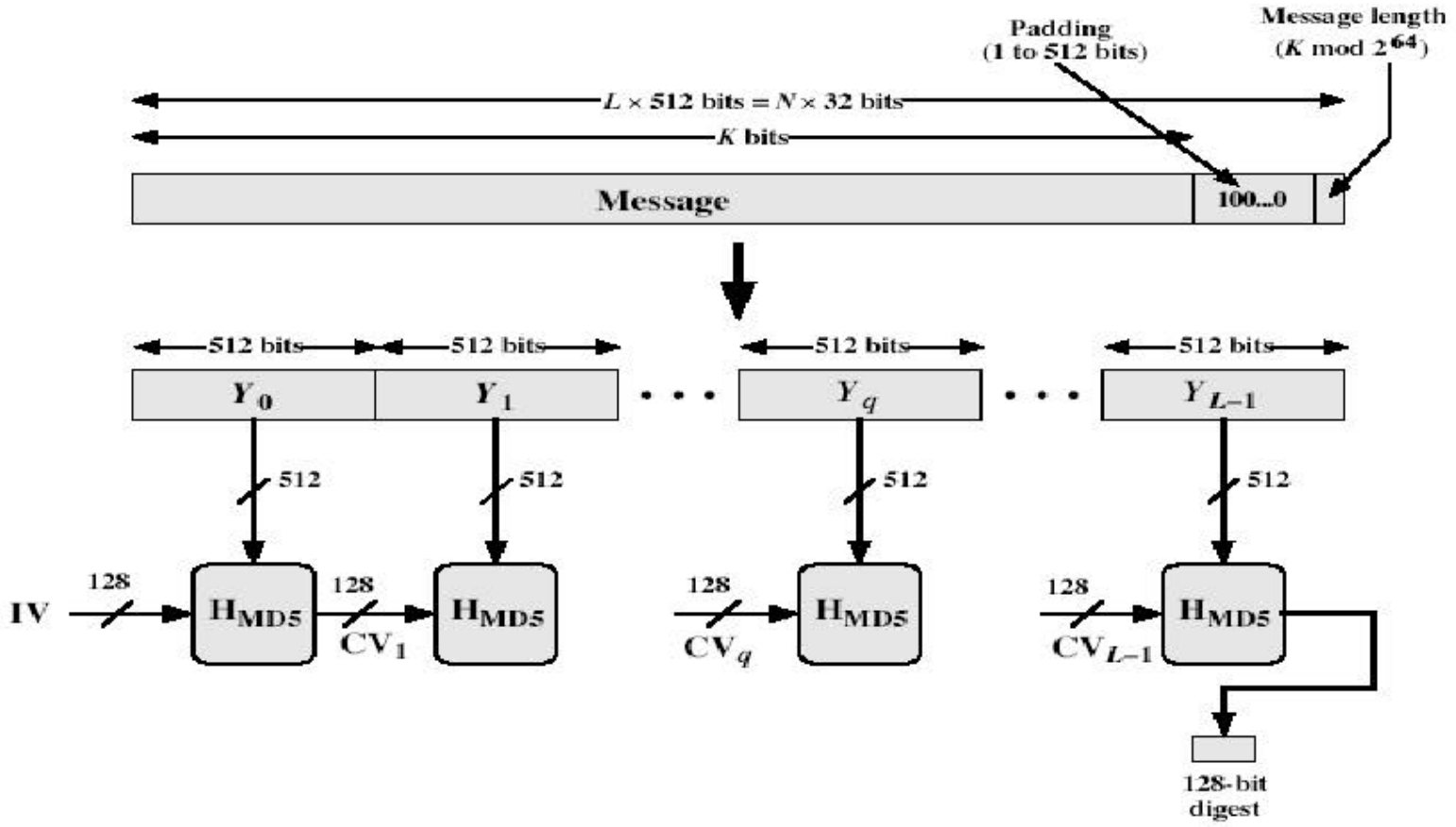
- Designed by Ronald Rivest (the R in RSA)
- Latest in a series of MD2, MD4
- Produces a 128-bit hash value
- Until recently was the most widely used hash algorithm in recent times have both brute-force & cryptanalytic concerns
- Specified as Internet standard RFC1321

MD5 Overview

1. Pad message so its length is 448 mod 512
2. Append a 64-bit length value to message
3. Initialize 4-word (128-bit) MD buffer (A,B,C,D)
4. Process message in 16-word (512-bit) blocks:
   - using 4 rounds of 16 bit operations on message block & buffer
   - add output to buffer input to form new buffer value
5. Output hash value is the final buffer value

kasun@cmb.ac.lk

MD5 Overview



kasun@cmb.ac.lk

Strength of MD5

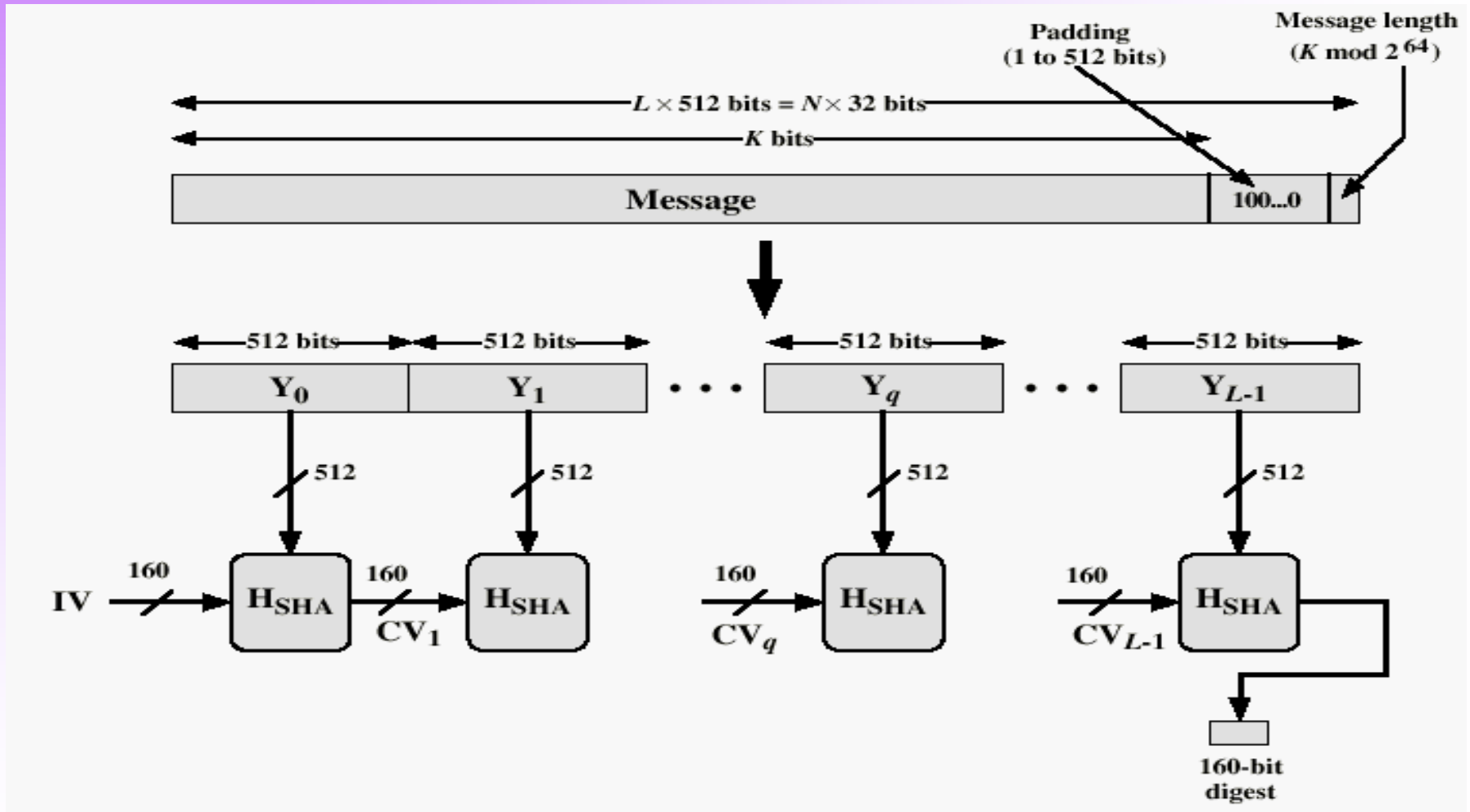
- MD5 hash is dependent on all message bits
- Rivest claims security is good as can be
- Known attacks are:
  - Berson (92) attacked any 1 round using differential cryptanalysis (but can't extend)
  - Boer & Bosselaers (93) found a pseudo collision (again unable to extend)
  - Dobbertin (96) created collisions on MD compression function (but initial constants prevent exploit)
  - Crypto 2004 attacks on SHA-0 and MD5
- Conclusion is that MD5 has been shown to be vulnerable

