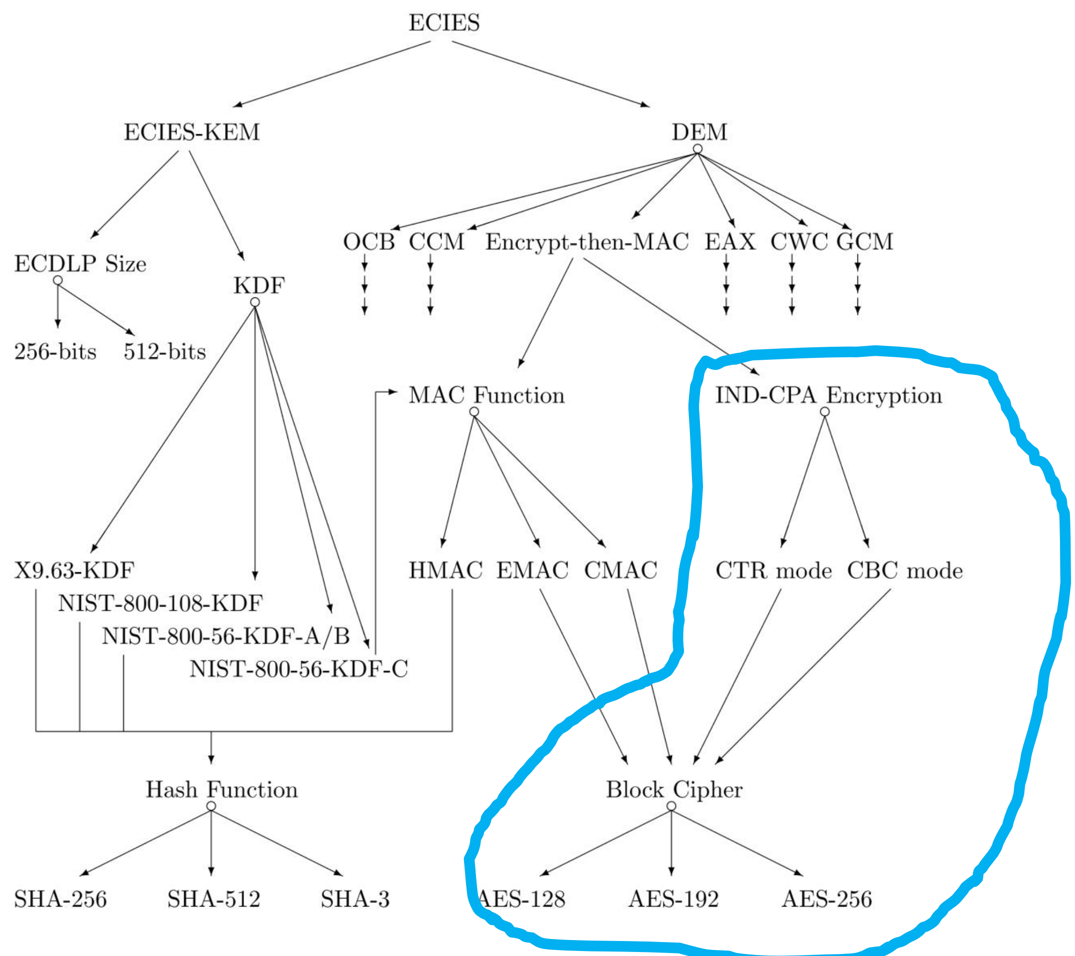
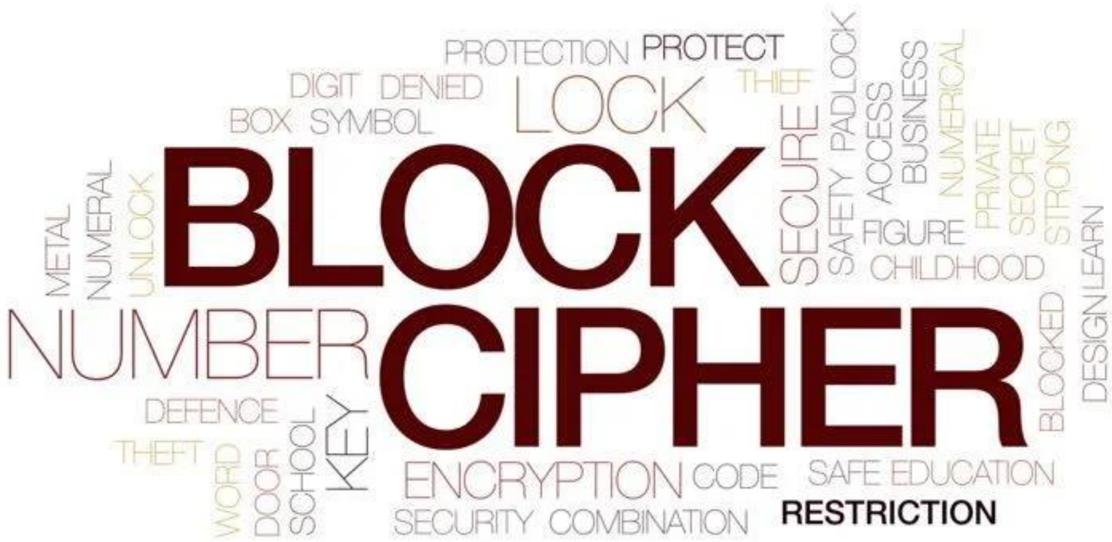
Cryptography
Lecture 2

