

# Basics of Cryptography

## (1) Introduction

- ❖ Expectation
  - Level one: know what they are, what they can achieve, and how to use them as tools.
  - Level two: know how they work, how secure they are, and how to analyze their strength (crypto analysis)
- ❖ It is not just encryption, other applications include
  - digital cash
  - fair exchange
  - online auction
  - zero knowledge
  - time stamping

## (2) Secret Key Encryption

- ❖ Convention and Terms
  - plaintext, ciphertext, encryption, decryption
  - cryptoanalysis
- ❖ Secret Key Encryption (Symmetric Encryption)

```

      k                               k
plaintext ---> encryption (ciphertext) ---> decryption (plaintext)
  
```

- ❖ Classical Cryptosystems
  - Substitution Cipher (e.g. Caesar Cipher)

```

Mapping: A-> D, B->E, so on.
Encryption: HELLO -> KHOOR
Attack: frequency analysis (1-char, 2-char, 3-char, ...)
  
```

- Transposition Cipher

```

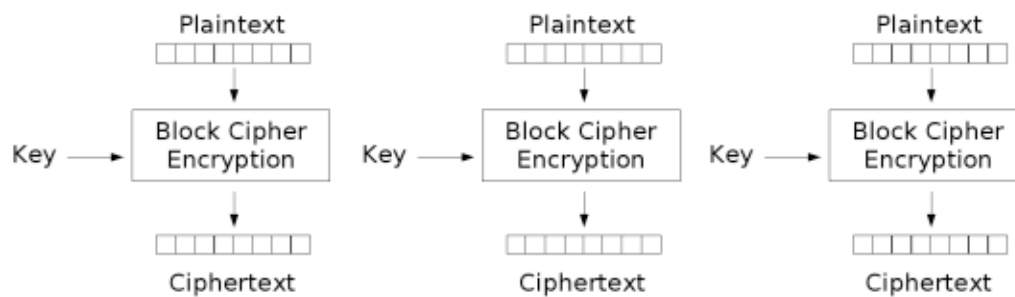
HLOOL
ELWRD
  
```

- One time pad
  - Is there a perfect encryption schemes?
    - Good news is: there is one, the one time pad.
    - Bad news is: it is not practical.
  - Invented in 1917
  - Used in low bandwidth
  - Used in military
  - Hotline between US and former Soviet Union
  
- ❖ DES (Data Encryption Standard) History
  - Horst Feistel (IBM) created the “Lucifer” block cipher as a result of research hobby.
  - Later, “Lucifer” became a major IBM initiative, and IBM revised it, named it DSD-1.
  - 1974: IBM decided to respond to the call for encryption standard issued by NBS (National Bureau of Standards). This means that IBM would be required to relinquish its patent rights, essentially giving—not selling—the algorithm to the world.
  - Early 1974, NSA offered to work with IBM on DSD-1. NSA’s all-star cryptanalysts would analyze DSD-1 and qualify the algorithm. IBM should allow NSA to control the implementation of the crypto system. IBM took the offer.
  - Horst Feistel’s Lucifer specified a 128-bit key, but NSA did not like that, and cut it into 64 bits, which is 8 bytes. IBM used one bit of each byte for “parity checks”. This reduced the size of the key to 56 bits.
  - 1977, the final algorithm was accepted as a standard, called DES.
  
- ❖ DES Technical Details
  - 56-bit key, 64-bit block, 16 rounds.
  - Block Cipher: 64-bit block
  - Brute-force attack  $2^{56}$
  - 1998, Electronic Frontier Foundation (EFF) built a "DES cracker" machine with \$250,000, it breaks DES in less 3 days.
  - RSA DES-III challenge: break in < 24 hours
  - Question: how to improve? --> 3DES
    - $C = E_1 [ D_2 [ E_1 [ P ] ] ]$
    - Why not  $E_1 [ E_2 [ p ] ]$ ? Attacks (although impractical) exists if done so.
    - 3DES use 2 keys (some use 3 keys), 112 bits are sufficient. \
    - Why 3DES did not become another standard?
      - Slow: 48 rounds, 64 block size.
  