kasun@cmb.ac.lk

Secure HASH Functions

- Purpose of the HASH function is to produce a ”fingerprint.
- Properties of a HASH function H :
  1. H can be applied to a block of data at any size
  2. H produces a fixed length output
  3. H(x) is easy to compute for any given x.
  4. For any given block x, it is computationally infeasible to find x such that H(x) = h
  5. For any given block x, it is computationally infeasible to find $y \neq x$ with H(y) = H(x).
  6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y)

Message Digest Generation Using SHA-1



kasun@cmb.ac.lk

Comparision-Secure HASH functions

	SHA-1	MD5	RIPEMD-160
Digest length	160 bits	128 bits	160 bits
Basic unit of processing	512 bits	512 bits	512 bits
Number of steps	80 (4 rounds of 20)	64 (4 rounds of 16)	160 (5 paired rounds of 16)
Maximum message size	2^{64}-1 bits		

Keyed Hash Functions (HMAC)

- Create a MAC using a hash function rather than a block cipher
  - because hash functions are generally faster
  - not limited by export controls unlike block ciphers
  - Hash includes a key along with the message
- Original proposal:

$KeyedHash = Hash(Key|Message)$

  - some weaknesses were found with this
- Eventually led to development of HMAC

kasun@cmb.ac.lk

HMAC Design Criteria

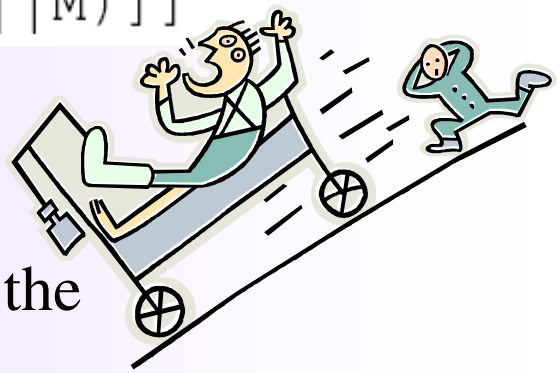
- To use, without modifications, available hash functions.
- To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required.
- To preserve the original performance of the hash function without incurring a significant degradation.
- To use and handle keys in a simple way.
- To have a well-understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions on the embedded hash function.

kasun@cmb.ac.lk

HMAC

- Specified as Internet standard RFC2104
- Uses hash function on the message:

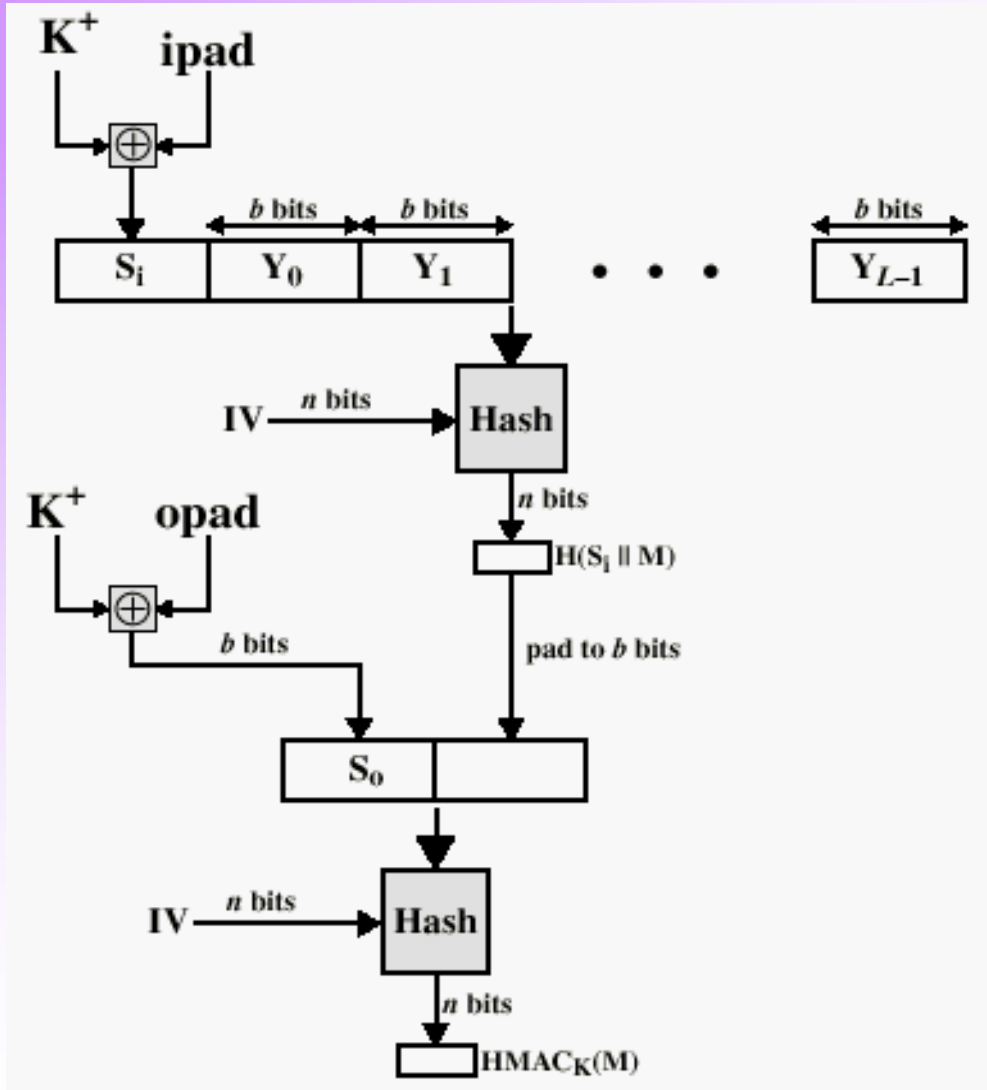
$$HMAC_K = Hash[(K^+ \text{ XOR } opad) \,||\, Hash[(K^+ \text{ XOR } ipad)||M)]]$$



- K+ is the key padded out to size
- opad, ipad are specified padding constants
- Overhead is just 3 more hash calculations than the message needs alone
- Any of MD5, SHA-1, RIPEMD-160 can be used

kasun@cmb.ac.lk

HMAC Structure



UCSC

kasun@cmb.ac.lk

HMAC Security

- know that the security of HMAC relates to that of the underlying hash algorithm
- attacking HMAC requires either:
  - brute force attack on key used
  - birthday attack (but since keyed would need to observe a very large number of messages)
- choose hash function used based on speed verses security constraints

kasun@cmb.ac.lk

Secure Encryption Systems

2.4 Secure Secret Key (Symmetric) Systems

- The Data Encryption Standard (DES)
- Analyzing and Strengthening of DES
- Key Escrow and Clipper
- Advance Encryption Standard (AES)
- Introduction to Quantum Cryptography

kasun@cmb.ac.lk

Requirements for Symmetric Key Cryptography

Two requirements for secure use of symmetric encryption:

- a strong encryption algorithm
- a secret key, K, known only to sender / receiver