Dr. Panagiotis Rizomiliotis

Roadmap



Algorithms, key size and parameters report. ENISA– Nov. 2014

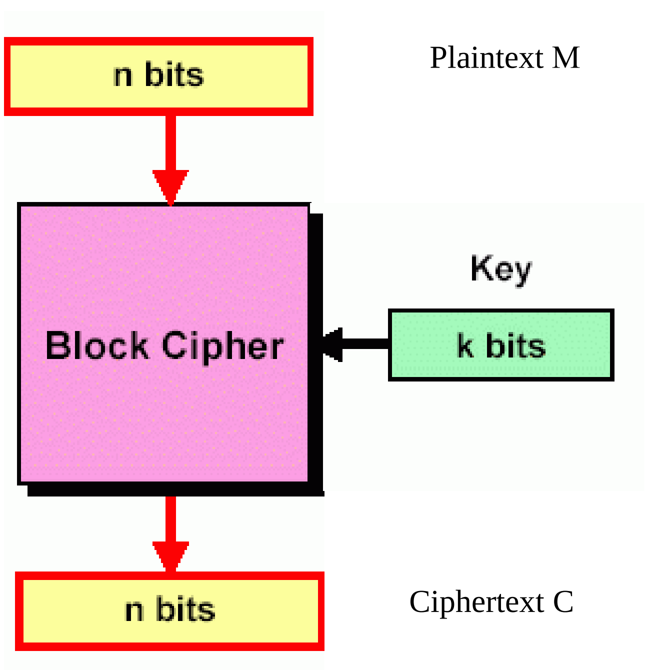


BLOCK CIPHERS

What is a block cipher?

