UNIVERSITY OF AMSTERDAM

Jaap van Ginkel

Security of Systems and Networks

9 September, 2024 Part 3 Modern Crypto Symmetric

Projects

- **Make groups today**
  - Groups of 3 exceptions ask Jaap
  - Normally within class room/ same TA
- **Brainstorm with your group**
  - Brainstorm some ideas
  - Diverge first: wild ideas personal fascination
  - Converge to few feasible ideas
- **Brainstorm with TA/Teacher**
  - Slots on whiteboard this afternoon 13:00-16:00

Projects

- Some examples
- Choose your project
  - Decide soon
  - Don´t switch last moment
  - Do proper related research
- Hand in project proposal
  - Intro/RQ/Experiments/Ethical paragraph/requirementsrelated research
  - 1 or 2 pages

University of Amsterdam

Pro

Taxonomy of Cryptography

- **Symmetric Key**
  - Same key for encryption and decryption
  - Two types: Stream ciphers, Block ciphers
- **Public Key** (or asymmetric crypto)
  - Two keys, one for encryption (public), and one for decryption (private)
  - And digital signatures — nothing comparable in symmetric key crypto
- **Hash algorithms**
  - Can be viewed as “one way” crypto

UNIVERSITY OF AMSTERDAM

Symmetric Encryption

Symmetric Encryption

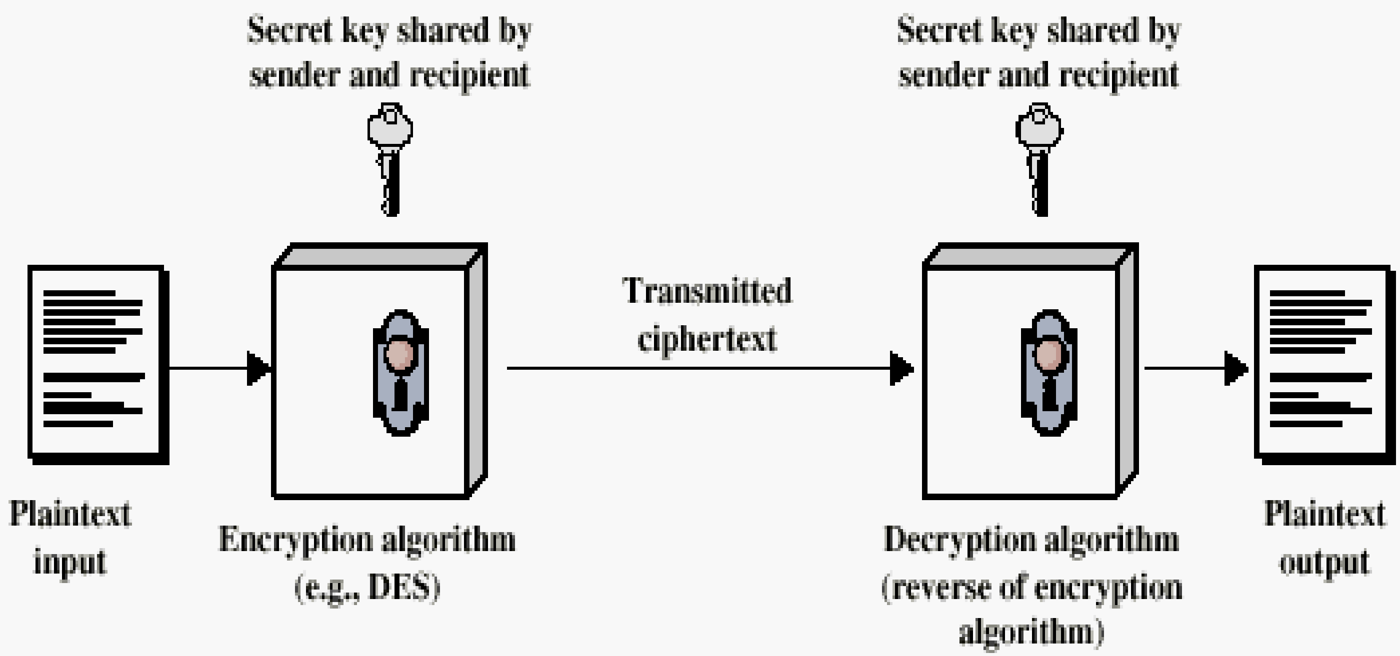
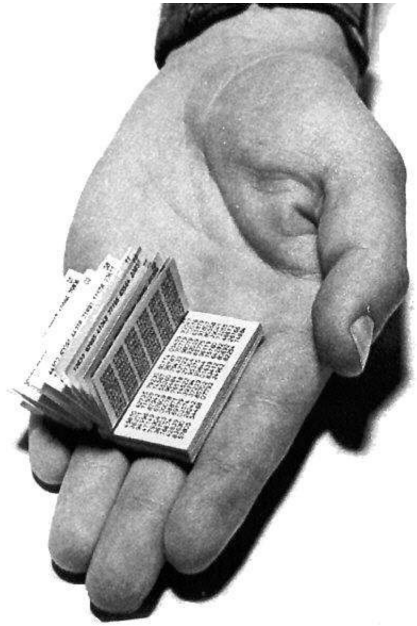


Figure 2.1 Simplified Model of Conventional Encryption

One Time Pad OTP



- Proven Secure by Shannon
- Implemented in the Vernam Cipher
- XOR data stream with pad
- Truly random pad data needed
- Hardware noise sources

Secret Key Encryption

- Symmetric Encryption
- DES (Triple DES)
- IDEA
- AES (Rijndael)
- RC6
- Blowfish

Key Distribution

- Expensive
- Vulnerable
- Difficult to scale

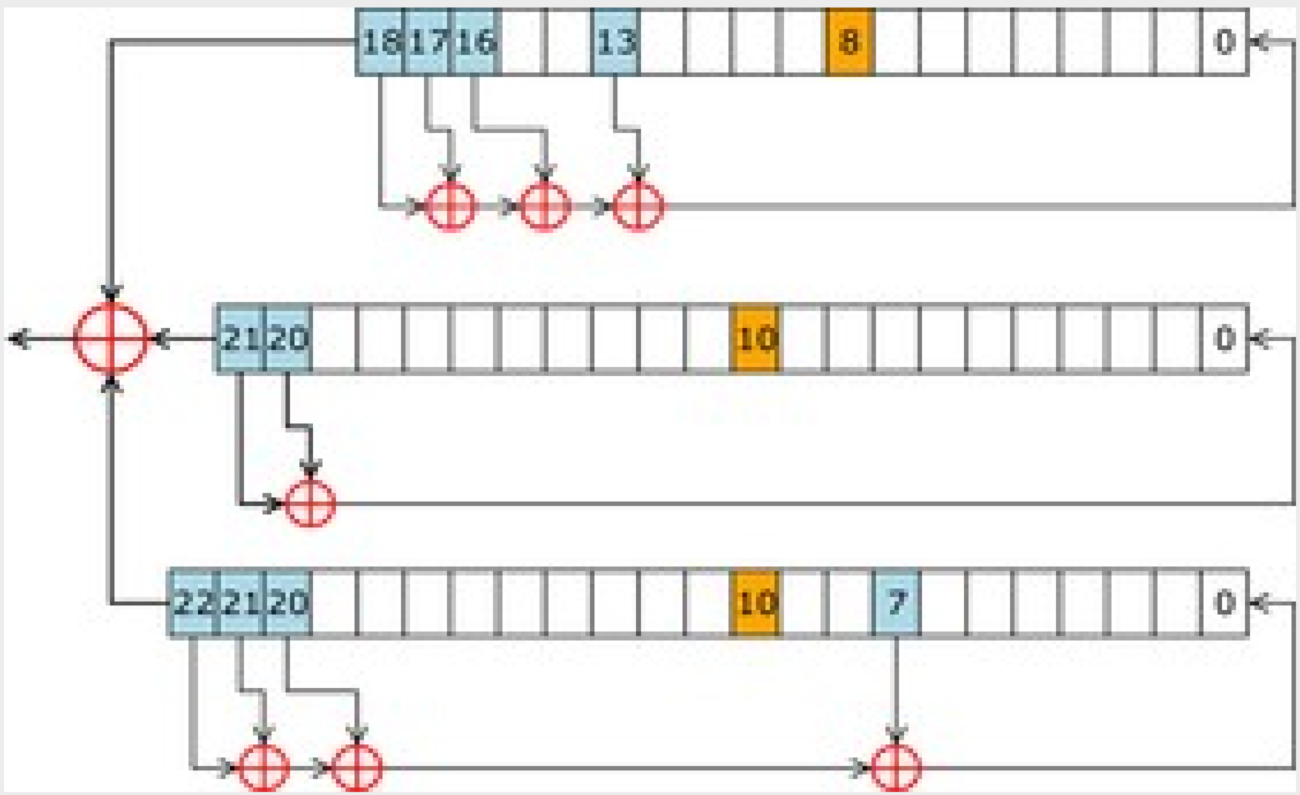
Symmetric Key Crypto

- Stream cipher — based on one-time pad
  - Except that key is relatively short
  - Key is stretched into a long **keystream**
  - Keystream is used just like a one-time pad
- Block cipher — based on codebook concept
  - Block cipher key determines a codebook
  - Each key yields a different codebook
  - Employs both "confusion" and "diffusion"