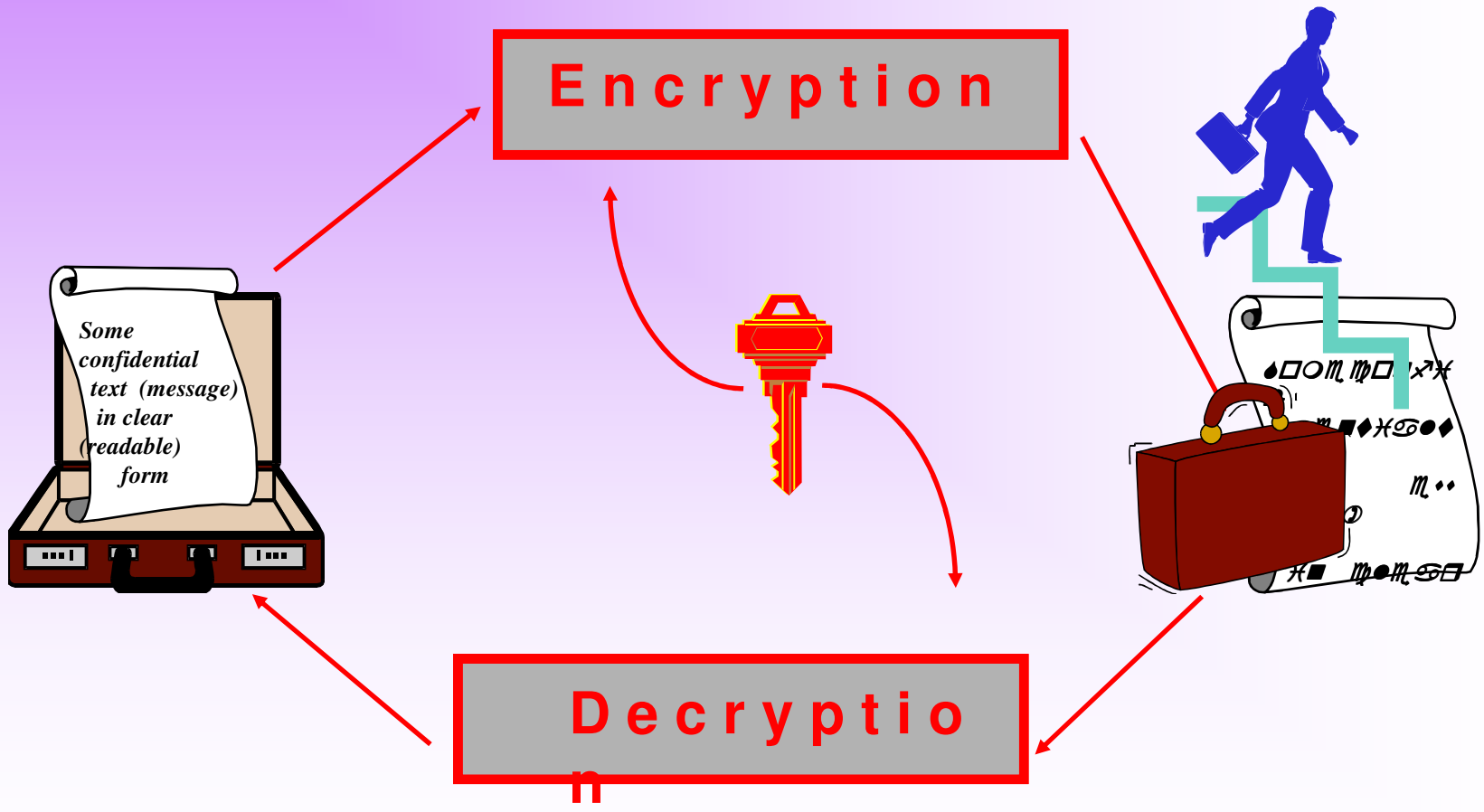
$Y = EK(X)$

$X = DK(Y)$

- Assume encryption algorithm is known
- Implies a secure channel to distribute key

kasun@cmb.ac.lk

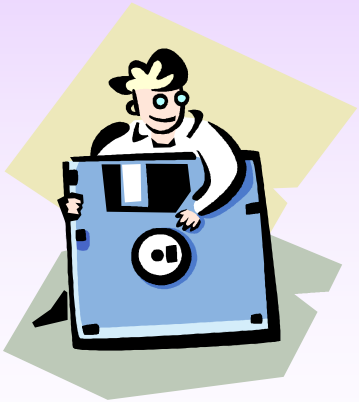
Symmetric key Cryptograms



kasun@cmb.ac.lk

Data Encryption Standard (DES)

- Most widely used block cipher in world
- Adopted in 1977 by NBS (now NIST) as FIPS PUB 46
- Encrypts 64-bit data using 56-bit key
- Has widespread use
- Has been the subject of considerable controversy over its security



kasun@cmb.ac.lk

The Data Encryption Standard- (DES)

Background and History

- National Bureau of Standards (NBS), issued call for a public encryption algorithm in August 1974
- DES based on Lucifer from IBM
- DES is officially adapted as a U.S. federal standard on 23rd November 1976

Overview of DES

Plain text is encrypted as block of 64 bits

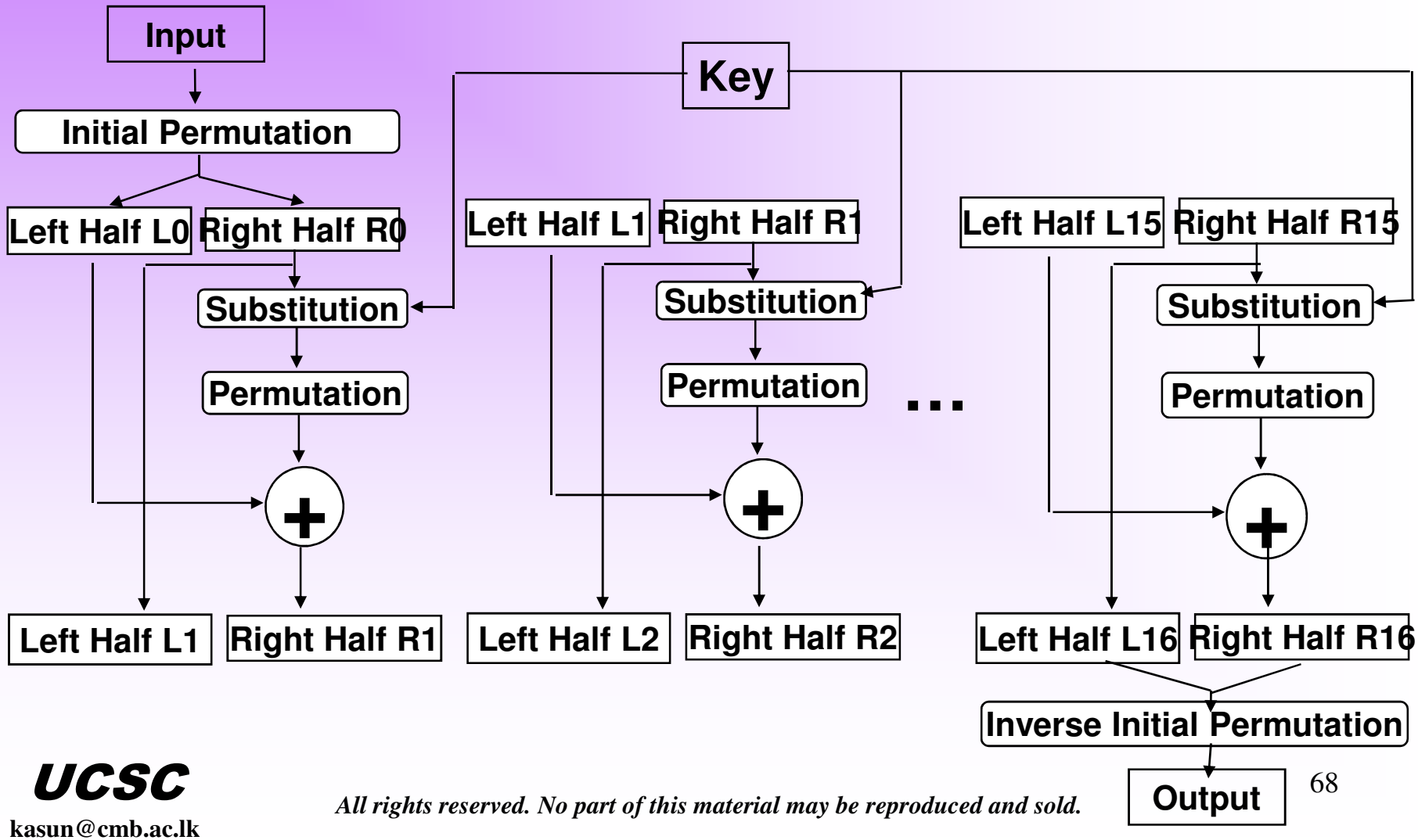
- Normal key length 56 bits
- Algorithm derived from two concepts of Shannon's theory, confusion and diffusion

