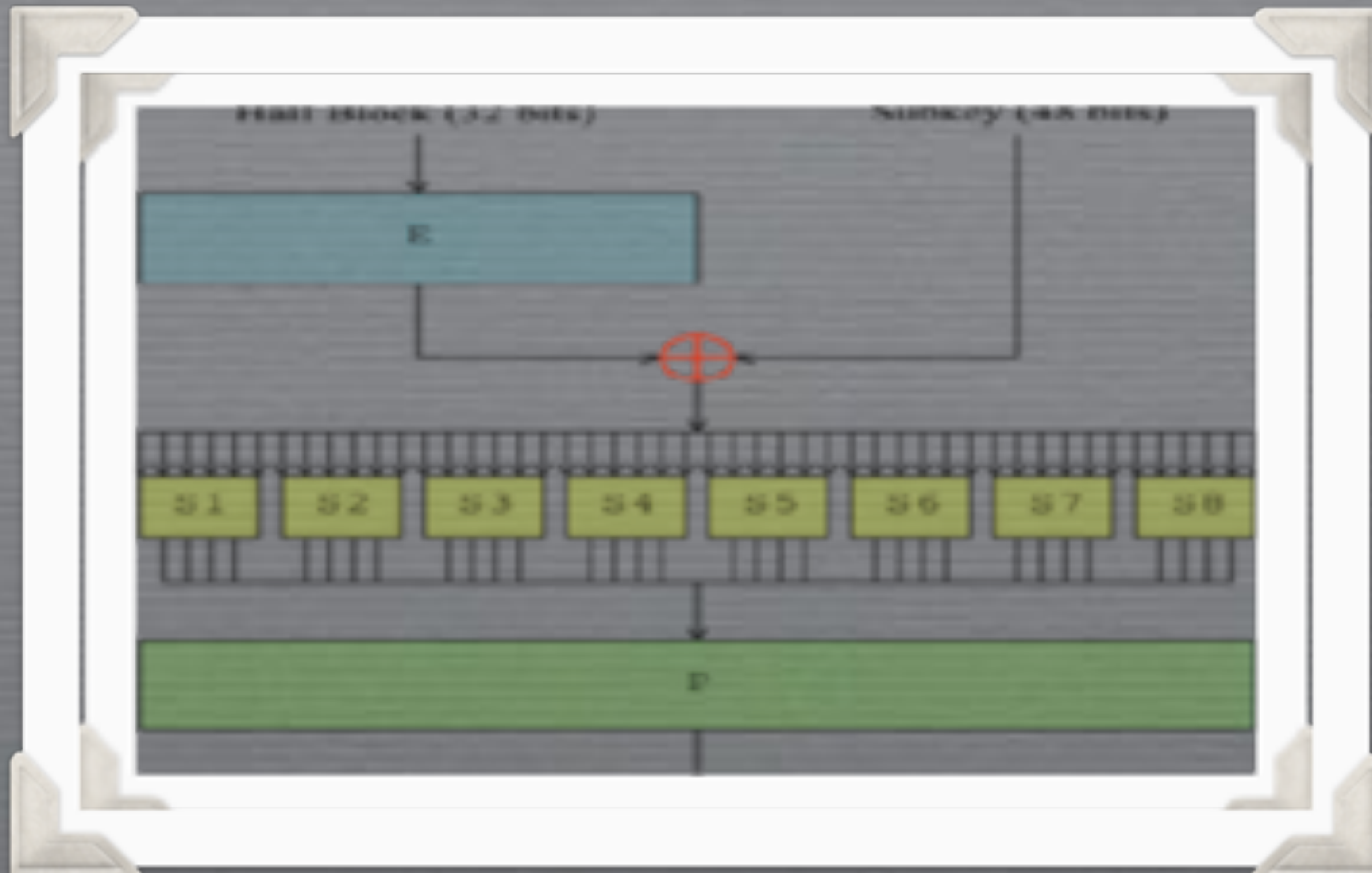
Symmetric Encryption

Phong Nguyễn

http://www.di.ens.fr/~pnguyen

INVENTEURS DU MONDE NUMÉRIQUE



Lesson 9: Randomize Encryption

Symmetric Encryption

- Encryption and decryption depend on the same (secret) key.

Plaintext m

Encryption

010001100100101

Ciphertext

$c=E_k(m)$

- Decryption: $m=D_k(c)$

Remarks

- Given a plaintext m and a ciphertext $c=E_k(m)$, it is not possible to know if they correspond, without knowing the secret key k.
- But this has the drawback of being deterministic: if you encrypt the same message m twice with the same key, you will get the same ciphertext.

Deterministic is Weak

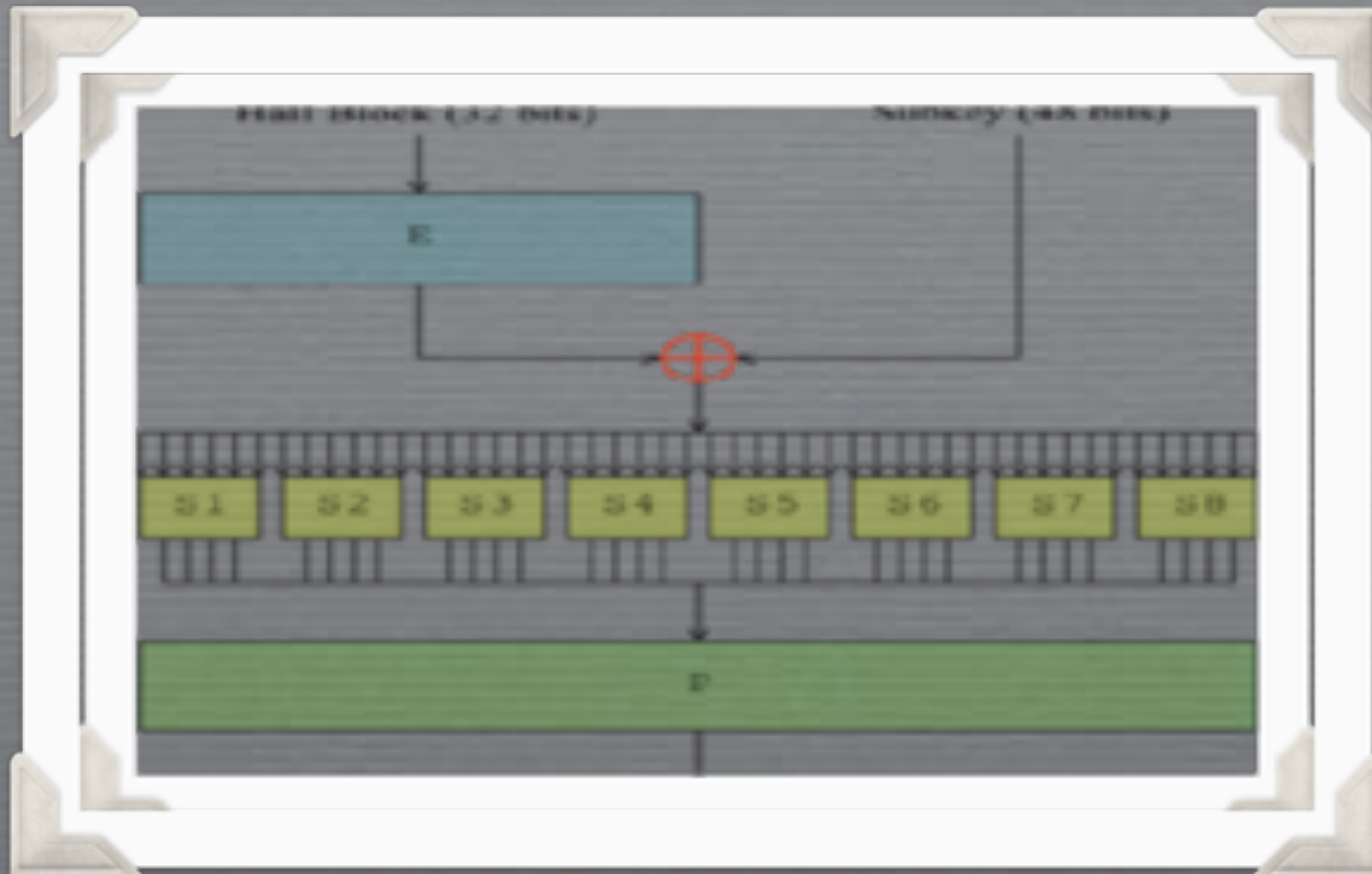
- Given two ciphertexts $c_1=E_k(m_1)$ and $c_2=E_k(m_2)$, one can check if $m_1=m_2$ by checking if $c_1=c_2$, without knowing the secret key.
  - This would be a disaster in electronic voting: just by looking at encrypted votes, you would know who voted the same way.
  - If you encrypt a message by blocks, the blocksize must be big enough to prevent statistical analysis.

Randomizing Encryption

- Strong encryption requires randomness, achieved by using an IV ("initial value"), which changes at each encryption.
- We have $c=E_{k,IV}(m)$ where IV may be:
  - a public random number sent with c
  - or a secret number that both parties can reconstruct, such as IV = E_k(public data).

Two Kinds of Symmetric Encryption

- The secret key is k.
- **Stream cipher**: k and the IV generate a stream of pseudo-random bits, which is XORed with the message.
- **Block cipher**: k defines a permutation over a block of bits (typically, 64 or 128). This permutation is used many times to encrypt arbitrary messages, through a mode of operation. The permutation is hopefully pseudo-random.



Lesson 10: Pseudo-random is as Good as Random

Stream Cipher

- Input:
  - a message stream M.
  - a secret key K, typically 128 bits.
  - an "initial value" IV, typically 64 to 128 bits.
- Output: a ciphertext stream C.
- Most common: synchronous stream cipher $C = M \oplus F$ where F is a stream of pseudo-random bits generated from K and IV.

Ex: Lorenz

- Used by the German army during WW2 for ultra-confidential communications.

Stream Ciphers Today

- Well-suited to radio-communications
  - Messages are streamed.
  - Limited error propagation: one wrong bit of ciphertext only affects one bit of message.
  - Hardware efficiency: few gates, low consumption.
  - Ciphertext and plaintext have same size.
- Many proprietary and confidential algorithms.

Famous Stream Ciphers

- RC4 used in SSL, WEP and WPA. Tailored for 8-bit soft.
- A5/1 used in GSM mobile phones. Based on LFSRs (very efficient in hard), like most stream ciphers.

- Similar construction for E0, used in Bluetooth.

The Need for IVs

- Suppose we only generate the stream from a secret key k: problems?
- In practice, we often use a session key, but we use several IVs, splitting the message into frames.
- The IV may be public, but it needs to be "random"

Stream Cipher Structure

- There is an internal state, which is initialized by the secret key and the IV.
- At each clock cycle:
  - Output a fixed number of pseudo-random bits
  - Update the internal state

The RC4 Stream Cipher

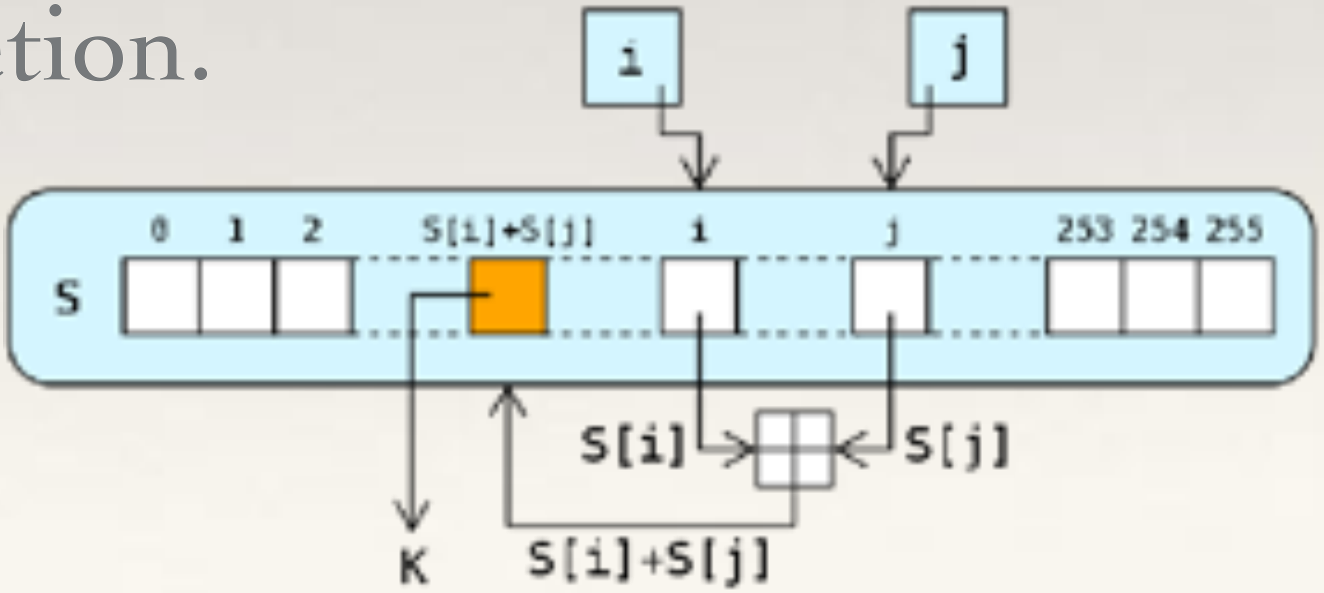
- Invented in 1987 by Ron Rivest: Ron's Cipher 4. Unrelated to the block ciphers RC5 and RC6.
- Disclosed anonymously in 1994.
- Widespread use: SSL, WEP, WPA, Windows, Lotus, Oracle, Skype, Bittorrent.
- Very simple to implement.
- More a pseudo-random number generator than a stream cipher, because IVs were not planned.