- ❖ AES (Advanced Encryption Standard)
  - In 1997, NIST calls for AES
  - AES: Rijndael (Pronuciation "Rain Doll")
    - 128 bits key size
    - 128 bit block size
    - 9 rounds
    - Other variant: 192 bit key, 256 bit key.
  - Performance:
    - Rijndael: 200M Pentium: 70.5Mbits/Sec, Java: 1.1Mbits/sec

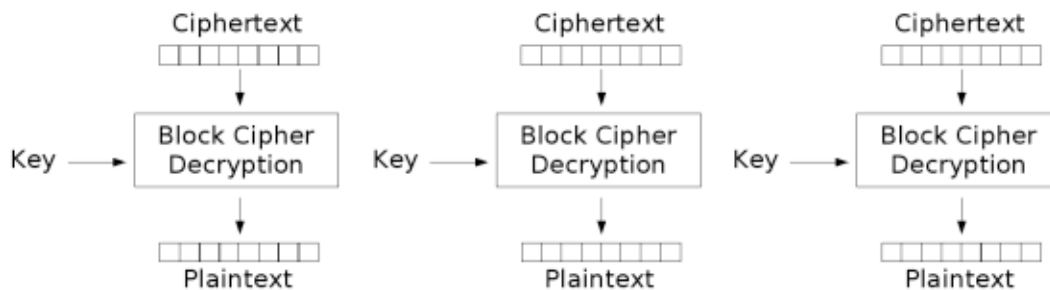
- RC6: see figures (<http://www.rsasecurity.com/rsalabs/rc6/>): Pentium 200M, Assembly Code: Encryption Rate: 100Mbits/sec. 8-bit platform: 9.2 kbits/sec
- Other AES Candidates: MARS (IBM), RC6 (Rivest), Rijndael, Serpent, Twofish (Schneier)
- Other existing encryption scheme:
  - IDEA (International Data Encryption Algorithm), used by PGP
  - Blowfish (Bruce Schneier).
  - RC5 (Rivest).
  - CAST-128, used by PGP.

### (3) Block Cipher Modes of Operation

- ❖ Encrypting a longer message
  - A block cipher operates on blocks of fixed length, often 64 or 128 bits
  - To encrypt longer messages, several **modes of operation** may be used
- ❖ Electronic Code Book Mode (ECB)
  - Problem: duplicate blocks and rearrange attack.

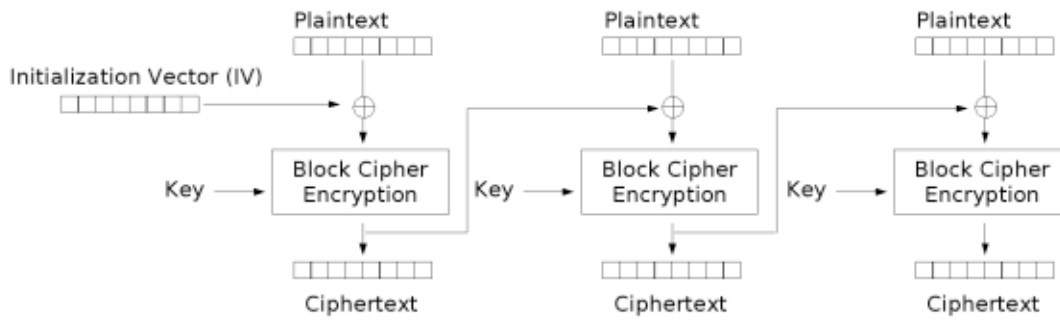


Electronic Codebook (ECB) mode encryption

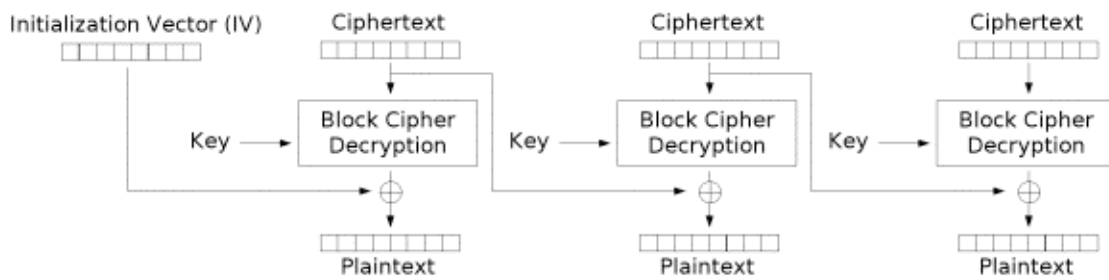


Electronic Codebook (ECB) mode decryption

- ❖ Cipher Block Chaining (CBC)
  - IV: Initialization Vector. Does not need to be secret.
  - Even if the same message is sent repeatedly, the ciphertext will be completely different each time due to IV.