Confusion - Piece of information changed, so that the output bits have no relationship to input bits

Diffusion - Attempts to spread the effect of one plain test bit to other bits in the cipher text

kasun@cmb.ac.lk

DES



Strength of DES – Key Size

- 56-bit keys have $2^{56} = 7.2 \times 1016$ values
- Brute force search looks hard
- Recent advances have shown that this is possible
  - in 1997 on Internet in a few months
  - in 1998 on DES Cracker dedicated h/w (EFF) in a less than 3 days (cost: $250,000)
  - in 1999 above combined in 22hrs!

Now we have alternatives to DES

kasun@cmb.ac.lk

Weakness of the DES

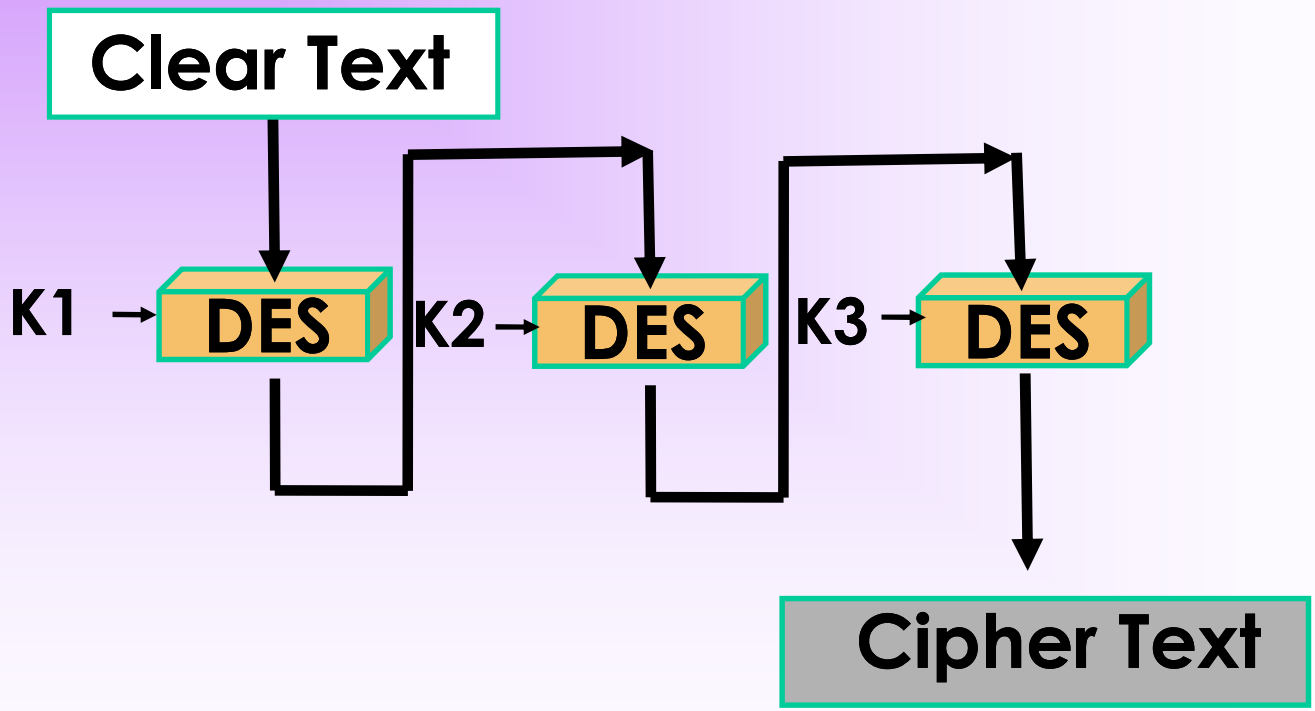
- Complements
- Weak Keys

Eg: 0101 0101 0101 0101
FEFE FEFE FEFE FEFE
1F1F 1F1F 1F1F 1F1F
E0E0 E0E0 E0E0 E0E0

- Semi-Weak Keys
- Design Weakness
- Key Clustering

UCSC
kasun@cmb.ac.lk

Triple DES



kasun@cmb.ac.lk

Triple-DES with Two-Keys

- Use 3 encryptions

would seem to need 3 distinct keys

But can use 2 keys with E-D-E sequence

C = EK1[DK2[EK1[P]]]

Note: encrypt & decrypt equivalent in security

if K1=K2 then can work with single DES

- Standardized in ANSI X9.17 & ISO8732
- No current known practical attacks

kasun@cmb.ac.lk

DES- AES

- Clearly, a replacement for DES was needed
  - have theoretical attacks that can break it
  - have demonstrated exhaustive key search attacks
- Can use Triple-DES – but slow with small blocks
- NIST issued a call for ciphers in 1997
- 15 candidates accepted in June 1998
- 5 were short listed in August 1999
- Rijndael was selected as the AES in October 2000
- Issued as FIPS PUB 197 standard in November 2001

kasun@cmb.ac.lk

AES Requirements

- Private key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger & faster than Triple-DES
- Active life of 20-30 years (+ archival use)
- Provide full specification & design details
- Both C & Java implementations
- NIST has released all submissions & unclassified analyses

kasun@cmb.ac.lk

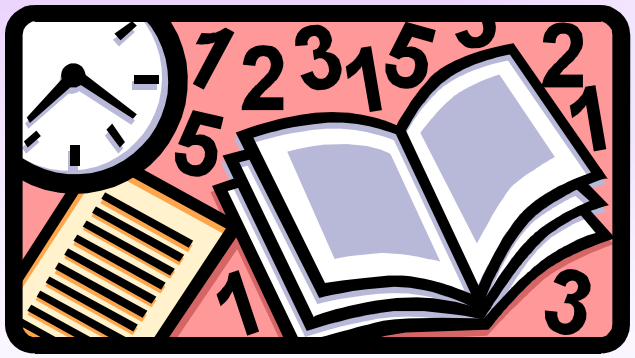
AES Shortlist

- After testing and evaluation, shortlist in August 1999:
  - MARS (IBM) - complex, fast, high security margin
  - RC6 (USA) - v. simple, v. fast, low security margin
  - Rijndael (Belgium) - clean, fast, good security margin
  - Serpent (Euro) - slow, clean, v. high security margin
  - Twofish (USA) - complex, v. fast, high security margin
- Then subject to further analysis & comment
- Saw contrast between algorithms with
  - few complex rounds verses many simple rounds
  - which refined existing ciphers verses new proposals

kasun@cmb.ac.lk

Advance Encryption Standard (AES)

- In 2001, National Institute of Standards and Technology (NIST) issued AES known as FIPS 197
- AES is based on Rijndael proposed by Joan Daemen, Vincent Rijmen from Belgium



kasun@cmb.ac.lk

Advance Encryption Standard (AES)

