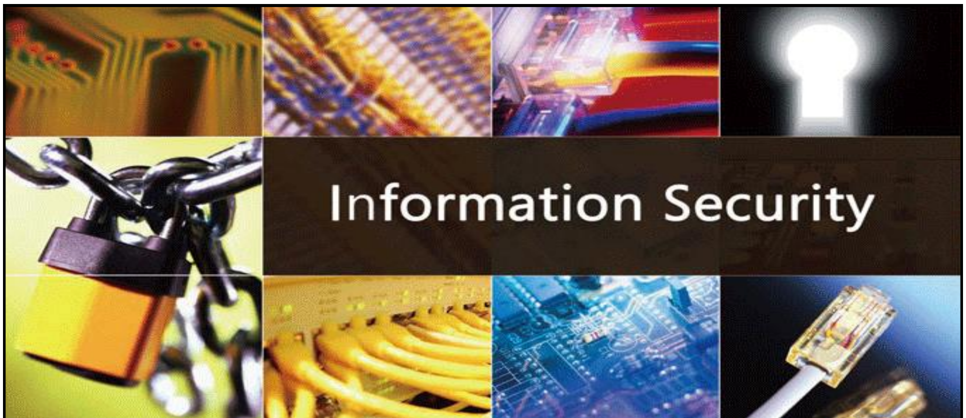


Asymmetric encryption

Lecturer: Nguyễn Thị Thanh Vân – FIT - HCMUTE

Objective

- Asymmetric encryption
- Modular arithmetic
- RSA
- Diffie-Hellman Key Exchange
- Message Authentication - MAC
- Cryptographic Hash Functions
- Digital Signatures

Asymmetric encryption

- ***Asymmetric encryption* is a form of cryptosystem in which *encryption* and *decryption* are performed using the different keys**
  - **a public key**
  - **a private key.**

- **It is also known as *public-key encryption***

Private – key Cryptography

- Private – key Cryptography also called **secret/ single key/ symmetric/ conventional.**
- It uses **ONLY ONE** key shared with both Recipient and Sender.
- *Private– key Cryptography looks like sealed box with message inside.*
- Private – key Cryptography's disadvantages:
  - Needs of secure channel to exchange keys
  - Each pair of users have to share one secret key. So the number of keys for N users should be N(N-1)/2 : so many keys!
  - Solution: **Using Public – key Cryptography**

Public – key Cryptography

- Cryptography with **public – key/2 keys/asymmetric** uses **TWO** keys that have one owner**:**
  - **Public - key**,
    - everyone can know and
    - use to **encrypt the message** or
    - to **check the signature** of key's owner.
  - **Private – key**:
    - only owner knows and
    - use to **decrypt the message** or
    - to **create the signature**

5

Public – key Cryptography

- In **asymmetric cryptography,** role of sender and recipient are **not** same:
  - *Person who encrypt message either check the signature*
  - *that can not be decrypted or create the signature.*

- Mathematical basis: *One-way functions*
  - ***$y = f(x)$ is the one – way function if $y = f(x)$ is easy to calculate but $x = f^{-1}(y)$ is difficult to find***
  - *$x = f^{-1}(y)$ might be easy to calculate if given additional information (key)*

6

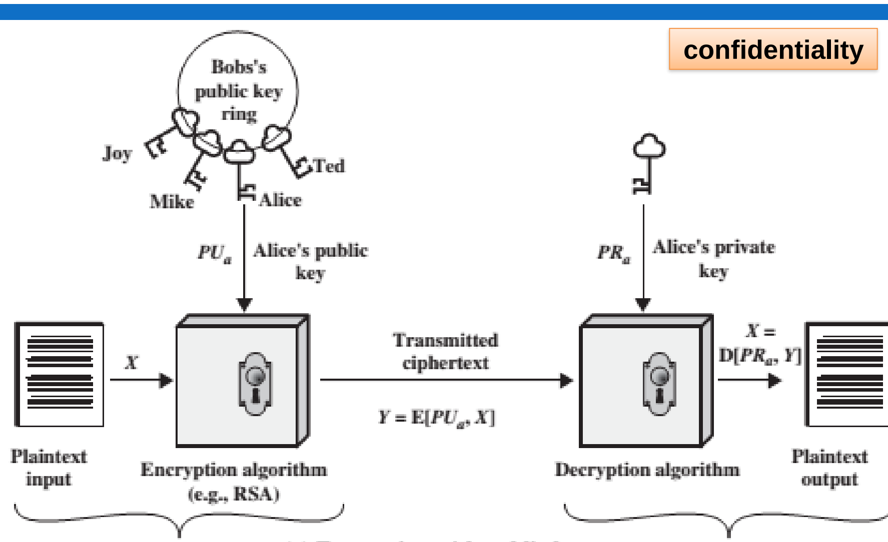
Public key theory

- **Public key** can be calculated from **private key** and other information of cryptography (**P problem**)
- However, if knowing the public key and the ciphertext cannot calculate the private key (**NP problem**)
- Public key needs to be distributed safely for everyone, who needs securely send message to key's owner
- Problem of public key distribution is important – that is **key distribution problem**

7

Encryption with public key

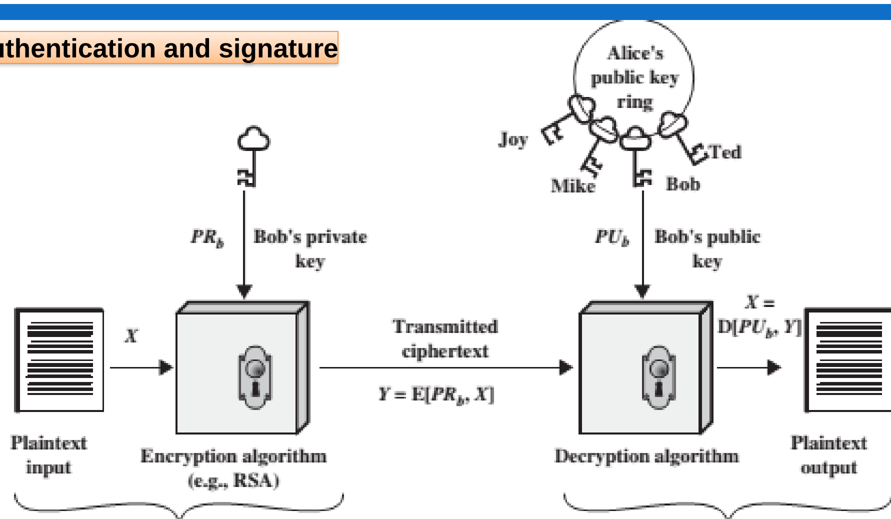
confidentiality



(a) Encryption with public key

Encryption with private key

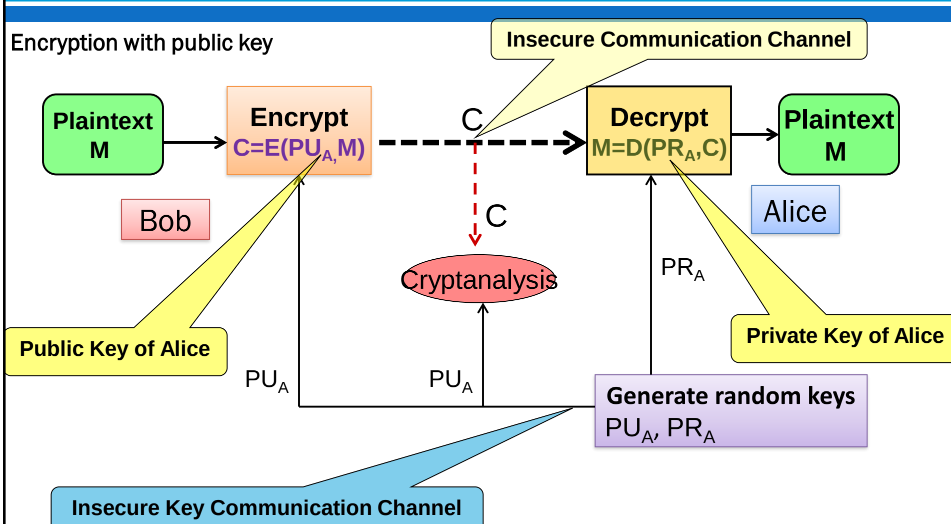
authentication and signature



(b) Encryption with private key

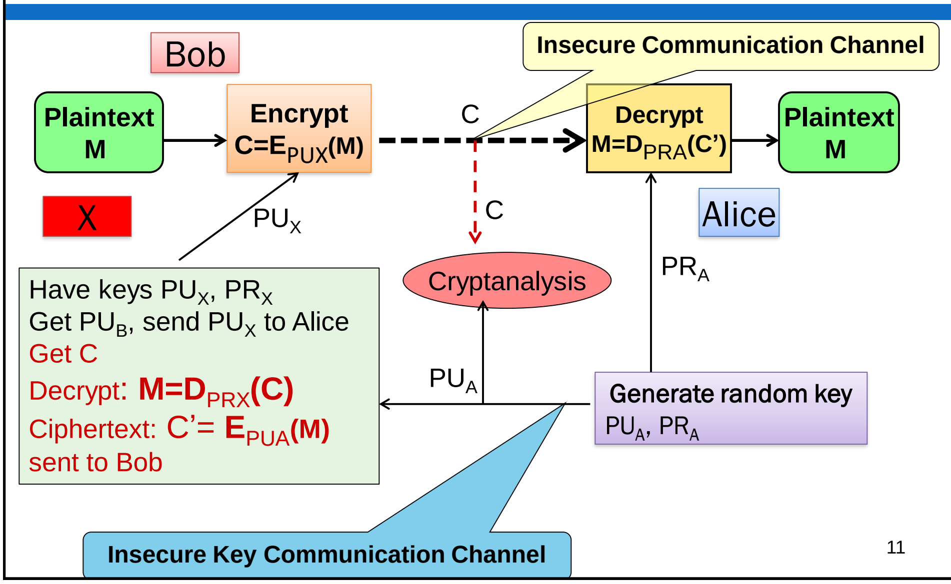
Public – key Cryptography: Model

Encryption with public key



10

Public – key Cryptography: Model



Public key cryptography Security

- Security based on **the difference** between the hardness of encryption/decryption problem (**easy**) and cryptanalysis problem (**hard**)
- Cryptanalysis using **key exhaustive key search** is always done theoretically. But in fact, the number of used keys is too large for it (>512 bit)
- To resist some other **advanced cryptanalysis methods**, need to use **the very large keys** (>>512 bit)
- Therefore implementation of public key cryptography is much slower than the secret key cryptography

