Cryptography PPT - 1

Classic ciphers

Compression: Extract the information from the data and encode as efficiently as possible with a public algorithm

Encryption: Diffuse a key into the information as much as possible & encode with a public algorithm.

[Frequency Analysis]

e → t → a → o → n

[Index of Coincidence] $IC \approx \sum_{i=a}^{z} P_i^2$ (0.066)

IC is defined to be the probability that two randomly selected letters from a text will be identical.

[Rail Fence cipher]

d . . . n
. e . e .
. . f . .

[Columnar Transposition Cipher]

vvongpruee iheep

3 1 5 2 4
v n r e e
v g u i e ⇒ never
o p e h p given
phope

[Steganography & Cryptography]

Cryptography: although encrypted and unreadable, the existence of data is not hidden.

Steganography: no knowledge of the existence of data
↑ Invisible inks
Microdots

A steganography consists of

① A cover-object is an original unaltered medium
② Embedding process in which the sender hides a message by embedding it into a cover-text.
③ stego-object
④ Recovering process in which the receiver extracts the hidden message from the stego text using the key

[Textual Steganography]

It consists of hiding messages in formatted texts

Symmetric Cryptographic System

→ The entropy of the information source.

$$H(X) = \sum_i P_i \cdot \log_2 \frac{1}{P_i}$$

→ How much can we compress

The minimum number of bits needed to binary code a message is given as follows:

$$B = n \cdot H(X)$$

↑ the number of symbols in the message

[Symmetric System]

Both sender and receiver use the same key which must remain private

[Block cipher]

A block of plaintext is treated as a whole and used to produce a ciphertext block of equal length

[Stream Cipher]

The plaintext is encrypted/decrypted one bit or one byte at a time.

[Problems with shared key systems]

1. Compromised key means interceptis can decrypt any ciphertext they have required. Keys can be changed frequently to limit damage.
2. Distribution of keys is problematic: keys must be transmitted securely. e.g. distribute key in pieces over separate channels.

[Shannon Perfect Secrecy]

A cipher (E,D) over (K, M, C) has perfect secrecy if:

$\forall m1, m2 \in M$, (length(m1) = length(m2)) and $\forall c \in C$.

$\Rightarrow Pr[E(K,m1) = c] = Pr[E(K,m2) = c]$

where k is a uniform in K ($k \xleftarrow{R} K$)

This means give a ciphertext, I won't be able to deduce whether the message is m1 or m2 or any other m, therefore the most powerful attacker can learn nothing about the plaintext from the ciphertext. This means there is no ciphertext only attack.

[One Time Pad]

OTP is defined over the following sets:

$M = C = \{0,1\}^n$
$K = \{0,1\}^n$

Encryption: $C = E(k, m) = k \text{ xor } m$
Decryption: $m = D(k, c) = k \text{ xor } c$

Is One Time Pad Secure?

$M = C = \{0,1\}^n$
$K = \{0,1\}^n$

$\forall m, c$ if $E(k,m) = c$ this means
$c = k \text{ xor } m$ therefore $k = c \text{ xor } m$
The size of $\{k \in K : E(k,m) = c\} = 1$

Therefore

$$\forall m, c \quad Pr[E(k,m) = c] = \frac{|\{k \in K : E(k,m) = c\}|}{|K|} = \frac{1}{|K|}$$

OTP has perfect secrecy

Semantic Security of PRP (one-time key)

Definition: E is semantically secure if for all "efficient" A $Adv[A,E]$ is negligible

$Adv[A,E] = |EXP_A[E(k,m1)] - EXP_A[E(k,m2)]|$ for all explicit $m_0, m_1 \in M$ $(len(m_1) = len(m_2))$

EXP the experiment that an adversary performs on encrypted messages.

In other words: the attacker cannot distinguish between encrypted messages.

To be secure against a chose-plaintext attack, can encryption scheme must be non-deterministic - that is, its output must include a random element, so that, e.g. encrypt the same plaintext twice will result in two different ciphertexts. (nonce IV)

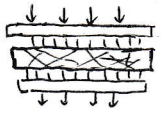
Encryption using a block cipher ~~such~~ such as AES by passing plaintext blocks directly to the encryption function is known as electronic code Book mode (ECB) and is ~~not~~ is not semantically secure. as it is entirely deterministic and two identical plaintext blocks will result in two identical ciphertext blocks.

Shannon's Confusion and Diffusion

① Diffusion: dissipates the redundancy of the plaintext by spreading it out over the ciphertext. (Permutation)

② Confusion: Obscures the relationship between ciphertext and the plaintext ~~in order to hide any statistical patterns~~. (Substitution)

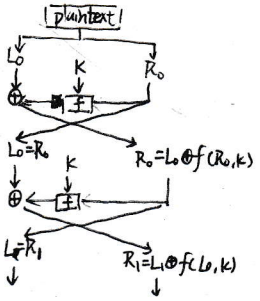
Ideal Block Cipher



$2^N!$ mappings

$\log_2(2^N!)$

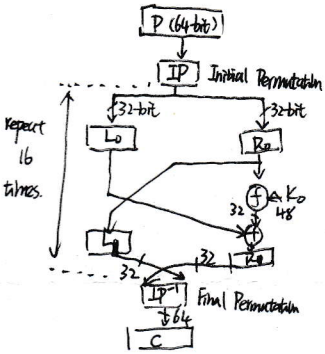
Feistel Cipher Structure



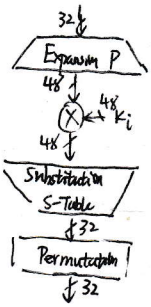
$$\begin{cases} R_i = F_i(R_{i-1}) + L_{i-1} \\ L_i = R_{i-1} \end{cases}$$

$$\begin{cases} R_{i-1} = L_i \\ L_{i-1} = R_i + F_i(L_i, K_i) \end{cases}$$

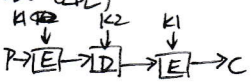
DES Course Structure



f-function



#3DES (EDE)



The advantage is that 3DES performs single DEC encryption if $k_3 = k_2 = k_1$ which is desired in implementations that should also support single DES for legacy reasons - backward compatibility.

Steps: ① for each possible key for K1, encrypt P to build produce a possible value for A.
② Using this A, and C, attack the 2DES to obtain a pair of keys (K2, K1')
③ If K1 = K1, try the key pair (K1, K2) on another (C', P')
④ If it works, (K1, K2) is the key pair with high probability
⑤ $O(2^{112})$ steps on average.

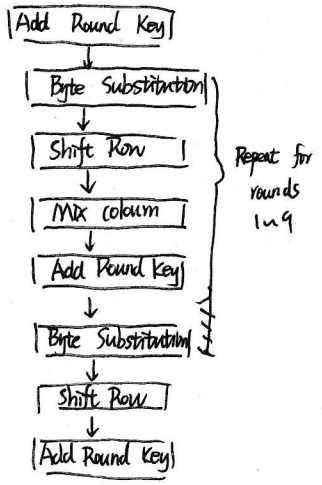
Invisible ink: write messages on fine silk, then was crunched into a tiny ball, covered in wax. The messager then ~~sat~~ swallowed the ball.

Null ciphers: the real message is camouflaged in an innocuous sounding message

[Avalanche Effect]

A small change in the plaintext or in the key result in a significant change in the ciphertext. An evidence of high degree of diffusion and confusion.

Cryptography PPT-2



AES is based on substitution-permutation network.

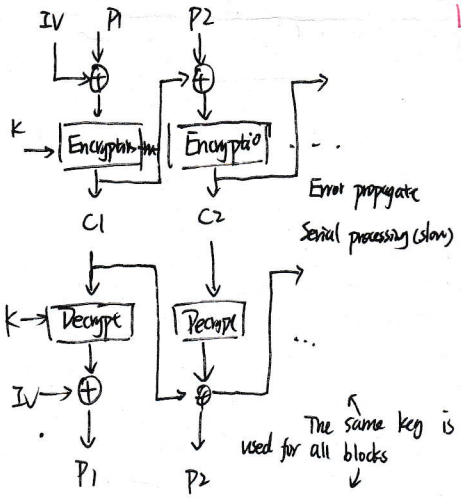
Electronic Code Book Weakness

Repetitive information contained in the plaintext may show in the ciphertext, if aligned with blocks.

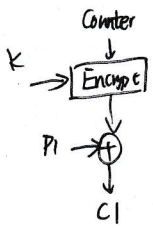
If the same message is encrypted (with the same key) and sent twice, their ciphertexts are the same. So an attacker can learn that two encrypted files are the same, two encrypted packets are the same.

ECB is only secure when the message length is equal to the data block length

CBC



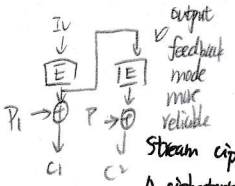
CTR

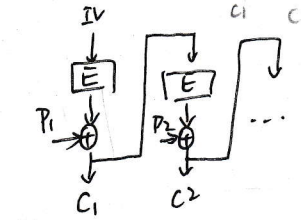


Only needs the encryption algorithm

Fast, can be processed in parallel

CFB





Stream cipher

A ciphertext segment depends on the current and all preceding plaintext segments

RSA Digital Signature Scheme

① Basic key Generation Algorithm

1. Choose random primes p, q
2. Set $N = p \cdot q$
3. Choose integers e, d s.t. $e \cdot d = 1 \pmod{\varphi(N)}$
4. Output $pk = (N, e)$, $sk = (p, q, d)$

② RSA Signing Algorithm

For each message x compute a signature y as follows

$F(sk, x) : (\mathbb{Z}_N)^* \to (\mathbb{Z}_N)^* : y = RSA(x, d) = x^d$

③ RSA Verification Algorithm

For each received message y

① Compute $F(pk, y) : (\mathbb{Z}_N)^* \to (\mathbb{Z}_N)^* : RSA(y, e) = y^e$

② If $y^e = x$ then output 1 else output 0

An example of an ~~existing~~ existential forgery attack.