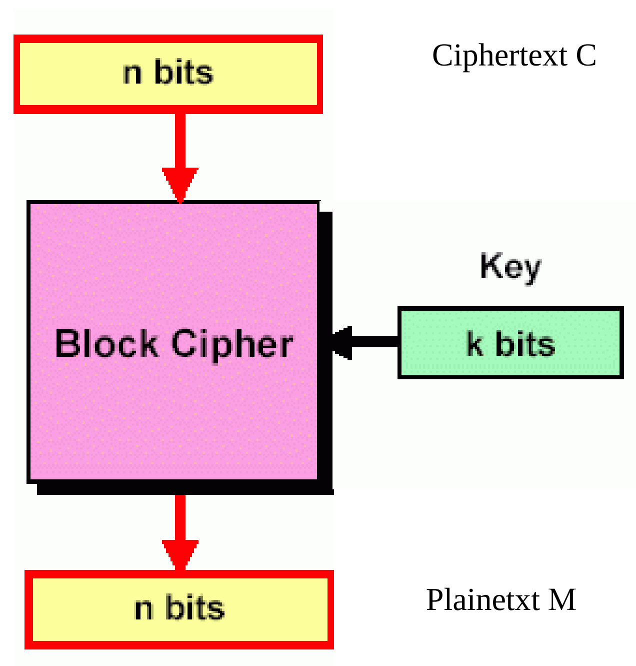
- $E_k : X \to X$ bijective for all k

A

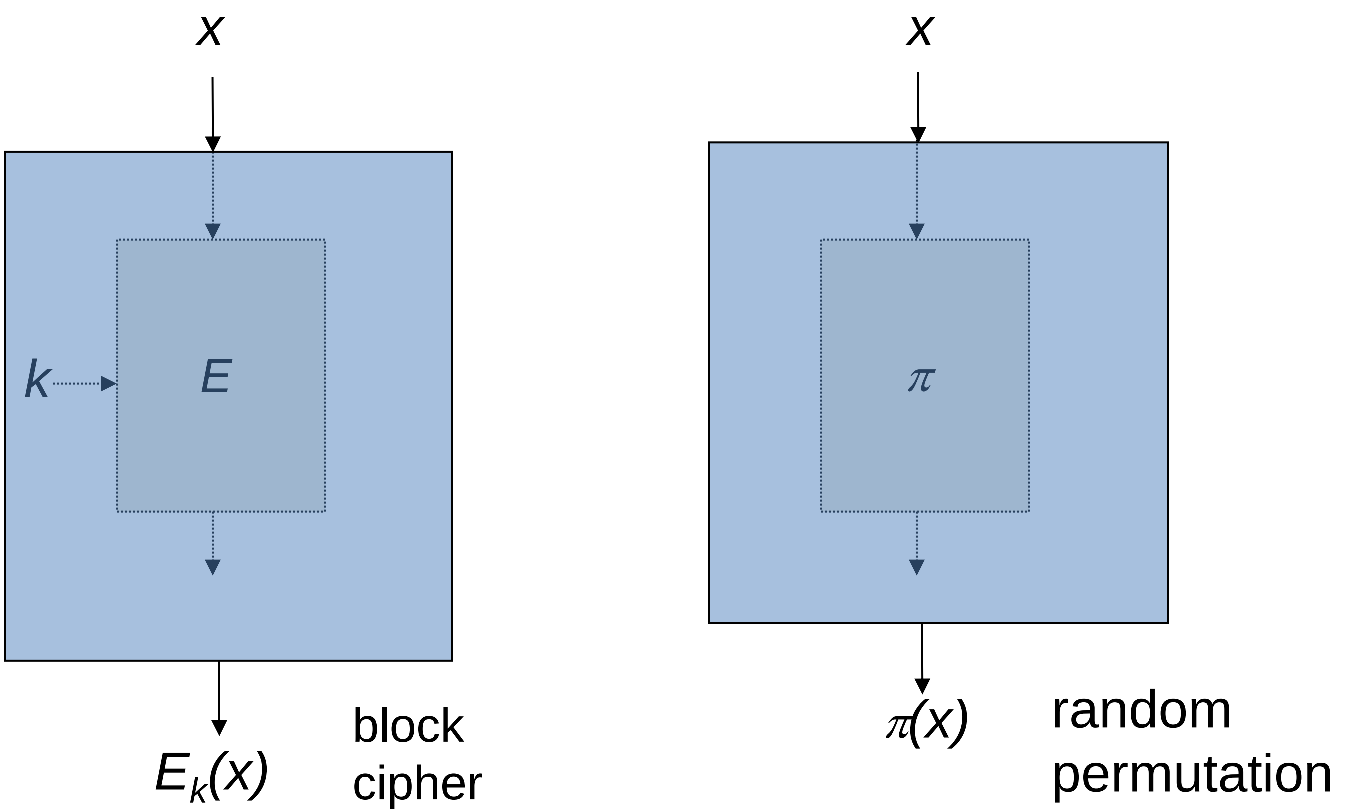


B

Inverse



When is a block cipher secure?