12

Problems of public key cryptography

Encryption using public key:
- Using to encrypt message then sent it to key owner
- Everyone can use **public key to encrypt**
- The owner uses **private key to decrypt**

○ **=> Ensuring the confidentiality of message**

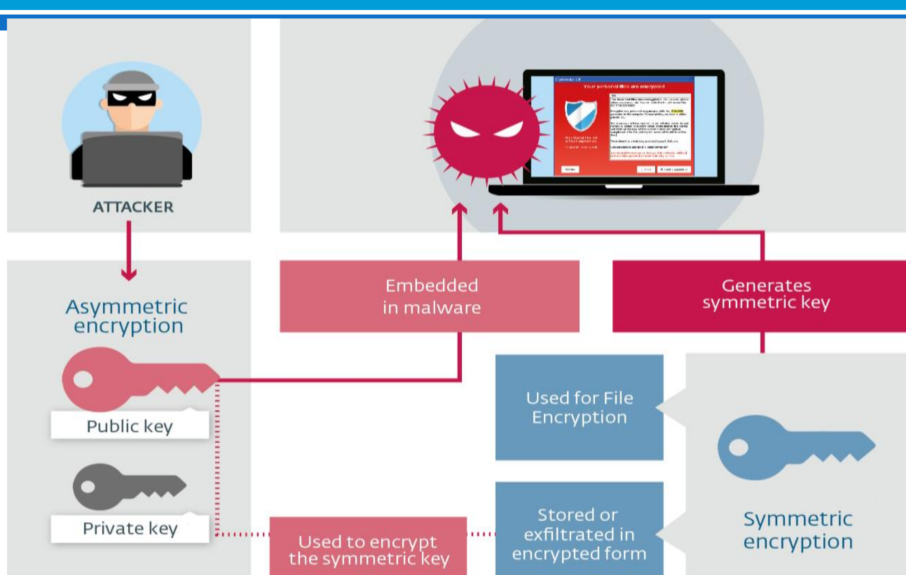
Encryption using private key:
- Using to create signature for message
- Owner uses **private key** to sign message
- Everyone uses **public key to check the signature**

○ **=> Ensuring the authentication of the message**

Public Key Distribution Scenario (PKDS):
- Methods of **public key distribution**
- Using **PK cryptography to exchange private keys**

13

Ex, Symmetric /Asymmetric encryption



Modular Arithmetic

- Modulo, modulus
- Congruence modulo
- Properties (addition, subtraction, multiplication, exponentiation)
- Modular inverse (additive, multiplicative)
- Totient (Euler's phi) function

Modular, congruence

- In mathematics, **modular arithmetic** is a system of arithmetic for integers, where numbers "wrap around" upon reaching a certain value—the **modulus** (plural **moduli**).
- The modern approach to modular arithmetic was developed by Carl Friedrich Gauss
- Ex: Instead of 13 = 1.
  - write 13 ≡ 1 (mod 12) and read it "13 is congruent to 1 modulo 12" or, to abbreviate, "13 is 1 modulo 12".
- Examples:
  - 12 ≡ 0 (mod 12); 17 ≡ 5 (mod 12)
    37 ≡ 1 (mod 12); -1 ≡ 11 (mod 12)
- In general, a ≡ b (mod n) if a-b is a multiple of n.
  Equivalently, a ≡ b (mod n) if a and b have the same remainder when divided by n (remainder modulo n)

Properties
(addition, subtraction, multiplication, exponentiation)

- mod n = 6
  - 4 + 3 = 7 - 6 = 1
  - 3 – 5 = -2 + 6 = 4
  - 4 * 5 = 20 % 6 = 2
  - 5^{13} (mod 6)
    - 5 - 6 ≡ -1, -1^{13} ≡ -1 (mod 6)

Associated with addition, multiplicative

- Additive

$$\begin{array}{rcll} \textit{If} \quad a & \equiv & b \pmod{m} & \textit{and} \\ c & \equiv & d \pmod{m}, & \textit{then} \\ a+c & \equiv & b+d \pmod{m}. & \end{array}$$

- Multiplicative

$$\begin{array}{rcll} \textit{If} \quad a & \equiv & b \pmod{m} & \textit{and} \\ c & \equiv & d \pmod{m}, & \textit{then} \\ a \times c & \equiv & b \times d \pmod{m}. & \end{array}$$

Ex, Additive

- Examples:
  - 7 + 8 ≡ 3 (mod 12); 10 + 2 ≡ 0 (mod 12)
    13 + 2 ≡ 3 (mod 12); -1 + 14 ≡ 1 (mod 12)
- Ex, 19 + 23 + 15 ≡ ? (mod 12)
  - First replace each number by its remainder mod 12:
    7 + 11 + 3. then do the sum: 21
  - and replace the sum by its remainder modulo 12: 9
    19 + 23 + 15 ≡ 9 (mod 12)
- If today is Sunday, what day will it be in 1000 days?
  We need to find the remainder of 1000 when divided by 7
  - 1000: = 700+280+20

ex

- Examples multiplicative:
  - 7 x 4 ≡ 8 (mod 10);
  - 19 x 28 ≡ 2 (mod 10): 9x8 = 72 ≡ 2 (mod 10)
  - -2 x 6 ≡ 8 (mod 10)
- Ex, exponentiation
  - $x=3^2$ mod 13 . $x = 3^2$ mod 13 = 9
  - $x = 3^{10}$ mod 13 = $3^4 \cdot 3^4 \cdot 3^2$ mod 13 = 9^2 . $9^2 \cdot 9$ mod 13 =
    (6.13+3). (6.13+3). 9 mod 13= 3.3.9 mod 13 = 3
- Ex, exponentiation: 17^2 (mod 19)
  - 17^2= 289, divide that by 19 and then take the remainder 4.
  - However, since we know that 17 ≡ -2 (mod 19), we can multiply this congruence equation by itself to obtain $17^2 \equiv -2^2 \equiv 4$ (mod 19).
- Ex: $x= 18^{489391312}$ (mod 19)

Greatest Common Divisor (GCD)

- GCD: a common problem in number theory
- GCD (a,b) of a and b is the largest number that divides evenly into both a and b
- Ex, GCD(60,24) = 12
- often want no common factors (except 1) and hence numbers are relatively prime
- Ex, GCD(8,15) = 1, hence 8 & 15 are relatively prime

Euclidean Algorithm

- an efficient way to find the GCD(a,b):
  GCD(a,b) = GCD(b, a mod b)
- Euclidean Algorithm to compute GCD(a,b) is:

```
EUCLID(a,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
```

- Example GCD(1970,1066)

Fermat's Theorem

- $a^{p-1} = 1 \pmod{p}$
  - where p is prime and gcd(a,p)=1
  - also known as Fermat's Little Theorem
  - Ex: $2^{16} \equiv 1 \pmod{17}$.
  - Ex: $2^{50} \equiv ? \pmod{17}$

  50=(16x3)+2 => $2^{(16)x3} \times 2^2 = 1^3 \times 4 \equiv 4 \pmod{17}$
- also $a^p = p \pmod{p}$
  - useful in public key and primality testing
  - Ex, if $a = 2$ and $p = 7$, then:

  $2^7 = 128$, and $128 - 2 = 126 = 7 \times 18$ is an integer multiple of 7.