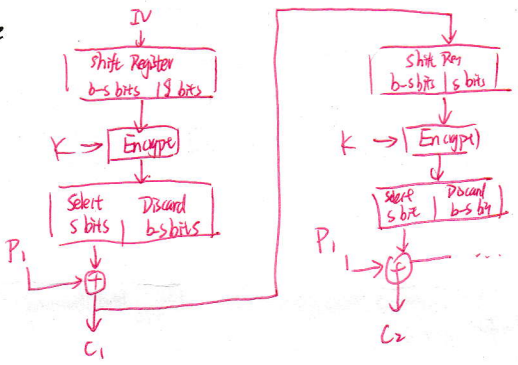
Simple Attack

If Bob signed two messages $y_1 = x_1^d$, $y_2 = x_2^d$

Then the signature for $x_3 = x_1 x_2$ can be easily forged as $y_3 = y_1 y_2$

Countermeasure.

Use collision resistant hash function before signing messages.



The block cipher is used as a stream cipher.

Appropriate when data arrives in bits/bytes

A ~~corrupted~~ corrupted ciphertext segment during transmission will affect the current and next several plaintext segments

PCL XL error
Error: IllegalAttribute
Operator: BeginPage
Position: 4

Cryptology { Cryptanalysis
Cryptography { Symmetric Ciphers { Block Cipher / Stream Cipher
Asymmetric Ciphers
Protocols

[Symmetric Algorithms]

Two parties have an encryption and decryption method for which they share a secret key.
Especially for data encryption and integrity check

[Asymmetric Algorithms]

In public-key cryptography, a user possesses a secret key as in symmetric cryptography but also a public key.
Can be used for applications such as { Digital Signatures / Key establishment

[Cryptographic Protocols]

Crypto protocols deal with the application of cryptographic algorithms.

Strictly speaking, hash functions, which will be introduced, form a third class of algorithms but at the same time, they share some properties with symmetric ciphers.

Symmetric Cryptography

Symmetric cryptographic schemes are also refered as "symmetric-key, secret-key and single-key schemes or algorithms.

Bad Guy { hack into an Internet router
listening to the radio signals of a WiFi

The type of unauthorized listening is called eavesdropping

Why not to keep the encryption algorithms secret?
→ Secret algorithm means untested algorithms.
The only way to find out whether an encryption method is strong, is to make it public and have it analyzed by other cryptographers.

Only consider the problem of confidentiality, that is hiding the contents of the message from an eavesdropper.

[Substitution Cipher]

① Brute Force Attack $26! \approx 2^{88}$
② letter frequency Analysis
Each plaintext symbol always maps to the same ciphertext symbol. That means that the statistical properties of the plaintext are preserved in the ciphertext.

Cryptanalysis

It is stressed that a cipher should be secure even if the attacker knows the details of the algorithm.

[General Thoughts on Breaking Cryptosystem]

Different ways of breaking cryptosystems in the real world.

Cryptanalysis { Social Engineering
Implementation Attacks
Classical Cryptanalysis { Mathematical Analysis / Brute-force attacks

Implementation Attacks

Side-channel analysis can be used to obtain a secret key
Measure the electrical power consumption of a processor which operates on the secret key.

This kind of attack is relevant against cryptosystems to which an attacker has physical access, such as smart cards.

Social Engineering Attack

Bribing (贿赂), blackmailing, tricking or espionage can be used to obtain a secret key by involving humans.
A attacker always looks for the weakest link in your cryptosystem. That means we have to choose strong algorithms and we have to make sure that social engineering and implementation attacks are not practical.

[How many key bits are enough?]

① The discussion of key for symmetric crypto algorithms is only relevant if a brute-force attack is the best known attack. If there is the probability of social engineering or implementation attacks, a long key also does not help.
② The key lengths for symmetric and asymmetric algorithms are dramatically different. Eg. 80-bit symmetric key provides roughly the same security as a 1024-bit RSA.

[Modular Arithmetic]

Modulo Operation

Let $a, r, m \in \mathbb{Z}$ (where $\mathbb{Z}$ is a set of all integers) and $m > 0$

$a \equiv r \bmod m$ if m divides $a-r$
(r: remainder, m: modulus)

"$x|y$" means "x divides y"

$3^{80} = 6561 \equiv 2 \bmod 7 \quad (81 \equiv 4 \bmod 7)$
$3^8 = 3^4 \cdot 3^4 = 81 \cdot 81 = 4 \cdot 4 \equiv 2 \bmod 7$

Shift Cipher

$e_k(x) \equiv x + k \bmod 26$
$d_k(y) \equiv y - k \bmod 26$

Affine Cipher

$e_k(x) = y \equiv a \cdot x + b \bmod 26$
$d_k(y) = x \equiv a^{-1}(y-b) \bmod 26$

a: (1,25) 1, (3,9) 3, (5,21) 5, 7, 9, 11, 15, 17, 19 (#12)

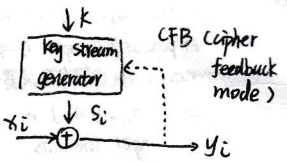
Is the affine cipher secure?
No, the key space is only a bit larger than in the case of the shift cipher.

key space = (# values for a) × (# values for b)
= 12 × 26 = 312

Stream Cipher vs Block Cipher

Stream ciphers encrypt bit individually. This is achieved by adding a bit from a key stream to plaintext bit.

① Synchronous stream ciphers
The key stream depends only on the key
② Asynchronous stream ciphers
The key stream also depends on the ciphertext.



Block ciphers encrypt an entire block of plaintext bits at a time with the same key. This means that the encryption of any plaintext bit in a given block depends on every other plaintext bit in the same block

Stream-ciphers are efficient ① for software-optimized stream ciphers means that they need fewer processor instructions to encrypt a bit of plaintext. ② for hardware-optimized stream ciphers, efficient means they need fewer gates (or smaller chip area) than a block cipher for encrypting at the same data rate. (Not always more efficient than block ciphers)

[Encryption and Decryption with Stream Ciphers]

Each bit x_i is encrypted by adding a secret key stream bit s_i modulo 2.

The plaintext, the ciphertext and the key stream consist of individual bits.
i.e., $x_i, y_i, s_i \in \{0, 1\}$.
Encryption: $y_i = e_{s_i}(x_i) \equiv x_i + s_i \bmod 2$
Decryption: $x_i = d_{s_i}(y_i) \equiv y_i + s_i \bmod 2$

Why are encryption and decryption the same function?

$d_{s_i}(y_i) \equiv y_i + s_i \bmod 2$
$\equiv (x_i + s_i) + s_i \bmod 2$
$\equiv x_i + 2s_i \bmod 2$
$\equiv x_i + 0 \bmod 2$
$\equiv x_i \bmod 2$

Modulo 2 addition is equivalent to the XOR operation

What exactly is the nature of the key stream?
↳ It turns out that the generation of the values s_i, which are called the key stream, is the central issue for the security of stream ciphers.
The central requirement for the key stream bits should be that they appear like a random sequence to an attacker.

Random Numbers and an Unbreakable Stream Cipher

[Pseudorandom Number Generators PRNG]

PRNGs generate sequences which are computed from an initial seed value. Often, they are computed recursively in the following way:

s_0 = seed $\quad s_{i+1} = f(s_i), i = 0, 1, \ldots$

A generalization of this are generators of the form $s_{i+1} = f(s_i, s_{i-1}, s_{i-2}, \ldots, s_{i-t})$ where t is a fixed integer. A popular example is the linear congruential generator: (线性同余生成器)

s_0 = seed $\quad s_{i+1} \equiv a s_i + b \bmod m, \ i = 0, 1, \ldots$

where a, b, m are integer constants.

Note that PRNGs are not random in a true sense because they can be computed and are thus completely deterministic.

[One-Time Pad] Unconditionally secure

- A stream cipher for which:

→① the key stream $s_0, s_1, s_2 \ldots$ is generated by a true random number generator

→② the key stream is only known to the legitimate communicating parties

→③ Every key stream bit s_i is only used once.

The best we can do in practice is to design crypto schemes for which it is assumed that they are computationally secure.

For symmetric ciphers this usually means one hopes that there is no attack method with a complexity better than an exhaustive key search.

[General Linear Feedback Shift Register]

$$s_m \equiv s_{m-1} \cdot p_{m-1} + \cdots + s_1 p_1 + s_0 p_0 \bmod 2$$

$$s_{m+1} = s_m \cdot p_{m-1} + \cdots + s_2 p_1 + s_1 p_0 \bmod 2$$

$$s_{i+m} = \sum_{j=0}^{m-1} p_j \cdot s_{i+j} \bmod 2 \quad \begin{cases} s_i, p_j \in \{0, 1\} \\ i = 0, 1, 2, \ldots \end{cases}$$

$\begin{cases} p_i = 0, \text{ output is not used for the feedback, open switch, the corresponding flip-flop} \\ p_i = 1, \text{ closed switch, the feedback is active,} \end{cases}$

The maximum sequence length generated by an LFSR of degree m is $2^m - 1$.

The Data Encryption Standard (DES)

[Confusion and Diffusion]

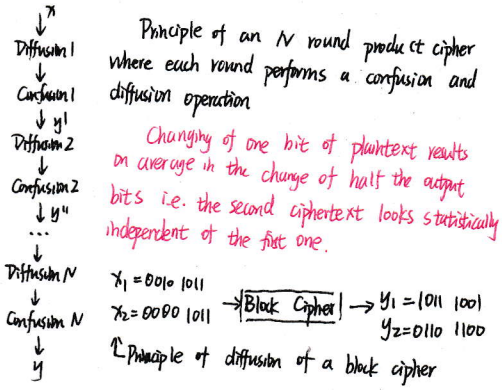
According to the famous information theorist Claude Shannon, there are two primitive operations with which strong encryption algorithms can be built.

① Confusion (混淆) is an encryption operation where the relationship between the key and ciphertext is obscured.

A common element for achieving confusion is substitution.

② Diffusion (扩散) is an encryption operation where the influence of one plaintext symbol is spread over many ciphertext symbols with the goal of hiding statistical properties of the plaintext.