Stream Ciphers

University of Amsterdam

Stream Ciphers



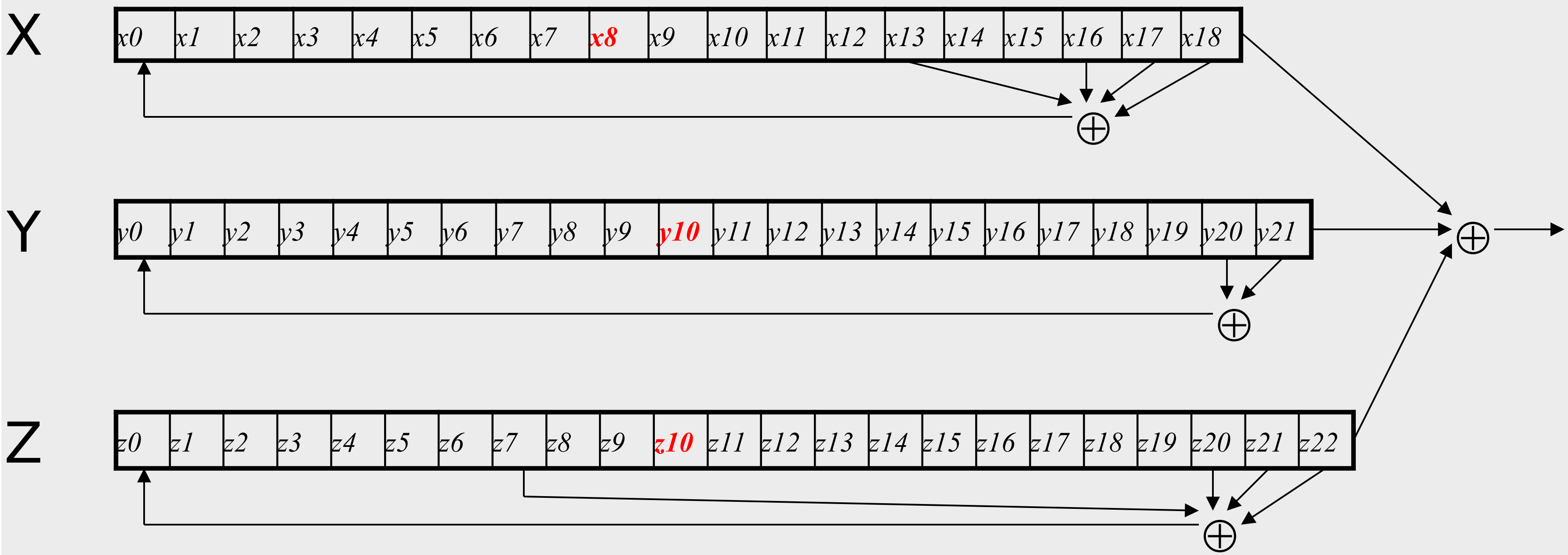
A5/1: Shift Registers

- A5/1 uses 3 *shift registers*
  - X: 19 bits $(x0, x1, x2, \ldots, x18)$
  - Y: 22 bits $(y0, y1, y2, \ldots, y21)$
  - Z: 23 bits $(z0, z1, z2, \ldots, z22)$

A5/1: Keystream

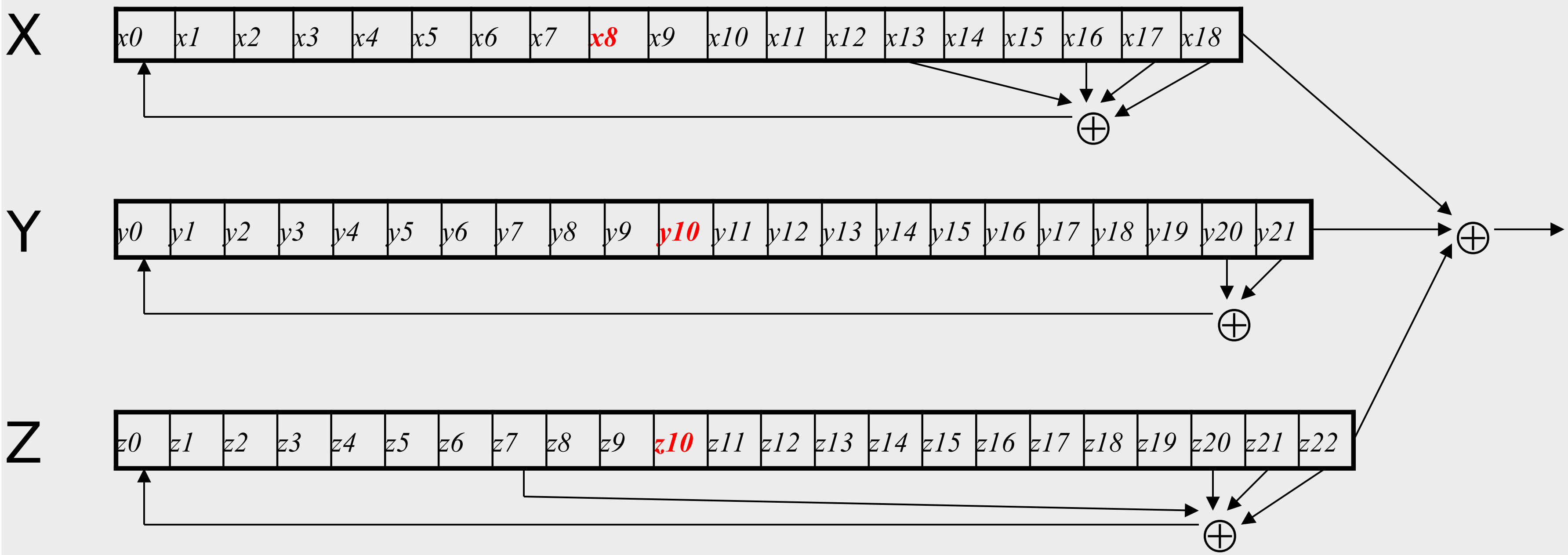
- At each step: $m = \mathrm{maj}(x8, y10, z10)$
  - Examples: $\mathrm{maj}(0,1,0) = 0$ and $\mathrm{maj}(1,1,0) = 1$
- If $x8 = m$ then X *steps*
  - $t = x13 \oplus x16 \oplus x17 \oplus x18$
  - $xi = xi-1$ for $i = 18,17,\ldots,1$ and $x0 = t$
- If $y10 = m$ then Y *steps*
  - $t = y20 \oplus y21$
  - $yi = yi-1$ for $i = 21,20,\ldots,1$ and $y0 = t$
- If $z10 = m$ then Z *steps*
  - $t = z7 \oplus z20 \oplus z21 \oplus z22$
  - $zi = zi-1$ for $i = 22,21,\ldots,1$ and $z0 = t$
- Keystream **bit** is $x18 \oplus y21 \oplus z22$

A5/1



- Each variable here is a single bit
- Key is used as **initial fill** of registers
- Each register steps (or not) based on $\mathrm{maj}(x8, y10, z10)$
- Keystream bit is XOR of rightmost bits of registers

A5/1



- Each variable here is a single bit
- Key is used as **initial fill** of registers
- Each register steps (or not) based on $\text{maj}(x8, y10, z10)$
- Keystream bit is XOR of rightmost bits of registers

Shift Register Crypto

- Shift register crypto efficient in hardware
- Often, slow if implement in software
- In the past, very popular
- Today, more is done in software due to fast processors
- Shift register crypto still used some
  - Resource-constrained devices

RC4 (ARC4)