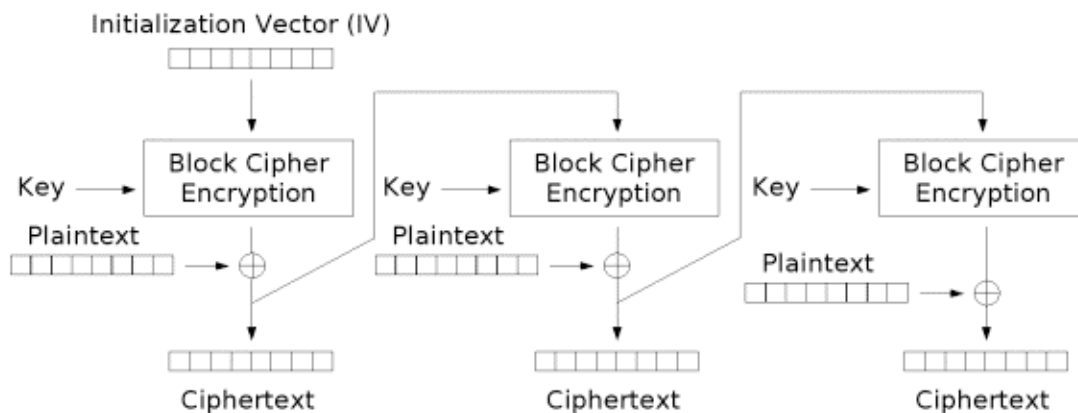


Cipher Block Chaining (CBC) mode encryption

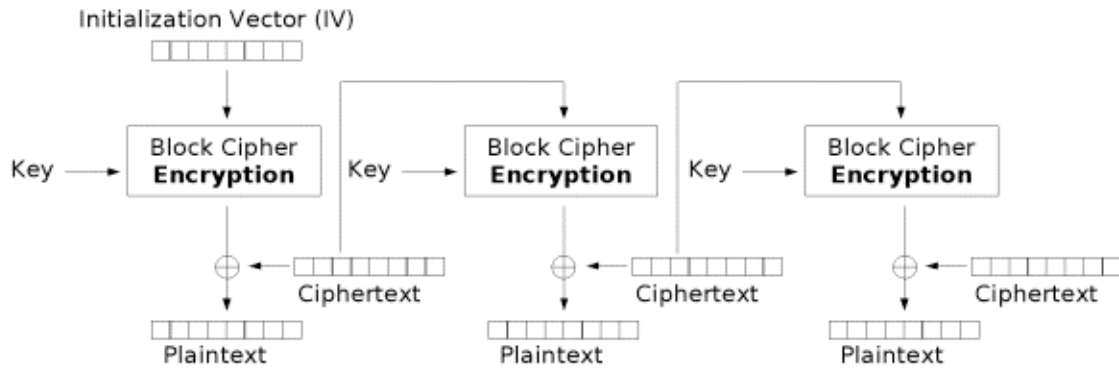


Cipher Block Chaining (CBC) mode decryption

- ❖ Cipher feedback (CFB)
  - Make the block cipher into a stream cipher



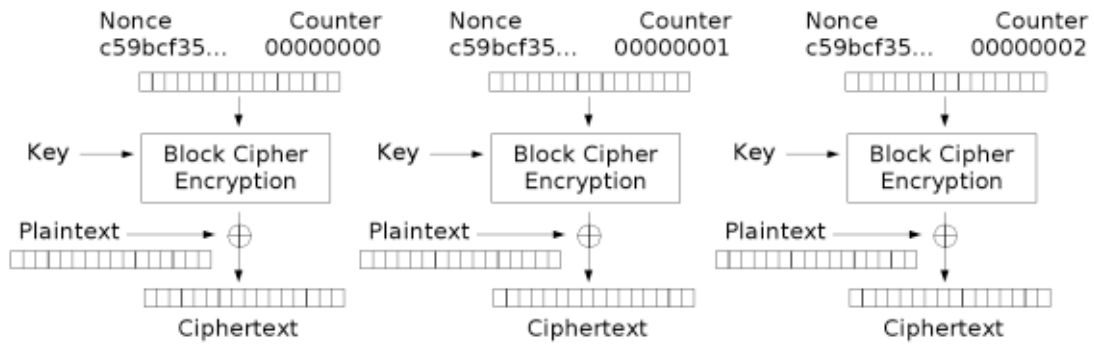
Cipher Feedback (CFB) mode encryption



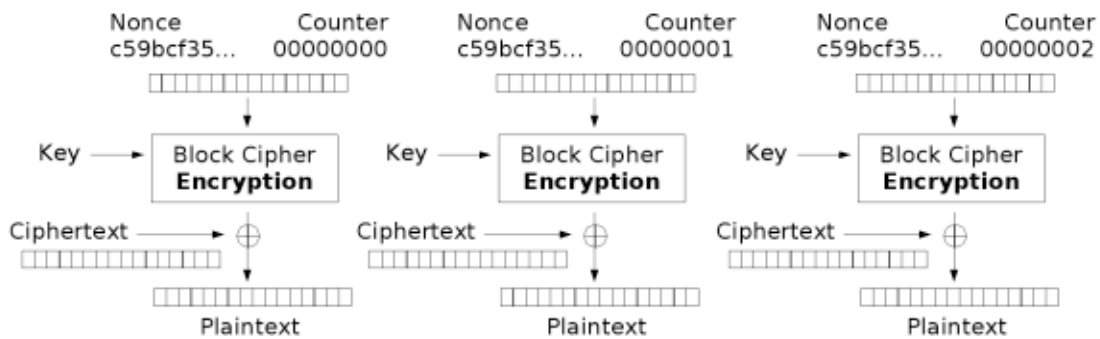
Cipher Feedback (CFB) mode decryption

❖ Counter (CTR)

- Also turns a block cipher into a stream cipher
- The counter can be any simple function which produces a sequence which is guaranteed not to repeat for a long time, although an actual counter is the simplest and most popular
- CTR allows a random access property for decryption



Counter (CTR) mode encryption



Counter (CTR) mode decryption

## (4) One-Way Hash Function

- ❖ One-way Hash Function
  - Reduce variable-length input to fixed-length (128 or 160 bit) output
  - Notation:  $M \rightarrow H(M)$
  - Properties:
    - One Way : input  $\rightarrow$  output (easy), output  $\rightarrow$  input (difficult)
    - Collision Free: impossible to find two messages that hash to the same hash value.
      - For  $m$ -bit hash, it takes only about  $2^{\{m/2\}}$  messages, chosen at random, before one would find two with the same value (like the birthday problem).
      - MD5 is broken: it is found not to be collision free!
    - Why do we want collision-free property?
      - Example: We can construct  $M1$  and  $M2$ , such that  $h(M1) = h(M2)$ . The meaning of  $M1$  and  $M2$  can be exactly the opposite.