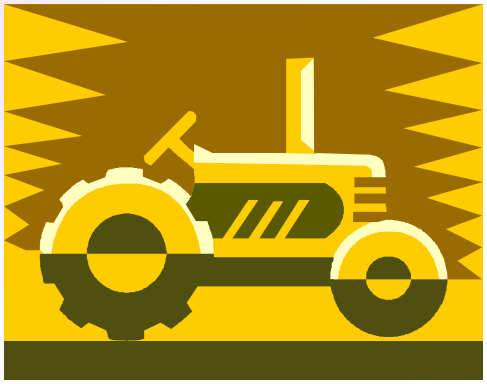
- AES has block length 128
- Supported key lengths are 128, 192 and 256
- AES requires 10 rounds of processing
- Key is expanded into 10 individual keys
- Decryption algorithm uses the expanded keys in reverse order
- Decryption algorithm is not identical to the encryption algorithm

kasun@cmb.ac.lk

Block Ciphers- Modes of Operation

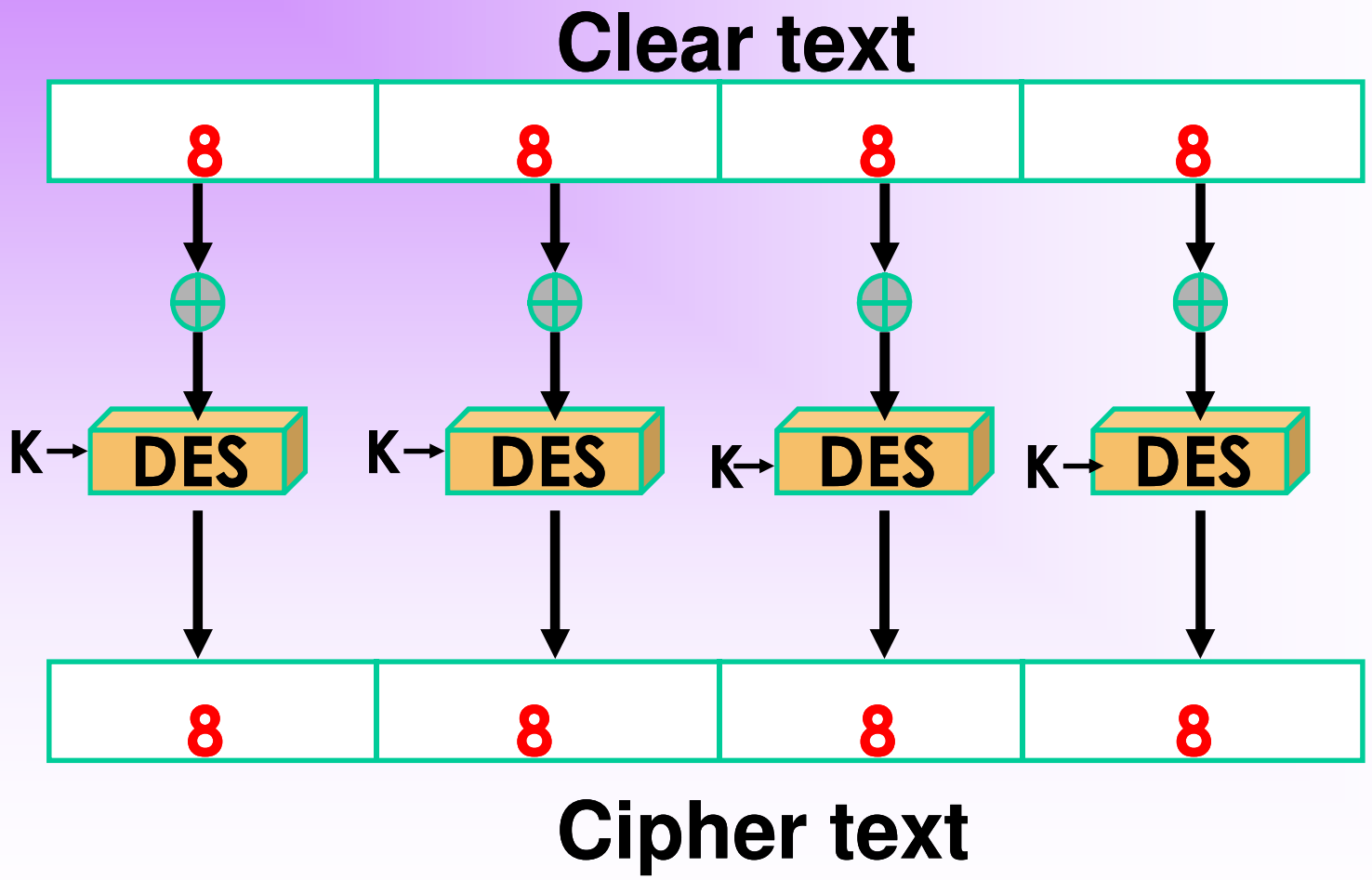
- Block ciphers encrypt fixed size blocks
  - E.g. DES encrypts 64-bit blocks, with 56-bit key

- Given that one needs to encrypt arbitrary amount of information, how do we use in practice,
  - Four modes were defined for DES in ANSI standard
  - **ANSI X3.106-1983 Modes of Use**
  - Subsequently now have 5 for DES and AES



kasun@cmb.ac.lk

Electronic Code Book Mode (ECB)



UCSC
kasun@cmb.ac.lk

Electronic Codebook Book (ECB)

- Message is broken into independent blocks which are encrypted
- Each block is a value which is substituted, like a codebook, hence name
- Each block is encoded independently of the other blocks

$$C_i = DES_K(P_i)$$

- Uses: secure transmission of single values

kasun@cmb.ac.lk

Advantages and Limitations of ECB

- Repetitions in message may show in ciphertext if aligned with message block particularly with data such graphics or with
- Messages that change very little
- Weakness due to encrypted message blocks being independent
- Main use is sending a few blocks of data

kasun@cmb.ac.lk

Cipher Block Chaining (CBC)

- Message is broken into blocks
- But these are linked together in the encryption operation
- Each previous cipher blocks is chained with current plaintext block, hence name
- Use Initial Vector (IV) to start process