A simple diffusion element is the bit permutation, which is used frequently within DES. AES uses more advanced MixColumn operation.



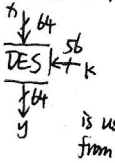
Principle of an N round product cipher where each round performs a confusion and diffusion operation

Changing of one bit of plaintext results on average in the change of half the output bits i.e. the second ciphertext looks statistically independent of the first one.

↑ Principle of diffusion of a block cipher

[Overview of the DES Algorithm]

DES is a cipher which encrypts blocks of length of 64 bits with a key of size of 56 bits



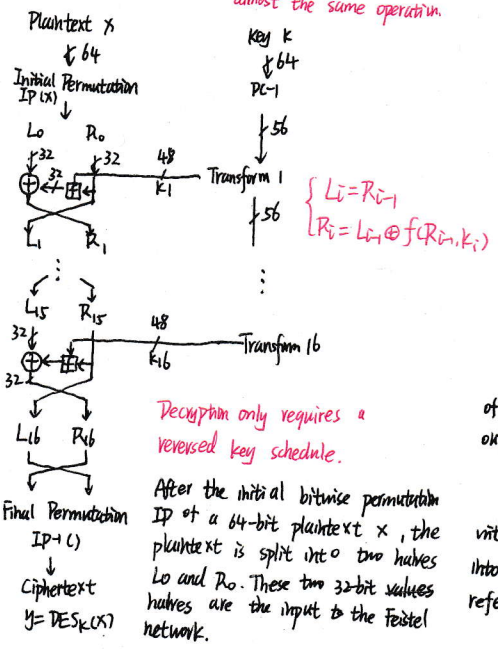
DES is a symmetric cipher

An iterative algorithm (16 rounds)

In every round, a different subkey is used and all subkeys k_i are derived from the main key k.

AES is not based on this

[Feistel network] Advantage: Encryption and Decryption are almost the same operation.



$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus f(R_{i-1}, k_i) \end{cases}$

Decryption only requires a reversed key schedule.

After the initial bitwise permutation IP of a 64-bit plaintext x, the plaintext is split into two halves L_0 and R_0. These two 32-bit values halves are the input to the Feistel network.

The right half R_i is fed into the function f. The output of the f function is XORed with the left 32-bit half L_i.

Finally, the right and the left half are swapped.

A round key k_i is derived from the main 56-bit key using what is called the key schedule.

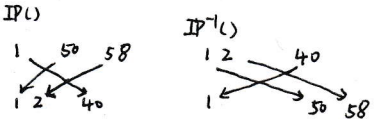
P76

Internal structure of DES

[Initial and Final Permutation]

A bitwise permutation can be viewed as simple bit crosswiring. Permutation can be very easily implemented in hardware but are not particularly fast in software.

Note that both permutations do not increase the security of DES at all.



IP (8×8)

58 50 42 34 26 18 10 2

⋮

IP⁻¹ (8×8)

40 8 48 ...

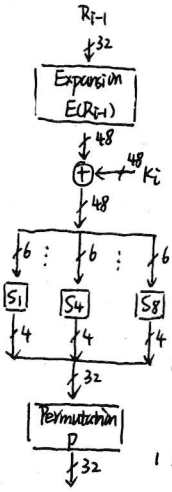
⋮

58 → 1

33 1 41 9

[The f-Function]

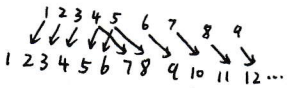
The f-function plays a crucial role for the security of DES.



Firstly, the 32-bit input is expanded to 48 bits by partitioning the input into eight 4-bit blocks and by expanding each block to 6 bits.

↑ This happens in the E-box which is a special type of permutation.

The first block consists of the bits (1,2,3,4), the second one of (5,6,7,8)



Expansion permutation E

E					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Exactly 16 of the 32 input bits appear twice in the output.

However, an input bit never appears twice in the same 6-bit output block

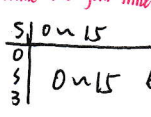
The expansion box increases the diffusion behavior of DES since certain input bits influence two different output locations.

Next, the 48-bit result of the expansion is XORed with the round key K_i and the 6-bit blocks are fed into eight different substitution boxes, which are often refered to as S-boxes.

Each S-box is a lookup table that maps a 6-bit input to a 4-bit output.

Each S-box contains $2^6 = 64$ entries, which are typically represented by a table with 16 columns and 4 rows. All S-box are different.

The most significant bit (MSB) and the least significant bit (LSB) of each 6-bit input select the row of the table, while the four inner bits select the column.



The S-boxes are the core of DES in term of cryptographic strength. They are the only nonlinear element in the algorithm and provide confusion.

Finally, the 32-bit output is permuted bitwise according to the P permutation, which is given. Unlike the initial permutation IP and IP^{-1}, the permutation P introduces diffusion because the four output bits of each S-box are permuted in such a way that they affect several different S-boxes in the following round.

扩散 雪崩效应

The diffusion caused by the expansion, S-boxes and the permutation P guarantees that every bit at the end of the fifth round is a function of every plaintext bit and every key bit. This behavior is known as the avalanche effect.

[Key Schedule]

The key schedule derives 16 round keys k_i, each consisting 48 bits, from the original 56-bit key. Another term for round key is subkey.

The DES input key is often started as 64-bit, where every eighth bit is used as an odd parity bit over the preceding seven bits.

In any case, the eight parity bits are not actual key bits and do not increase the security. DES is a 56-bit cipher, not a 64-bit one.

The parity bits are stripped in the initial PC-1 permutation. The name PC-1 stands for "permuted choice one"

key permutation without map to k·8 (64 − 8 = 56-bit key)

In rounds i = 1, 2, 9, 16, the two halves are rotated left by 1 bit

In the other rounds, the two halves are rotated left by 2 bits

The resulting 56-bit key is split into two halves C_0 and D_0.

Note that the rotations only take place within either the left or the right half. The total number of rotation positions is

4 · 1 + 2 · 12 = 28. ⇒ $C_0 = C_{16}$ and $D_0 = D_{16}$

$K_{16} = PC\text{-}2(C_{16}, D_{16}) = PC\text{-}2(C_0, D_0) = PC\text{-}2(PC\text{-}1(k))$

$K_{15} = PC\text{-}2(C_{15}, D_{15})$

$= PC\text{-}2(RS_2(C_{16}), RS_2(D_{16}))$

$= PC\text{-}2(RS_2(C_0), RS_2(D_0))$

Triple DES (3DES)

$y = DES_{k3}(DES_{k2}(DES_{k1}(x)))$

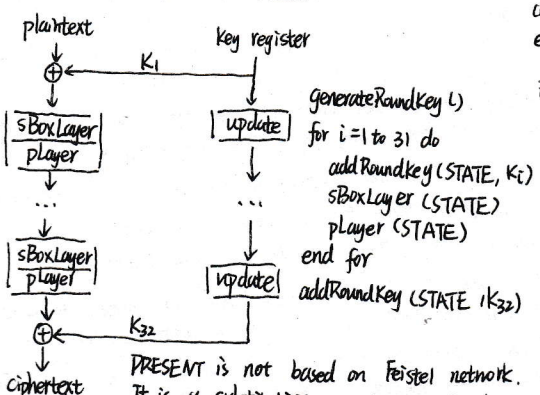
$y = DES_{k3}(DES_{k2}^{-1}(DES_{k1}(x)))$

The advantage here is that 3DES performs single DES encryption if $K_3 = K_2 = K_1$, which is sometimes desired in implementations that should also support single DES for legacy reasons.

A different approach for strengthening DES is to use key whitening. For this, two additional 64-bit key K_1, K_2 are XORed to the plaintext and ciphertext, respectively, prior to and after the DES algorithm. This yields the following encryption scheme:

$y = DES_{k,K_1,K_2}(x) = DES_k(x \oplus K_1) \oplus K_2$

Lightweight Cipher – Present



PRESENT is not based on Feistel network. It is a substitution-permutation network (SP-network) and consists of 31 rounds.

The block length is 64 bits (support 80 & 128 bits key)

Each of the 31 rounds consists of an XOR operation to introduce a round key K_i for $1 \le i \le 32$, where K_{32} is used after round 31.

The nonlinear layer uses a single 4-bit Sbox S, which is applied 16 times in parallel in each round.

[addRoundKey]

At the beginning of each round, the round key K_i is XORed to the current STATE.

[sBoxLayer]

PRESENT uses a single 4-bit to 4-bit Sbox.

This is a direct consequence of the pursuit of hardware efficiency, since such an S-Box allows a much more compact implementation than, eg., an 8-bit S-box. (p94)

[pLayer]

The mixing layer was chosen as a bit permutation, which can be implemented extremely compactly in hardware.

$$P(i) = \begin{cases} i \cdot 16 \bmod 63 & i \in \{0, \dots, 62\} \\ 63, & i = 63 \end{cases}$$

[Key Schedule]

The user-supplied key is stored in a key register K and is represented as $k_{79}k_{78}\dots k_0$.

At round i the 64-bit round key $K_i = k_{63}k_{62}\dots k_0$ consists of the 64 leftmost bits of the current content of the register K.

$$K_i = k_{63}k_{62}\dots k_0 = k_{79}k_{78}\dots k_{16}$$

The first subkey K_1 is a direct copy of 64 bit of the user supplied key.

For the following subkeys $K_2, \dots, K_{32}$, the key register $K = k_{79}k_{78}\dots k_0$ is updated as follows.

① $[k_{79}k_{78}\dots k_1k_0] = [k_{18}k_{17}\dots k_{20}k_{19}]$ $(K \ll 61)$

② $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$

③ $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus$ round_counter

Operations:
(1) the key register is rotated by 61 bit position to the left
(2) the leftmost four bits are passed through S-box PRESENT
(3) the round_counter value i is XORed with bits $k_{19}k_{18}k_{17}k_{16}k_{15}$ of K. (LSB is on the right)
(00001, 00010, ..., 11111) ($K_2 \leftarrow 00001$, $K_3 \leftarrow 10$)

Area of cryptanalytical computers in general should take a look at the SHARCS (Special-purpose Hardware for Attacking Cryptographic Systems).