      - $M1 = \text{"Alice owes Kevin \$100"}$ .  $M2 = \text{"Alice owes Kevin \$1M"}$ . Alice signs  $h(M1)$ , but not  $h(M2)$ . If  $h(M1) = h(M2)$ , Alice will be in trouble. If the hash is not collision free, it might be possible to find  $h(M1, r1) = h(M2, r2)$ , where  $r1$  and  $r2$  are random numbers.
- ❖ Hash Algorithms
  - MD2 (Message Digest) -- by Rivest
  - MD3 does exist, but it was superseded by MD4 before it was ever published or used.
  - MD4 (Message Digest) -- by Rivest
    - Faster than MD2, but a version of MD4 is found to be weak.
  - MD5 (Message Digest) -- by Rivest
    - A little slower than MD4.
    - 128-bit hash
  - SHA (Secure Hash Algorithm)
    - 1993 NIST published SHA
    - 1995 a never published flaw is found in SHA.
    - SHA-1 is proposed: most popular one.
    - SHA-1: 160-bit hash.
- ❖ MD5, SHA-0, SHA-1 are broken (by Xiaoyun Wang et al. 2005)
  - Wang et al found collisions on the MD5 hash algorithm in summer 2004.
  - Collisions in the full SHA-1 in  $2^{69}$  hash operations, much less than the brute-force attack of  $2^{80}$  operations based on the hash length. Later, the result was improved to  $2^{63}$ .
  - Collisions in SHA-0 in  $2^{39}$  operations.
  - Collisions in 58-round SHA-1 in  $2^{33}$  operations.
- ❖ The Performance of hash algorithms
  - SUN SPARC (33Mhz): MD5: 60.02Mbits/sec
- ❖ MAC: Message Authentication Code
  - Using secret key encryption: CBC residue
  - Using hash (keyed hash, HMAC)