Structure of RC4

- Internal state:
  - Two bytes i and j: operations mod 256.
  - A permutation over bytes = {0,...,255}, represented as an array S[0..255].
- The number of internal states is huge $\approx 2^{1700}$.
- At each clock cycle, RC4 outputs one pseudo-random byte.

Description of RC4

- Initialization by a key K = array of m bytes.
  - S := (0,1,....,255); j := 0
  - For i :=0 to 255
    - j := j + S[i] + K[i mod m]
    - Swap S[i] and S[j]
- Update and Byte Extraction.
  - i++; j := j + S[i]
  - Swap S[i] and S[j]
  - Output S[S[i]+S[j]]



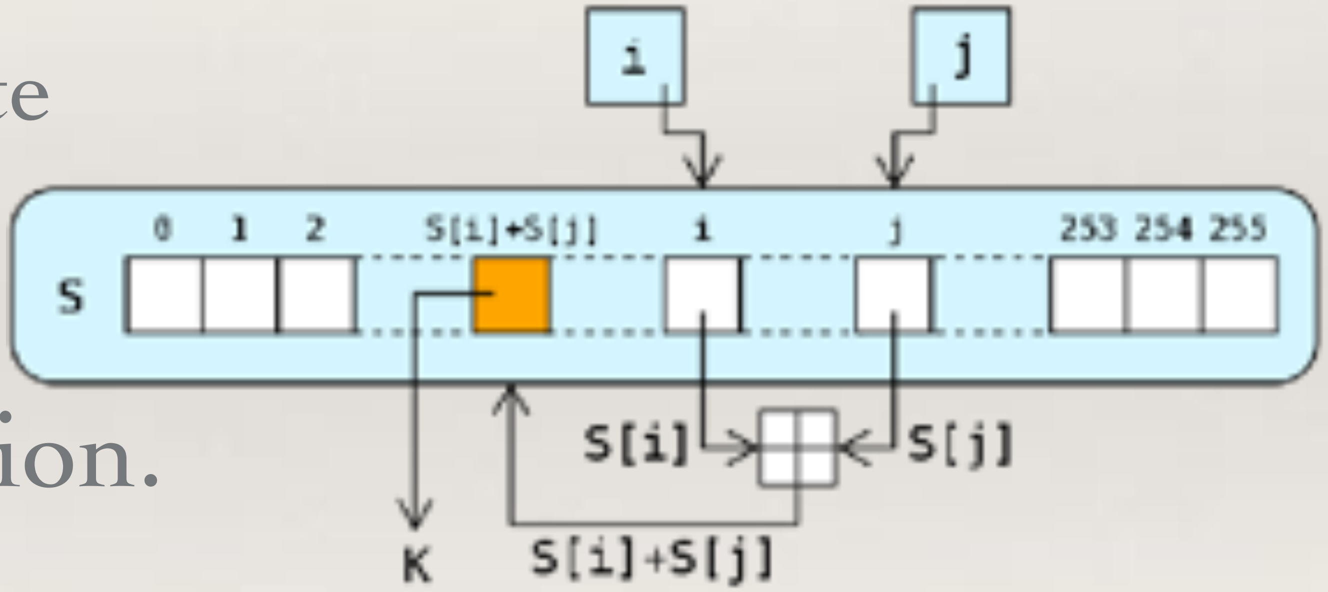
Intuition behind RC4

- RC4 is based on shuffling "cards". Instead of 52 cards, RC4 deals with 256 cards.

- Initialization shuffles the deck S.
- Update slightly shuffles the deck S.

Idealized RC4

- Initialization.
  - S := (0,1,....,255); j := 0
  - For i :=0 to 255
    - j := pseudo-random byte
    - Swap S[i] and S[j]
- Update and Byte Extraction.
  - i++; j := pseudo-random byte
  - Swap S[i] and S[j]
  - Output S[pseudo-random byte]



Security of RC4

- The first output bytes are not perfectly random:
  - The 1st byte is not uniformly distributed.

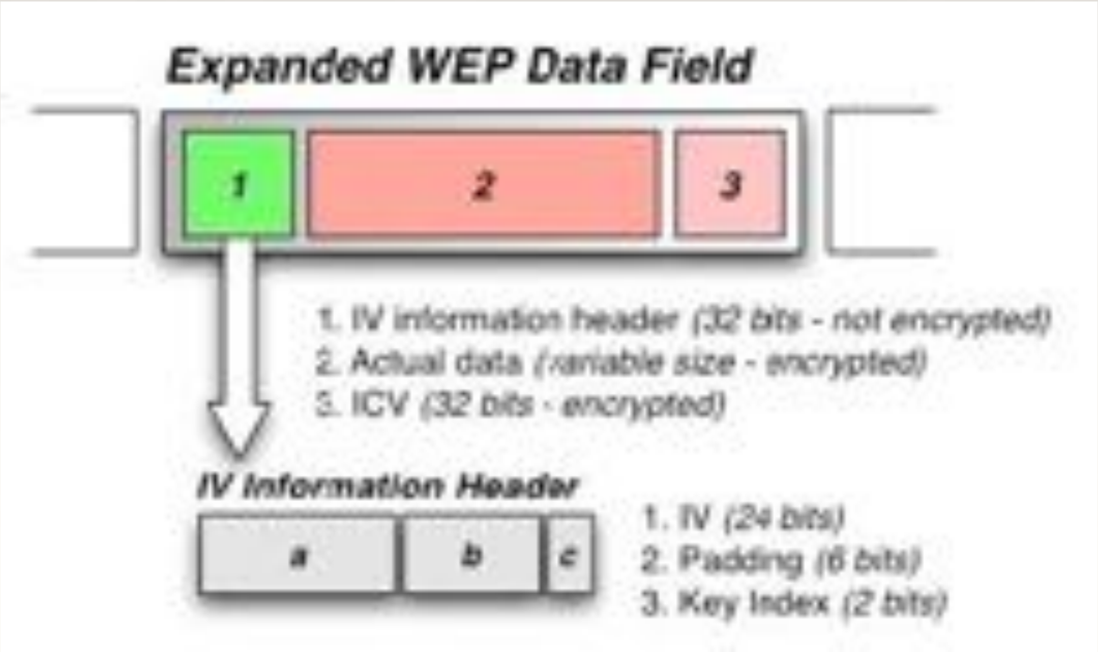
  - The 2nd byte is twice more likely to be zero than a random byte.

The Case of WEP

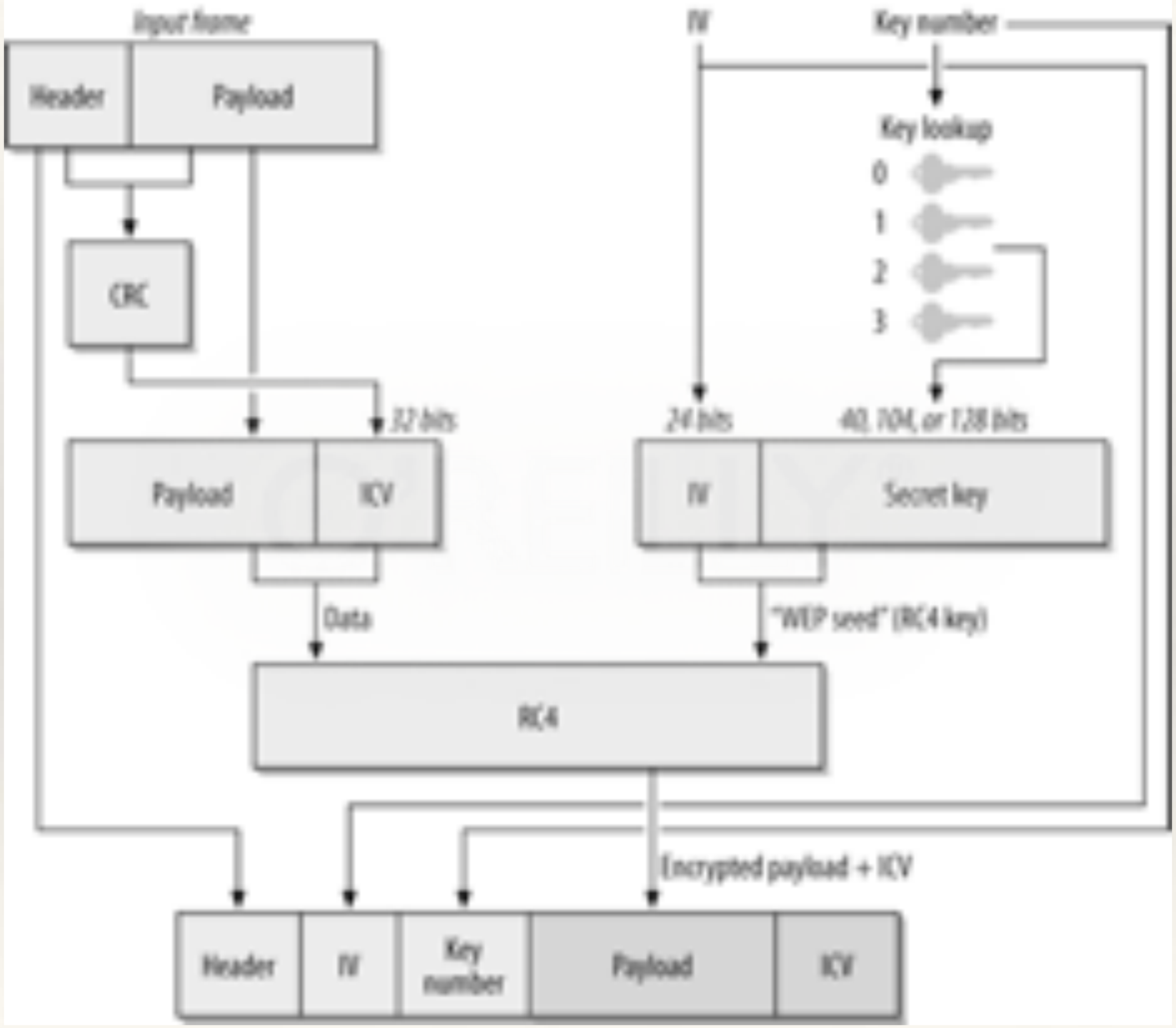
- Adopted in 1999 and replaced in 2003 by WPA. But still in (rare) use.

RC4 in WEP

- WEP uses a secret password between 40 and 104 bits, shared between the router and the users.
- For each packet, the sender selects a random public 24-bit IV: the RC4 secret key is the concatenation k = IV || password, between 64 and 128 bits (8 to 16 bytes).



RC4 in WEP



Input frame
Header
Payload
IV
Key number
Key lookup
0
1
2
3
CRC
32 bits
24 bits
40, 104, or 128 bits
Payload
ICV
IV
Secret key
Data
"WEP seed" (RC4 key)
RC4
Encrypted payload + ICV
Header
IV
Key number
Payload
ICV

Security of RC4

- It is known how to distinguish RC4 from a random stream from 2^{30} consecutive bytes.
- In the general case, the best attack is an exhaustive search over the key.
- However, the initialization is weak, especially when related keys are used. This is how WEP got broken.
- The main weaknesses disappear if sufficiently many bytes are discarded.

Attacks on RC4-WEP

- We are in the (partially) known-plaintext setting:
  - IP packets: 9 bytes are known.
  - ARP packets: 22 bytes are known.
- Thus, the first keystream bytes are known for many IVs: one can simulate the beginning of the initialization.
- Used by several attacks (implemented in **aircrack-ng**) to recover the key byte by byte, using "weak" IVs.
  - 2001: FMS, using the 1st byte.
  - 2004: Korek, using the 1st and 2nd bytes.
  - 2007: PTW, using more bytes.
  - 2008: Beck and Tews.

Practical Impact

- When the FMS attack was first implemented, it required one night in a lab.
- Today, the best attacks recover the WEP secret key after collecting only hundreds of thousands encrypted packets, rather than dozens of millions initially.
- Active attacks allow to generate such traffic very quickly in a few minutes: either by replaying well-chosen packets, or by generating new encrypted packets based on former encrypted packets.

Countermeasures

- WPA uses RC4 more securely:
  - The IV is longer
  - The secret key is hash(IV || password).
- In WPA2, RC4 encryption is replaced by a stream cipher based on a block cipher: AES in counter mode.

Attacks on TLS-RC4

- TLS is a security protocol for the Internet
- RC4 biases have been exploited to mount nearly-practical attacks on TLS and WPA-TKIP: RC4 is no longer recommended.