- A self-modifying lookup table
- Table always contains a permutation of the byte values 0,1,…,255
- Initialize the permutation using key
- At each step, RC4 does the following
  - Swaps elements in current lookup table
  - Selects a keystream byte from table
- Each step of RC4 produces a **byte**
  - Efficient in software
- Each step of A5/1 produces only a bit
  - Efficient in hardware

RC4 Initialization

- S[] is permutation of 0,1,...,255
- key[] contains N bytes of key

```
for i = 0 to 255
    S[i] = i
    K[i] = key[i (mod N)]
next i
j = 0
for i = 0 to 255
    j = (j + S[i] + K[i]) mod 256
    swap(S[i], S[j])
next i
i = j = 0
```

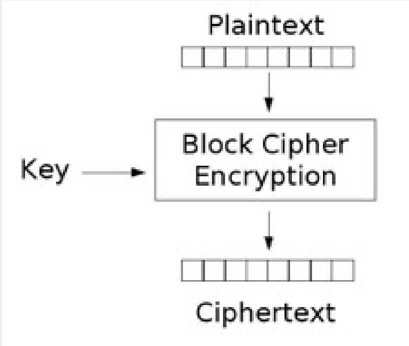
RC4 Keystream

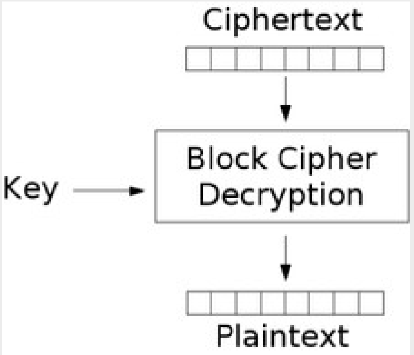
- For each keystream byte, swap elements in table and select byte

```
i = (i + 1) mod 256
j = (j + S[i]) mod 256
swap(S[i], S[j])
t = (S[i] + S[j]) mod 256
keystreamByte = S[t]
```

- Use keystream bytes like a one-time pad
- **Note:** first 256 bytes should be discarded
  - Otherwise, related key attack exists