12/5/2018 23

Euler Totient Function ø(n)

- Euler Totient Function ø(n)
  - when doing arithmetic modulo n
  - **complete set of residues** is: 0..n-1
  - **reduced set of residues** is those numbers (residues) which are relatively prime to n
  - eg for n=10,
    – complete set of residues is {0,1,2,3,4,5,6,7,8,9}
    – reduced set of residues is {1,3,7,9}
  - number of elements in reduced set of residues is called the **Euler Totient Function ø(n)**

12/5/2018 24

Euler Totient Function ø(n)

- to compute ø(n) need to count number of residues to be excluded
  - in general need prime factorization, but
    – for p (p prime): ø(p) = p-1
    – for p.q (p,q prime): ø(pq) =(p-1)x(q-1)
  - eg.
    ø(37) = 36
    ø(21) = (3–1)x(7–1) = 2x6 = 12

12/5/2018 25

RSA (Rivest, Shamir, Adleman)

- RSA is a well – known and widely popular public key cryptography.
- Firstly published by the authors in 1977 (MIT)
- **Its based on exponentiation on Galos' Field of the integers of modulo prime number**
  - Exponentiation has complexity $O((\log n)^3)$ (**easy**)
- RSA security is based on hardness of the factor analysis and the discrete logarithm problem:
  - Analysis problem has complexity $O(e^{\log n \log \log n})$ (**difficult**)
  - Similarly, discrete logarithm is very hard
- RSA has been copyrighted in North America and in some other countries.

26

RSA Algorithm

- **Users create pair of public/private keys :**
  - Choose 2 random prime numbers $p \neq q$ (>120 digits)
  - Calculate $N = p \times q$,
  - Calculate $\varphi(N) = (p-1) \times (q-1)$
  - Choose integer e, $1< e < \varphi(N)$ such as: $gcd(e,\varphi(N)) = 1$
  - Calculate $d = e^{-1} \bmod \varphi(N)$ and $0 < d < \varphi(N)$
  - **Public key is the pair: $K_u = \{e,N\}$**
  - **Private key is the pair: $K_r = \{d,N\}$**
- **Encryption**: $c = m^e \bmod n,\ m < n$
- **Decryption**: $m = c^d \bmod n$
- **Signature**: $s = m^d \bmod n,\ m < n$
- **Verification:** $m = s^e \bmod n$

27

RSA Example - Key Setup

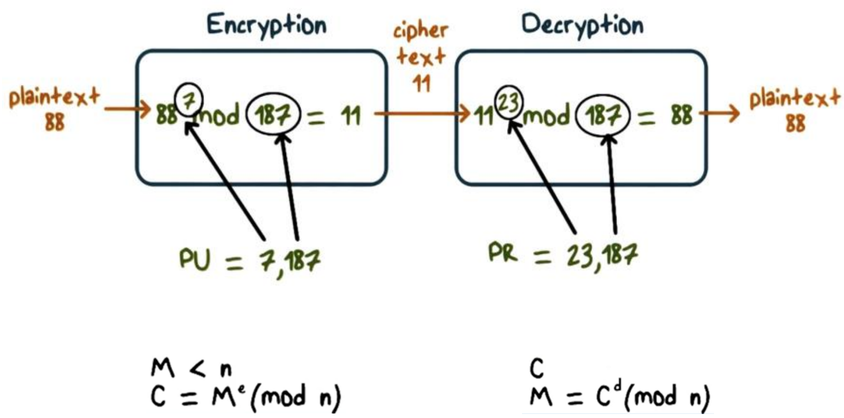
1. Select primes: $p = 17$ & $q = 11$
2. Calculate $n = pq = 17 \times 11 = 187$
3. Calculate $ø(n)=(p–1)(q-1)=16x10=160$
4. Select e: gcd(e,160) = 1; choose $e = 7$
5. Determine d: $de = 1 \bmod 160$ and $d < 160$

Value is d = 23 since 23x7 = 161 = 1x160 + 1

1. Publish public key PU = {7,187}
2. Keep secret private key PR = {23,187}

12/5/2018 28

Why does RSA Work?



RSA Quiz

Fill in the text boxes:

Given p = 3 and q = 11

1. Compute n: n= ☐

2. Compute φ(n):
φ(n) = ☐

3. Assume e = 7
Compute the value of d:
d = ☐

4. What is the public key

(e,n) = (☐ , ☐)

5. What is the private key

(d,n) = (☐ , ☐)

RSA Encryption Quiz

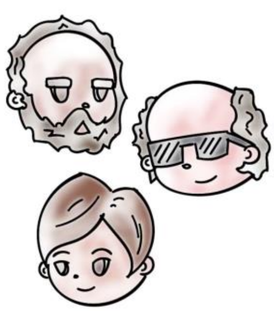
Given:
- Public key is (e, n) => (7, 33)
- Private key is (d, n) => (3, 33)
- Message m = 2

What is the encryption of m:

What formula is used to decrypt m?

(Use ** for denoting an exponent)

RSA Characteristics



- **Variable key length**
- **Variable plaintext block size**
  - Plaintext treated as an integer, and must be “smaller” than the key
  - Ciphertext block size is the same as the key length

RSA Security

- Four possible approaches to attacking the RSA algorithm are:

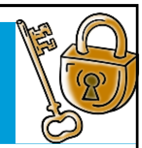
1. Brute force: This involves trying all possible private keys.

2. Mathematical attacks: There are several approaches, all equivalent in effort to factoring the product of two primes.

3. Timing attacks: These depend on the running time of the decryption algorithm.

4. Chosen ciphertext attacks: This type of attack exploits properties of the RSA algorithm.

Why RSA is Secure?

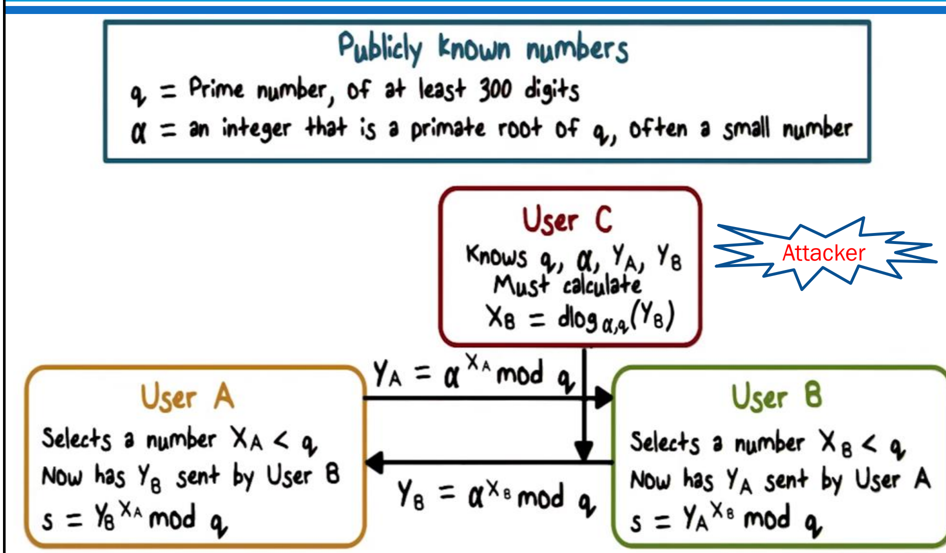


- Factoring an integer with at least 512-bit is **very hard**!
- **But if you can factor big number n then given public key <e,n>, you can find d, and hence the private key by:**
  - Knowing factors p, q, such that, $n = p \times q$
  - Then compute $ø(n) = (p-1)(q-1)$
  - Then find d such that $e \times d = 1 \bmod ø(n)$

Diffie and Hellman Key Exchange

- **First published** public-key algorithm
- By Diffie and Hellman in 1976 along with the **exposition of public key concepts**
- Used in a number of commercial products
- **Practical method to exchange a secret key** securely that can then be used for subsequent encryption of messages
- Security **relies on difficulty of computing discrete logarithms**

Diffie and Hellman Key Exchange



Diffie-Hellman Example

Have
- Prime number $q = 353$
- Prime number $\alpha = 3$

A and B each computer their public keys
- A computes $Y_A = 3^{97} \bmod 353 = 40$
- B computes $Y_B = 3^{233} \bmod 353 = 248$

Then exchange and computer secret key:
- For A: $K = (Y_B)^{XA} \bmod 353 = 248^{97} \bmod 353 = 160$
- For B: $K = (Y_A)^{XB} \bmod 353 = 40^{233} \bmod 353 = 160$

Attacker must solve:
- $3^{\alpha} \bmod 353 = 40$ which is hard
- Desired answer is 97, then computer key as B does

Diffie-Hellman Quiz

Alice and Bob agree to use
prime q = 29 and primitive root α = 5

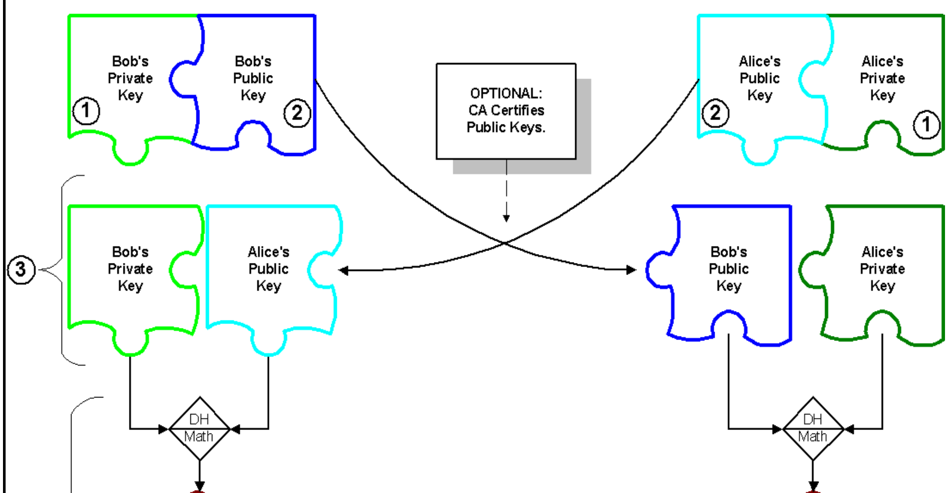
Alice chooses secret a = 6
Bob chooses secret b= 5

What number does Alice send Bob?