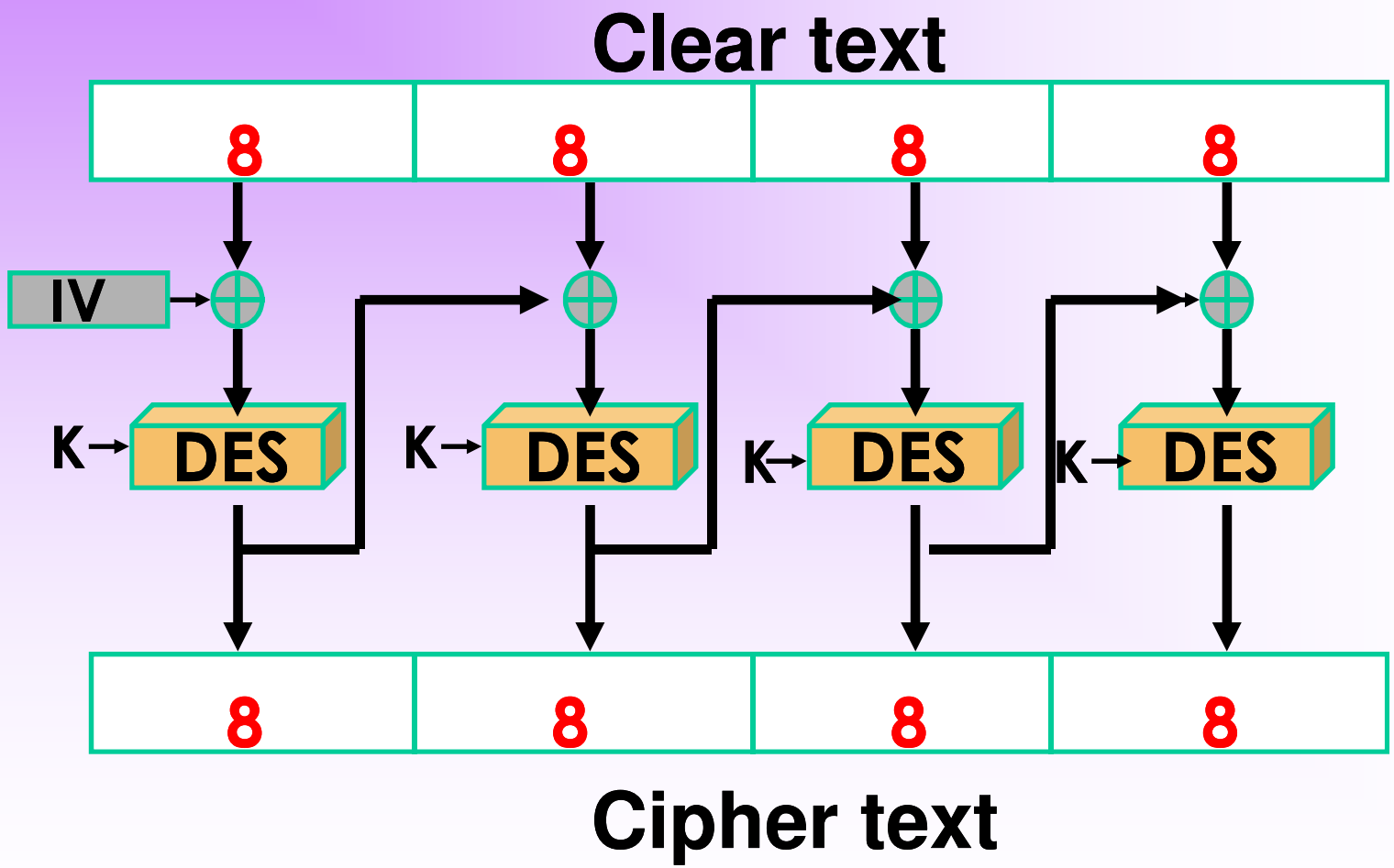
$$C_i = DES_K(P_i \; XOR \; C_{i-1})$$

$$C_{-1} = IV$$

- Uses: bulk data encryption, authentication

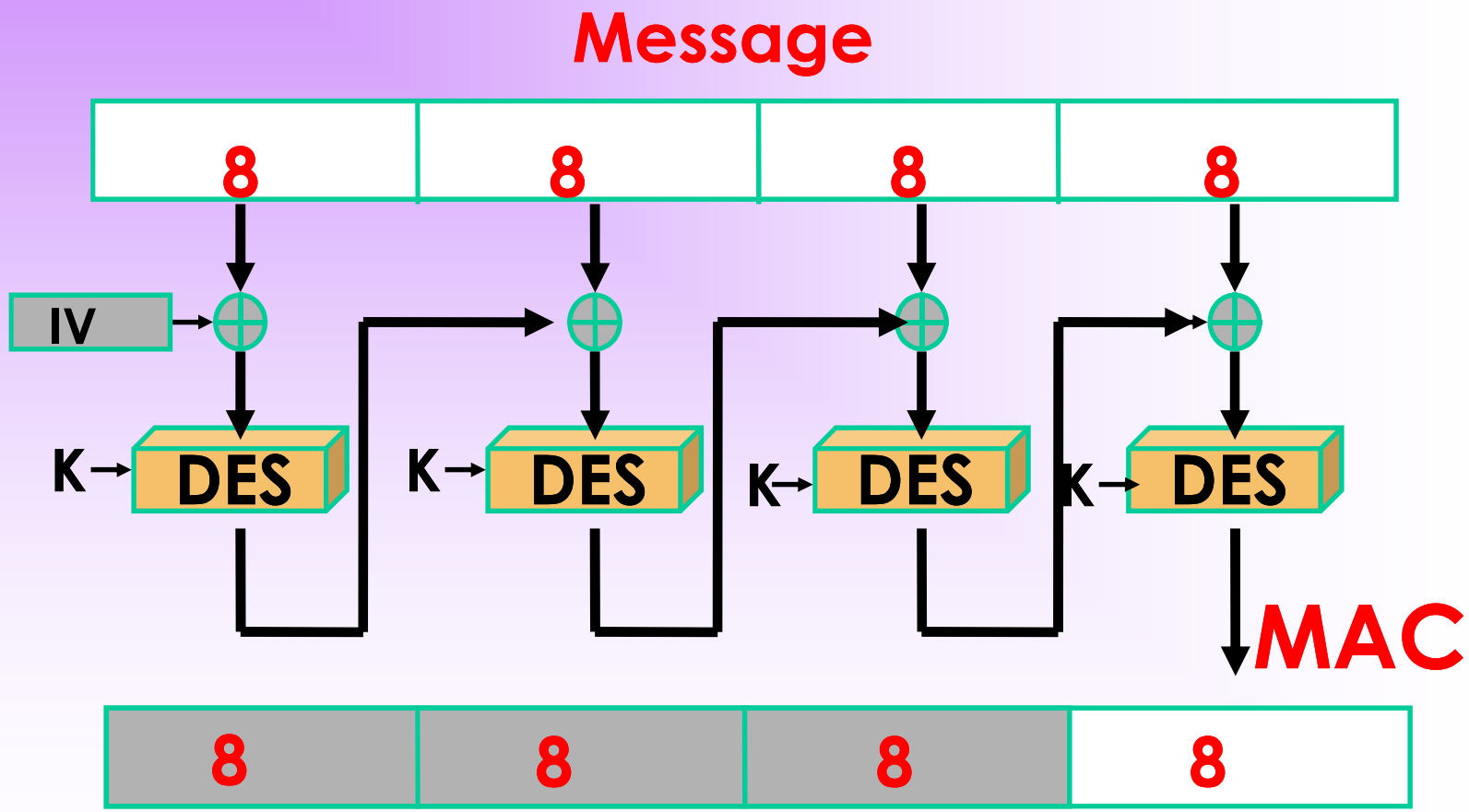
kasun@cmb.ac.lk

Cipher Block Chaining Mode (CBC)



kasun@cmb.ac.lk

MAC based on CBC



kasun@cmb.ac.lk

Advantages and Limitations of CBC

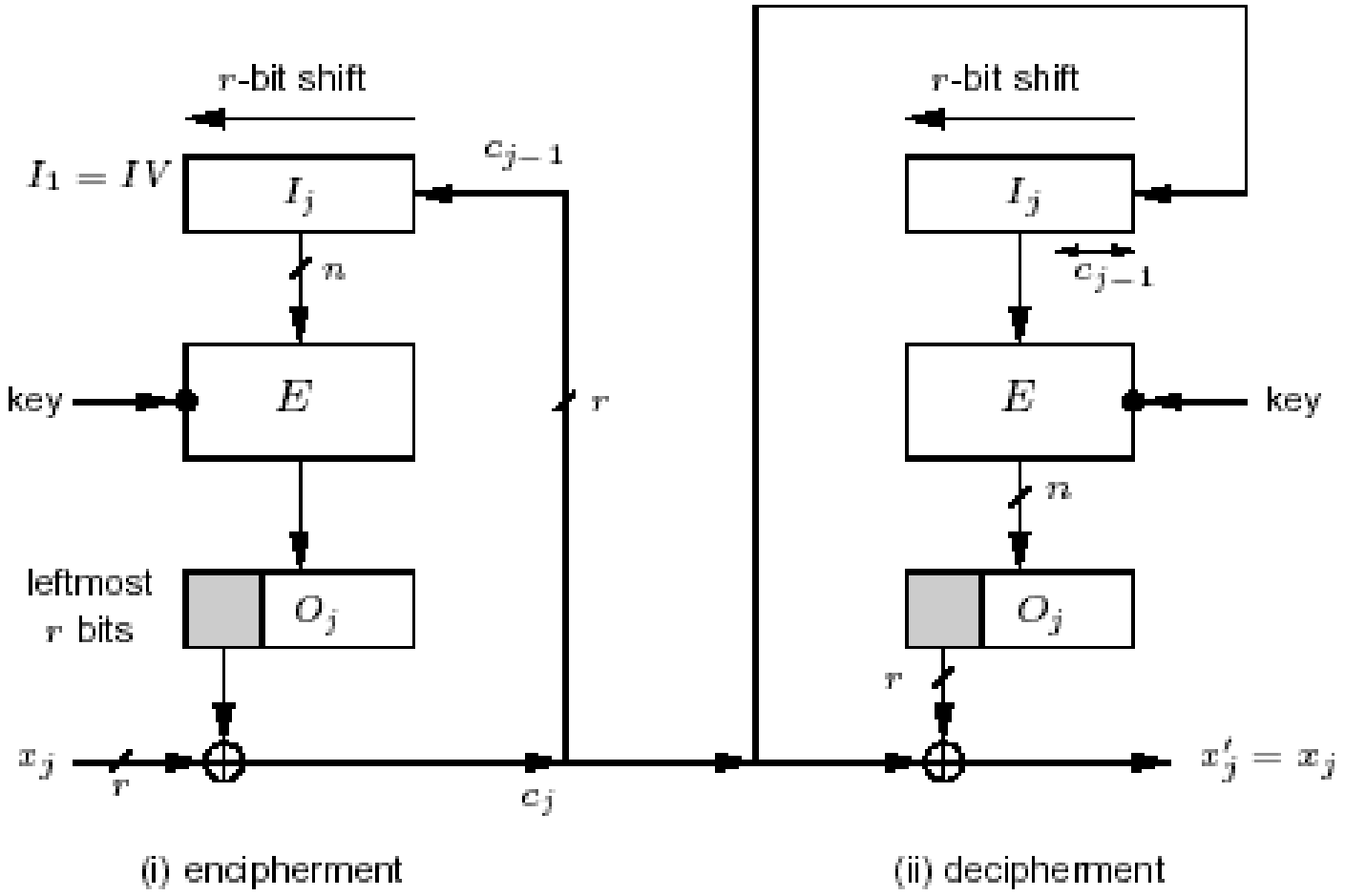
- Each ciphertext block depends on **all** preceding message blocks thus a change in the message affects all ciphertext blocks after the change as well as the original block