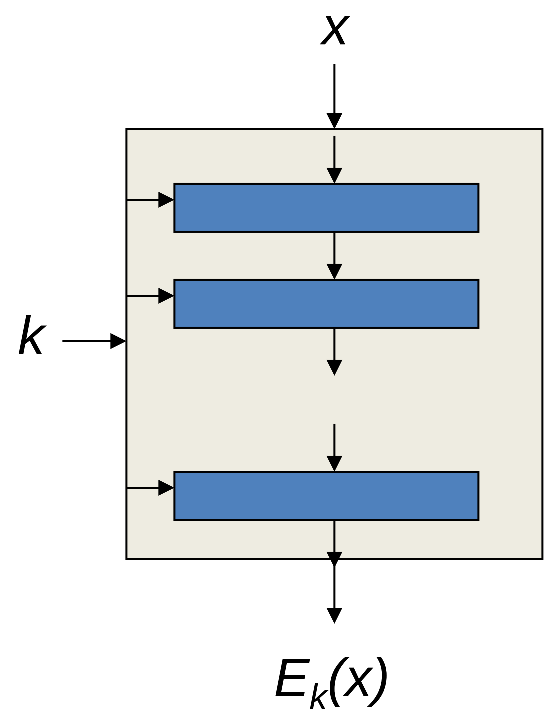


when these two black boxes are indistinguishable

Round constructions

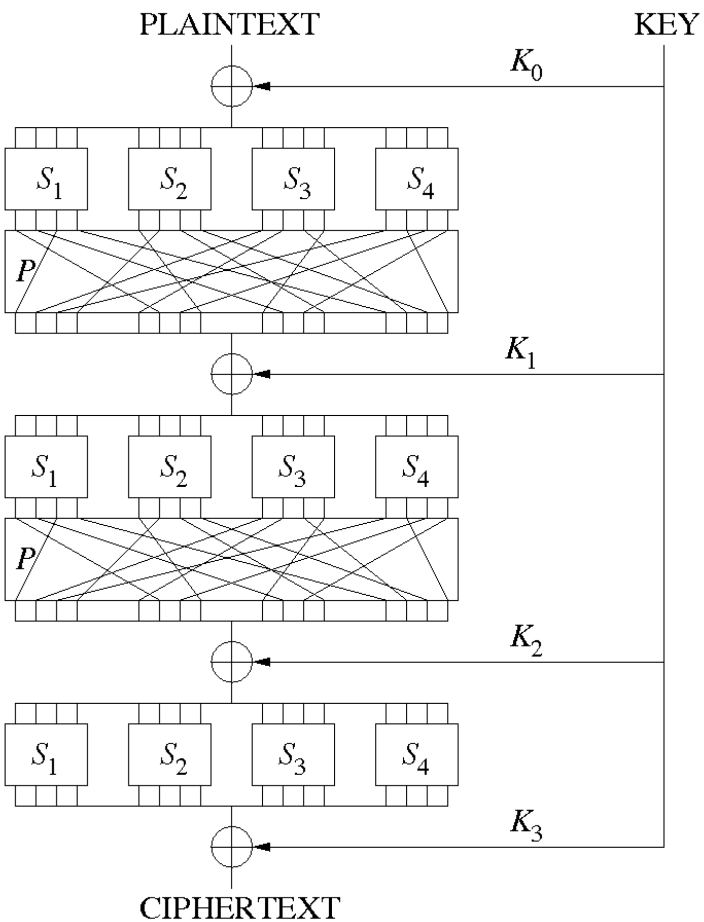
Two main constructions

1. Substitution-permutation Network (SPN)
   – AES = Advanced Encryption Standard
2. Feistel network
   – DES = Data Encryption Standard
   – Camellia