- ❖ HMAC: Hashed MAC (or keyed hash), used as signature.
    - Concatenation approach:  $MD(K \parallel m)$
    - This doesn't work: extension attacks: Attacker is able to compute  $MD(K \parallel m \parallel m')$ , without knowing  $K$ .
      - How: Use  $MD(K \parallel m)$  to initialize the message digest computation.
    - Q: How to solve this problem?
      - Put the secret at the end
      - Use only half the bits of the message digest as the MAC. This is not less secure because the key is unknown.
    - HMAC: prepends the key to the data, digests it, and then prepends the key to the result and digests that.
  
  - ❖ One-way hash chain
    - One-way password
    - Broadcast authentication
  
  - ❖ Merkle Tree
    - Timestamping
    - Broadcast authentication in lossy channels
      - Signing each packet is expensive
      - Hashing all of them together and then hash the result cannot tolerate the loss of packet.
  
  - ❖ Applications of hash:
    - Integrity: message authentication code
    - Tripwire
    - How to save password? --- Save hash only.
  
  - ❖ Message Authentication Code (MAC)
    - Q: Can you conduct message authentication?
      - 1: public-key approach
      - 2: symmetric encryption approach
        - Disadvantage: encryption cost, small blocks (overhead is significant), export control.
      - 3: secret value approach
        - Q: How would you design a MAC without using encryption?
        - Hint: secret value, the property of hash:  $\rightarrow H(M \parallel K)$
      - A variation of 3, called HMAC, is the one adopted.
- $$HMAC_K(m) = h((K \oplus opad) \parallel h((K \oplus ipad) \parallel m)),$$
- HMAC (Keyed-Hashing for MAC)
  - Need a secret key (SHA, MD5 do not need a secret key)
  - Use one-way Hash function  $h$  as a black-box building block.
  - $ipad = 0x363636...3636$  and  $opad = 0x5c5c5c...5c5c$
  - $K$  is a secret key padded to length  $L$  with extra zeros ( $L$  is the size of hash block)
  - HMAC-MD5, HMAC-SHA.
  - Need to know the secret key in order to verify.
- 
- ❖ Key Exchange

- Goal: Alice and Bob want to agree upon a key  $K$ , which can be used as session key.
- Session key: only exists for the duration of the communication.
- (1) Using Trent (Key Distribution Center: KDC): both symmetric and public key encryption.
- (2) Without using Trent
  - Diffie-Hellman Key Exchange Protocol
  - Alice  $x$   $X=g^x \bmod n$ , Bob  $y$ ,  $Y=g^y \bmod n$
  - Discrete logarithm:  $p$  is a prime and  $g$  and  $M$  are integers, find  $x$ , s.t.  $g^x = M \pmod{p}$ .
- (3) Using Public Key: will talk about this later.