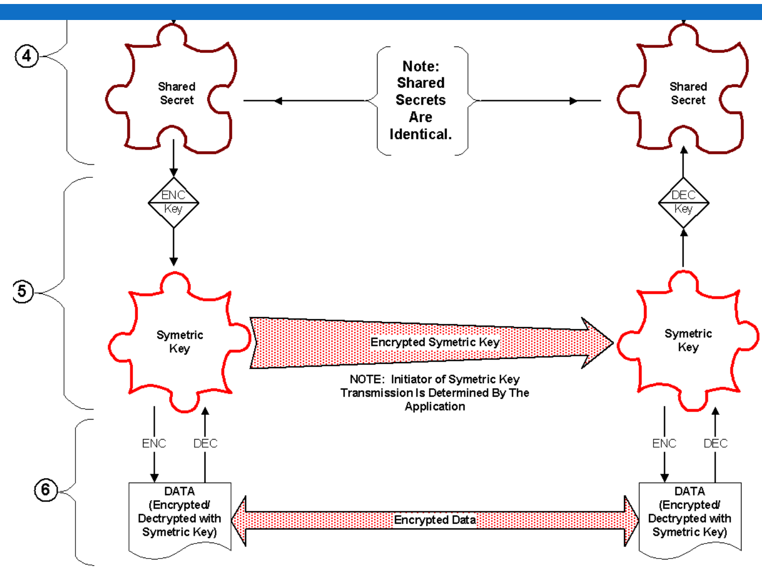
What number does Bob send Alice?

Implementation

Diffie-Helman Key Exchange



Implementation



Diffie-Hellman Security

- **Shared key (the secret) itself never transmitted**
- Discrete logarithm is very hard
- $Y = \alpha^X \bmod q$
- **Conjecture:** given Y, α, and q, it is extremely hard to compute the value of X because q is a very large prime (discrete logarithm)

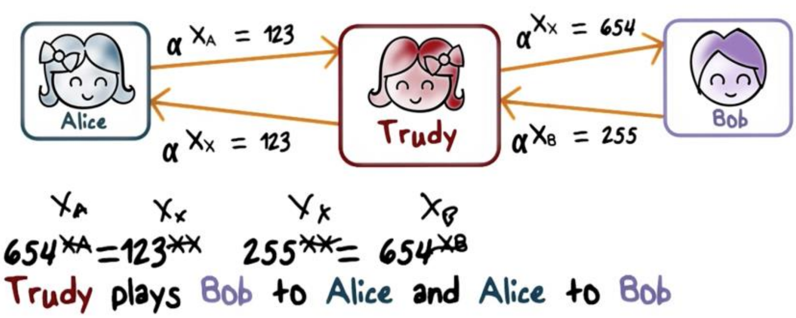
Applications

- Diffie-Hellman is currently used in many protocols, namely:
  - Secure Sockets Layer (SSL)/Transport Layer Security (TLS)
  - Secure Shell (SSH)
  - Internet Protocol Security (IPSec)
  - Public Key Infrastructure (PKI)

Diffie-Hellman Limitations

- Expensive exponential operation
  - DoS possible
- The scheme itself **cannot be used to encrypt anything –** it is for secret key establishment
- **No authentication**, so you cannot sign anything

Bucket Brigade Attack, Man-in-the-Middle(MIM)



Other Public-Key Algorithms

Digital Signal Standard:

- Makes use of SHA-1 and the Digital Signature Algorithm (DSA)
- Originally proposed in 1991, revised in 1993 due to security concerns, and another minor revision in 1996
- **Cannot be used for encryption or key exchange**
- Uses an algorithm that is designed to provide only the digital signature function

Other Public-Key Algorithms

Elliptic-Curve Cryptography (ECC):

- Equal security for smaller bit size than RSA
- Seen in standards such as IEEE P1363
- Confidence level in ECC is not yet as high as that in RSA
- **Based on a mathematical construct known as the elliptic curve**

Part 3: Cryptographic data integrity algorithms

Nguyen Thi Thanh Van - Khoa CNTT

Contents

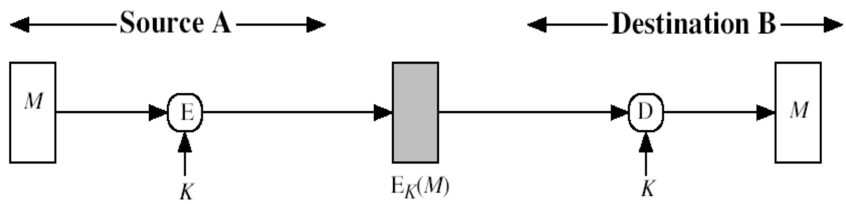
- Message Authentication
- **Cryptographic Hash Functions**
- Digital Signatures

Message Authentication

- **Authentication has purpose:**
  - Ensure **message sequentiality**
  - Assure **message integrity**
  - Confirm **sender's validity**
- **Mechanisms for message authentication**
  - Message encryption (in symmetric, asymmetric)
  - Hash function
  - Message Authentication Code – MAC

Message encryption

Symmetric Cryptography

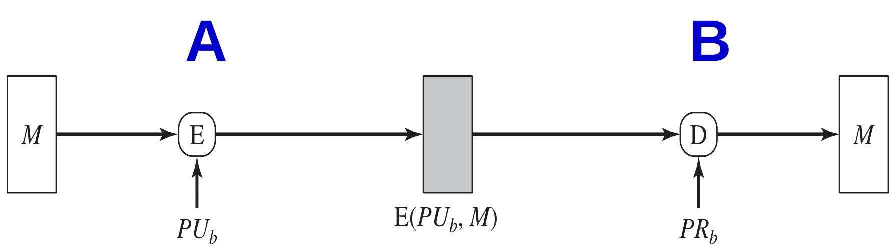


Symmetric encryption: confidentiality and authentication

- **Confidentiality:** Only A & B have key K to decrypt
- **Authentication:** M must only from A. M cannot be changed without detection. Need more constraints *(in case M is binary)*
- **No signature:** Recipient can fabricate message, and sender can deny message.

52

Asymmetric Cryptography

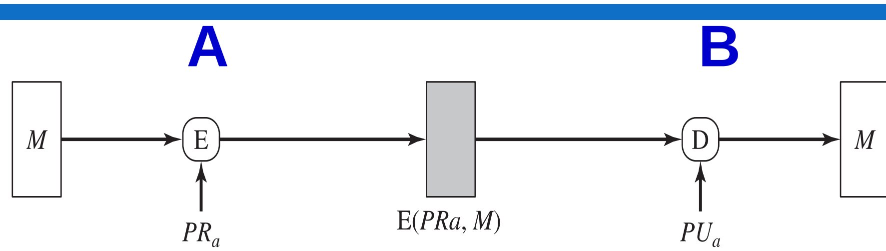


Public-key encryption: confidentiality

- **Confidentiality:** Only **B** has PR_b to decrypt
- **No authentication:** everyone can use PU_b to encrypt M then blame on **A**

53

Asymmetric Cryptography (2)

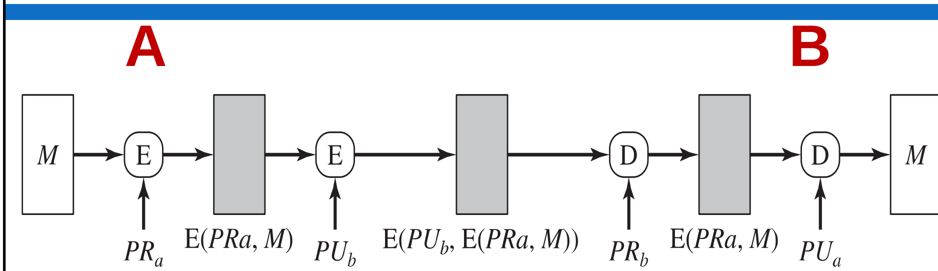


Public-key encryption: authentication and signature

- Only **A** has PR_a to encrypt
- M cannot changed
- Need more constraints
- Everyone can use PU_a to check signature

54

Asymmetric Cryptography (3)

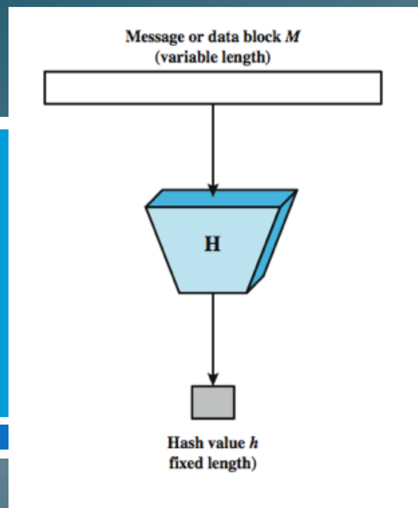


Public-key encryption: confidentiality, authentication, and signature

- **Confidentiality:**
  - Because of using PUb to encrypt
- **Authentication & Signature:**
  - Because of using PRa to encrypt