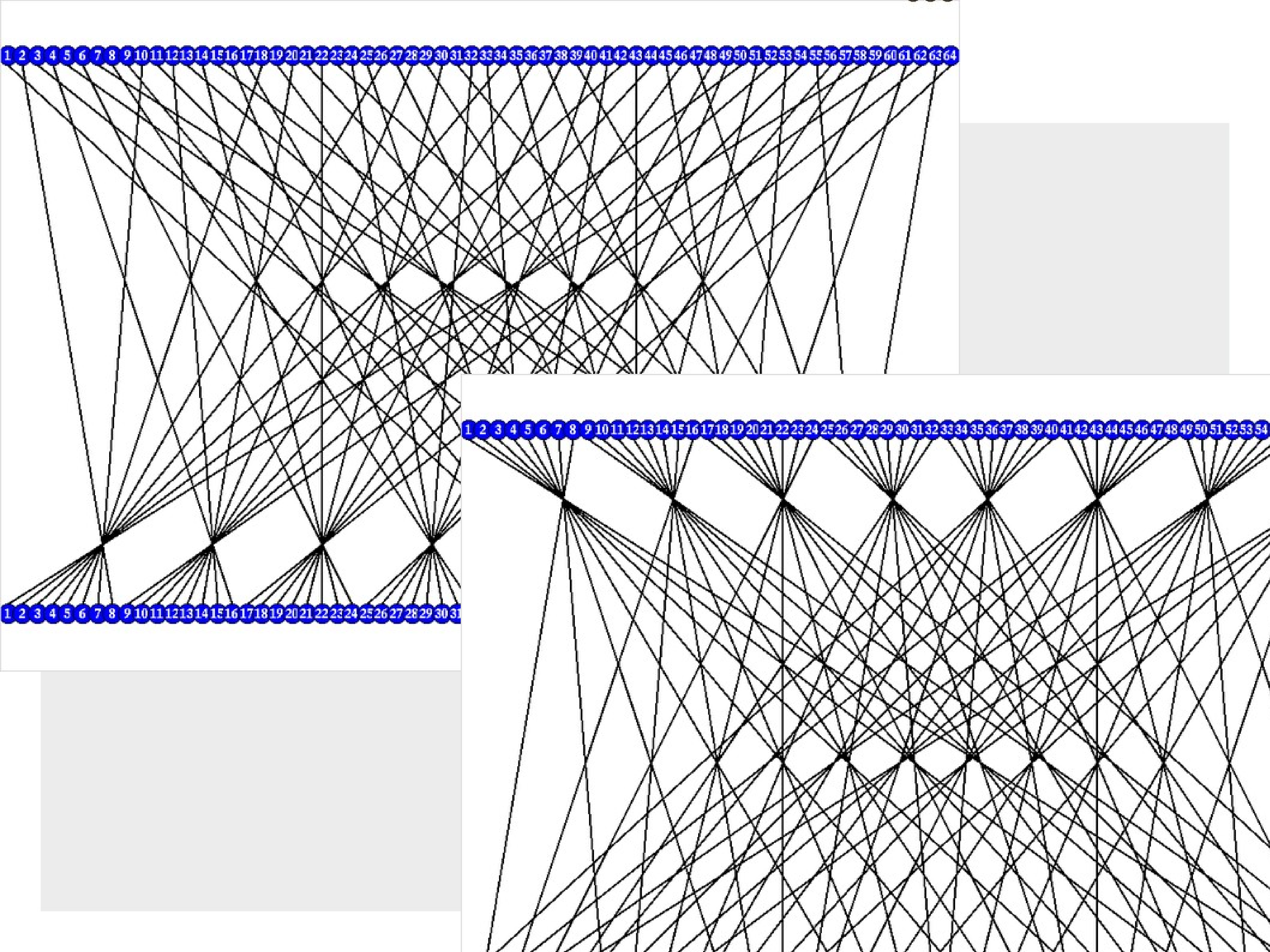
Block Ciphers

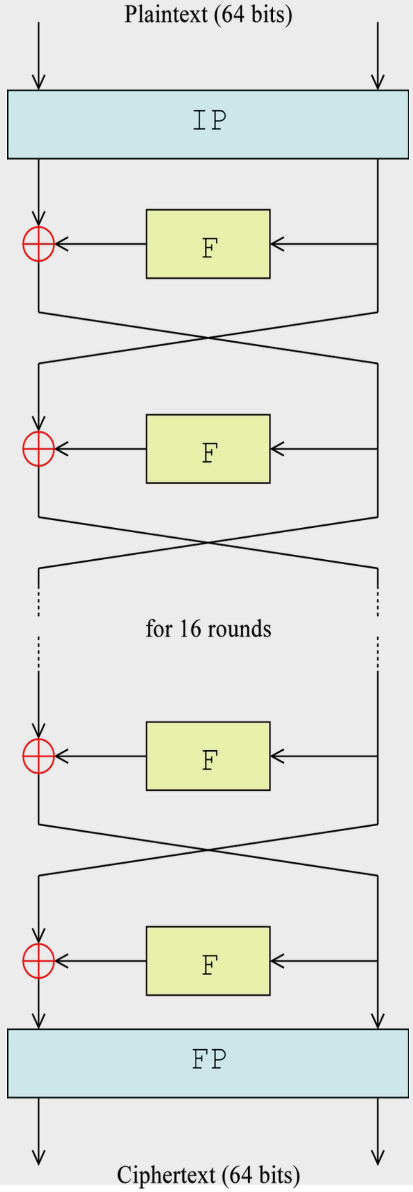


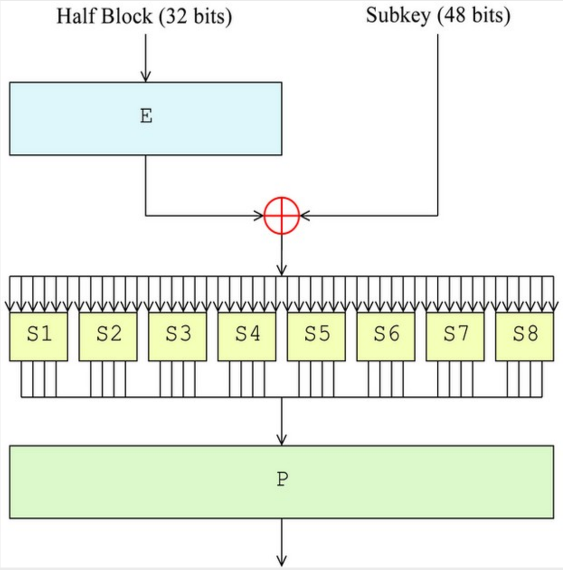


DES

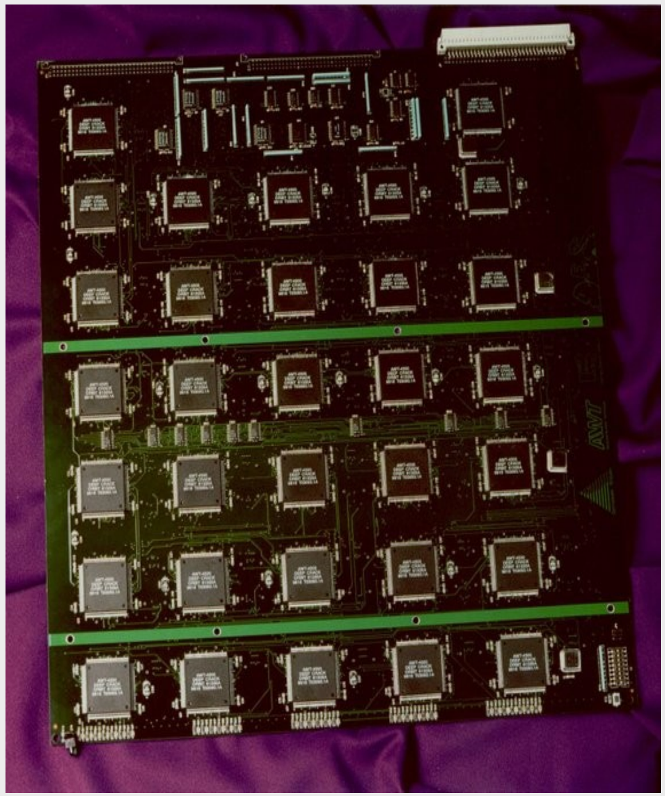
- Feistel Cipher
- DEA is algorithm
- 64 bits key with parity
- Effectively 56 bits
- Theoretically and practically considered cracked


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


Plaintext (64 bits)
IP
F
F
for 16 rounds
F
F
FP
Ciphertext (64 bits)


Half Block (32 bits)
Subkey (48 bits)
E
S1
S2
S3
S4
S5
S6
S7
S8
P

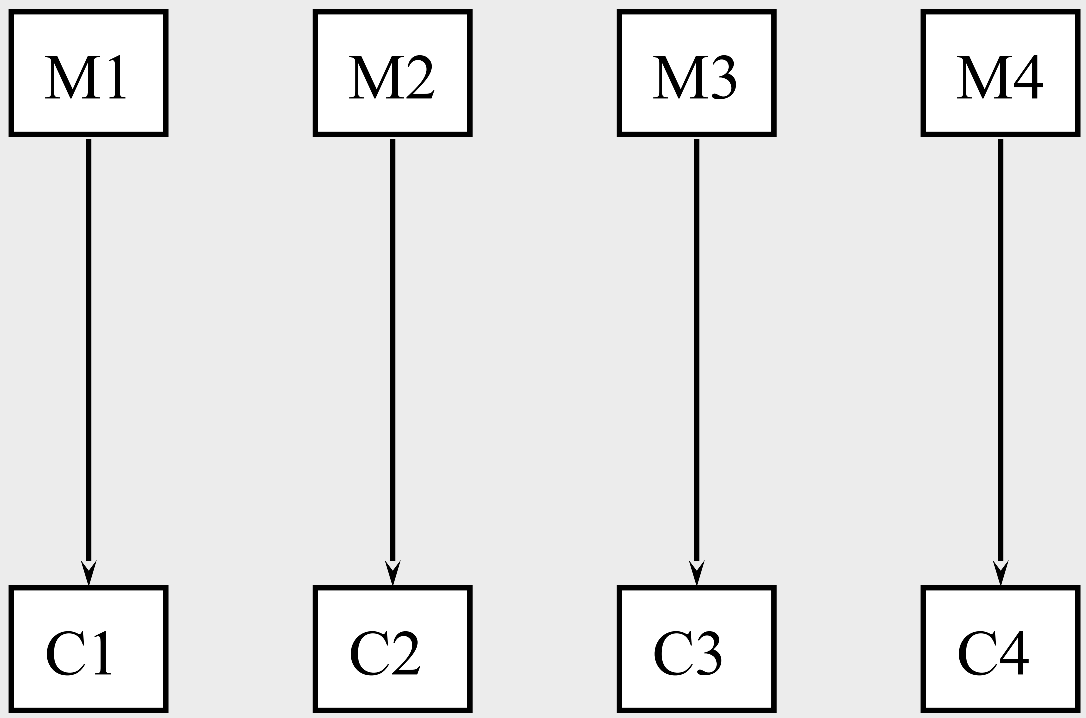
Deep Crack



Triple DES

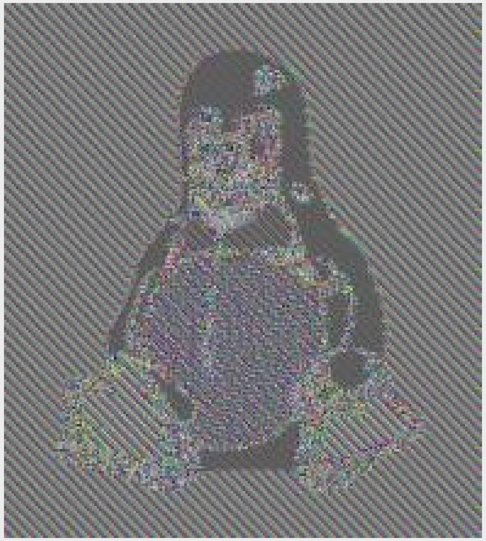
- 3 times ?
- In a smart way
- Key length between 80 en 112 bits
- EEE
- EDE with K1, K2, K3, often K1 equals K3.