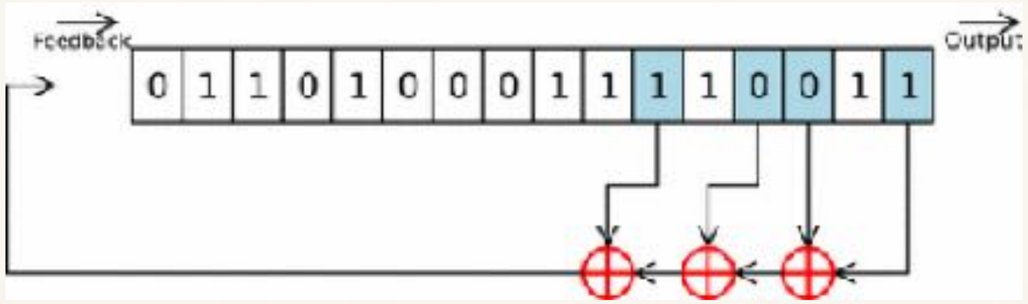
Conclusion on RC4

- Perhaps the simplest and smallest cipher to implement: many virus use it!
- Only average security.
- Slow on modern platforms: tailored for 8-bit architectures.

RC4 Reloaded

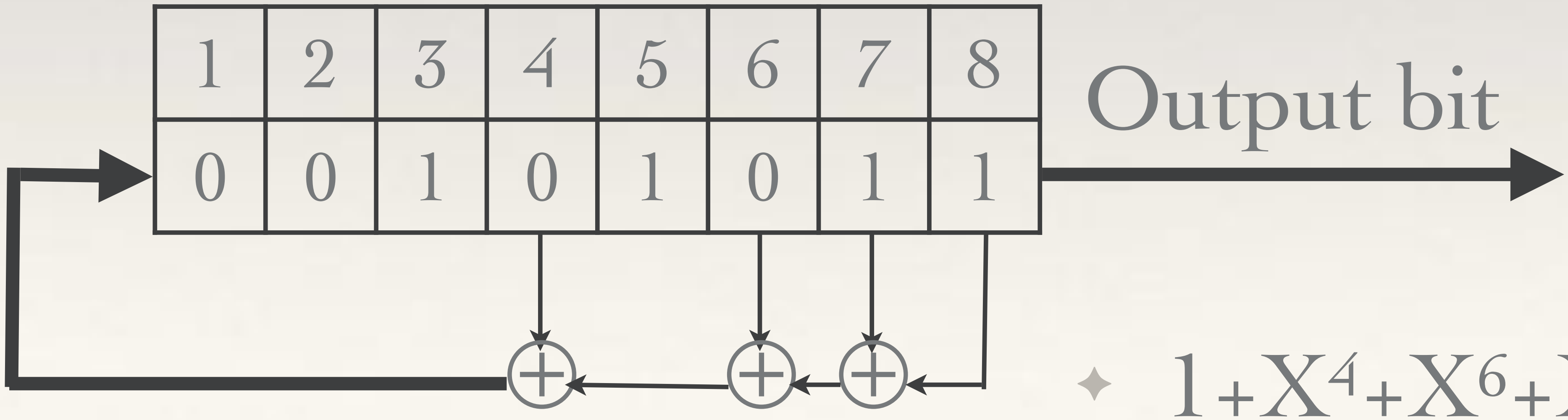
- In 2014, Rivest and Schuldt proposed Spritz: an RC4 variant targetting better security.
  - Twice as slow.

```
All arithmetic is performed modulo 256
while GeneratingOutput:
    i := i + w
    j := k + S[j + S[i]]
    k := k + i + S[j]
    swap values of S[i] and S[j]
    output z := S[j + S[i + S[z + k]]]
endwhile
```



LFSR

- **Linear Feedback Shift Register**
  - Register of n bits
  - Update by linear feedback
  - The feedback polynomial $1+\sum^{n}_{i=1} a_i X^i$ where a_i=1 or 0 if bit i belongs to the feedback.



- $1+X^4+X^6+X^7+X^8$

Security of LFSR

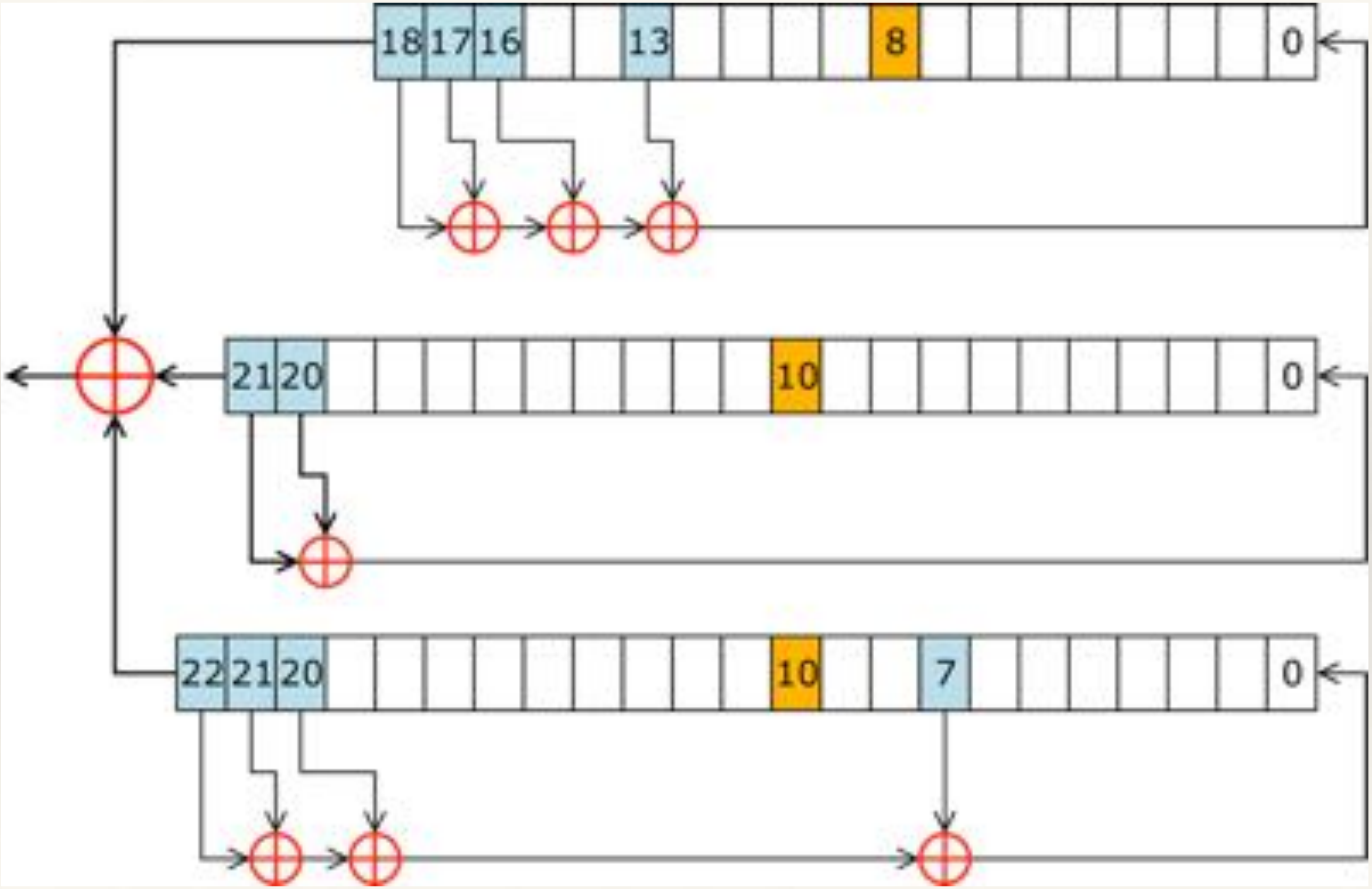
- A LFSR generates pseudo-random bits which are **not cryptographically secure**: the Berlekamp-Massey algorithm recovers the feedback and the register, given only 2n consecutive output bits.


- Hardware stream ciphers often **combine several LFSRs**:
  - The output bit is a function of the LFSR output bits, usually non-linear.
  - At each clock cycle, we may not update all LFSRs.

The GSM Standard

- Dominant standard for mobile phone in Europe.
- Each SIM card has an individual secret key shared with the operator.
- Protocol between phone and mobile station
  - Session key agreement (often 54 bits), using A3A8.
  - Symmetric encryption using A5
    - A5/0 = Identity : Iran, Syria, etc.
    - A5/1 “strong” : Europe and USA
    - A5/2 “weak” : Asia

The A5/1 Stream Cipher



The E0 Stream Cipher

- Used in Bluetooth.
- Keysize = 128 bits.
- Based on 4 LFSR.

Popular Stream Ciphers

- In software: Chacha (variant of Salsa).
- Block ciphers (like AES) in stream cipher mode.

Block Ciphers Today

- Many algorithms considered "secure", many patent-free.
- Very efficient in soft and hard.
- Theory well-understood.
- Most famous algorithms
  - DES (1976): former US standard.
  - IDEA (1992) by Lai-Massey, used in PGP.
  - RC5 (1994) by Rivest.
  - AES (2000): current US standard by Daemen-Rijmen.
  - KASUMI (2002) used in UMTS.

Differences with Stream Ciphers

- Block ciphers are more powerful than stream ciphers, which can only provide symmetric encryption.

- Block ciphers can provide:
  - symmetric encryption: it can even simulate a stream cipher with certain modes of operation.
  - integrity: hash function
  - authentication: message authentication code.

Principles of Block Ciphers

- A block cipher is a family of permutations, indexed by a secret key.
- For any key k, it defines a permutation over a block. Ex: for any 56-bit key k, DES_k is a permutation over 64 bits.
- Alone, a block cipher does not encrypt: we need a padding and a mode of operation.
- It is recommended to additionally use a MAC to authenticate ciphertexts.

Security of Block Ciphers

- We hope that the permutations defined by a block cipher are pseudo-random permutations: an attacker should not be able to distinguish a random permutation from the block cipher with a random key.

Pseudo-Random Permutation

- DES is a **pseudo-random permutation** if you cannot distinguish the two following blackboxes in less than 2^{56} operations:
  - A box implementing DES_k where k is a fixed 56-bit random key.
  - A box implementing a random permutation, chosen uniformly from all $(2^{64})!$ permutations.

Caesar is not Pseudo-Random

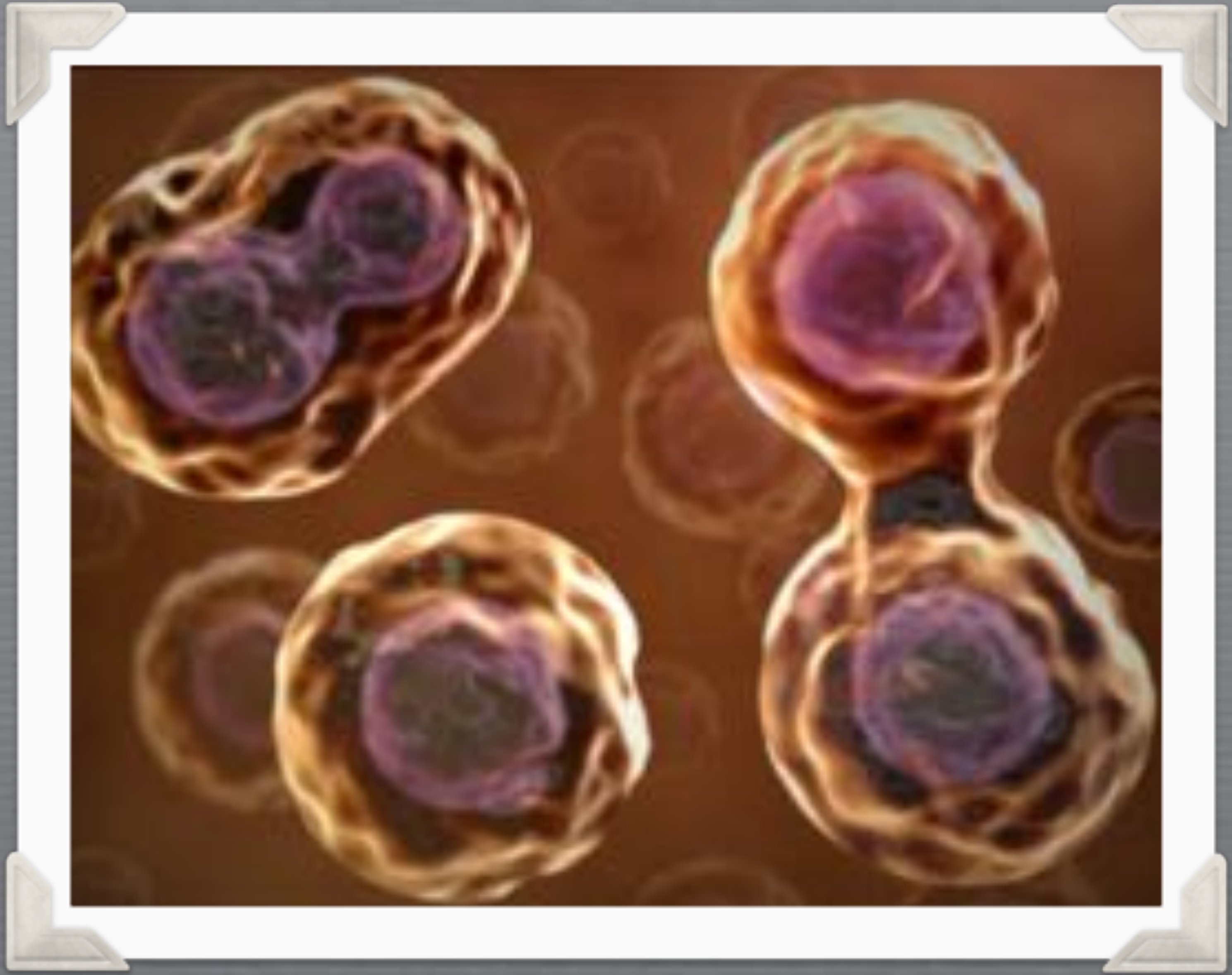
- Show that a shift cipher can be distinguished from a random permutation in much less than 26 operations.
- A substitution cipher is pseudo-random by definition, but the way it was used was not secure.