55

Hash function



Cryptographic Hash Functions

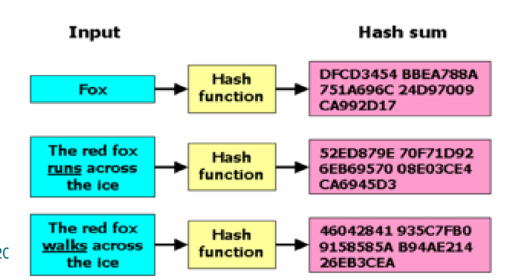
- What is Hash Functions
- Hash function Requirement
- Hash Functions in Message authentication
- Attacks on Hash Functions
- Secure Hash Algorithm (SHA)

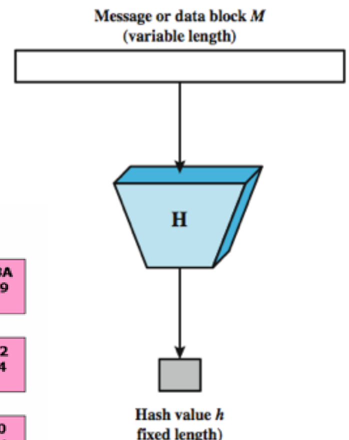
What is Hash Functions

- A hash function maps a *variable-length message* into a *fixed-length hash value*,

  or message digest

  ***h* = H(*M*)**
- The *principal object:*
  - *data integrity*





12/5/20 58

Hash function Requirement

- Variable input size
- Fixed output size
- Efficiency: H (M) is easily calculated with arbitrary M
- For any given value h, it is difficult to find M such that H(M) = h
  - One-way function
- For any M1, it is very difficult to find M2#M1 such that H(M2) = H(M1)
  - collision resistant: weak
- Very difficult to find any pair (M1, M2) such that H (M1) = H(M2)
  - collision resistant: Strong

A Strong hash function: satisfied all 6 reqs (weak: 5 reqs)

12/5/2018 59

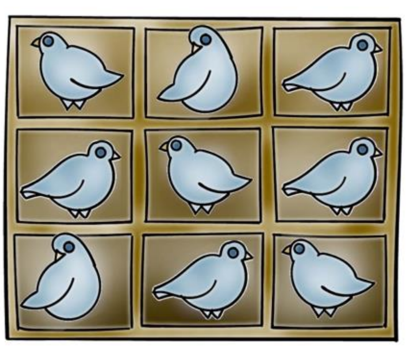
Hash Functions - Issues

Pigeonhole Principle

The Birthday Paradox

Hash Function Weaknesses

Pigeonhole Principle



n = number of pigeons
m = number of holes

n = m There is one pigeon per hole

n > m Then at least one hole must have more than one pigeon

Hash Function Weaknesses

Hash Functions:

- There are many more 'pigeons' than 'pigeonholes'
- Many inputs will be mapped to the same output. That is, ***many input messages will have the same hash.***

Conclusion: The longer the length of the hash, the fewer collisions.

Determining Hash Length

Hash Length	Possible # of hash values
1	2
64	2^{32}

Hash Size Quiz

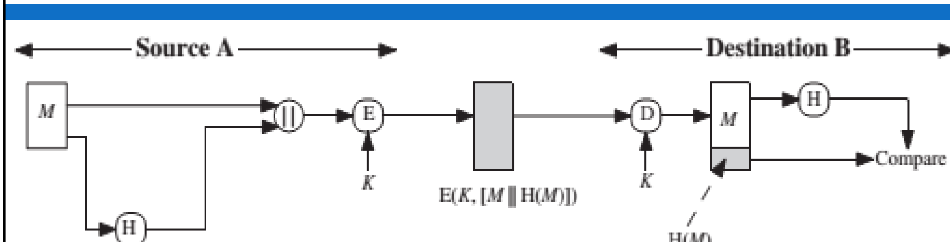
Choose the correct answer:

If the length of hash is 128 bits, then how many messages does an attack need to search in order to find two that share the same hash?

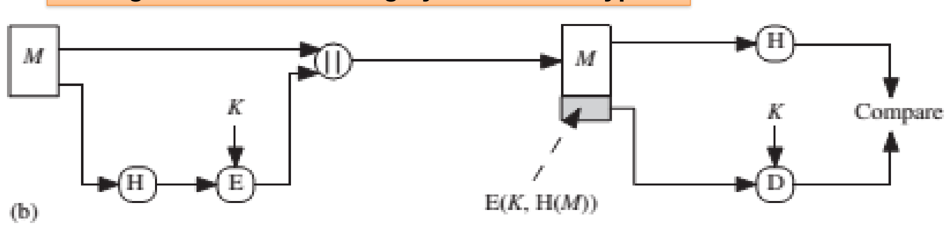
- ☐ 128
- ☐ 2^{127}
- ☐ 2^{128}
- ☐ 2^{64}

Exs of the Use of a Hash Function

for Message Authentication

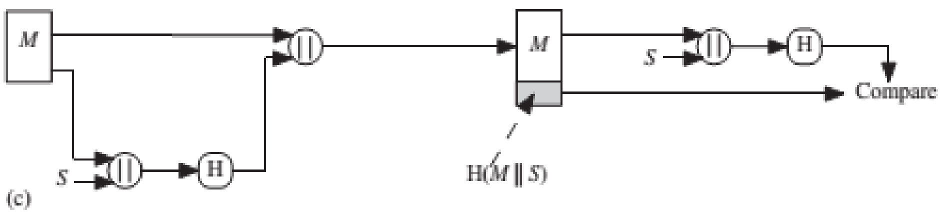


Message Auth & Conf using symmetric encryption

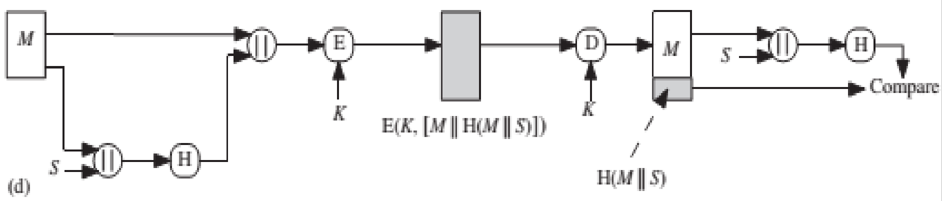


Message Authentication using symmetric encryption

Exs of the Use of a Hash Function
for Message Authentication



Message Auth & Integrity, no encryption



Message Auth & Conf using encryption

keyed hash function

- More commonly, message authentication is achieved using a **message authentication code (MAC)**, also known as a **keyed hash function**.
- Typically, MACs are used between two parties that share a secret key to authenticate information exchanged between those parties.
- A MAC function takes as input:
  - a secret key and
  - a data block
  - and produces a hash value, referred to as the MAC

Hash function: MD5

- **MD5 creates hash value of 128-bit from message**
  - Calculations in 32 – bit numbers is fast and widely used with **the acceptable security** (RFC1321 standard)
  - it is fast, simple and small => used in many cases even collision was found
- **Calculation Process of MD5:**
  - Add to message 1→512 bits to get length of **448 mod 512**
  - **Add one 64-bit value to the message**
  - Begin with **4-word 32-bit (128-bit) block**, that is (A,B,C,D)
  - In 16-word (512-bit) blocks: use **4 rounds** to calculate 16- bit numbers in the buffer and blocks. **Add outputs into inputs** to create new buffer values
  - **Hash value** is the final result of (A,B,C,D)

SHA - Secure Hash function

- SHA (Secure Hash Algorithm) originally designed by NIST & NSA in 1993, was revised in 1995 as SHA-1
- US standard for use with DSA signature scheme
  - standard is FIPS 180-1 1995, also Internet RFC3174
  - Note that, the algorithm is SHA, the standard is SHS
- based on design of MD4 with key differences
- produces 160-bit hash values
- recent 2005 results on security of SHA-1 have raised
- concerns on its use in future applications
- adds 3 additional versions of SHA: SHA-256, SHA-384, SHA-512

Comparison of SHA Parameters

	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	80	80	80
Security	80	128	192	256

Notes:
1. All sizes are measured in bits.
2. Security refers to the fact that a birthday attack on a message digest of size n produces a collision with a work factor of approximately $2^{n/2}$.