Pervasive computing (ubiquitous computing)

↳ is the growing trend of embedding computational capability (generally in the form of microprocessors) into everyday objects to make them effectively communicate and perform useful tasks in a ~~everyday objects~~ way that minimizes the end user's need to interact with computers as computers.
Pervasive computing devices are network-connected and constantly available.

Unlike desktop computing, pervasive computing can occur with any device, at any time, in any place, and in any data format across any network and can hand tasks from one computer to another.
For example, a user moves from his car to his office. Thus persasive computing devices have evolved to include not only laptops, notebooks and smart phones, but also tablets, wearable devices, fleet management and pipeline components, lighting systems, appliances and sensors.

The goal of pervasive computing is to make devices "smart", thus creating a sensor network capeable of collecting, processing and sending data. and, ultimately, communicating as a means to adapt to the data's context and activity; In essence, a network that can understand its surroundings and improve the human experience and quality of life

Example: Apple Watch informs a user of phone call and allowing his to complete the call through the watch.

The Advanced Encryption Standard (AES)

Disadvantage of DES:

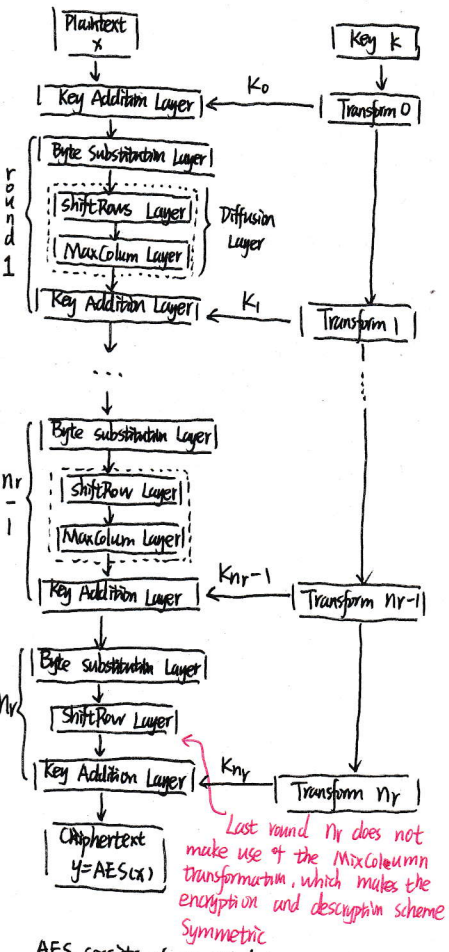
① DES is not particularly well suited for software implementations.
3DES is three times slower than DES

② Relatively short block cipher size of 64 bits, which is a drawback in certain applications.

key length	# rounds = n_r
128	10
192	12
256	14

In contrast to DES, AES does not have a Feistel structure. Feistel networks do not encrypt an entire block per iteration, e.g., in DES, 64/2=32 bits are encrypted in one round.

AES on the other hand, encrypts all 128 bits in one iteration. This is one reason why it has a comparably small number of rounds.



AES consists of so-called layers. Each layer manipulates all 128 bits of data path. The data path is also refered to as the state of the algorithm.

There are only different three types of layers.

① Key Addition Layer.

A 128-bit round key, or subkey, which has been drived from the main key in the key schedule, is XORed to the state.

② Byte Substitution Layer (S-Box)

Each element of the state is nonlinearly transformed using lookup tables. with special mathematical properties. This introduces confusion to the data.

③ Diffusion Layer

It provides diffusion over all state bits. It consists of two sublayers both of which perform linear operations:

1) The ShiftRows layer permutes the data on a byte level.

2) The MixColumn layer is a matrix operation which combines (mixes) blocks of four bytes.

Number Theory

[Congruence]

$a_1 \bmod b \equiv a_2$

$= a_2 \pmod b$

[Modular Inversion]

The inverse of x in Z_n is an element y in Z_n such as $x \cdot y = 1$, y is denoted x^{-1}

Theorem:

x in Z_n has an inverse if and only if $\gcd(x, N) = 1$

[Find an multiplicative inverse using the extended Euclidean Algorithm]

$\frac{1}{8} \bmod 11$ $\quad \gcd(11, 8)$

$11 = 8(1) + 3 \quad 3 = 11 - 8(1)$

$8 = 3(2) + 2 \quad 2 = 8 - 3(2)$

$3 = 2(1) + 1 \quad 1 = 3 - 2(1)$

$2 = 2(1)$

$\gcd(11, 8) = 1$

$1 = 3 - 2(1) = 3 - 8 + 3(2)$

$= 3(3) - 8 = 11(3) - 8(3) - 8$

$= 11(3) - 8(4)$

Therefore $1 \equiv 8(-4) \bmod 11$

$1 \equiv 8(7) \bmod 11$

Hence 7 is the inverse of 8 mod 11

[Modular Inversion using Fermat's Little theorem]

Let p be a prime

$\forall x \in (Z_p)^*$: $x^{p-1} = 1$ in Z_p $\quad x^{p-2}$

$\Rightarrow x \cdot x^{p-2} = 1 \Rightarrow$ $x^{-1} = x^{p-2}$

$\frac{1}{8}$ in $Z_{11} \Rightarrow 8^{-1} = 8^9 = (8^3)^3 = (6)^3 = 7$

[Computing roots in Z_p]

Case 1: if $\gcd(e, p-1) = 1$

if $\gcd(e, p-1) = 1$, d is the inverse of e in Z_{p-1} ($d = e^{-1}$ in Z_{p-1})

To compute the eth root of C in this case:

① Find the inverse of e in Z_{p-1}

② Compute $C^{\frac{1}{e}} = C^d$ in Z_p

Case 2: $\gcd(e, p-1) \neq 1$ Hard

[Quadratic Residue Q.R.] $\quad \frac{p-1}{2}$

Euler's theorem: Let p be an odd prime, if x in $(Z_p)^*$ is a Q.R, then $x^{\frac{p-1}{2}} = 1$ in Z_p

Z_{11}: 1^5 2^5 3^5 4^5 5^5 6^5 7^5 8^5 9^5 10^5

1 −1 1 1 1 −1 −1 −1 1 −1

[Computing the square root modular odd prime]

Case 1: $p = 3 \pmod 4$ $\quad \frac{p+1}{4}$

Theorem: if $c \in (Z_p)^*$ is Q.R. Then $\sqrt{c} = c^{\frac{p+1}{4}}$ in Z_p

[Groups]

① Closure 封闭性: $\forall a, b \in G$ then $(a * b) \in G$

② Associativity 结合性: $\forall a, b, c \in G \Rightarrow (a*b)*c = a*(b*c)$

③ Identity 单一性: $\exists e \in G \Rightarrow a*e = a = e*a$ for all $a \in G$

④ Invertability 可逆性: $\forall a \in G$, $\exists a_i \in G \Rightarrow a * a_i = e = a_i * a$

[Group Order]

The order of a group G is its size $|G|$, meaning the number of elements in it.

[Euler's generalization of Fermat]

For $N = p$ (p prime) $\quad \varphi(N) = p - 1$

For $N = pq$ $\quad \varphi(N) = (p-1)(q-1)$

Euler's Theorem: a generalization of Fermat's Theorem

$\forall x \in (Z_N)^*$: $x^{\varphi(N)} \bmod N = 1$

e.g. $5^{\phi(12)} = 5^4 = 625 = 1$ in Z_{12}

Discrete Logarithm Problem

Definition: fix a prime $p > 2$ and g in $(Z_p)^*$ of order q.

Consider the function: $y \mapsto g^x$ in Z_p

Consider the inverse form function:

$\text{Dlog}_g(g^x) = x$ where x in $\{0, \ldots q-2\}$

Given g, x it is relatively easy to compute y.

Given g, y it is hard to compute x

Asymmetric Cryptographic System

A trapdoor function $X \to Y$ is a triple of efficient algorithms (G, F, F^{-1})

① A key generation algorithm (G):
randomized algorithm outputs a key pair (pk, sk)

② An encryption func $(pk, \cdot)$
deterministic algorithm that defines a function $X \to Y$

③ A decryption function $(sk, \cdot)$
defines a function $Y \to X$ that inverts $F(pk, \cdot)$

Intuitively: (G, F, F^{-1}) is secure if $F(pk, \cdot)$ is a "one-way" function. e.g. TDF can be evaluated, but cannot be inverted without sk.

RSA

Key generation

① Select primes: $p = 17$ & $q = 11$

② Compute $n = pq = 17 \times 11 = 187$

③ Compute $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$

④ Select e: $\gcd(e, 160) = 1$; choose $e = 7$

⑤ Determine d: $de = 1 \bmod 160$

$\Rightarrow d = 23$ since $7 \cdot 23 \equiv 1$

⑥ Publish publish key: $pk = (N, e) = (7, 187)$

⑦ Keep secret key private key

$sk = (p, q, d) = (17, 11, 23)$

If an adversary could factorize N she would know p and q and hence also $\phi(n)$

Since e is public knowledge she could then find d, and easily compute $x = y^d$

RSA security depends on the fact that it is so far very difficult to factorize large integers

[Fermat's Factorization Method]

Fermat's factorization method is based on the representation of an odd integers as the difference of two squares: $N = p \cdot q$

$N = i^2 - j^2 = (i-j)(i+j)$

Test whether $i^2 - N$ is a perfect square j^2 for some integer j.

Computational complexity theory is a branch of the theory of computation in mathematics that focuses on classifying computational problems according to their inherent difficulty.

A computational problem is said to be a task that is in principle amenable to being solved by a computer. In other words, the problem may be solved by automatic application of mathematical steps, such as an algorithm.