Paddings

- A padding makes the message length divisible by the blocksize.
- The exact padding rule is specified by the standard.
- Usual rules:
  - Append bit 1, then as many 0s as necessary.
  - Add a block which encodes the bitlength.

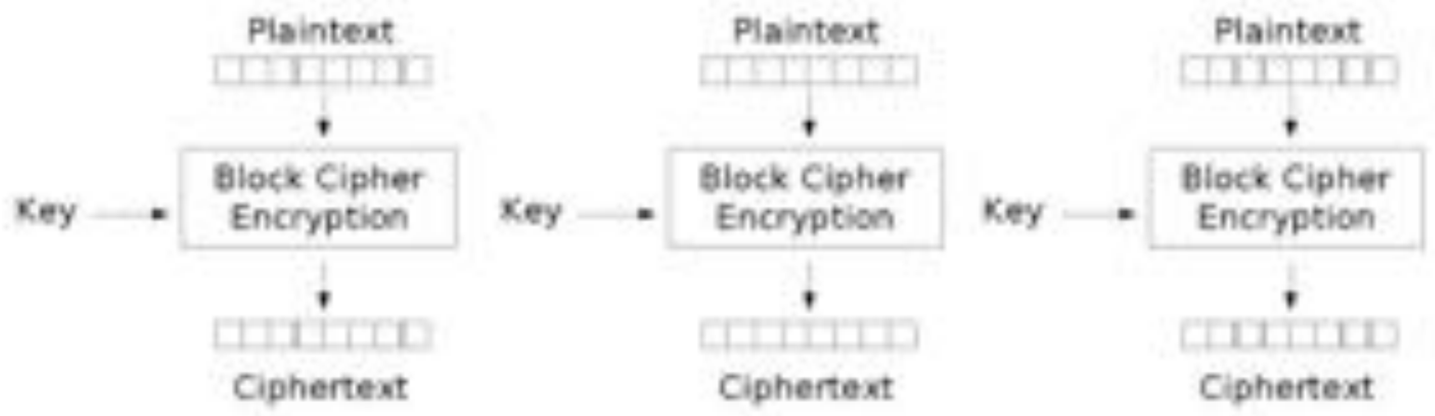
Classical Modes of Operation

- A mode of operation specifies how to encrypt arbitrary messages of length divisible by the blocksize.
- The DES appeared with 5 modes of operation:
  - 4 randomized modes, including 3 stream modes.
  - 1 deterministic mode.

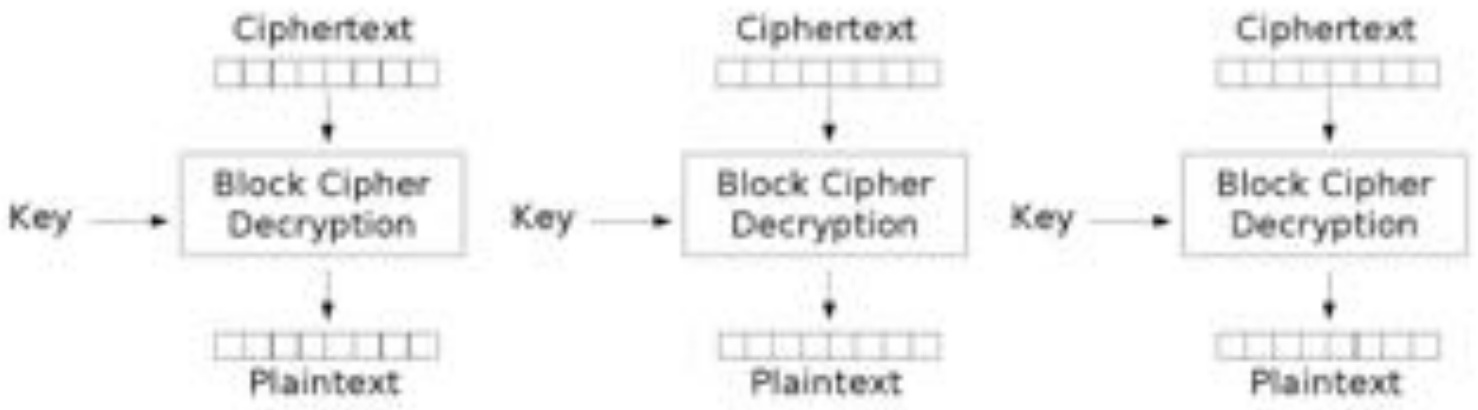


Lesson 11: Divide and Conquer

ECB



Electronic Codebook (ECB) mode encryption

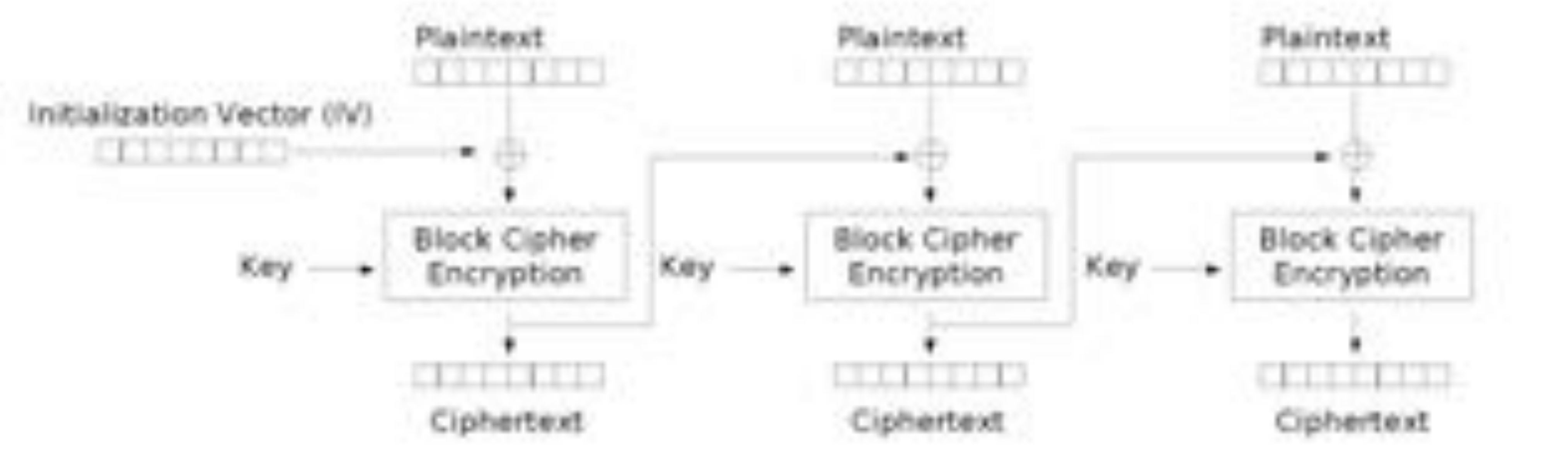


Electronic Codebook (ECB) mode decryption

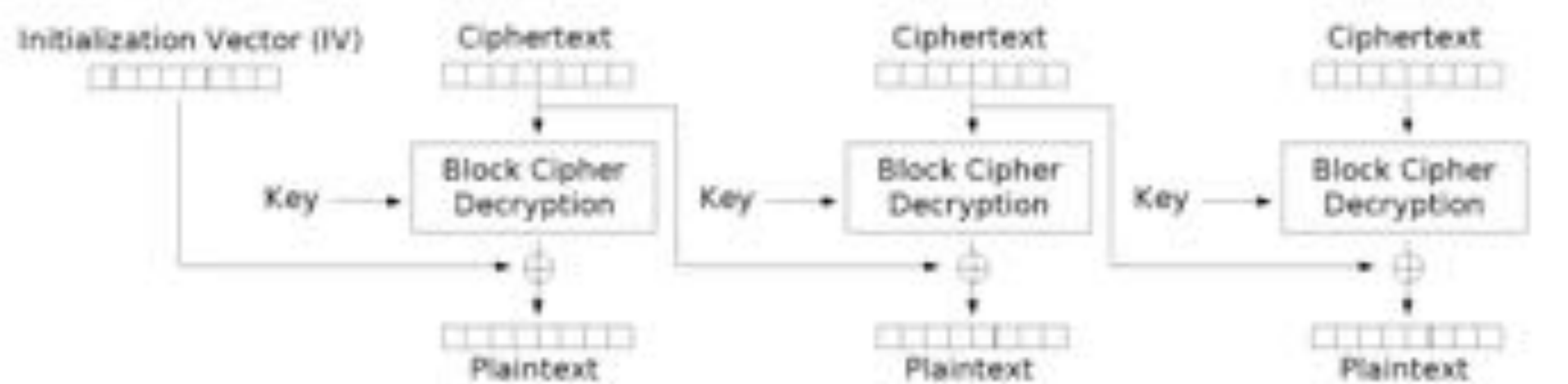
Security of ECB

- ECB is the only deterministic mode: it cannot be very secure.
- We saw that a substitution cipher (over the alphabet) in ECB mode is insecure.