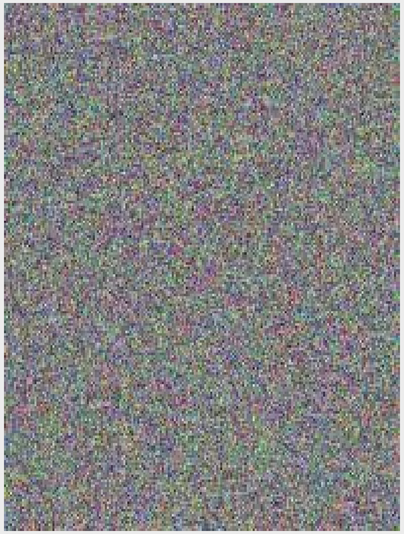
ECB



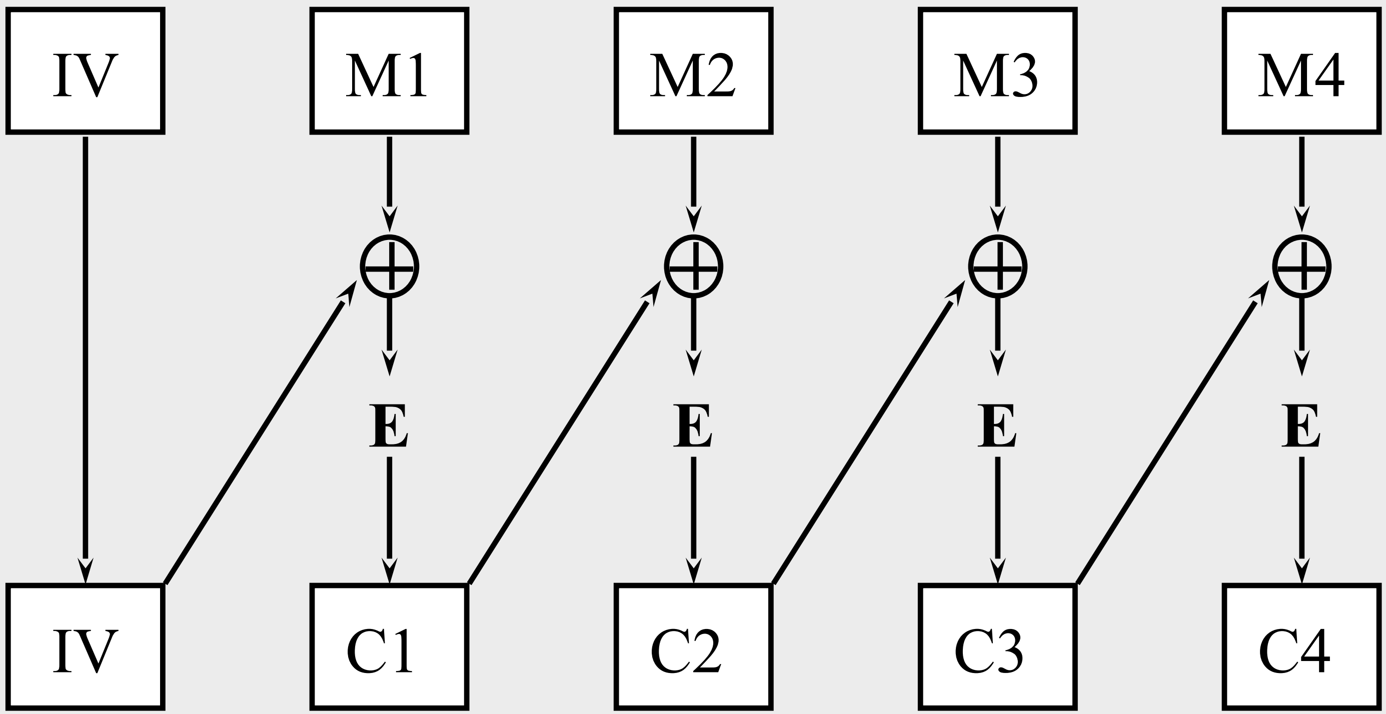
University of Amsterdam

ECB effect

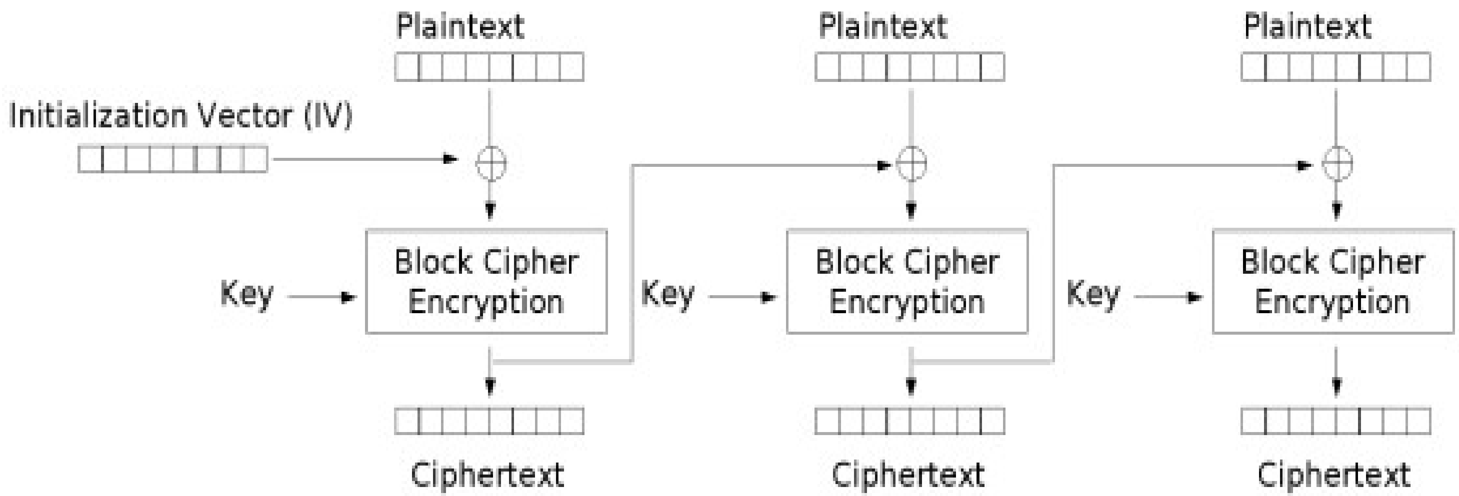




CBC (Cipher Block Chaining)

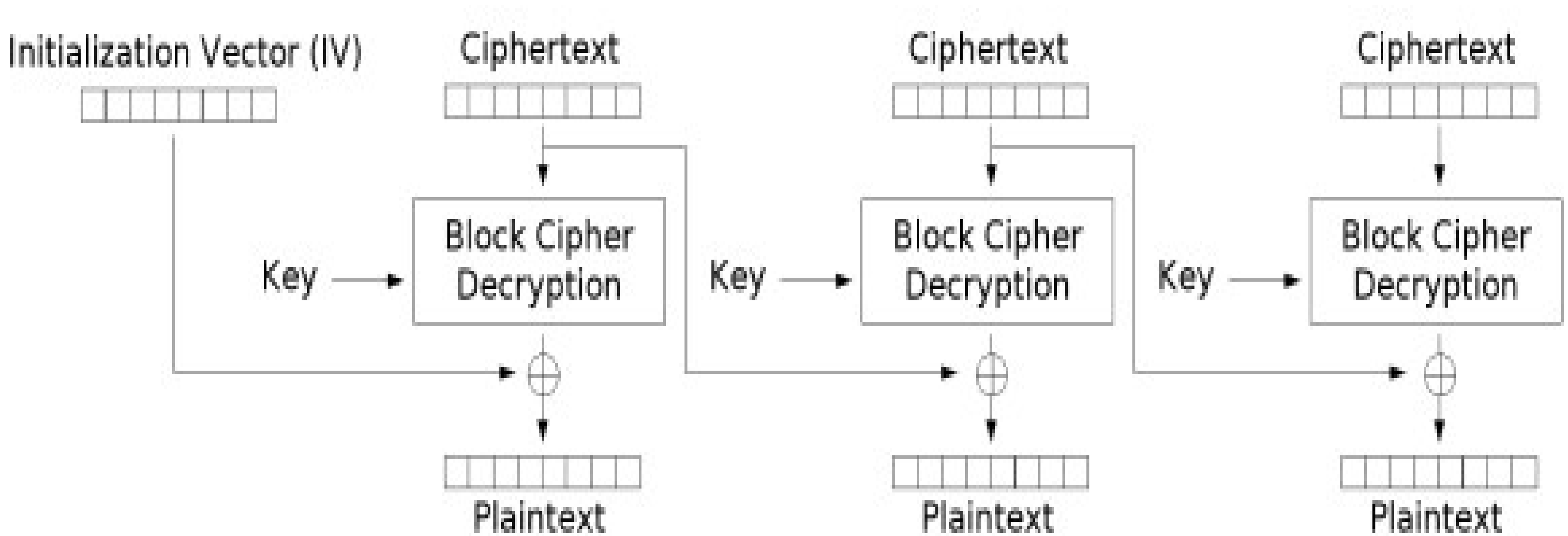


Cipher Block Chaining



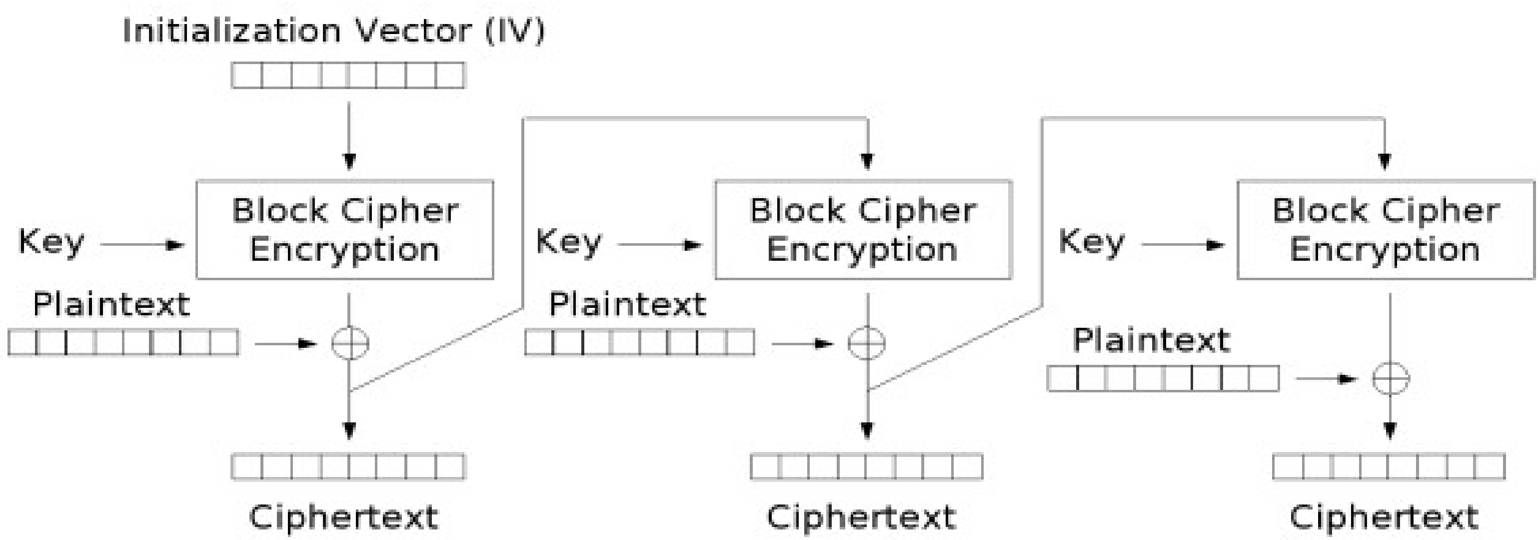
Cipher Block Chaining (CBC) mode encryption