Attacks on Hash function

- two categories of attacks on hash functions:
  - Brute-force attack:
    - depend only on bit length of the hash value (not specific algorithm)
    - Attack to: One-way function; collision resistant - weak

    wishes to find a value y such that H(y)=h, try 2m-1 values
    - Attack to: collision resistant - strong

    wishes to find 2 messages: x,y, that yield H(y)=H(x), try 2m/2 values
  - Cryptanalysis:
    - based on weaknesses in a particular cryptographic algorithm.
    - require a cryptanalytic effort greater than or equal to the BF effort

MAC –
Message Authentication Code

Nguyen Thi Thanh Van - Khoa CNTT

12/5/2018

Message Authentication Code (MAC)

- **Message Authentication Code** *(MAC)*
  - attached to message
  - depends on **both message** and **private key** that only sender and recipient know
  - Message length can be arbitrary, but MAC often has certain fixed length (Ex: 128 bit)
  - To create MAC we can use hash function
    - To reduce message length
    - To keep message integrity

73

Message Authentication Code - MAC

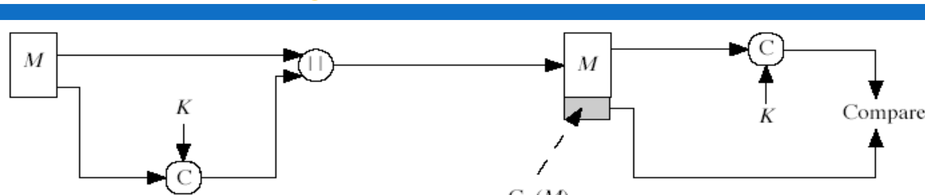
- When A has a message to send to B, it calculates the MAC (checksum) as a function of the message M and the key K:

MAC = C(M,K)

where
M = input message
K = shared secret key
C = MAC function

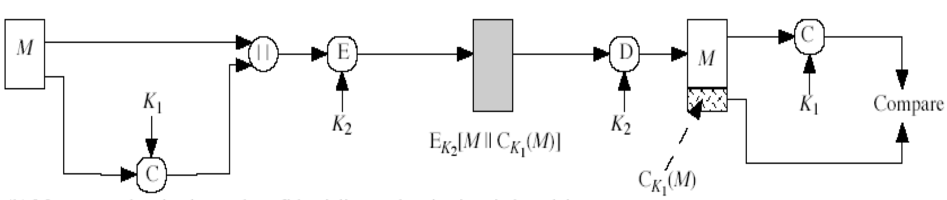
- MAC = message authentication code, is attached to M
- When B receive MAC & M, B calculates *MAC'* = C(**M,K**);
- If *MAC* = *MAC'* we can conclude:
  - **M** is not changed
  - **A** is the one who sent **M**

Basic Uses of Message Authentication Code



(a) Message authentication

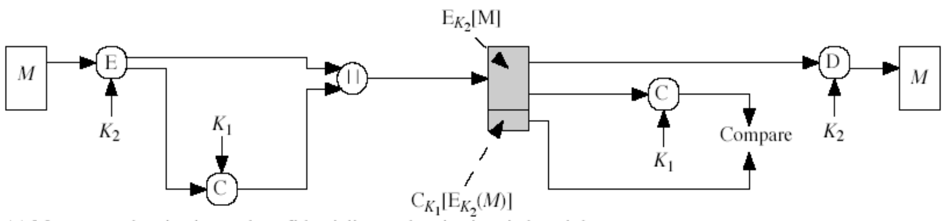
Authentication: Only **A** & **B** have key **K**



(b) Message authentication and confidentiality; authentication tied to plaintext

Confidentiality: Only **A** & **B** have key $\mathbf{K_2}$
Authentication: Only **A** & **B** have key $\mathbf{K_1}$

Basic Uses of Message Authentication Code



(c) Message authentication and confidentiality; authentication tied to ciphertext

Confidentiality: Due to **K_1.**
Authentication: Due to **K_2**

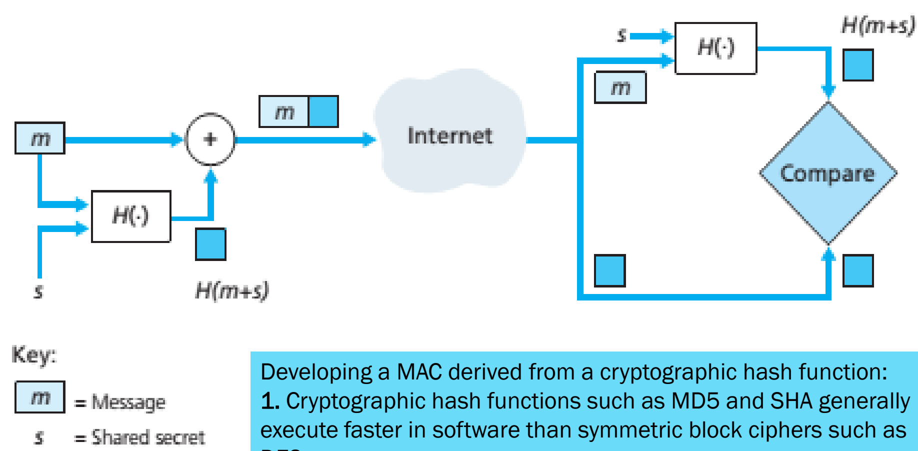
76

Security of MAC

- two categories of attacks on MAC:
  - Brute-force attack:
    - depends on the relative size of the key and the tag
    - more difficult undertaking than BF attack on a hash function because it requires known message-tag pairs.
  - Cryptanalysis:
    - based on weaknesses in a particular cryptographic algorithm.
    - require a cryptanalytic effort greater than or equal to the BF effort
    - There is much more variety in the structure of MACs than in hash functions, so it is difficult to generalize about the cryptanalysis of MACs.

12/5/2018 77

MAC based on Hash Function: HMAC



Developing a MAC derived from a cryptographic hash function:
1. Cryptographic hash functions such as MD5 and SHA generally execute faster in software than symmetric block ciphers such as DES.
2. Library code for cryptographic hash functions is widely available.

12/5/2018

MAC based on Hash Function: HMAC

1. Append zeros to the left end of to create a b-bit string K+ (e.g., if is of length 160 bits and , then will be appended with 44 zeroes).
2. XOR (bitwise exclusive-OR) with ipad to produce the b-bit block Si.
3. Append M to Si
4. Apply H to the stream generated in step 3.
5. XOR with opad to produce the b-bit block S0.
6. Append the hash result from step 4 to S0.
7. Apply H to the stream generated in step 6 and output the result.

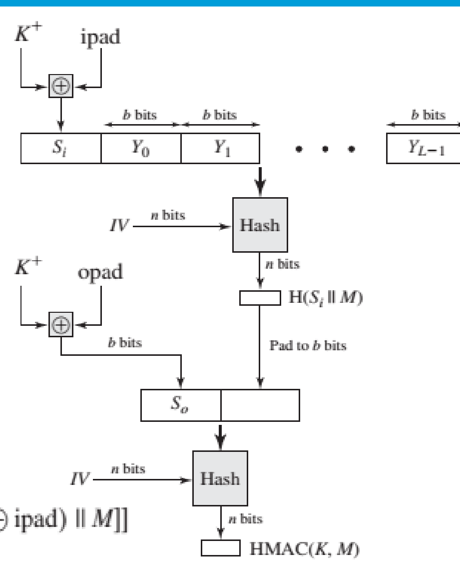


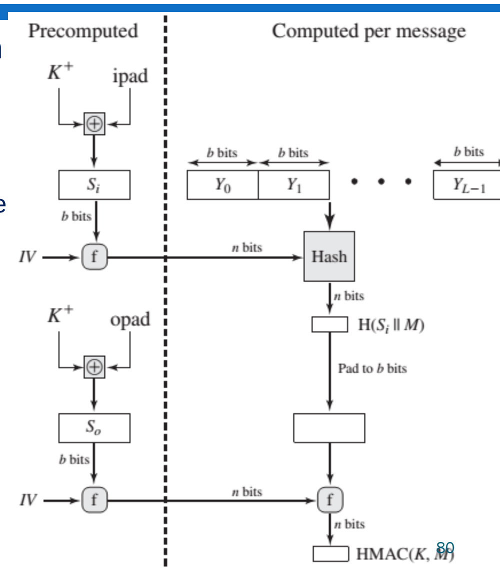
Figure 12.5 HMAC Structure

HMAC:

$$\mathrm{HMAC}(K, M) = \mathrm{H}[(K^+ \oplus \mathrm{opad}) \| \mathrm{H}[(K^+ \oplus \mathrm{ipad}) \| M]]$$

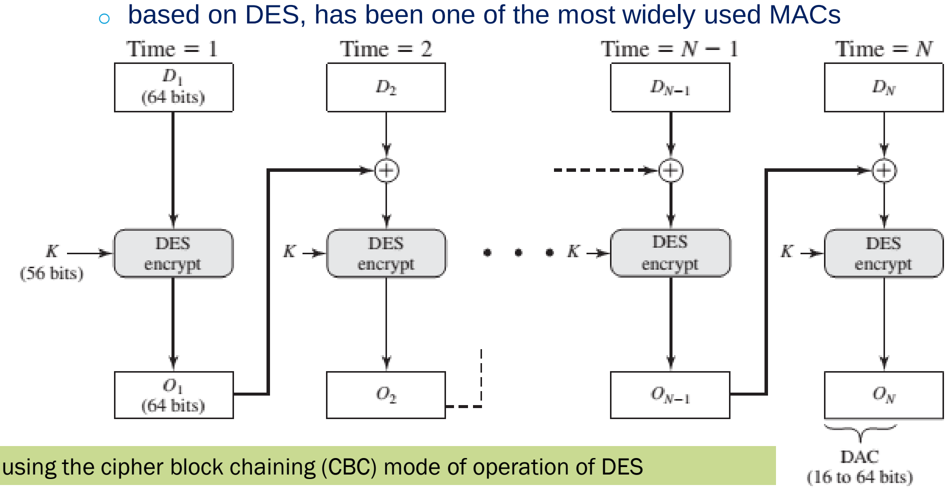
Security of HMAC

- based on an embedded hash function
  - depends on strength of the core hash function.
  - the probability of successful fake with time spent and some message–tag pairs created with the same key.
- Attack:
  - compute an output of the compression function
  - finds collisions in the hash function