CBC

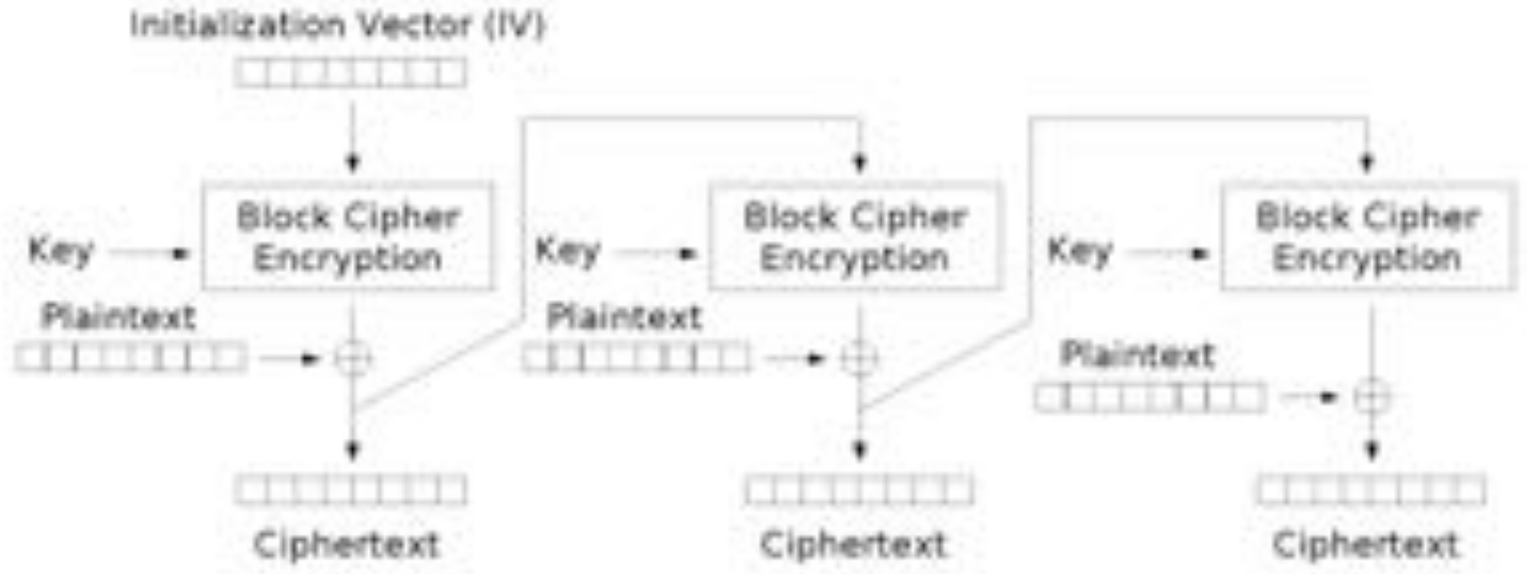


Cipher Block Chaining (CBC) mode encryption

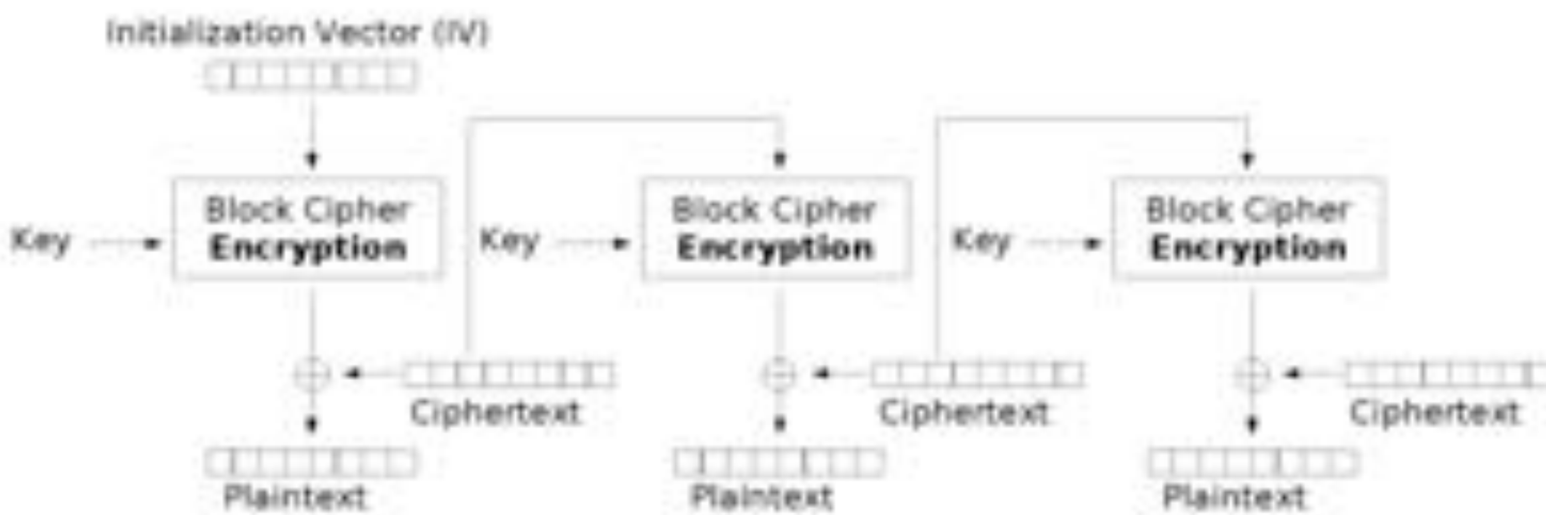


Cipher Block Chaining (CBC) mode decryption

CFB

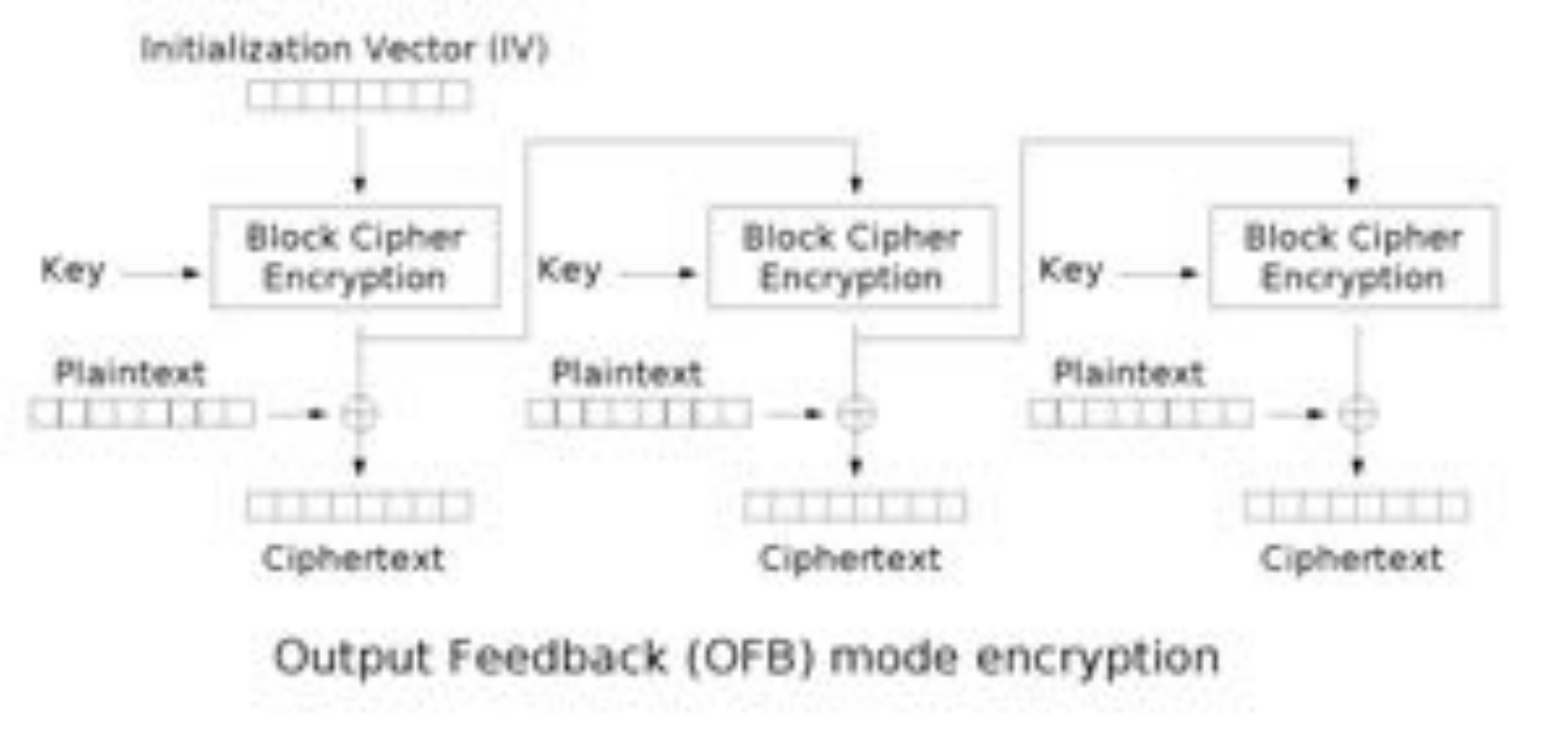


Cipher Feedback (CFB) mode encryption

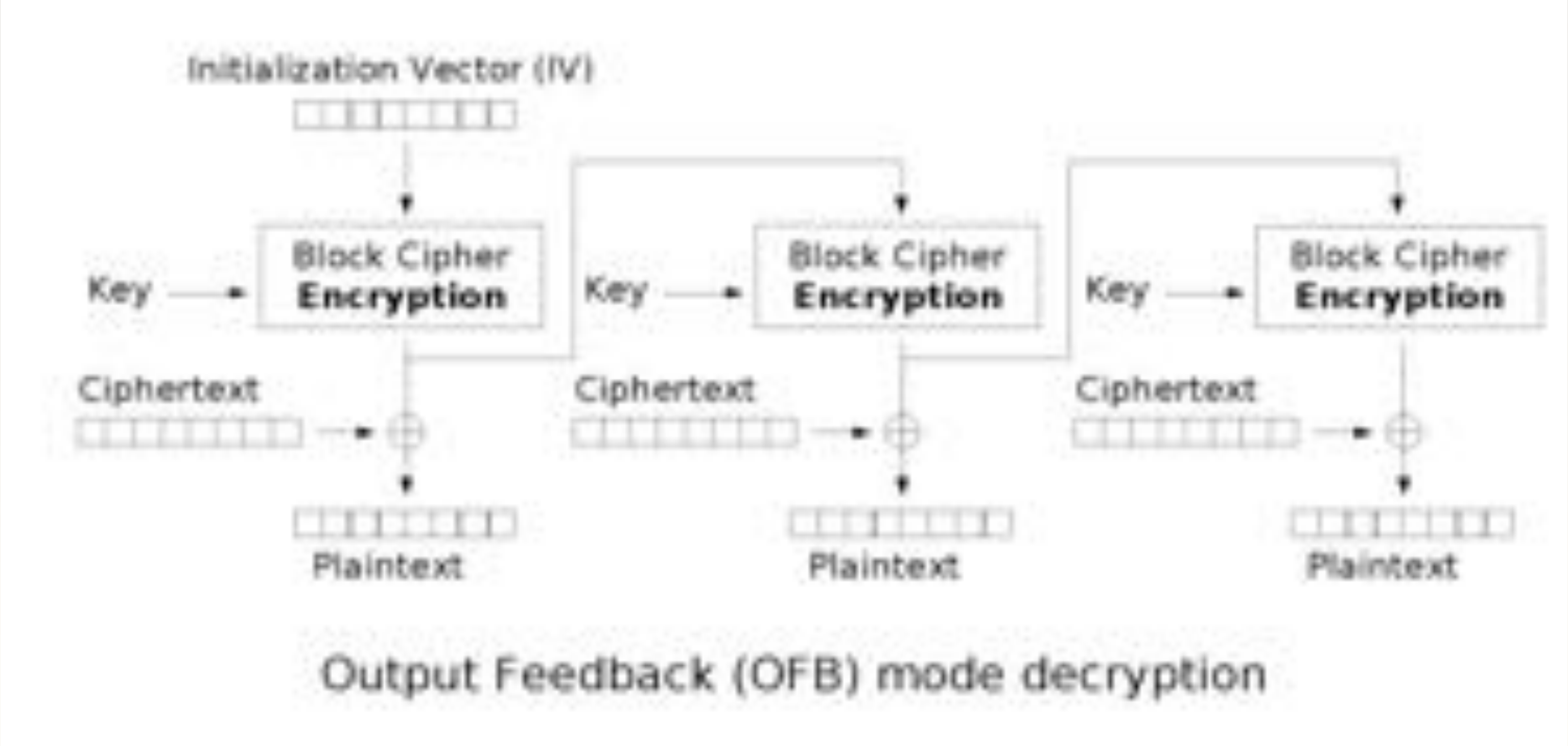


Cipher Feedback (CFB) mode decryption

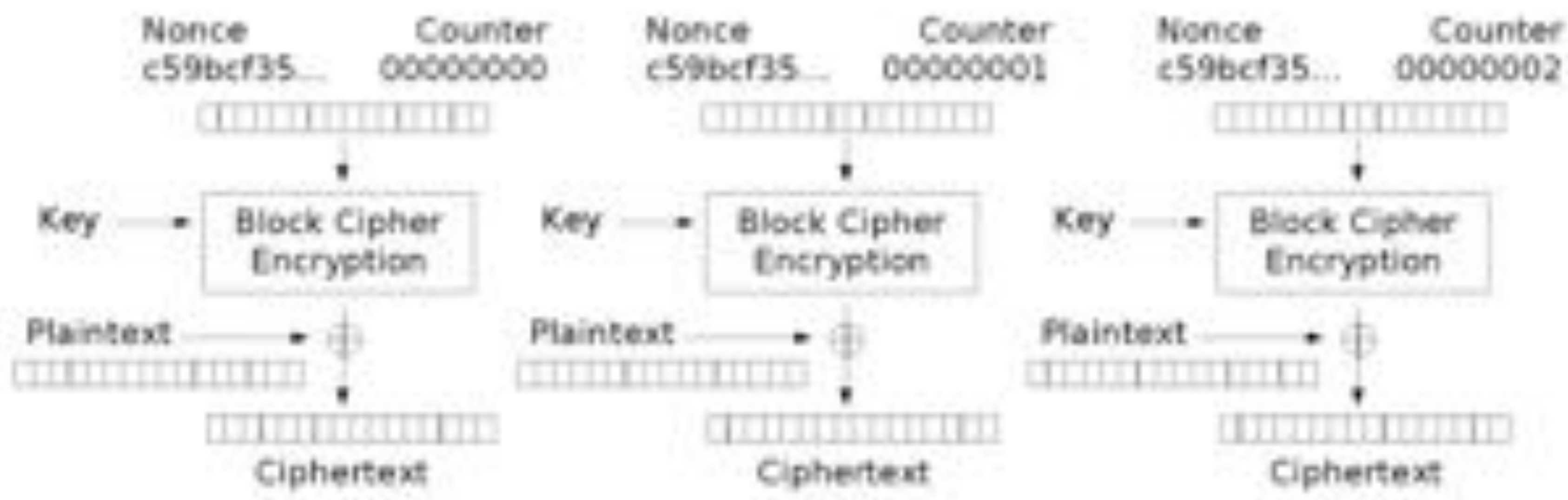
OFB



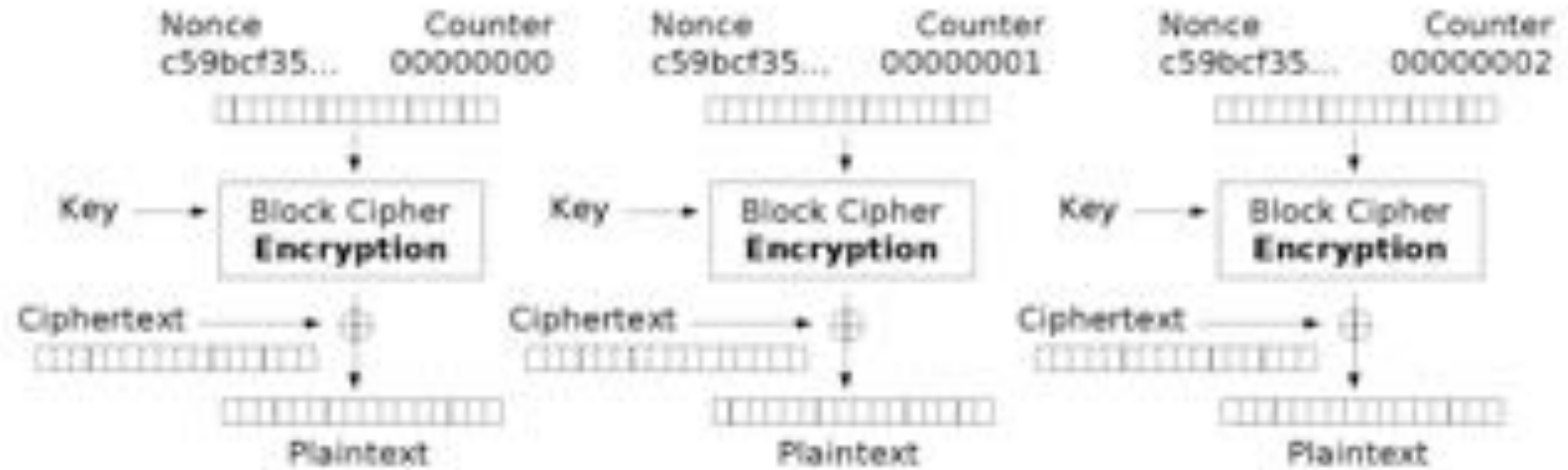
Used in UMTS with the block cipher KASUMI