CBC decryption



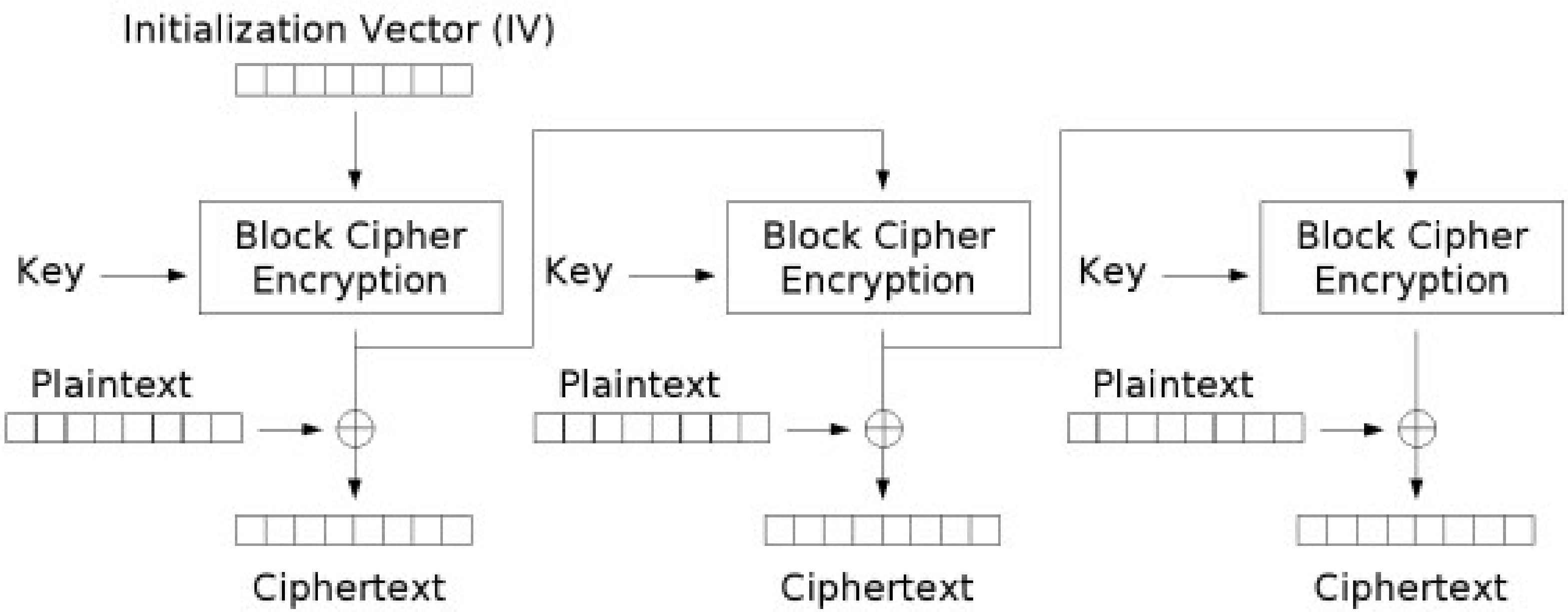
Cipher Block Chaining (CBC) mode decryption

Cipher feedback (CFB)

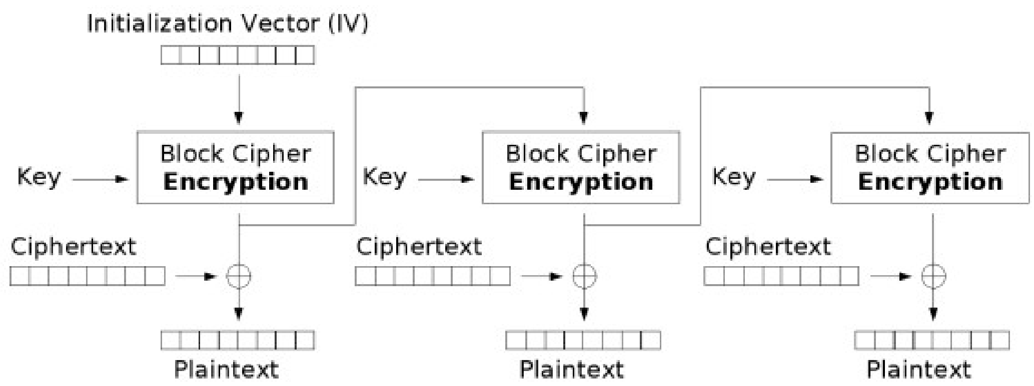


Cipher Feedback (CFB) mode encryption

Output Feedback Mode (OFB)