## (5) Public-Key Cryptography

### ❖ Introduction

- Motivation
- Public key and Private key
- $M = D [ E(M) ] = E [ D(M) ]$

### ❖ History

- First asymmetric key algorithm was invented, secretly, by Clifford Cocks (then a recent mathematics graduate and a new staff member at GCHQ in the UK) early in the 1970s.
- 1976, Diffie and Hellman postulated this system without demonstrating that such algorithms exist.
- 1978, Rivest, Shamir and Adleman all then at MIT invented RSA, which is a reinvention of Cocks scheme.
- Since then, several other asymmetric key algorithms have been developed, but the most widely known remains Cocks/RSA.
- Another algorithm is ElGamal (Taher ElGamal), which relies on the (similar and related) difficulty of the discrete logarithm problem.
- A third is a group of algorithms based on elliptic curves, first discovered by Neal Koblitz in the mid '80s.
- NSA has also claimed to have invented public-key cryptography, in the 1960s; however, there is currently (as of 2004) little supporting evidence for their claims.
- Merkle-Hellman (MH) was one of the earliest public key cryptosystems invented by Ralph Merkle and Martin Hellman in 1978. Although its ideas are elegant, and far simpler than RSA, it has been broken. (Merkle-Hellman Knapsacks).
- Stories behind RSA: Steven Levy's Crypto book

### ❖ Diffie-Hellman Key Exchange

- The algorithm was first published by Whitfield Diffie and Martin Hellman in 1976
- Discrete Logarithms in a finite field
  - Hard problem: find  $x$  where  $a^x = b \pmod{n}$
- Alice sends  $g^x \pmod{p}$ .
- Bob sends  $g^y \pmod{p}$ .
- Alice and Bob both get  $g^{xy} \pmod{p}$ .
- Good for key exchange, but not good for public key encryption (the key  $g^{xy}$  cannot change)
- $g$  can be small (e.g. a one-digit number)
- $x$  and  $y$  must be large.
- Q: Why isn't Diffie-Hellman considered as the first public key encryption?
  - A: It can't do encryption and signature.
- How to use Diffie-Hellman to do encryption?
  - Alice's public key  $g^x \pmod{p}$
  - Bob generates  $y$ , and generates a key  $k = g^{xy}$
  - Bob encrypts  $m$  using  $AES_k(m)$ , and sends to Alice the ciphertext along with  $g^y$ .

### ❖ ElGamal algorithm

- An asymmetric key encryption algorithm for public key cryptography
- Based on discrete logarithms
- $h = g^x \bmod p$
- Public key:  $(p, g, h)$
- Private key:  $x$
- Encryption: random  $k$ ,  $c_1 = g^k$ ,  $c_2 = m \cdot h^k$
- Decryption:  $c_2 / c_1^x$

#### ❖ RSA background

- *Extended Euclidean algorithm*: Given  $x$ , find  $y$ , such that  $xy = 1 \bmod m$
- *Euler's theorem*: For any  $a$  relatively prime to  $n = pq$ ,  $a^{(p-1)(q-1)} = 1 \bmod pq$ .  
Why is this very useful?  
Let  $z = k(p-1)(q-1) + r$   
 $a^z = a^{k(p-1)(q-1)} * a^r = a^r \bmod n$   
In other words:  
If  $z = r \bmod (p-1)(q-1)$ , then  $a^z = a^r \bmod n$
- Encryption:  $M^e \bmod n$ , let's derive  $d$ , s.t.  $(M^e)^d = M \bmod n$   
i.e.  $M^{ed} = M \bmod n$   
i.e.  $M^{ed-1} = 1 \bmod n$
- Q: how to get  $M^{ed-1} = 1 \bmod n$  ?
  - Can we let  $ed = 1$ ?
  - Can we let  $ed = 1 \bmod n$ ?
  - How about  $ed = 1 \bmod (p-1)(q-1)$
- Q: How to get  $d$ ?
  - Using the Extended Euclidean Algorithm
- Security of RSA depends on the hardness of factoring: *factoring  $n=p*q$  is hard when  $n$  is large.*

#### ❖ RSA Scheme:

- Public key  $(e, n)$
- Private key  $(d, n)$
- Encryption:  $m^e \bmod n$
- Decryption:  $(m^e)^d = m \bmod n$
- Find  $n$ : product of two prime numbers  $p$  and  $q$ , i.e.  $n = p*q$ .
- Find  $e$ :  $e$  is random, but relative prime to  $(p-1)(q-1)$
- Find  $d$ : find  $ed = 1 \bmod (p-1)(q-1)$  using Euclid's algorithm