CTR



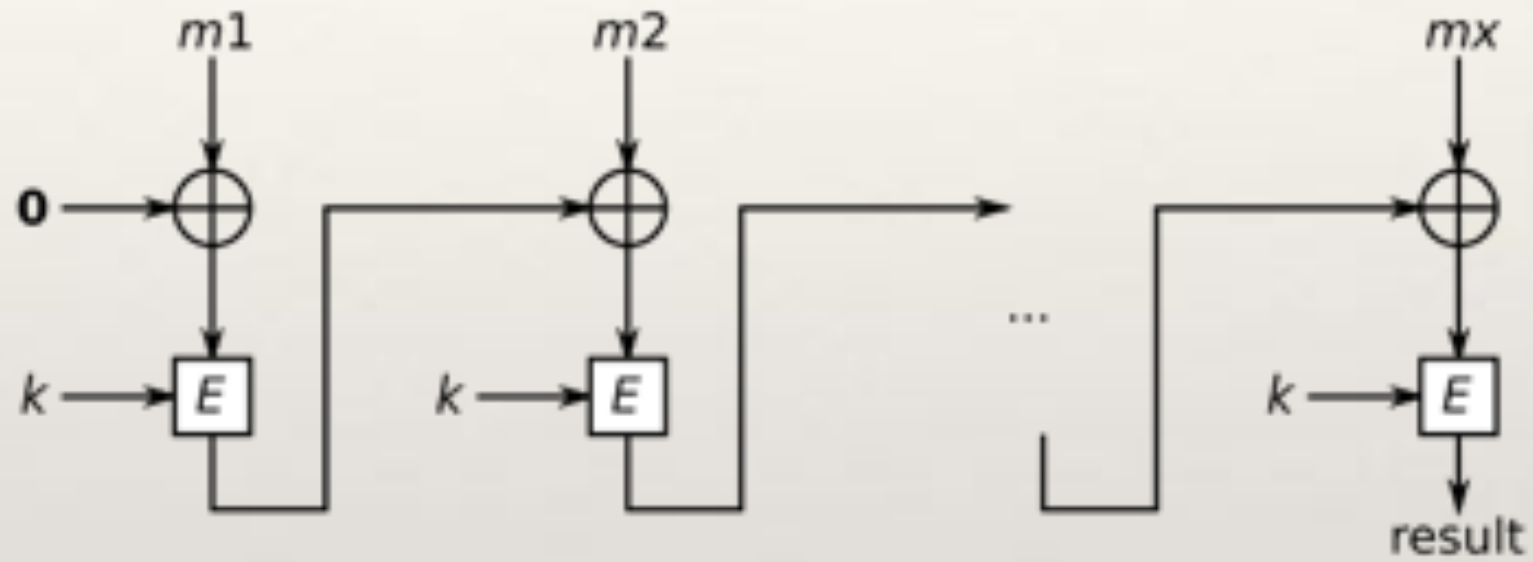
Counter (CTR) mode encryption



Counter (CTR) mode decryption

Authentication Modes

- CBC-MAC



- PMAC and OMAC are recent and have better security properties.

New Modes

- OCB achieves both encryption and authentication at the same time, without much additional cost compared to single encryption.
- Ex: CCM, GCM.

Summary of Modes

- 1 deterministic encryption mode: ECB.
- 4 randomized encryption modes:
  - CBC: not //, requiring to implement the inverse permutation.
  - 3 stream modes: CFB, OFB and CTR. Only CTR is //.
- authentication modes: CBC-MAC, OMAC, PMAC.
- dual modes: encryption + authentication without any overhead.

Lesson 12: Security by iteration

Iterative Encryption

- Modern block ciphers iterate a simple cipher many times: at each iteration, we apply the same simple cipher with a different key, the subkey.
  - The key-schedule algorithm specifies how to generate all the subkeys from the secret key.
  - An iteration is called a round.
- What differs is the philosophy of the simple cipher. There are two big strategies: DES and AES.

Examples

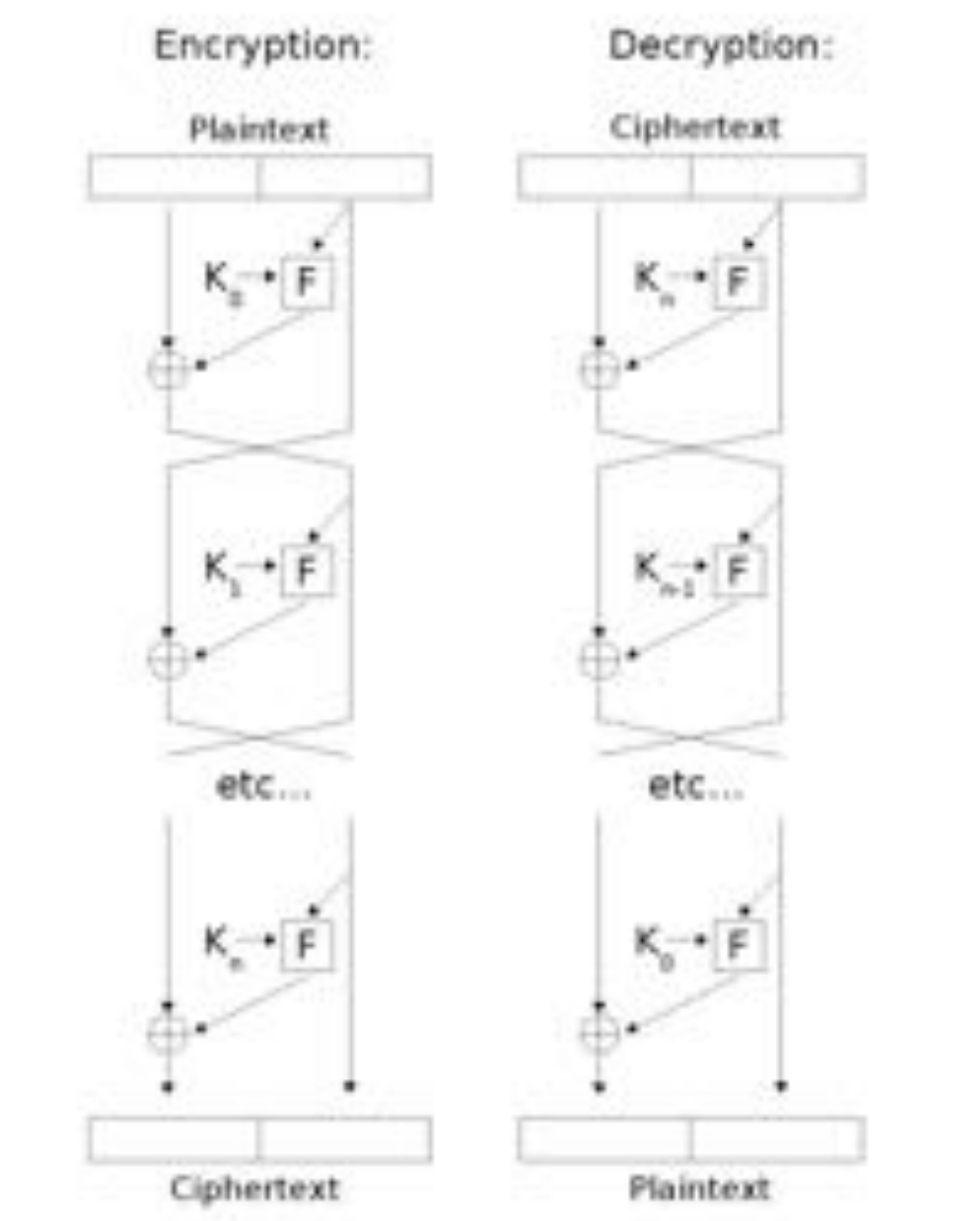
- DES has 16 rounds, and each subkey has 48 bits.
- AES has 10, 12 or 14 rounds, depending on the keylength (128, 192 or 256).
- The more rounds there are, the more secure and the less efficient.

Design of a Round

- Two Families:
  - Feistel Networks like the DES.
  - Substitution Permutation Networks (SPN) like the AES.

Feistel Networks

- Invented by **Feistel** in 1973.
- At round i:
  - The block is $L_i \| R_i$.
  - The subkey is K_i.
- Then $L_{i+1} = R_i$
  and $R_{i+1} = L_i \oplus F(R_i, K_i)$
- F is an **arbitrary function**.



Feistel Cipher

Security of Feistel Networks

- [LubyRackoff1988]: if the F-function is a pseudo-random function, then 3-round Feistel is a pseudo-random permutation, and 4-round Feistel is a strong pseudo-random permutation.
- [HKT2010]: if the F-function is a random oracle, then 10-round Feistel is an ideal cipher.
- Meaning: if the F-function is a good « random » function, then Feistel with sufficiently many rounds will be a good permutation.