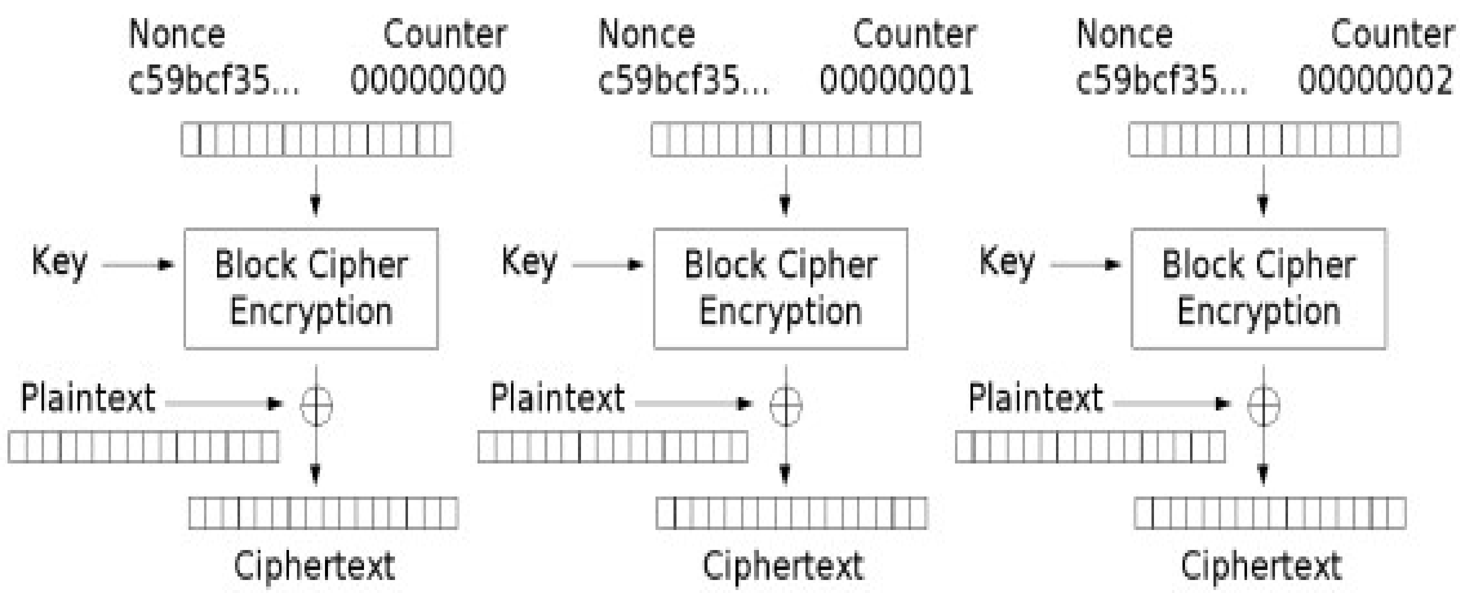


Output Feedback (OFB) mode encryption



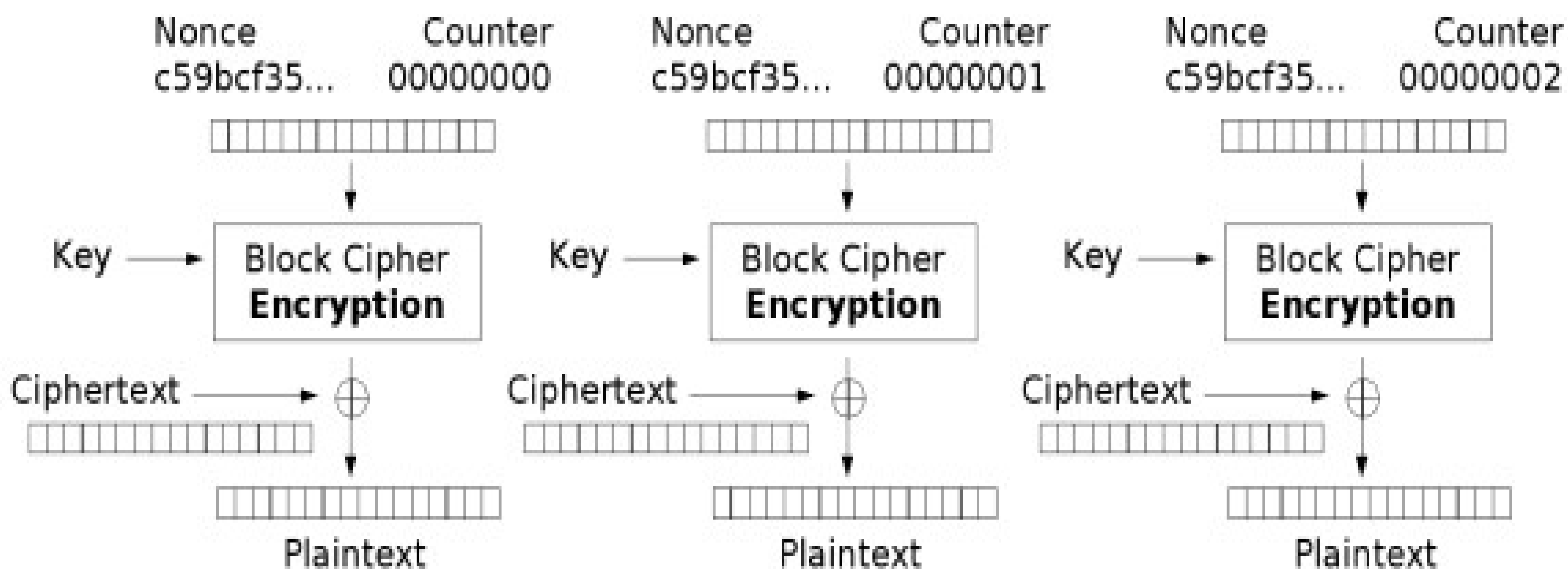
Output Feedback (OFB) mode decryption

Use with error correcting codes



Counter (CTR) mode encryption

Counter (CTR) Mode



Counter (CTR) mode decryption

Also known as Segmented Integer Counter (SIC) mode

Random Access possible
properties OFB

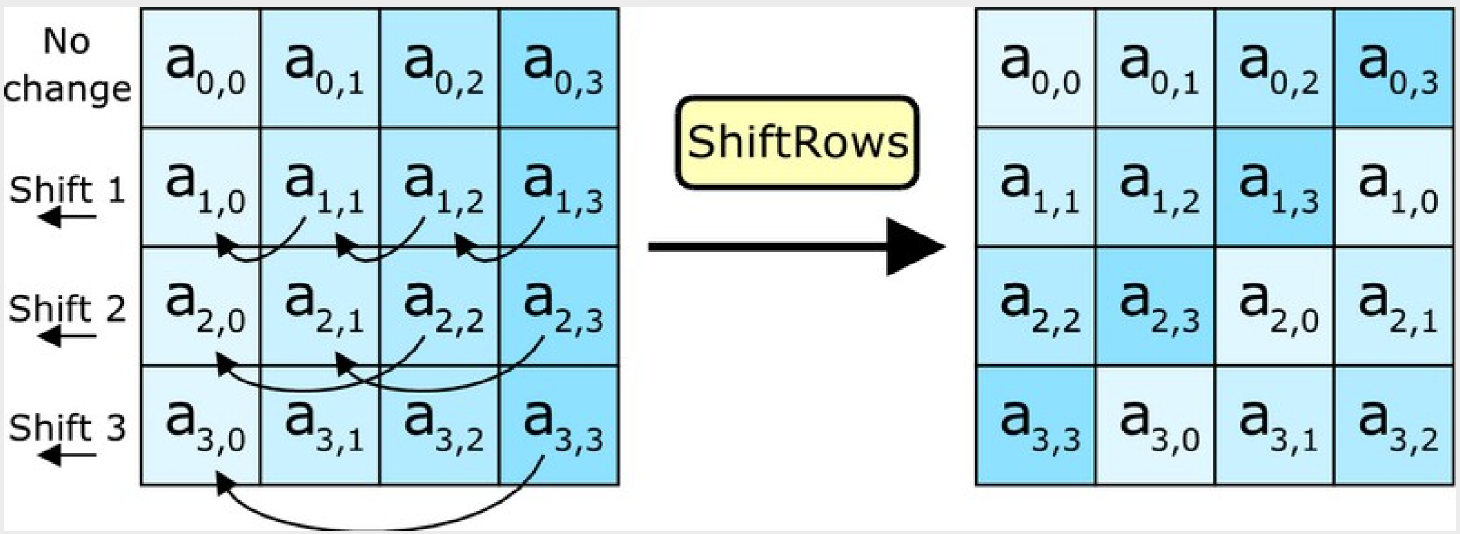
AES Competition

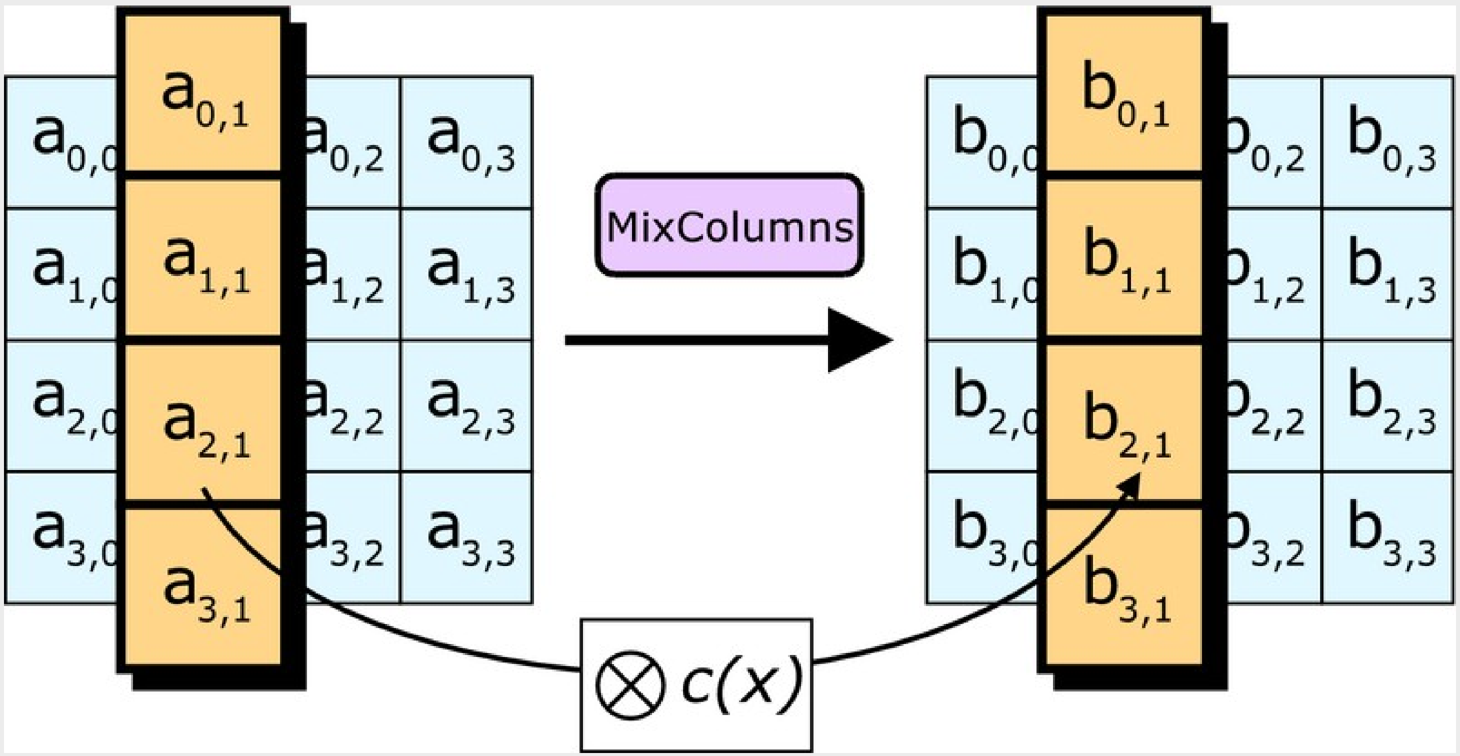
- NIST 1997-2001
  - MARS
  - RC-6
  - Rijndael
  - Twofish
  - Serpent