MACS based on block ciphers: DAA & CMAC

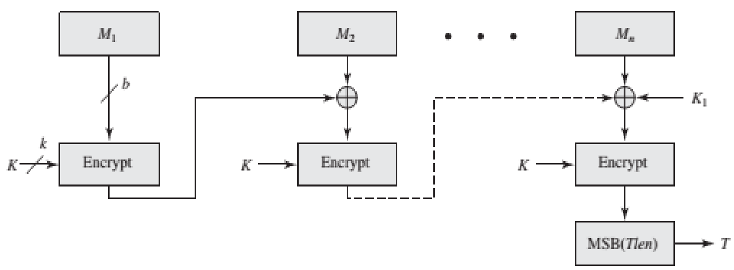
- **Data Authentication Algorithm** (DAA) (obsolete – old)
  - based on DES, has been one of the most widely used MACs



using the cipher block chaining (CBC) mode of operation of DES

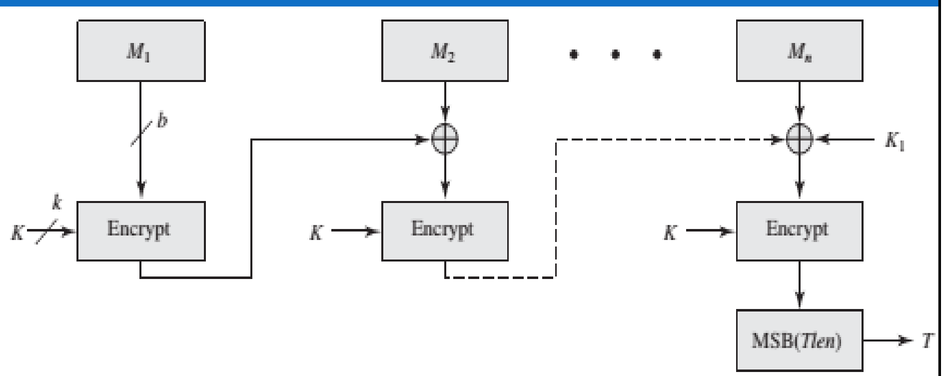
Cipher-Based Message Authentication Code (CMAC)

- operation for use with AES and triple DES:
- using three keys:
  - one key of length to be used at each step of the cipher block chaining and
  - two keys of length , where is the key length and is the cipher block length.
- This proposed construction: the two -bit keys could be derived from the encryption key, rather than being provided separately



(a) Message length is integer multiple of block size

Cipher-Based Message Authentication Code (CMAC)



(a) Message length is integer multiple of block size

$$C_n = E(K, [M_n \oplus C_{n-1} \oplus K_1])$$
$$T = MSB_{Tlen}(C_n)$$

T = message authetication code, also referred to as the tag
$Tlen$ = bit length of T
$MSB_s(X)$ = the s leftmost bits of the bit string X

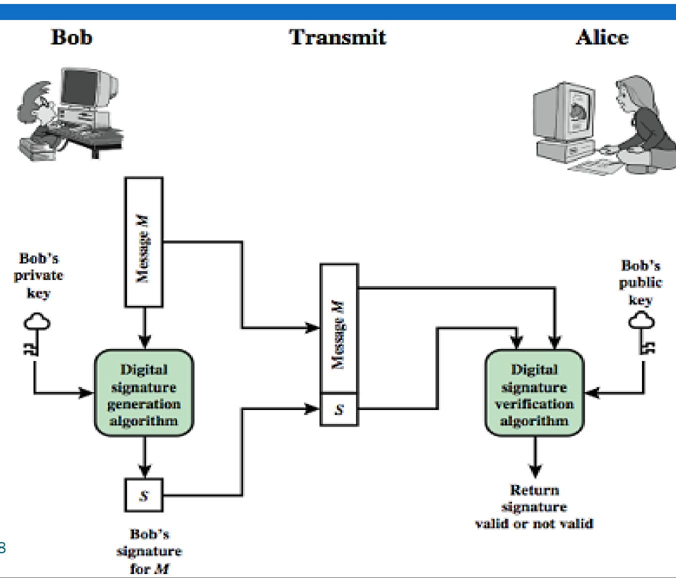
Digital Signature

Nguyen Thi Thanh Van - Khoa CNTT

Digital signature

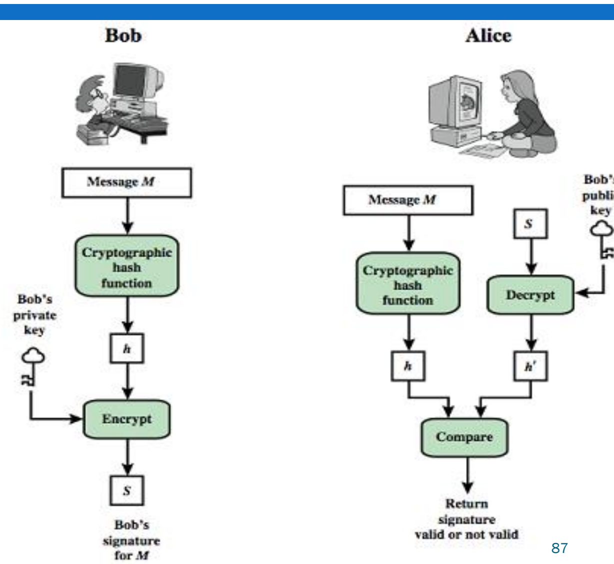
- A digital signature:
  - enables the creator of a message to attach a code that acts as a signature.
  - is formed by taking the hash of the message and encrypting the message with the creator’s private key.
- digital signature properties:
  - verify the author and time of the signature.
  - authenticate the contents at the time of the signature.
  - It must be verifiable by third parties, to resolve disputes.

Generic Model of Digital Signature Process



Simplified Depiction of Essential Elements of Digital Signature Process

Hash code is used to provide a digital signature



Ex of Digital Signatures

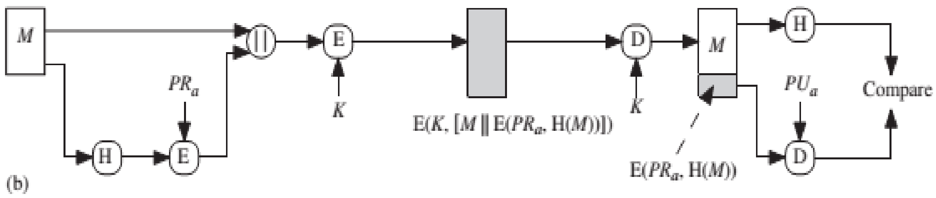


Figure 11.3 Simplified Examples of Digital Signatures

- hash code is used to provide a digital signature:
  - E(K,[M,E(PR_a, H(M))]): confidential
  - This is a common technique

Digital Signature Standard DSS

- DSS: Digital Signature Standard
  - US Govt approved signature scheme
  - designed by NIST & NSA in early 90's
  - published as FIPS-186 in 1991, revised in 1993, 1996, 2000
  - Use RSA to create the digital signature process
- DSA: Digital Signature Algorithm
  - new digital signature technique
  - is a public-key technique
- SHA: Secure Hash Algorithm
  - Is American standard in Digital Signature Algorithm DSA

Digital Signature using RSA

- RSA is used to create the digital signature process
- Assume we have the process RSA {(e,N), (d,N)}
- To **sign** the message **M** we calculate:

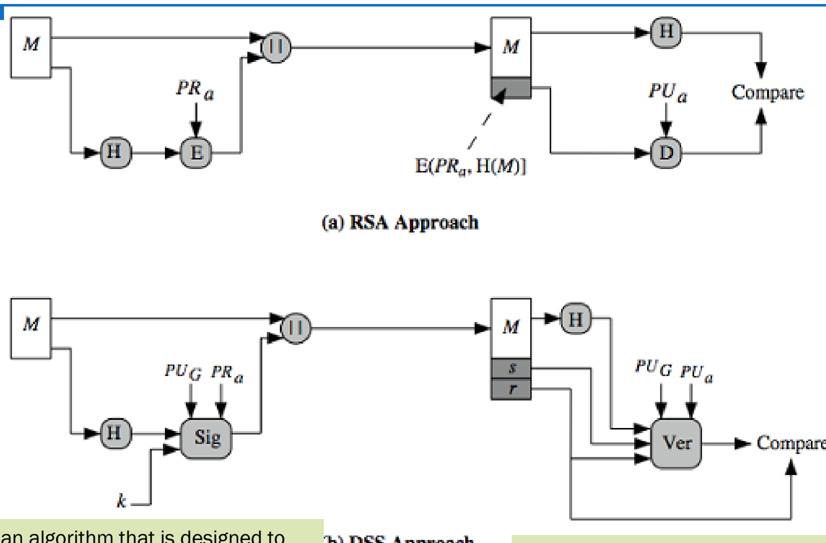
  S = $\mathbf{M}^{d}$(mod N).
- **Signature S should be attached to message M:**

  {M,S}
- To **check** signature we have to verify the equality of **M** and $\mathbf{S}^{e}$:

  $\mathbf{S}^{e}$(mod N) = $\mathbf{M}^{e.d}$(mod N) = **M**(mod N)