Elliptic Curve Cryptography $\quad p+1-2\sqrt{p} \le |E| \le p+1+2\sqrt{p}$

$2P$: $\lambda = \frac{3x_1^2 + a}{2y_1}$ $\quad \begin{cases} x_R = \lambda^2 - 2x_1 \\ y_R = -y_1 + \lambda(x_1 - x_R) \end{cases}$

$P+Q$: $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ $\quad \begin{cases} x_R = \lambda^2 - x_1 - x_2 \\ y_R = -y_1 + \lambda(x_1 - x_R) \end{cases}$

Euler Theorem: (Z_p^*) is a cyclic group, that is

$\exists g \in (Z_p)^*$ such that $\{1, g, g^2, g^3, \ldots, g^{p-2}\} = (Z_p)^*$

g is called a generator of $(Z_p)^*$

The order of $g \in (Z_p)^*$ is the size of $\langle g \rangle$

$\text{ord}_p(g) = |\langle g \rangle|$ $\quad 4a^3 + 27b^2 \neq 0$

Elliptic Curve Discrete Logarithm Problem

Problem: given two points P and Q in an elliptic curve E over a finite field Z_p, find an integer i satisfying:

$Q = i \cdot P$

The security of ECC depends on how difficult it is to determine i given iP and P. This is referred to as the elliptic curve logarithm problem.

Compared to factoring integers or polynomials, one can use much smaller numbers for equivalent levels of security

Elliptic Curve Diffie-Hellman Key exchange.

Public knowledge: A group $E(Z_p)$ and a point g of ordern n.

Bob		Alice
Choose secret $0 < b < n$		Choose secret $0 < a < n$
Compute $Q_{Bob} = bg$		Compute $Q_{Alice} = ag$
Send Q_{Bob}	→	to Alice
to Bob	←	Send Q_{Alice}
Compute bQ_{Alice}		Compute aQ_{Alice}

Cyclic Elliptic Curve Groups

Theorem: A finite abelian group G is cyclic if and only if, for each prime p divisor $|G|$, it has fewer than $p^2 - 1$ elements of order p (in which case it has exactly $p-1$ of them)

Lemma: if the order of an elliptic curve E over a finite field F_q denoted $|E|$ is a prime number then the group is a cyclic and every element is a generator

Discrete Logarithm Problem

Define: fix a prime ($p > 2$) and g in $(Z_p)^*$ of order q (or q)

Consider the function: $y \mapsto g^x$ in Z_p

Consider the inverse function

$\text{Dlog}_g(g^x) = x$ where $x = \{0, \ldots, q-2\}$

Given g, x, it is relatively easy to compute y

Given g, y, it is hard to compute x (Discrete Logarithm problem)

The big O notation is used to classify algorithms by how they respond to changes in input size in terms of processing time or working space requirement.

Message Authentication Code and Digital Signature

Message integrity is achieved by ensuring that a received message has actually originated from the intended party, and was not modified even if an attacker controls the channel.

[Message Authentication Code MAC]

① Signing algorithm (tag generation): takes as input a message m and a key k and outputs a tag t.

② Verification algorithm: takes a key k, message m, and tag t as input; outputs 1 ("accept") or 0 ("reject").

What does forgery means?

If an attacker A is able to produce a pair (m', t'), such that m' did not originate from the sender, and $V(m', k, t') = 1$. Such a pair is called "forgery" and the attacker is said to have forged.

Security goal: external unforgeability

A message authentication code is said to be secure if and only if MAC is able to detect any attempt by the adversary to modify the transmitted data.

Attacker should not be able to produce a new valid (message, tag) pair or even produce a new tag for an old message.

$Adv[A, MAC] = Pr[V(k, m', t') = 1]$ is "negligible"

Replay attacks

Replay attack is when an attacker re-send old messages which have valid tags.

Note that replay attacks are not prevented by the use of MACs

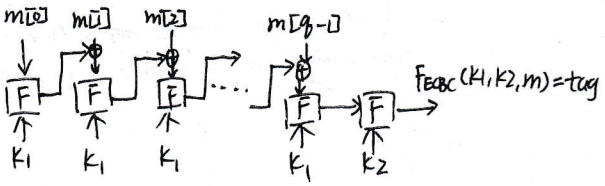
No stateless mechanism can prevent them

ECBC-MAC

Definition: Let $F: K \times X \to X$ be a PRP, m is a long message which is divided into q segment $m[0] \dots m[q-1]$,

$S(k1, k2, m) = F_{ECBC}(k1, k2, m)$

$V(k1, k2, m, t)$: output '1' if $t = F_{ECBC}(k1, k2, m)$ and '0' otherwise

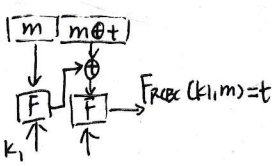


Simple Attack on Raw CBC

① Adversary chooses an arbitrary one-block message $m \in X$

② He requests tag for m. Get $t = F_{ECBC}(k, m)$ (Chose Message Attack)

③ He simply outputs t as MAC forgery for the 2-block message $(m, t \oplus m)$



$F_{ECBC}(k, (m, t \oplus m)) =$

$F(k, F(k, m) \oplus (t \oplus m)) =$

$F(k, t \oplus (t \oplus m)) =$

$F(k, m) = t$

$F_{ECBC}(k, (m, m \oplus t))$

$= F_{ECBC}(k, F(k, m) \oplus m \oplus t)$

$= F_{ECBC}(k, m)$

$= t$

Hash Functions $H: \{0,1\}^N \to \{0,1\}^n$

A hash function: maps arbitrary length inputs to short, fixed-length digest:

→ Pre-image: if $y = H(x)$, x is a pre-image of y. Each hash value typically has multiple pre-images

A collision is a pair of distinct inputs x, x' such that $H(x) = H(x')$

[Security Requirements]

① Pre-image resistant: if it is computationally infeasible to find a pre-image of a hash value (given a value y it is infeasible to find an x such that $h(x) = y$).

② Collision resistant: if it is computationally infeasible to find a collision.

- given x and $h(x)$, infeasible to find $y \neq x$ such that $h(y) = h(x)$
- Infeasible to find any x and y, with $x \neq y$ such that $h(x) = h(y)$

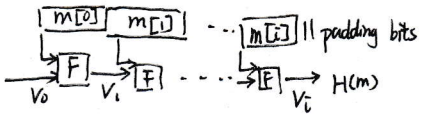
[Application]

① Message Authentication Codes

② Create a one-way password file in order to store hash of password not actual password.

③ For instruction detection and virus detection by creating hashes of files on system and monitoring these for any changes

Collision Resistant Hash Function (Merkle-Damgard Scheme)



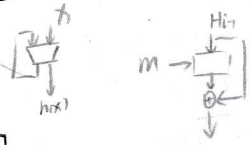
The message is broken into blocks of size K

Padding bits are appended to the last block if its size is smaller than K

If the message happens to be multiple of K, then an extra padding block will be added.

F is a collision resistant compression function

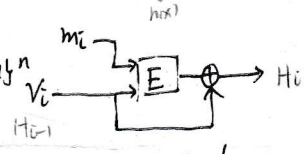
↖ If F is a collision-resistant, then is H collision-resistant



The Davies-Meyer compression function:

Consider a block cipher $E(k \times \{0,1\}^n) \to \{0,1\}^n$

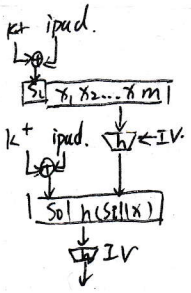
$F(V, m) = E(m, H) \oplus H$



Hash Message Authentication Codes (HMAC)

HMAC: $S(k, m) = H(k \oplus opad \,||\, H(k \oplus ipad \,||\, m))$ ← Signing Algorithm

- ipad 防止 collision 替代攻击，但是攻击者可以添加已经算过hash
- opad 防止 添加包经算hash，但是无法防 collision 替代攻击



Digital Signature

① Data origin authentication of the signer

A digital signature validates the message in the sense that assurance is provided about the integrity of the message and ~~of~~ the identity of the entity that signed the message.

② Non-repudiation

A digital signature can be sorted by anyone who receives the signed message as evidence that the message was sent and of who sent it. This evidence could later be presented to a third party who could use the evidence to resolve any dispute 纠纷 that related to the contents and/or origin of the message.

Definition: a digital signature system (DSS) consists of a triple of efficient algorithm (G, S, V)

① A key generation algorithm that generate a private key at random from a set of possible private keys and a corresponding public key (pk, sk)

② A signing algorithm that, given a message and a private key, produces a signature. $S(sk, m)$ outputs t

③ A signature verifying algorithm that, given a message, public key and a signature, either accepts or rejects the message's claim to authenticity. $V(pk, m, t)$ outputs '1' or '0'.

More About Block Ciphers

The different ways of encryption are called modes of operation. Block ciphers can also be used for constructing hash functions, message authentication codes (MACs), or key establishment protocols.

Encryption with Block Ciphers : Modes of Operation

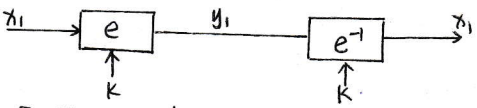
In practice one wants typically to encrypt more than one single 8-byte or 16-byte ~~of~~ block of ~~ciphers~~ plaintext. There are several ways of encrypting long plaintexts with a block cipher.

Electronic Code Book (ECB).
Cipher Block Chaining mode (CBC)
Cipher Feedback mode (CFB)
Output Feedback mode (OFB)
Counter mode (CTR)

All of the five modes have one goal: They encrypt data and thus providing confidentiality for a message

ECB and CFB modes require that the length of the plaintext be an exact multiple of the block size of the cipher used. otherwise, the plaintext must be padded.

Electronic Codebook mode (ECB)



The Electronic Code Book mode is the most straightforward way of encrypting a message.

$e_K(x_i)$ denotes the encryption of plaintext block x_i with key k. $e_K^{-1}(y_i)$ denotes the decryption of cipher block y_i.

Let e() be a block cipher of block size b, and let x_i and y_i be bit string of length b

Encryption: $y_i = e_K(x_i)$, $i \geq 1$
Decryption: $x_i = e_K^{-1}(y_i) = e_K^{-1}(e_K(x_i))$; $i \geq 1$

Advantage: Block synchronization between the encryption and decryption parties Alice and Bob is not necessary. i.e. If the receiver does not receive all encrypted block due to transmission problems, it is still possible to decrypt the received blocks. Similarly, bit errors, e.g., caused by noisy transmission lines, only affect the corresponding block but not succeeding blocks.