- Need **Initial Value** (IV) known to sender & receiver however if IV is sent in the clear, an attacker can change bits of the first block, and change IV to compensate hence either IV must be a fixed value or it must be sent encrypted in ECB mode before rest of message

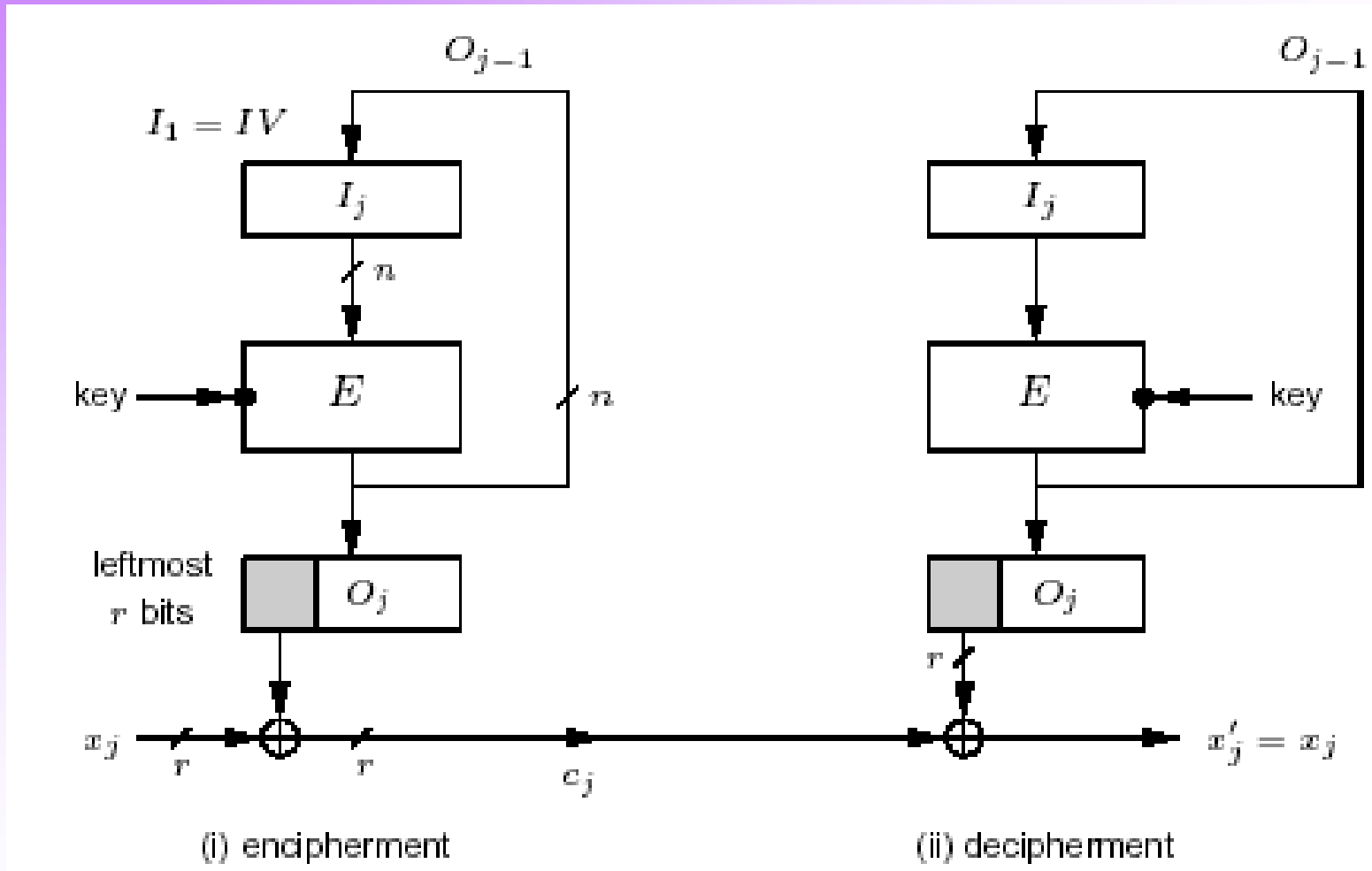
- At end of message, handle possible last short block by padding either with known non-data value (e.g. nulls) or pad last block with count of pad size

kasun@cmb.ac.lk

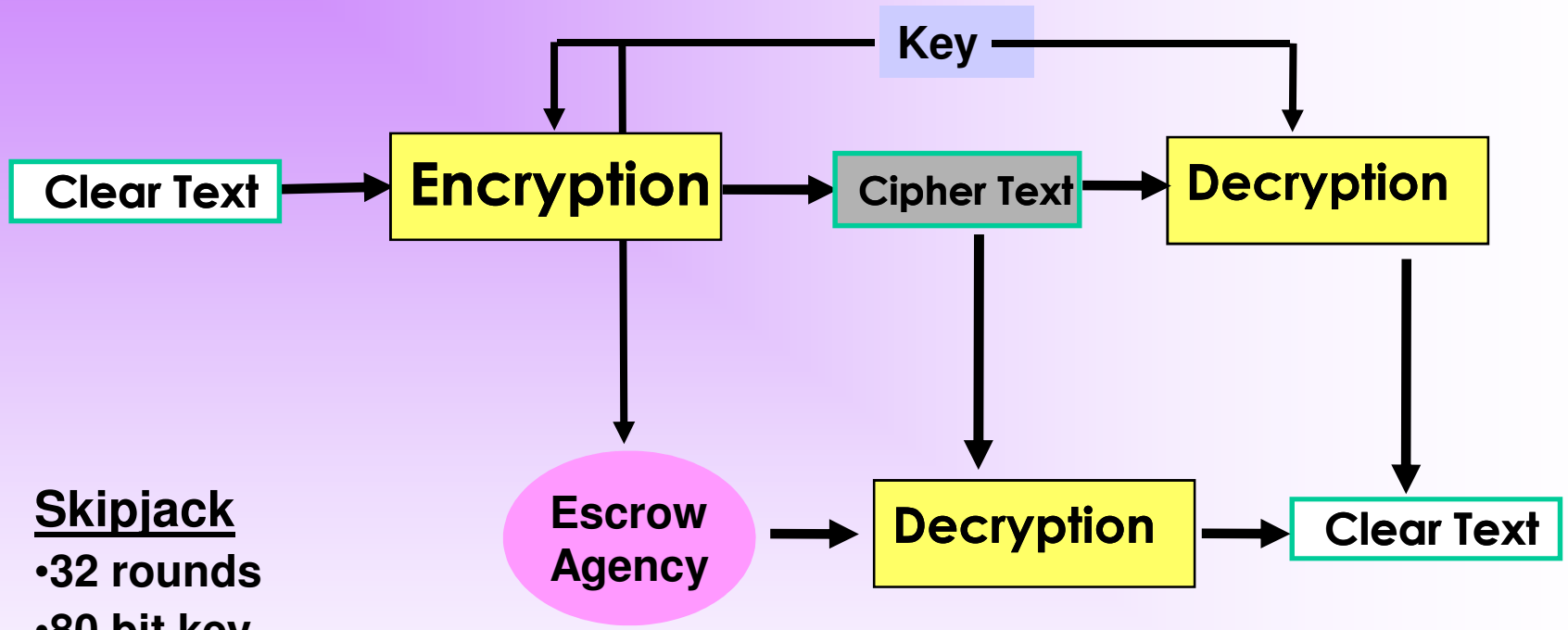
Cipher Feedback Mode (CFB)