90

RSA vs. DSS



DSS uses an algorithm that is designed to provide only the digital signature function

it cannot be used for encryption

Digital Signature Algorithm - DSA

The DSA is based on the difficulty of computing discrete logarithms and is based on schemes originally

Global Public-Key Components

p prime number where $2^{L-1} < p < 2^L$ for $512 \le L \le 1024$ and L a multiple of 64; i.e., bit length of between 512 and 1024 bits in increments of 64 bits

q prime divisor of $(p-1)$, where $2^{159} < q < 2^{160}$; i.e., bit length of 160 bits

g $= h^{(p-1)/q} \bmod p$, where h is any integer with $1 < h < (p-1)$ such that $h^{(p-1)/q} \bmod p > 1$

User's Private Key

x random or pseudorandom integer with $0 < x < q$

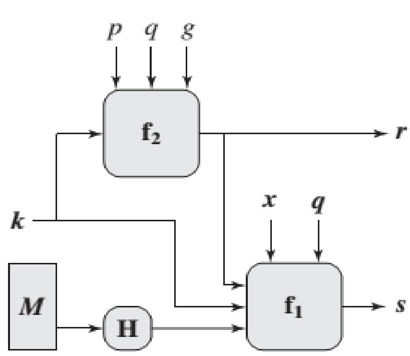
User's Public Key

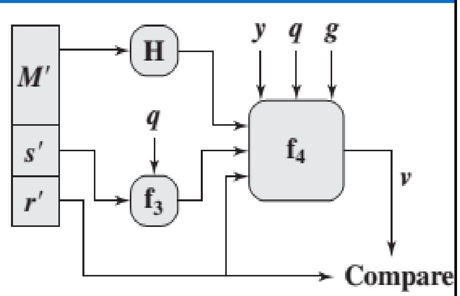
$y \;= g^x \bmod p$

User's Per-Message Secret Number

$k \;=$ random or pseudorandom integer with $0 < k < q$

Digital Signature Algorithm - DSA





Signing

$r \;= (g^k \bmod p) \bmod q$

$s \;= [k^{-1}(H(M) + xr)] \bmod q$

Signature $= (r, s)$

Verifying

$w \;= (s')^{-1} \bmod q$

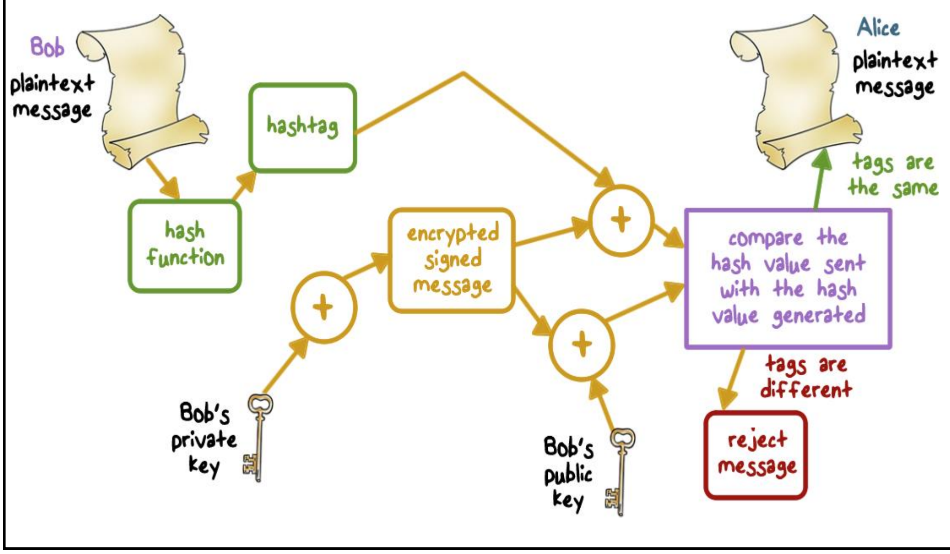
$u_1 = [H(M')w] \bmod q$

$u_2 = (r')w \bmod q$

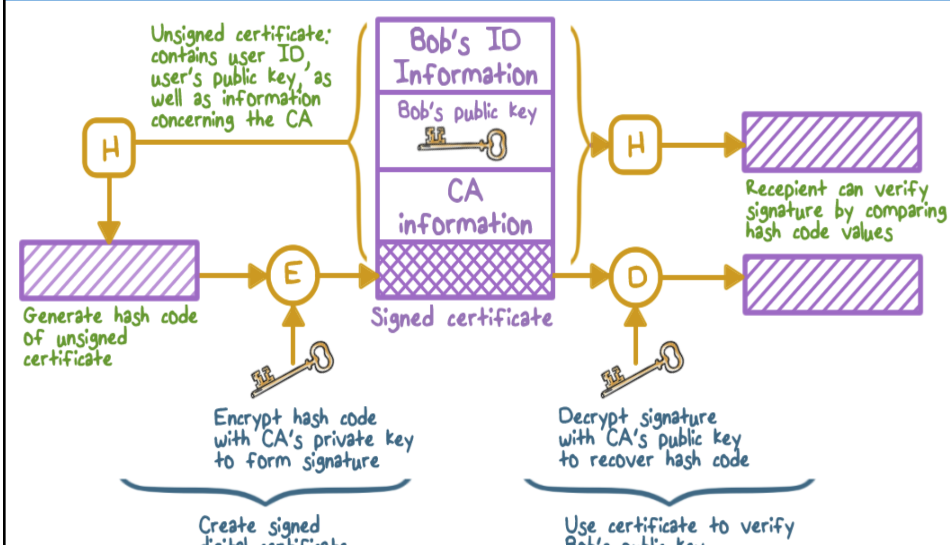
$v \;= [(g^{u1} y^{u2}) \bmod p] \bmod q$

TEST: $v = r'$

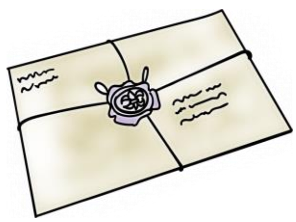
Digital Signatures



Digital Signatures

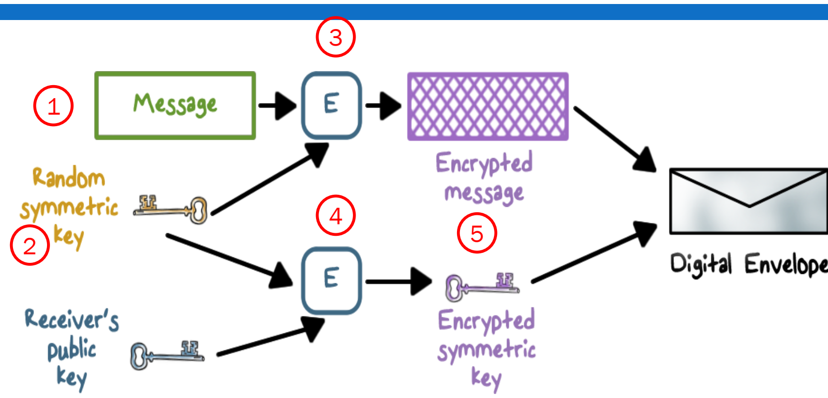


Digital Envelopes



- Means: public-key encryption is used to protect a symmetric key
- Protects a message **without needing** to first arrange for sender and receiver to have the same secret key
- Compares to the same thing as a **wrapped envelope containing an unsigned letter**

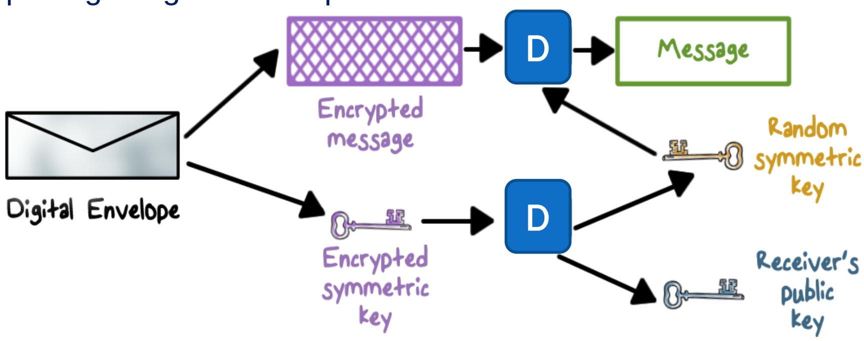
Digital Envelopes



1. Prepares a message.
2. Generates a random symmetric key that will be used this one time only.
3. Encrypts that message using symmetric encryption the one-time key.
4. Encrypts the one-time key using public-key encryption with Alice's public key.
5. Attaches the encrypted one-time key to the encrypted message and sends it

Digital Envelopes

Opening a digital envelope



Receiver is capable of decrypting the one-time key and recovering the original message. If Bob obtains Alice's public key by means of Alice's public-key certificate, then Bob is assured that it is a valid key.

Practice openSSL

- **Secure Sockets Layer (SSL)** is an application-level protocol which was developed by the Netscape Corporation for the purpose of transmitting sensitive information, such as Credit Card details, via the Internet
- **OpenSSL** is a robust, commercial-grade implementation of SSL tools, and related general-purpose library based upon SSL, developed by Eric A. Young and Tim J. Hudson
- OpenSSL is already installed on SEEDUbuntu

Q & A