Can be parallelized, adv for high-speed implementations.

The main problem of ECB mode is that it encrypts highly deterministically. This means, identical plaintext blocks result in identical ciphertext blocks as long as the key does not change.

The ECB mode can be viewed as a gigantic code book which maps every input to a certain output. (as long as the key is static the book is fixed). ✓ (traffic analysis)
↳ Consequences: an attack recognizes if the same message has been sent twice simply be looking at the ciphertext. For instance, if there is a fixed header that always precedes a message, the header always results in the same ~~plaintext~~ cipher. (can learn when a new message has been sent.

Secondly, plaintext blocks are encrypted independently of previous blocks. If an attacker recorders the ciphertext blocks, this might result in valid plaintext and the reordering may not be detected.

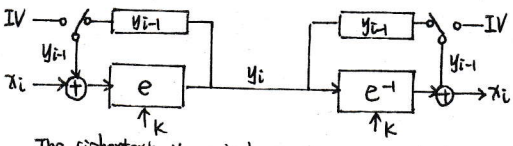
① Substitution attacks. $x_i \to y_i$ (P140) 替换 ciphertext block
② Bit maps. (图片) identical plaintexts are mapped to identical ciphertexts. The information (text in picture) can still be read out from the encrypted picture even through AES with a 256-bit key is used.
↳ Not an attack that breaks the block cipher itself. Messages still remain confidential.

Both attacks were examples of the weakness of a deterministic encryption scheme.

Preferable that different ciphertexts are produced every time we encrypt the same plaintext. This behavior is called probabilistic encryption.

Cipher Block Chaining Mode (CBC)

① Ciphertext y_i depends not only on block x_i but on all previous plaintext block as well.
② The encryption is randomized by using an initialization vector (IV)



The ciphertext y_i, which is the result of the encryption of plaintext block x_i, is fed back to the cipher input and XORed with the succeeding plaintext block x_{i+1}.

The XOR sum is then encrypted, yielding the next ciphertext y_{i+1}, which can then be used for encrypting x_{i+2}.

For the first plaintext block x_1, there is no previous ciphertext. For this, an IV is added to the first plaintext, which also allows us to make each CBC encryption nondeterministic. The last ciphertext is a function of all plaintext blocks and the IV.

Let e() be a block cipher of block size b; let x_i and y_i be bit strings of length b; and IV be a nonce of length b (nonce: 临时的, 只用一次的)

Encryption (first block): $y_1 = e_K(x_1 \oplus IV)$
Encryption (general block): $y_i = e_K(x_i \oplus y_{i-1})$, $i \geq 2$
Decryption (first block): $x_1 = e_K^{-1}(y_1) \oplus IV$
Decryption (general block): $x_i = e_K^{-1}(y_i) \oplus y_{i-1}$, $i \geq 2$

$d(y_1) = e_K^{-1}(y_1) \oplus IV = e_K^{-1}(e_K(x_1 \oplus IV)) \oplus IV = x_1$
$d(y_i) = e_K^{-1}(y_i) \oplus y_{i-1} = e_K^{-1}(e_K(x_i \oplus y_{i-1})) \oplus y_{i-1} = x_i$

If we choose a new IV every time we encrypt, the CBC mode becomes a probabilistic encryption scheme. Note that we do not have to keep the IV secret. But we want the IV to be a nonce (a number used only once).

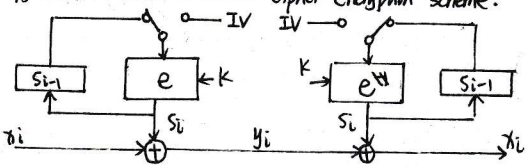
If the IV is properly chosen for every wire transfer (电汇), the substitution attack will not work at all since Oscar will not recognize any patterns in the ciphertext. ✗ If the IV is kept the same for several transfers, he would recognize the transfers from his account at bank A to his account at bank B. However, if he substitutes ciphertext block 4, which is his encrypted account number, in other wire transfers going from bank A to bank B, bank B would decrypt block 4 and 5 to some random value. Even though money would not be redirected into Oscar's account, it might be redirected to some other random account.

This example shows that even though Oscar cannot perform specific manipulations, ciphertext alternations by him can cause random changes to the plaintext which can have major negative consequences.

Hence in many, if not in most, real-world sys, encryption itself is not sufficient: we also have to protect the integrity of the message.

Output Feedback Mode (OFB)

In the Output Feedback (OFB) mode a block cipher is used to build a stream cipher encryption scheme.

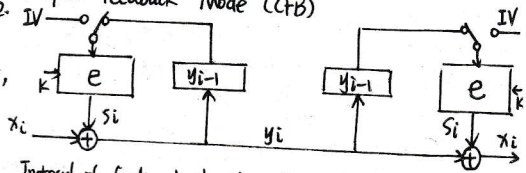


We start with encrypting an IV with a block cipher. The cipher output gives up the first set of b key stream bits. The next block of key stream bits is computed by feeding the previous cipher output back into the block cipher and encrypting it.

The OFB mode forms a synchronous stream cipher as the key stream does not depend on the plain or ciphertext.

Encryption (first block): $s_1 = e_K(IV)$ and $y_1 = s_1 \oplus x_1$
Encryption (general): $s_i = e_K(s_{i-1})$ and $y_i = s_i \oplus x_i$ $i \geq 2$
Decryption (first block): $s_1 = e_K(IV)$ and $x_1 = s_1 \oplus y_1$
Decryption (general): $s_i = e_K(s_{i-1})$ and $x_i = s_i \oplus y_i$

Cipher Feedback Mode (CFB)

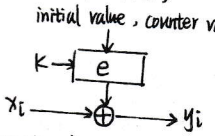


Instead of feeding back the output of the block cipher, the ciphertext is fed back.

$y_1 = e_K(IV) \oplus x_1$ $\quad x_1 = e_K(IV) \oplus y_1$
$y_i = e_K(y_{i-1}) \oplus x_i$ $\quad x_i = e_K(y_{i-1}) \oplus y_i$

As a result of the use of an IV, the CFB encryption is also nondeterministic, hence, encrypting the same plaintext twice results in different ciphertext.

Counter Mode (CTR)



The input to the block cipher is a counter which assumes a different value every time the block cipher computes a new key stream block.

The counter is incremented but the IV stays the same

$y_i = e_K(IV \| CTR\,i) \oplus x_i$, $i \geq 1$
$x_i = e_K(IV \| CTR\,i) \oplus y_i$, $i \geq 1$

One of attractive feature of the counter mode is that it can be parallelized because it does not require feedback.

Remarks.

CBC: Error propagate — Stream cipher
Serial processing (slow) ~~(feedback)~~ ✗
CTR: Needs only the encryption algorithm ✓
Fast (parallel)
CFB: Error propagate ✗✓
Serial processing (slow)
OFB: More reliable ✓
ECB: ↙

Double Encryption and Meet-in-the Middle Attack

Oscar first brute-force-attacks the encryption on the left-hand side, which requires 2^k cipher operations, and then the right encryption, which again requires 2^k operations. If he succeeds with this attack, the total complexity is $2^k + 2^k = 2 \cdot 2^k = 2^{k+1}$.

The attack has two phases. In the first one, the left encryption is brute-forced and a lookup table is computed. In the second phase the attacker tries to find a match in the table which reveals both encryption keys.

Triple Encryption

$$y = e_{k3}(e_{k2}(e_{k1}(x)))$$

In practice, often a variant of the triple encryption EDE: $y = e_{k3}(e_{k2}^{-1}(e_{k3}(x)))$ (Encryption-Decryption-Encryption)

The reason for this has nothing to do with security

If $K_1 = K_2$, the operation effectively performed is

$$y = e_{k3}(x)$$

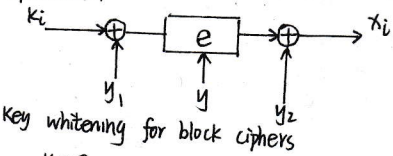
Again, we assume K bits per key. The problem for an attacker is that she has to compute a lookup table either after the first or after the second encryption. In both cases, the attacker has to compute two encryptions or decryptions in a row in order to reach the lookup table.

3DES: 2^{112} key tests ($2^{2 \cdot 56} + 2^{56} \approx 2^{112}$)

Key Whitening

More resistant against brute-force attacks

In addition to the regular cipher key k, two whitening key k_1 and k_2 are used to XOR-mast the plaintext and ciphertext.



Key whitening for block ciphers

$$y = e_{k,k_1,k_2}(x) = e_k(x \oplus k_1) \oplus k_2$$

$$x = e^{-1}_{k,k_1,k_2}(x) = e_k^{-1}(y \oplus k_2) \oplus k_1$$

$e_k^{-1}(e_k(x \oplus k_1) \oplus k_2) \oplus k_2) \oplus k_1 = x$

It is important to stress that key whitening does not strengthen block ciphers against most analytical attacks such as linear and differential cryptanalysis.

This is in contrast to multiple decryption, which often also increase the resistance to analytical attacks. Hence, key whitening is not a "cure" for inherently weak ciphers. It's main application is ciphers that are relatively strong against analytical attacks but posess too short a key space. The prime example of such a cipher is DES. A variant of DES which uses key whitening is DESX. It the case of DES X, the key k_2 is derived from K and k_1.

Most modern block ciphers such as AES already apply key whitening internally by adding a subkey prior to the first round and after the last round.

Currently (2010), there are 8 approved block cipher modes:

① Five for confidentiality (ECB, CBC, CFB, OFB, CTR)

② One for authentication (CMAC)

③ Two combined modes (CCM, GCM)

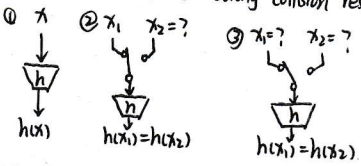
Hash Functions

Hash Functions compute a digest of a message which is a short, fixed-length bit-string.