Rijndael

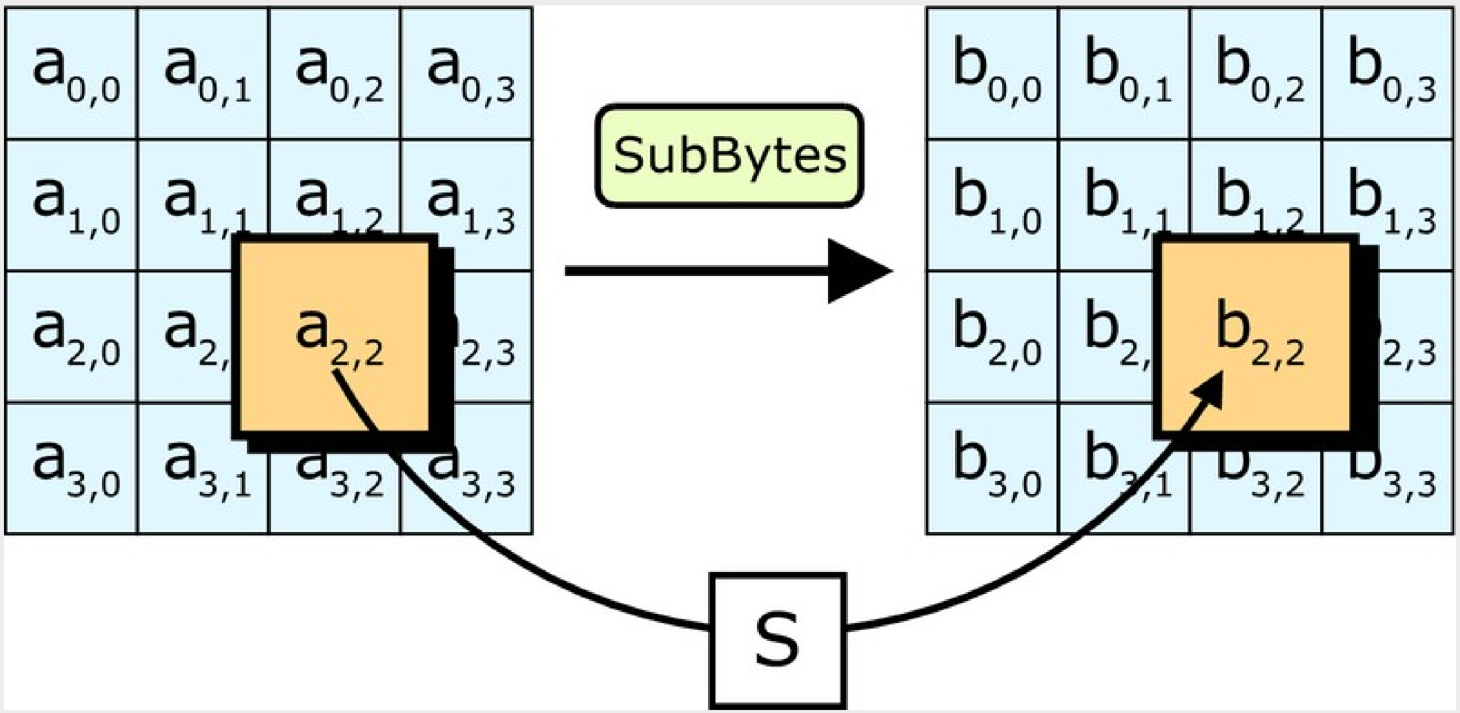
- Winner AES
- SPN (substitution–permutation network) cipher
- Joan Daemen and Vincent Rijmen


Welkom bij ESAT
Departement Elektrotechniek
10
0:00 / 3:00


No change
Shift 1
Shift 2
Shift 3
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
ShiftRows
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$ $a_{3,2}$



$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
MixColumns
$\otimes c(x)$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$


$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,2}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
SubBytes
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,2}$
$b_{3,0}$ $b_{3,2}$ $b_{3,3}$
S

Advanced Encryption Standard

- Replacement for DES
- AES competition (late 90's)
  - NSA openly involved
  - Transparent process
  - Many strong algorithms proposed
  - Rijndael Algorithm ultimately selected (pronounced like "Rain Doll" or "Rhine Doll")
- Iterated block cipher (like DES)
- Not a Feistel cipher (unlike DES)

AES Overview

- **Block size:** 128 bits (192 or 256)
- **Key length:** 128, 192 or 256 bits (independent of block size)
- 10 to 14 rounds (depends on key length)
- Each round uses 4 functions (3 "layers")
  - ByteSub (nonlinear layer)
  - ShiftRow (linear mixing layer)
  - MixColumn (nonlinear layer)
  - AddRoundKey (key addition layer)

AES ByteSub

- Treat 128 bit block as 4x6 byte array

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \longrightarrow \texttt{ByteSub} \longrightarrow \begin{bmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{10} & b_{11} & b_{12} & b_{13} \\ b_{20} & b_{21} & b_{22} & b_{23} \\ b_{30} & b_{31} & b_{32} & b_{33} \end{bmatrix}.$$

- ByteSub is AES's "S-box"
- Can be viewed as nonlinear (but invertible) composition of two math operations

AES "S-box"

Last 4 bits of input

First 4 bits of input

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

AES ShiftRow

- Cyclic shift rows

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \longrightarrow \texttt{ShiftRow} \longrightarrow \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{11} & a_{12} & a_{13} & a_{10} \\ a_{22} & a_{23} & a_{20} & a_{21} \\ a_{33} & a_{30} & a_{31} & a_{32} \end{bmatrix}$$

AES MixColumn

- Nonlinear, invertible operation applied to each column

$$\begin{bmatrix} a_{0i} \\ a_{1i} \\ a_{2i} \\ a_{3i} \end{bmatrix} \longrightarrow \texttt{MixColumn} \longrightarrow \begin{bmatrix} b_{0i} \\ b_{1i} \\ b_{2i} \\ b_{3i} \end{bmatrix} \quad \text{for } i = 0, 1, 2, 3$$

- Implemented as a (big) lookup table

AES AddRoundKey

- XOR subkey with block

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \oplus \begin{bmatrix} k_{00} & k_{01} & k_{02} & k_{03} \\ k_{10} & k_{11} & k_{12} & k_{13} \\ k_{20} & k_{21} & k_{22} & k_{23} \\ k_{30} & k_{31} & k_{32} & k_{33} \end{bmatrix} = \begin{bmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{10} & b_{11} & b_{12} & b_{13} \\ b_{20} & b_{21} & b_{22} & b_{23} \\ b_{30} & b_{31} & b_{32} & b_{33} \end{bmatrix}$$

Block Subkey

- RoundKey (subkey) determined by **key schedule** algorithm

AES Decryption

- To decrypt, process must be invertible
- Inverse of MixAddRoundKey is easy, since "$\oplus$" is its own inverse
- MixColumn is invertible (inverse is also implemented as a lookup table)
- Inverse of ShiftRow is easy (cyclic shift the other direction)
- ByteSub is invertible (inverse is also implemented as a lookup table)

AES Practicalities

- For now safe
  - For AES 256 best attack effective to 254.6
  - Seems Quantum computing resistant
- Hardware support (offload) in generic processors
  - Use Salsa20 or ChaCha as alternatives