Output Feedback Mode (OFB)



Key Escrow Standard



Skipjack

- 32 rounds
- 80 bit key
- 64 bit block of plain text

kasun@cmb.ac.lk

Other Symmetric Block Ciphers

- International Data Encryption Algorithm (IDEA)
  - 128-bit key
  - Used in PGP
- Blowfish
  - Easy to implement
  - High execution speed
  - Run in less than 5K of memory

kasun@cmb.ac.lk

Other Symmetric Block Ciphers

- RC5
  - Suitable for hardware and software
  - Fast, simple
  - Adaptable to processors of different word lengths
  - Variable number of rounds
  - Variable-length key
  - Low memory requirement
  - High security
  - Data-dependent rotations
- Cast-128
  - Key size from 40 to 128 bits
  - The round function differs from round to round

kasun@cmb.ac.lk

Stream Ciphers

- Process the message bit by bit (as a stream)
- Typically have a (pseudo) random **stream key**
- Combined (XOR) with plaintext bit by bit
- Randomness of **stream key** completely destroys any statistically properties in the message

$C_i = M_i$ XOR $StreamKey_i$

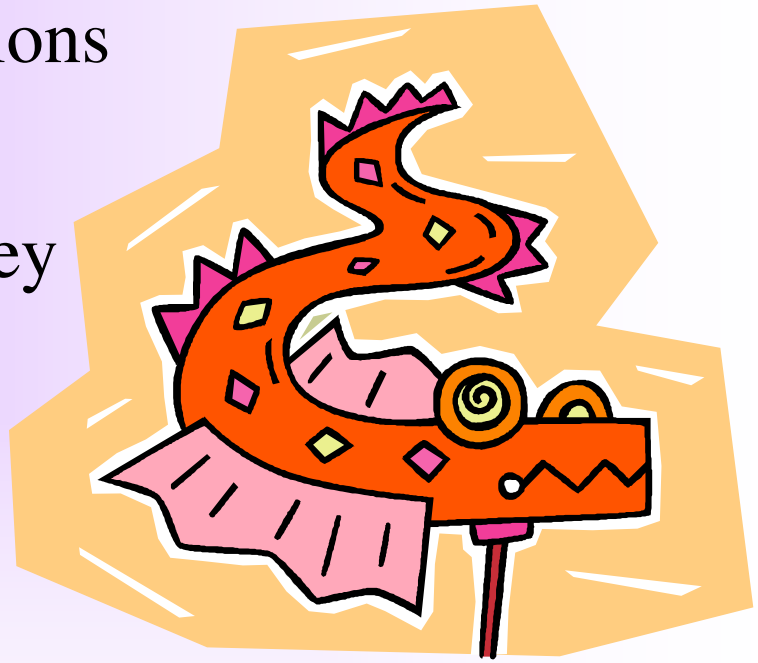
- But must never reuse stream key

otherwise can remove effect and recover messages

kasun@cmb.ac.lk

Stream Cipher Properties

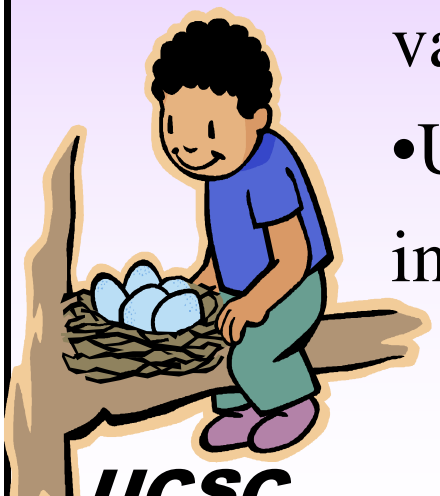
- Some design considerations are:
  - long period with no repetitions
  - statistically random
  - depends on large enough key
  - large linear complexity
  - correlation immunity
  - confusion
  - diffusion
  - use of highly non-linear Boolean functions



kasun@cmb.ac.lk

RC4

- A proprietary cipher owned by RSA DSI
- Another Ron Rivest design, simple but effective
- Variable key size, byte-oriented stream cipher
- Widely used (web SSL/TLS, wireless WEP)
- Key forms random permutation of all 8-bit values
- Uses that permutation to scramble input information processed a byte at a time



kasun@cmb.ac.lk

RC4 Security

- Claimed secure against known attacks
  - have some analyses, none practical
- Result is very non-linear
- Since RC4 is a stream cipher, must **never reuse a key**

kasun@cmb.ac.lk

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*

kasun@cmb.ac.lk

The Main contribution of Quantum Cryptography.

- It solved the key distribution problem.
- Unconditionally secure key distribution method proposed by Charles Bennett and Gilles Brassard in 1984. The method is called BB84.
- Once key is securely received it can be used to encrypt messages transmitted by conventional channels.

UCSC
kasun@cmb.ac.lk

Security of quantum key distribution

- Quantum cryptography obtains its fundamental security from the fact that information is carried by a single photon, and each photon will be altered as soon as it is read.
- This makes impossible to intercept message without being detected.

UCSC
kasun@cmb.ac.lk

Quantum Computing algorithm for factoring.

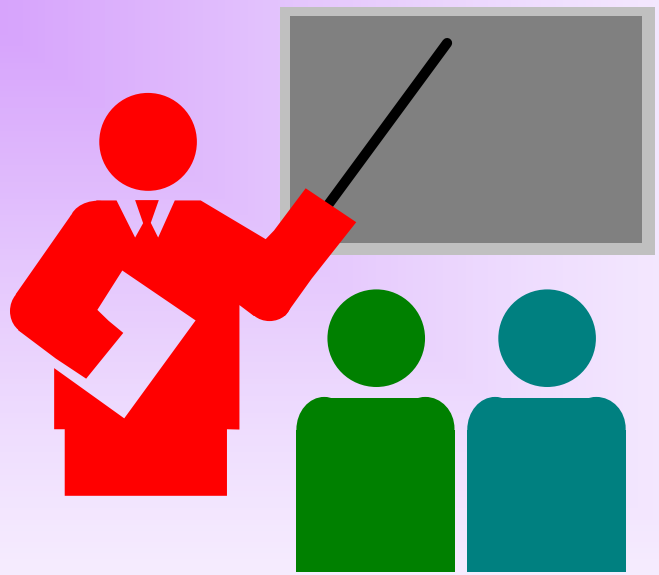
- In 1994 Peter Shor from the AT&T Bell Laboratory showed that in principle a quantum computer could factor a very long product of primes in seconds.
- Shor's algorithm time computational complexity is

$$T(n) = O[(\ln n)^3]$$

Once a quantum computer is built the RSA method would not be safe.

UCSC
kasun@cmb.ac.lk

Questions???



kasun@cmb.ac.lk