For a particular message, i.e., a unique representation of message. the hash function value or the message digest (摘要) can be seen as the fingerprint of a message.

Three central properties which hash functions need to process in order to be secure:

① Preimage resistance (or one-wayness)
② Second preimage resistance (or weak collision resistance)
③ Collision resistance (or strong collision resistance)



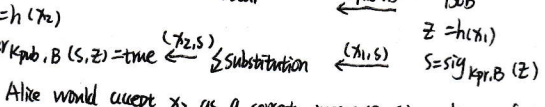
[Preimage Resistance or One-Wayness]

If the hash funch is not a one-way, Oscar can compute the message x from $h^{-1}(z)=x$. Thus the symmetric encryption of x is circumvented (规避) by the signature, which leaks the plaintext.

[Second Preimage ~~Attack~~ Resistance or Weak Collision Resistance]

x_1 is given and we try to find x_2.

Assumes Bob hashes and signs a message x_1. If Oscar is capable of finding a second message x_2 such that $h(x_1)=h(x_2)$ he can run the following substitution attack:



Alice would accept x_2 as a correct message since the verification gives her the statement "true".

Impossible to have a hash function for which weak collisions do not exist due to pigeonhole (鸽巢) principle.

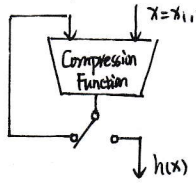
Overview of hash functions

Two general types of hash functions.

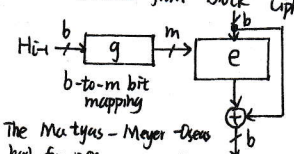
① Dedicated hash functions
These are algorithms that are specifically designed to serve as a hash f.

② Block cipher-based hash functions.

Hash functions can process an arbitrary-length message and produce a fixed-length output. In practice, this is achieved by segmenting the input into a series of blocks of sequal size. These blocks are processed sequentially by the hash function, which has a compression function at its heart. This iterated design is known as Merkle-Damgard construction.



Hash function from Block Ciphers



The Matyas-Meyer-Oseas hash function construction from block ciphers

The function can be expressed as:

$$H_i = e_{g(H_{i-1})}(x_i) \oplus x_i$$

$$\begin{cases} H_i = H_{i-1} \oplus e_{x_i}(H_{i-1}) & \text{(Davies-Meyer)} \\ H_i = H_{i-1} \oplus x_i \oplus e_{g(H_{i-1})}(x_i) & \text{(Miyaguchi-Preneel)} \end{cases}$$

Birthday attack reduces the security level (merely 50%)

The Secure Hash Algorithm SHA-1

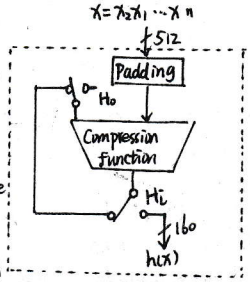
SHA-1 is based on a Merkle-Damgard construction.

[High-level diagram of SHA-1]

SHA-1 produces a 160-bit output of a message with a maximum length of 2^{64} bit

During the actual computation, the compression function processes the message in 512-bit chunks.

The compression function consists of 80 rounds which are divided into four stages of 20 rounds each.



[Preprocessing]

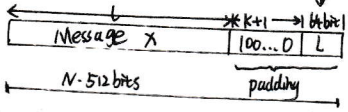
Before the actual hash ~~function~~ computation, the message x has to be padded to fit a size of a multiple of 512 bit. For the internal processing, the padded message must then be divided into blocks. Also, the initial value H_0 is set to a predefined constant.

Padding

Assume we have a message x with a length of L bit. To obtain a overall message size of a multiple of 512 bits, we append a single '1' followed by k zero bits and the binary 64-bit representation of L.

Consequently, the number of required zeros k is given by

$$k = 512 - 64 - 1 - L$$
$$= 448 - (L+1) \bmod 512$$



For example: message "abc" L=24

k=512-64-1-24=423

01100001 01100010 01100011 1 00...00 00...011000
a b c 423 L=24

Dividing the padded message

Divide the message into 512-bit blocks $\{x_1, x_2, \ldots, x_n\}$. Each 512-bit block can be subdivided into 16 words of size of 32 bits.

Initial value

A 160-bit buffer is used to hold the initial & hash value for the first iteration. The five 32-bit words are fixed and given in hexadecimal notations as:

$$A = H_0^{(0)} = 67452301 \quad P(321)$$

[Hash Computation]

A message schedule which computes a 32-bit word $W_0, W_1, \ldots, W_{79}$ for each of the 80 rounds. The words W_j are derived from the 512-bit message block as follows:

$$W_j = \begin{cases} x_i^{(j)} & j \in 0..15 \\ (W_{j-16} \oplus W_{j-14} \oplus W_{j-8} \oplus W_{j-3}) \lll 1 & 16..79 \end{cases}$$

SHA-1 was designed to be especially amenable to software implementations.

Message Authentication Codes (MACs)

Also known as a cryptographic checksum or a keyed hash function. In terms of security functionality, MACs share some properties with digital signatures, since they also provide message integrity and message authentication. However, unlike digital signatures, MACs are symmetric-key schemes and they do not provide nonrepudiation.

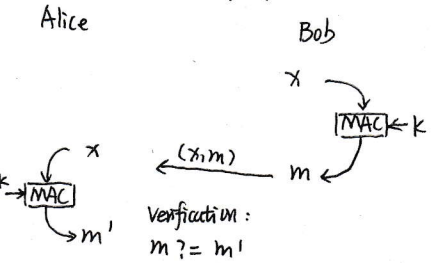
One advantage of MACs is that they are much faster than digital signatures since they are based on either block ciphers or hash functions.

Principles of MACs

MACs append an authentication tag to a message. MACs use a symmetric key k for both generating the authentication tag and verifying it.

A MAC is a function of the symmetric key K and the message X.

$$m = MAC_k(X)$$



The motivation for using MAC is typically that Alice and Bob want to be assured that any manipulations of message X in transit are detected.

Any malicious or accidental forgery of the message will be detected by the receiver due to a failed verification of the MAC.

Properties of MACs

① Cryptographic checksum

A MAC generates a cryptographically secure authentication tag for a given message.

② Symmetric MACs are based on secret sym. keys.

The signing and verifying parties must share a secret key.

③ Arbitrary message size ~~MACs~~

Accept messages of arbitrary length.

④ Fixed output length

Generate fixed-size authentication tag.

⑤ Message integrity

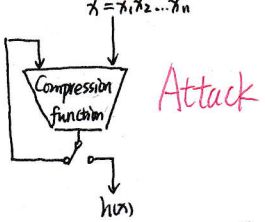
Any manipulations of a message during transit will be detected by the receiver.

⑥ Message authentication.

The receiving party is assured of the origin of the message.

⑦ ~~Non-rep~~ No nonrepudiation

Since MACs are based on symmetric principles, they do not provide nonrepudiation.



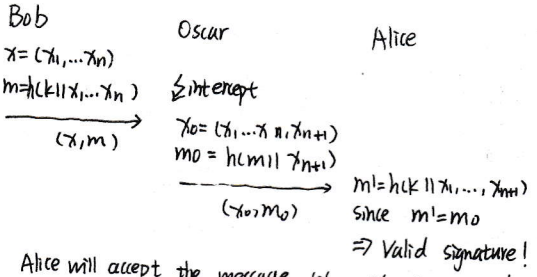
[Attacks Against Secret Prefix MACs]

$m = h(k \| x)$ || means concatenation

For attack we assume that the cryptographic checksum m is computed using a hash function construction as shown above.

This iterated approach is used in the majority of today's hash function. Bob computes an authentication tag as: $m = MAC_k(x) = h(k \| x_1, x_2, \ldots, x_n)$

The problem is that the MAC for the message $x = (x_1, x_2, \ldots, x_n, x_{n+1})$, where x_{n+1} is an arbitrary additional block, can be constructed from m without knowing the secret key.



Alice will accept the message $(x_1, \ldots x_{n+1})$ as valid, even though Bob only authenticated $(x_1, \ldots x_n)$.

The attack is possible since the MAC of the additional message block only needs the previous hash output, which is equal to Bob's m, and x_{n+1} as input but not the key k.

[Attacks against Secret 后缀 Suffix MACs] $m = h(x \| k)$ 上述Attack不适用

A different weakness occurs: $h(x) = h(x_0)$

Construct a collision in the hash function.

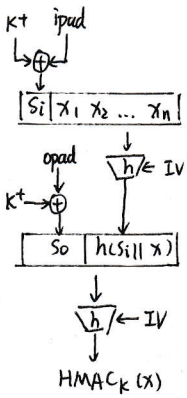
If Bob signs x with a message authentication code

$m = h(x \| k)$

m is also a valid checksum for x_0 $m = h(x \| k) = h(x_0 \| k)$

HMAC

A hash-based message authentication code which does not show the security weakness described above is the HMAC construction. The scheme consists of an inner and outer hash



The MAC computation starts with expanding the symmetric key k with zeros on the left such that the result k^+ is b bits in length, where b is the input block width of the hash function.

The expended key is XORed with the inner pad (ipad)

$$HMAC_k(X) = h[(k^+ \oplus opad) \| h[(k^+ \oplus ipad) \| x]]$$

Public-Key Cryptography

There are several shortcomings associated with symmetric-key schemes.

[Key Distribution Problem]

The key must be established between Alice and Bob using a secure channel.

As the communication like for the message is not secure, so sending the key over the channel directly – which would be the most convenient way of transporting it – can't be done.

[Number of Keys]

Even if we solve the key distribution problem, we must potentially deal with a very large number of keys.

If each pair of users needs a separate pair of keys in the network with n users, there are $\frac{n(n-1)}{2}$ key pairs. And every user has to store $n-1$ keys securely.

Even for a mid-size networks, say, a corporation with 2000 people, this requires more than 4 million key pairs that must be generated and transported via secure channel.