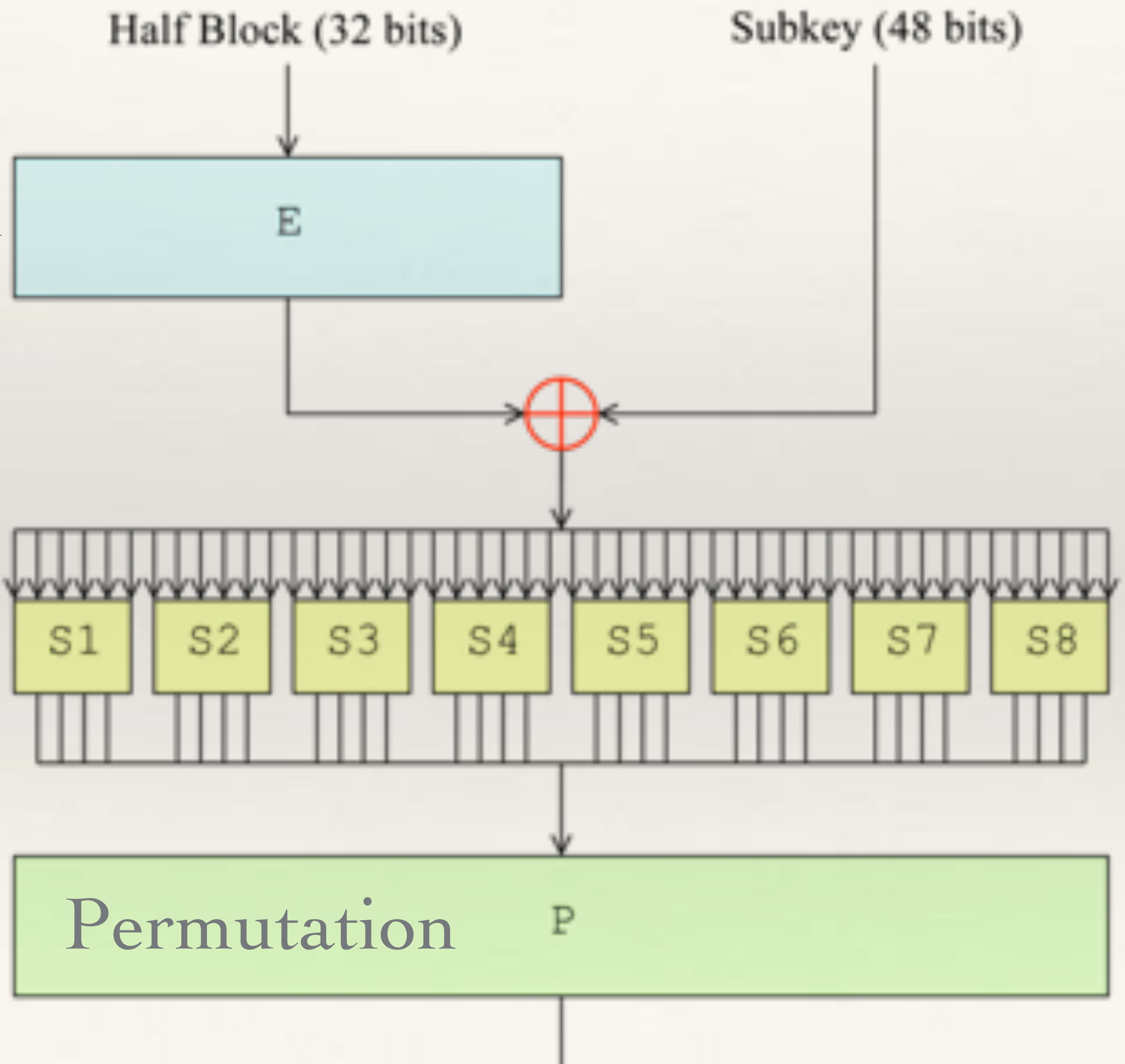
The Case of DES

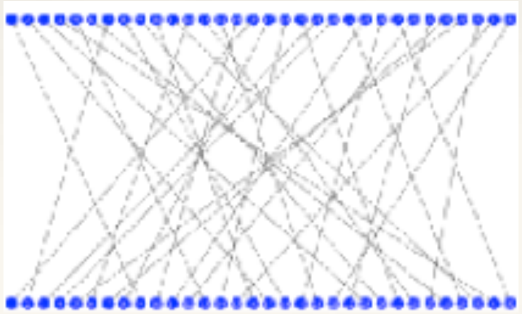
- Characteristics:
  - 64-bit block
  - 16 rounds
  - 56-bit key
  - 48-bit subkeys
- Chosen as a US standard in 1977.

DES’ F-function

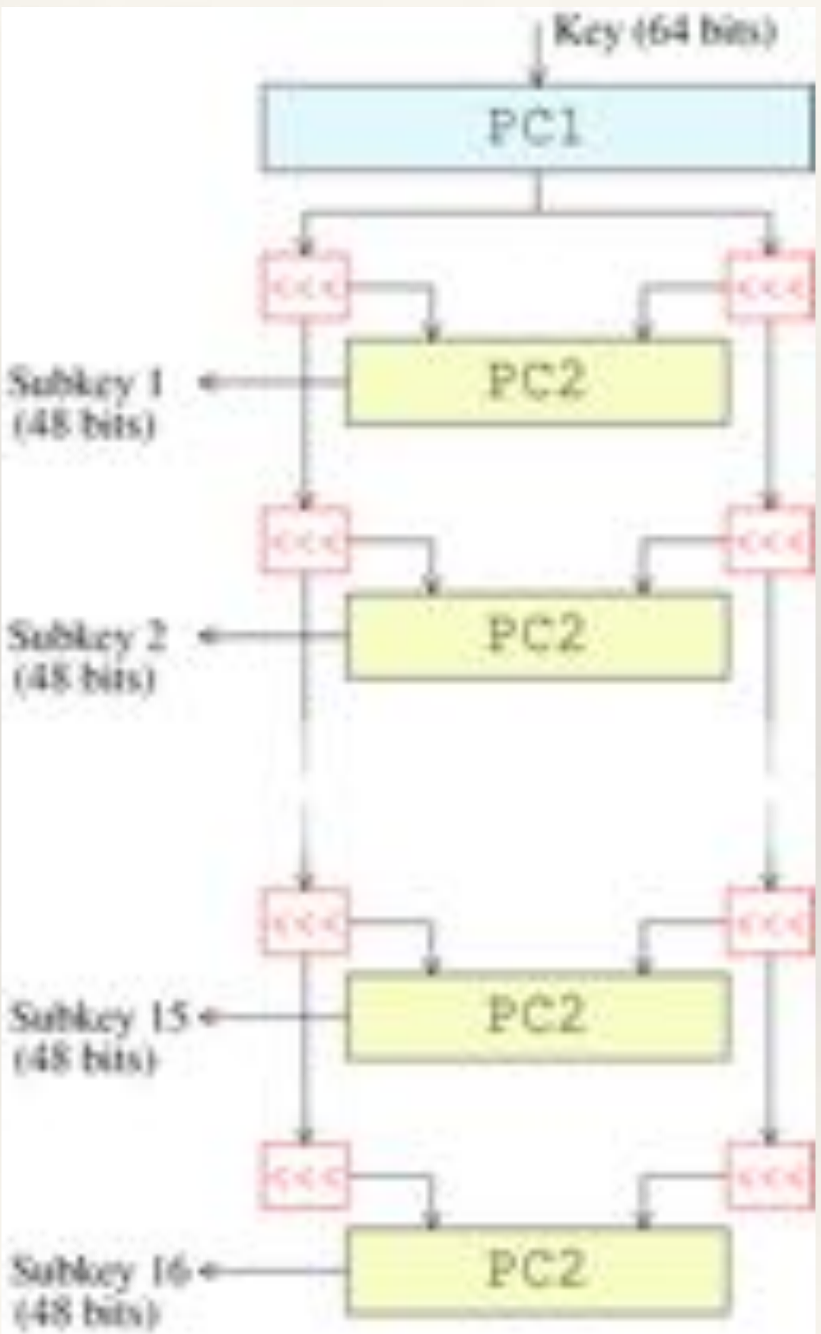
Expansion

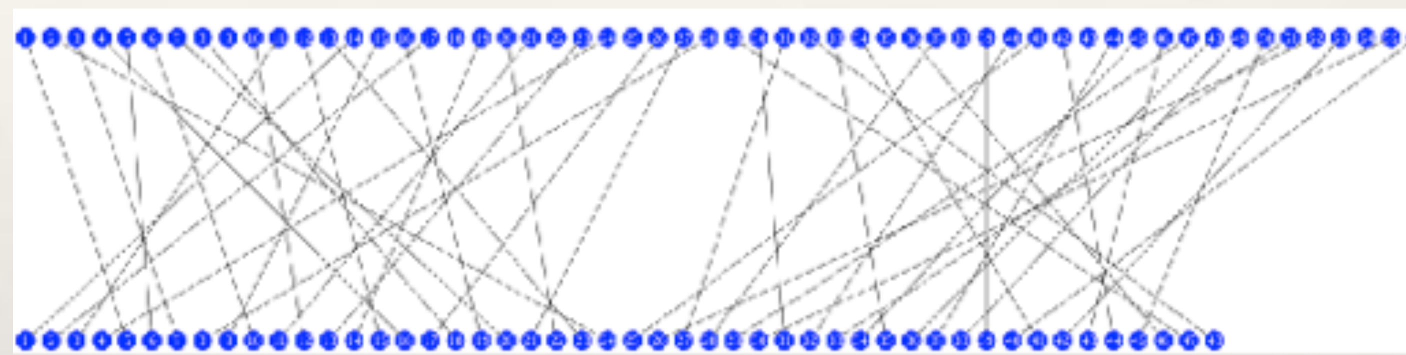
S-boxes





DES' key schedule





DES' Modifications

- Keysize
  - Lucifer had a 64-bit secret key.
  - The NSA argued for a 48-bit secret key.
  - 56-bit was a compromise.
- S-boxes
  - The NSA modified IBM's original S-boxes.

DES in Software

```
// Key schedule tables

const int PC1[56] = {
  57, 49, 41, 33, 25, 17,  9,
   1, 58, 50, 42, 34, 26, 18,
  10,  2, 59, 51, 43, 35, 27,
  19, 11,  3, 60, 52, 44, 36,
  63, 55, 47, 39, 31, 23, 15,
   7, 62, 54, 46, 38, 30, 22,
  14,  6, 61, 53, 45, 37, 29,
  21, 13,  5, 28, 20, 12,  4
};
const int Rotations[16] = {
   1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1
};
const int PC2[48] = {
  14, 17, 11, 24,  1,  5,
   3, 28, 15,  6, 21, 10,
  23, 19, 12,  4, 26,  8,
  16,  7, 27, 20, 13,  2,
  41, 52, 31, 37, 47, 55,
  30, 40, 51, 45, 33, 48,
  44, 49, 39, 56, 34, 53,
  46, 42, 50, 36, 29, 32
};

// Permutation tables

const int InitialPermutation[64] = {
  58, 50, 42, 34, 26, 18, 10,  2,
  60, 52, 44, 36, 28, 20, 12,  4,
  62, 54, 46, 38, 30, 22, 14,  6,
  64, 56, 48, 40, 32, 24, 16,  8,
  57, 49, 41, 33, 25, 17,  9,  1,
  59, 51, 43, 35, 27, 19, 11,  3,
  61, 53, 45, 37, 29, 21, 13,  5,
  63, 55, 47, 39, 31, 23, 15,  7
};
const int FinalPermutation[64] = {
  40,  8, 48, 16, 56, 24, 64, 32,
  39,  7, 47, 15, 55, 23, 63, 31,
  38,  6, 46, 14, 54, 22, 62, 30,
  37,  5, 45, 13, 53, 21, 61, 29,
  36,  4, 44, 12, 52, 20, 60, 28,
  35,  3, 43, 11, 51, 19, 59, 27,
  34,  2, 42, 10, 50, 18, 58, 26,
  33,  1, 41,  9, 49, 17, 57, 25
};

// Rounds tables

const int DesExpansion[48] = {
  32,  1,  2,  3,  4,  5,  4,  5,
   6,  7,  8,  9,  8,  9, 10, 11,
```

DES in Software

```
void addbit(uint64_t *block, uint64_t from,
            int position_from, int position_to)
{
   if(((from << (position_from)) & FIRSTBIT) != 0)
      *block += (FIRSTBIT >> position_to);
}
void Permutation(uint64_t* data, bool initial)
{
   uint64_t data_temp = 0;
   for(int ii = 0; ii < 64; ii++)
   {
      if(initial)
         addbit(&data_temp, *data, InitialPermutation[ii] - 1, ii);
      else
         addbit(&data_temp, *data, FinalPermutation[ii] - 1, ii);
   }
   *data = data_temp;
}
```

```
bool key_parity_verify(uint64_t key)
{
   int parity_bit = 0; // Parity helper

   for(int ii = 0; ii < 64; ii++)
   {
      // Test the parity bit (8-th bit)
      if(ii % 8 == 7)
      {
         if(parity_bit == 0)
         {
            // Test if 8-th bit != 0
            if( ((key << ii) & FIRSTBIT) != (uint64_t)0)
            {
               printf("parity error at bit #%i\n", ii + 1);
               return false;
            }
         }
```

DES in Software

```
void key_schedule(uint64_t* key, uint64_t* next_key, int round)
{
   // Init
   uint64_t key_left = 0;
   uint64_t key_right = 0;

   uint64_t key_left_temp = 0;
   uint64_t key_right_temp = 0;

   *next_key = 0; // Important !

   // 1. First round => PC-1 : Permuted Choice 1
   if(round == 0)
   {
      for(int ii = 0; ii < 56; ii++)
      {
         if(ii < 28)
            addbit(&key_left, *key, PC1[ii] - 1, ii);
```

```
void rounds(uint64_t *data, uint64_t key)
{
   uint64_t right_block = 0;
   uint64_t right_block_temp = 0;

   // 1. Block expansion
   for(int ii = 0; ii < 48; ii++)
      addbit(&right_block, *data, (DesExpansion[ii] + 31), ii);

   // 2. Xor with the key
   right_block = right_block ^ key;

   // 3. Substitution
   int coordx, coordy;
   uint64_t substitued;

   for(int ii = 0; ii < 8; ii++)
   {
      coordx = ((right_block << 6 * ii) & FIRSTBIT) == FIRSTBIT ? 2 : 0;
      if( ((right_block << (6 * ii + 5)) & FIRSTBIT) == FIRSTBIT)
```

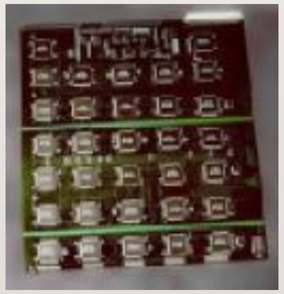
Security

- In practice, the best attack against DES remains a 56-bit exhaustive search, which can be done today at low-cost in hardware or software.
- There are several "theoretical" attacks that require much less than 2^{56} operations, but significant data.
  - Differential cryptanalysis
  - Linear cryptanalysis
- The choice of the S-boxes increase the resistance of DES to these attacks.

Exhaustive Search on DES

- 1970s: a supercomputer could perform about 10^8 flops = 2^{58} operations every 100 years.

- 1997: 96 days on the Internet with up to 78,000 PCs; 200-MHz CPU = 10^6 keys/sec.



- 1998: 2 days on EFF's $250,000 machine.
- 2006: 1 week on Germany's $10,000 FPGA.

- 2011's quadcore has 2200 times more cycles/s than 1981's IBM PC.