#### ❖ Example

- Let  $n = 22 = 2*11$ .
- Let  $e = 3$ , find  $d$ , such that  $e*d = 1 \bmod 10$ . We get  $d = 7$ .
- Encrypting  $M = 7$ :  $7^3 = 49 * 7 = 5 * 7 = 13 \bmod 22$
- Decrypting:  $13^7 = 13^2 * 13^2 * 13^2 * 13 = (15 * 15) * (15 * 13)$   
 $= 5 * 19 = 7 \bmod 22$
- Sign  $M = 9$ :  $9^7 = 9^2 * 9^2 * 9^2 * 9 = 15 \bmod 22$

## ❖ RSA Performance Issues:

. How efficient are the RSA operations?

- Exponentiating with big numbers

e.g.  $123^{32} \bmod 678$

$$123^2 = 15129 = 213 \bmod 678,$$

$$123^4 = 621 \bmod 678, 123^8 = 537 \bmod 678, 123^{16} = 219 \bmod 678$$

$$123^{32} = 501 \bmod 678$$

e.g.  $123^{54}$

$$54 = 1100110 = (((((1)2+1)2)2+1)2+1)2$$

$$123^{54} = 123^{(2^2)} * 123^{(2^2)^2} * 123^{(2^2)^2} * 123^{(2^2)^2}$$

Cost: 8 multiplies and 8 divides.

- Performance: if the exponent is 512 bit, the number of multiplication and divides is linear to 512.

## ❖ Can we choose small e?

➤ How about  $e = 3$ ?

➤ Attacks:

▪ (1) when  $m^3$  is small than  $n$

▪ (2) when  $m^3$  is sent to three person (with  $n_1, n_2, n_3$ , respectively). We can get  $m^3 \bmod n_1 * n_2 * n_3$ , and we know  $m^3$  is smaller than  $n_1 * n_2 * n_3$ .

➤ How about 65537?

## ❖ Efficiency: very costly

➤ 100 times slower than DES (software)

➤ 1000 times slower than DES (hardware)

## ❖ Misconceptions:

➤ Public-key encryption is more secure than conventional encryption.

➤ Public-key encryption makes the conventional encryption obsolete.

Combination: RSA encrypts a symmetric key.

➤ Key distribution is trivial.

## ❖ Digital Signature

➤ Motivation: physical signature

➤ Properties: Authenticity, unforgeable, not reusable, unalterable, can't be repudiated.

➤ Sign using asymmetric cryptosystems (public key)

$$M = E[ D(M) ]$$

$$S_k(M) \text{ and } V_k(M)$$

The signature:  $D(M)$ .

➤ Sign with timestamp (Q: how to avoid reusing digital signature?)

➤ Sign the hash (Q: public key crypto is slow): The signature:  $D[H(M)]$

➤ Algorithm: RSA, DSS (Digital Signature Standard, using DSA)

➤ ElGamal scheme

➤ DSA (Digital Signature Algorithm) Story:

- 1991, Proposed by NIST as a DSS
  - Criticism from RSA and its supporters
  - Developed by NSA
  - Royalty-free
  - DSA is slower than RSA
- 
- ❖ Associating public keys with identities
    - Man-in-the-middle attack
      - How to avoid?
      - How to make sure Bob's Public Key is really Bob's?
      - Binding the identity with the public key
        - How? Can we put them together?
        - No, it should be unalterable
        - It should be verifiable
        - Certificate
    - Hierarchical certificate authority (eg, X.509),
    - A local trust model (eg, SPKI),
    - A statistical web of trust (eg, PGP and GPG).
- 
- ❖ Certificates
    - consist of: holder's id, holder's public key, CA's signature, expiration date
    - X.509 certificate
      - The certificate can be obtained from a public known database
      - Tree Structure
      - If you trust one, you can verify all of them.
- 
- ❖ Key Exchange using public keys
    - Goal: Alice and Bob want to agree upon a key  $K$ , which can be used as session key.
    - Session key: only exists for the duration of the communication.
    - Alice sends her public key to Bob, and Bob sends his to Alice.
- 
- ❖ Challenge-Response Authentication Using Public keys