[No Protection Against Cheating by Alice or Bob]

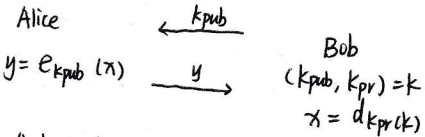
Alice and Bob have the same key capabilities, since they possess the same key. As a consequence, symmetric cryptography cannot be used for applications where we would like to prevent cheating by either Alice and Bob.

For instance, in e-commerce applications it is often important to prove that Alice actually sent a certain message, say, an online order for a flat screen TV. If we only use symmetric cryptography and Alice changes her mind latter, she can claim that Bob, the vendor, has fusely generated the electronic purchase order. Preventing this is called nonrepudiation

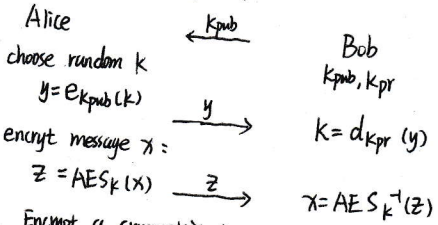
Principles of Asymmetric Cryptography

Basic idea: It is not necessary that the key possessed by the person who encrypts the message is secret. The crucial part is the receiver can only decrypt using a secret key.

In order to realize such a system, Bob publishes a public encryption key which is known to everyone. Bob also has a matching *secret key, which is used for decryption.



A basic key transport protocol:



Encrypt a symmetric key using the public-key algorithm. Once the symmetric key has been decrypted by Bob, both parties can use it to encrypt and decrypt messages using symmetric ciphers.

The advantage of this protocol is that the payload is encrypted with a symmetric cipher, which tends to be much faster than a asymmetric algorithm.

Most asymmetric schemes are based on one common principle, the one-way function.

A function $f(x)$ is a one-way function if:

1. $y = f(x)$ is computationally easy
2. $x = f^{-1}(y)$ is computationally infeasible.

Security Mechanisms

The main functions that public-key cryptography can provide are listed below:

[Key Establishment]

Protocols for establishing secret keys over

- DHKE: Diffie-Hellman key exchange
- RSA key transport protocols

[Nonrepudiation]

Providing nonrepudiation and message integrity can be realized with digital signature algorithms

RSA, DSA (digital signature algorithm)

ECDSA (elliptic curve digital signature algorithm)

[Identification]

Identify entities using challenge-and-response protocols together with digital signature

[Encryption]

...

Even through public-key schemes can provide all function required by modern security protocols, the major drawback in practice is that encryption of data is very computationally intensive, extremely slow – with public-key algorithm.

In order to use the best of both worlds, most practical protocols are hybird protocols which incorporate both symmetric and public-key algorithms.

SSL/TLS protocol that is commonly used for secure Web connections, or IPsec, the security part of the Internet communication protocol.

An algorithm is said to have a "security level of n bit" if the best known attack requires $\geq 2^n$ steps.

Fermat's Little Theorem

Let a be an integer and p be a prime:

$$a^p \equiv a \bmod p$$

$$a^{p-1} \equiv 1$$

$$a \cdot a^{p-2} \equiv 1 \pmod p$$

$$a^{-1} \equiv a^{p-2} \pmod p$$

Euler's Phi Function

The number of integers in $\mathbb{Z}_m$ relatively prime to m is denoted by $\Phi(m)$

e.g. $\gcd(0,5) = 1$ ✓
$\gcd(2,5) = 1$ ✓
$\gcd(3,5) = 1$ ✓
$\gcd(4,5) = 1$ ✓

$\Phi(5) = 4$

Digital Signatures

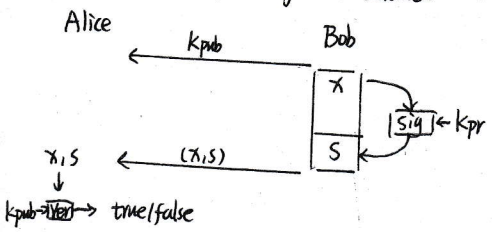
So far we've always assumed that the bad guy is an external party that we often named Oscar. However, in practice it is often the case that Alice and Bob do want to communicate securely with each other, but at the same time they might be interested in cheating each other. It turns out that symmetric-key schemes do not protect the two parties against each other.

Priciples of Digital Signatures

As with conventional hand-written signatures, only the person who creates a digital message must be capable of generating a valid signature.

Public-key cryptography. The basic idea is that the person who signs the message uses a private key, and the receiving party uses the matching public key.

The principle of a digital signature scheme:



The process starts with Bob singing the message x. The signature algorithm is a function of Bob's private key, Kpr. Hence, assuming he in fact keeps his private key private, only Bob can sign a message x on his behalf.

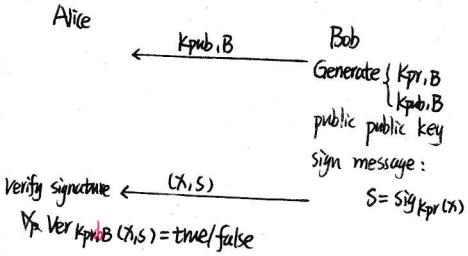
In order to relate a signature to the message, x is also an input to the signature algorithm. After signing the message, the signature is s is appended to the message x and their pair (x,s) is sent to Alice.

It is important to note that a digital signature by itself is of no use unless it is accompanied by the message.

The digital signature itself is merely a (large) integer value, for instance, a string of 2048 bits. This signature is only useful to Alice if she has means to verify wheather the signature is valid or not. For this, a verification func is needs which takes both x and the signature s as inputs. In order to like the signature to Bob, the func also requires his public key. Even though the verification function has long inputs, its only output is the binary statement "true" or "fake". If x was actually signed with the private key that belongs to the public verification key, the output is true. otherwise, it is fake.

General digital signature protocol.

[Basic Digital Signature Protocol]



The core property of digital signatures follows: A signed message can unambiguously be tracked back to its originator since a valid signature can only be computed with the unique signer's private key. Hence we can prove that the signing party has actually generated the message.

Alice receives the signed message and RSA-decrypts s using Bob's public key Kpub, yielding x. If x and x' match, Alice knows two important things: First, the author of the message was in possession of Bob's secret key, and if only Bob has had access to the key, it was in fact Bob who signed the message. (Message authentication) Second, the message has not been changed in transmit (message integrity).

Security Services

What are possible security objectives that a security systems one-called might process? More accurately the objectives of a security systems are called security services.

There exist many security services, but the most important ones which are desirable in many applications are:

① Confidentiality

Information is kept secret from all but authorized parties.

② Integrity

Messages have not been modified in transit

③ Message Authentication

The sender of a message is authentic. An alternative term is data origin authentication.

④ Non repudiation

The sender of a message can not deny the creation of the message.

Different applications call for different sets of security services. For instance, for private-email the first three functions are desirable, whereas a corporate e-mail system might also require non repudiation.

⑤ Identification / entity authentication.

Establish and verify the identity of an entity.

⑥ Access control

Restrict access to the resources to privileged entities

⑦ Availability

Asures that the electronic system is reliably available.

⑧ Auditing (審記)

Provide evidence about security-relevant activities
eg. by keeping logs about certain events

⑨ Physical Security

Provide protection against physical tampering and/or responses to physical tampering attempts.

⑩ Anonymity 匿名

Provide protection against discovery and misuse of id

The RSA Signature Scheme

↑ base on RSA encryption whose security is to relies on the difficulty of factoring a product of two large primes

Schoolbook RSA Digital Signature

Suppose Bob wants to send a signed message x to Alice. He generates At the end of the set-up he has the following parameters:

RSA Keys { Bob's private key: $k_{pr} = (d)$
Bob's public key: $k_{pub} = (n, e)$

Basic RSA Digital Signature Protocol

As can be seen from the protocol, Bob computes the signature s for a message x by RSA-encrypting x with his private key Kpr. Bob is only part who can apply Kpr and hence the ownship Kpr authenticates him as the author of the signed message. Bob sends appends the signature s to the message x and sends both to Alice.

The Elliptic Curve Digital Signature Algorithm

The ECDSA Algorithm

The ECDSA standard is defined for elliptic curves over prime field Z_p and Galois field $GF(2^m)$

[Key Generation]

1. Use an elliptic curve E with
   modulus p
   coefficients a and b
   a point A which generates a cyclic group of prime order q
2. Choosen a random integer d with $0 < d < q$.
3. Compute $B = d \cdot A$

$$\begin{cases} K_{pub} = (p, a, b, q, A, B) \\ K_{pr} = d \end{cases}$$

The cyclic group has an order q which should have a size of at least 160 bit or more for higher security levels.

[Signature and Verification]

Like DSA, an ECDSA signature consists of a pair of integers (r,s). Each value has the same bit length as q, which makes for fairly compact signatures. Using the public and private key, the signature for a message x is computed as follows

ECDSA Signature Generation

1. Choose an integer as random ephemeral key k_E $0 < k_E < q$ 短暫的
2. Compute $R = k_E A$
3. Let $r = x_R$
4. Compute $s \equiv (h(x) + d \cdot r)\, k_E^{-1} \bmod q$

In step 3 the x-coordinate of the point R is assigned to the variable r. The message x has to be hashed using the function h in order to compute s. The hash function output length must be at least as long as q.

The hash function hash compresses x and computes a fingerprint which can be viewed as a representative of x.

ECDSA Signature Verification

1. Compute auxiliary value $w \equiv s^{-1} \bmod q$
2. Compute auxiliary value $u_1 \equiv w \cdot h(x) \bmod q$
3. Compute auxiliary value $u_2 \equiv w \cdot r \bmod q$
4. Compute $P = u_1 A + u_2 B$
5. The verification $Ver_{K_{pub}}(x, (r,s))$ follows from:

$$x_P = \begin{cases} \equiv r \bmod q \Rightarrow \text{valid signature} \\ \not\equiv r \bmod q \Rightarrow \text{invalid signature} \end{cases}$$

In the last step, the notation x_P indicates the x-coordinate of the point P. The verifier accepts a signature (r,s) only if the x_P has the same value as the signature parameter r modulo q. otherwise, the signature should be considered invalid.