Faster Attacks on DES


- 1992: Biham-Shamir's differential cryptanalysis, 2^{47} chosen plaintexts (= 1 petabyte), similar time.

- 1993: Matsui's linear cryptanalysis. The best variant only requires 2^{41} cleartext/ciphertext pairs (= 16 Tb) and has been implemented.

Differential Cryptanalysis

- “Discovered” by Biham and Shamir in 1990.
- Known by IBM in 1974, and already known by NSA, but kept secret: DES was designed to resist differential cryptanalysis.
- It’s a chosen-plaintext attack:
  - One selects a well-chosen pattern Δ.
  - Submit plaintexts (m,m’) s.t. $m \oplus m' = \Delta$.
  - Guess subkey bits by observing $c \oplus c'$.

Linear Cryptanalysis

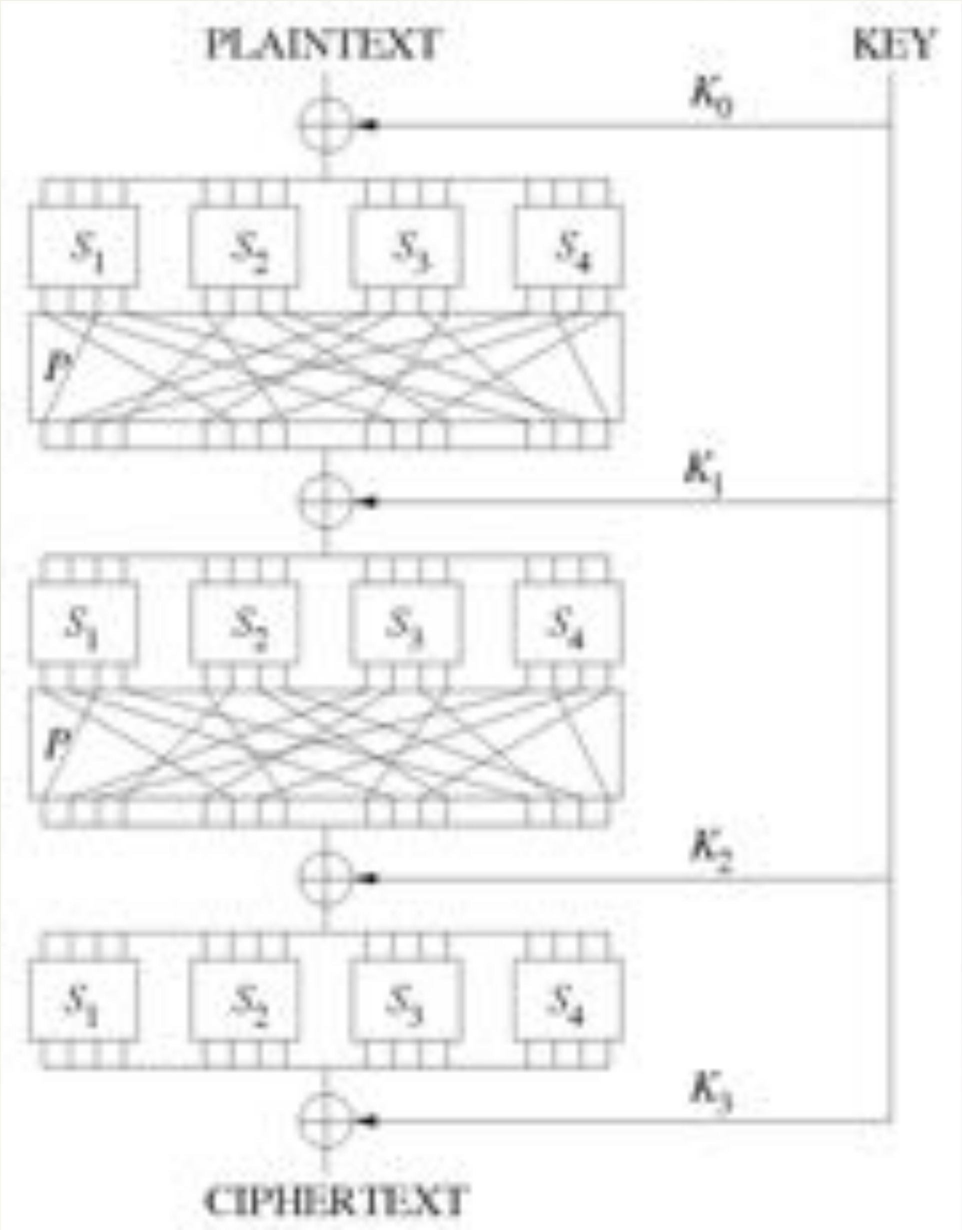
- It's a known-plaintext attack.
- Consider linear equations between bits of subkeys, cleartexts and ciphertexts; and try to recover subkey bits.
  - Easy if the equations always hold.
  - Since encryption is not linear, the equations only hold with probability close to 1/2.

SPN

- Substitution Permutation Network

Substitution

Permutation



The Case of AES

- It's an SPN.
- Number of rounds: 10, 12 or 14.
- Keysize: 128, 192 or 256 bits.
- Blocksize: 128 bits viewed as a 4x4 matrix of bytes.

Finite Fields

- Who knows what is a finite field?
- Why is there a finite field with 256 elements?

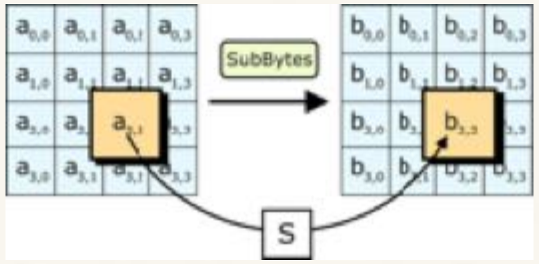
Finite Fields

- For any prime power $q=p^n$, there exists a "unique" finite field with exactly q elements, called GF(q).
  - Two operations + and x with neutral elements 0 and 1.
  - Every element y has an opposite -y.
  - Every nonzero element y has an inverse 1/y.
- AES uses the finite field $GF(256)=GF(2^8)$.

Implementing GF(256)

- Each element is represented by 8 bits.
- Addition = bitwise XOR.
- Multiplication
  - View the 8 bits as a bit-polynomial of degree ≤ 7.
  - Perform a multiplication of polynomials
  - And reduce modulo a special bit-polynomial of degree 7.

The AES Substitution



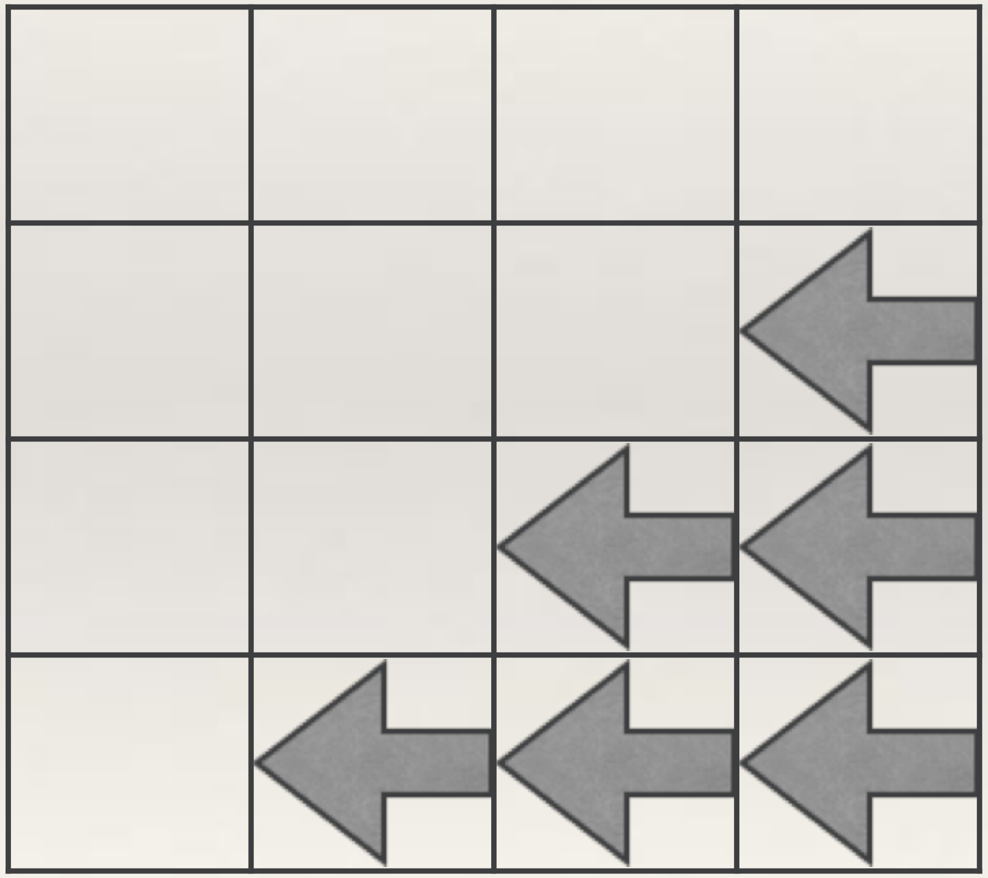
- Apply the same substitution to each block byte.
- This substitution is the **only non-linear operation** in AES.
- This substitution is essentially the inversion in GF(256), typically implemented by table-lookup, which is known to lead to cache-timing attacks.

The AES Permutation

- ShiftRows
- MixColumns

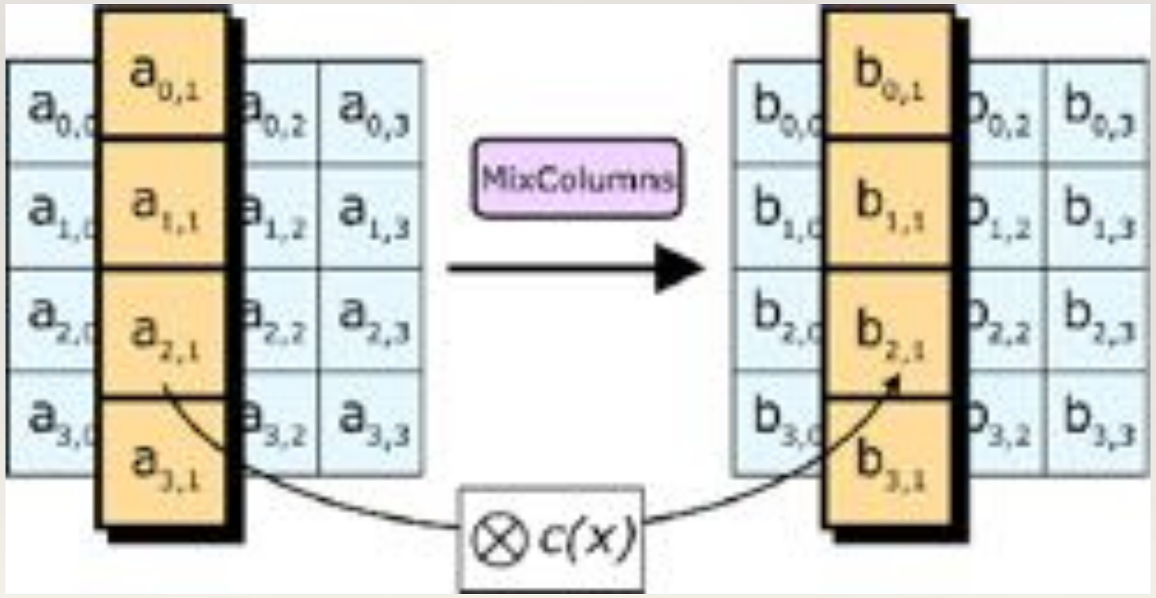
AES Shift Rows

- Each row (except the 1st) is circularly shifted.



AES MixColumns

- Each column is viewed as a GF(256)-polynomial of degree ≤ 3, then multiplied by a special polynomial, modulo $1+x^4$.



Benchmarks

- On 500-MHz Pentium:
  - DES: 500,000 keys/s and encrypts 10Mb/s.
  - AES: 800,000 keys/s and encrypts 30Mb/s.
- On 8-bit smartcard:
  - DES: RAM 13+8bytes and encrypts 450 cycles/byte.
  - AES: RAM 33+16bytes and encrypts 200 cycles/byte
- On FPGA: DES=10Gbit/s and AES=5Gbit/s
- AES is faster and more secure than DES in software, especially with new Intel processors.

AES Inside

- Since 2010, AES has been available directly in Intel processors.
  - Decrease side-channel attacks like cache attacks.
  - Speed-up by a factor 8.

Security of AES

- AES and DES are the most studied block ciphers.
- No serious attack has ever been found but:
  - There are efficient side-channel attacks against basic implementations of AES. This has led Intel to develop AES instructions for their processors, available since 2010.
  - In 2009, researchers found the first related-key attacks on AES (slightly) faster than exhaustive search. Controversial.

Future of Block Ciphers

- Keep studying the most important algorithms:
  - AES
  - KASUMI used in UMTS.
- Propose block ciphers for extreme conditions (RFID or multiparty computation/homomorphic encryption)
- The most important problems are considered solved.