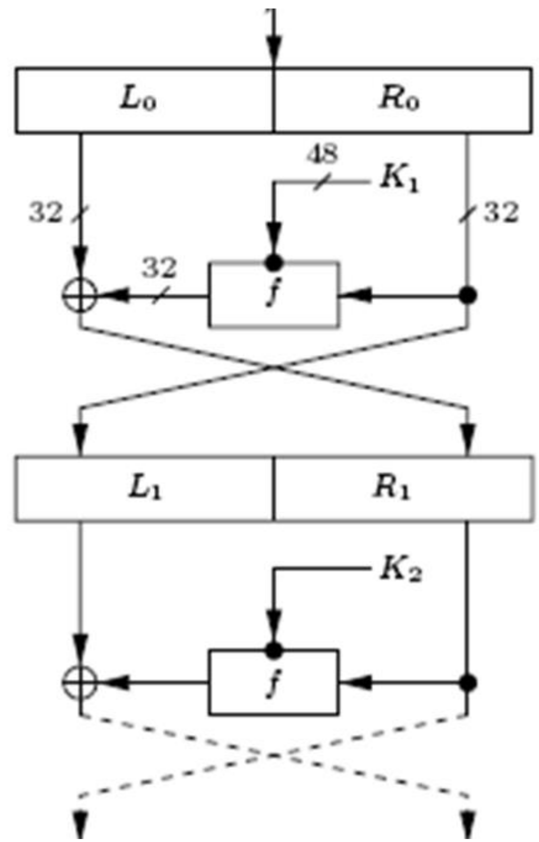


Block cipher Architectures

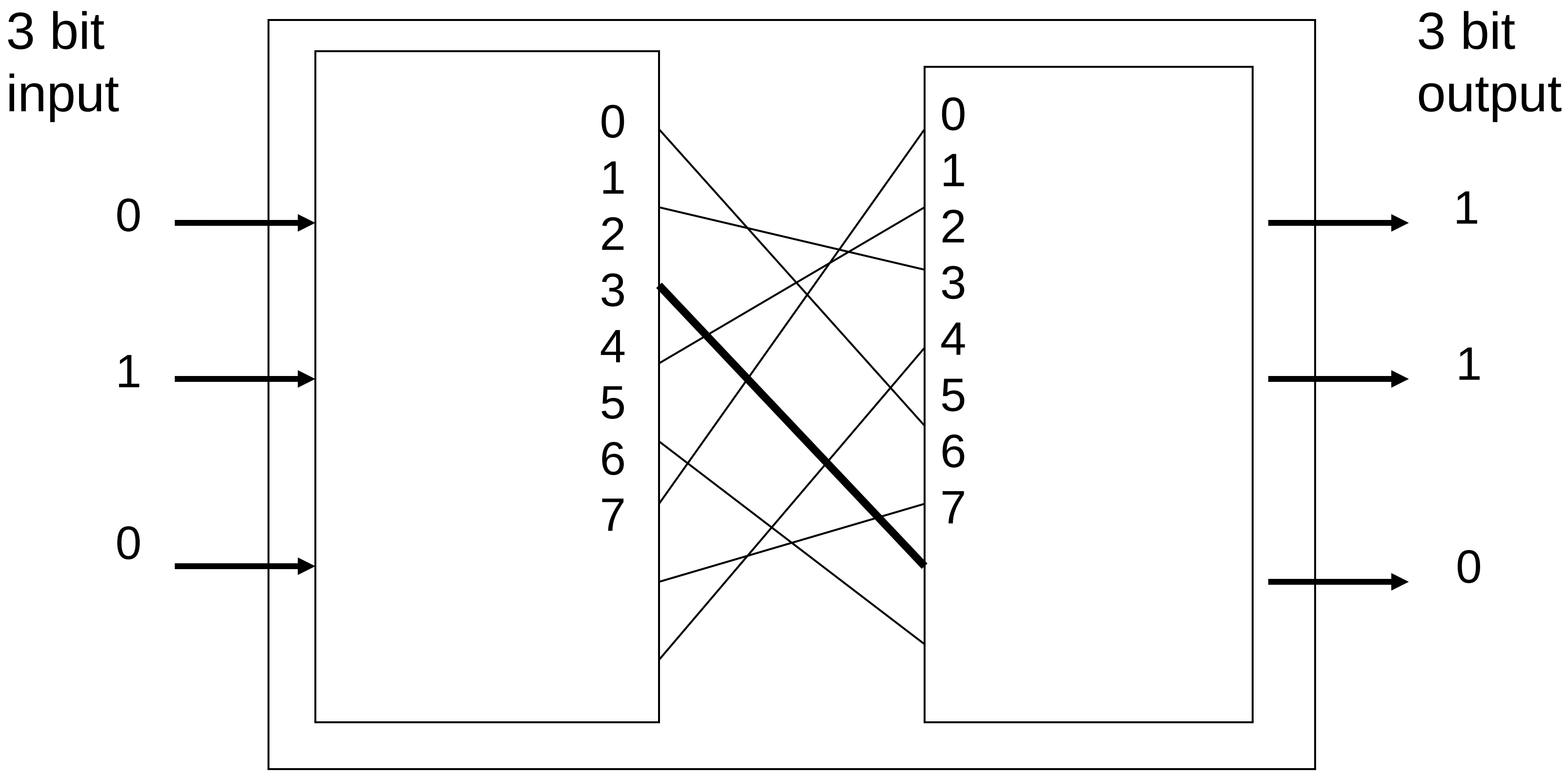
SPN



FEISTEL



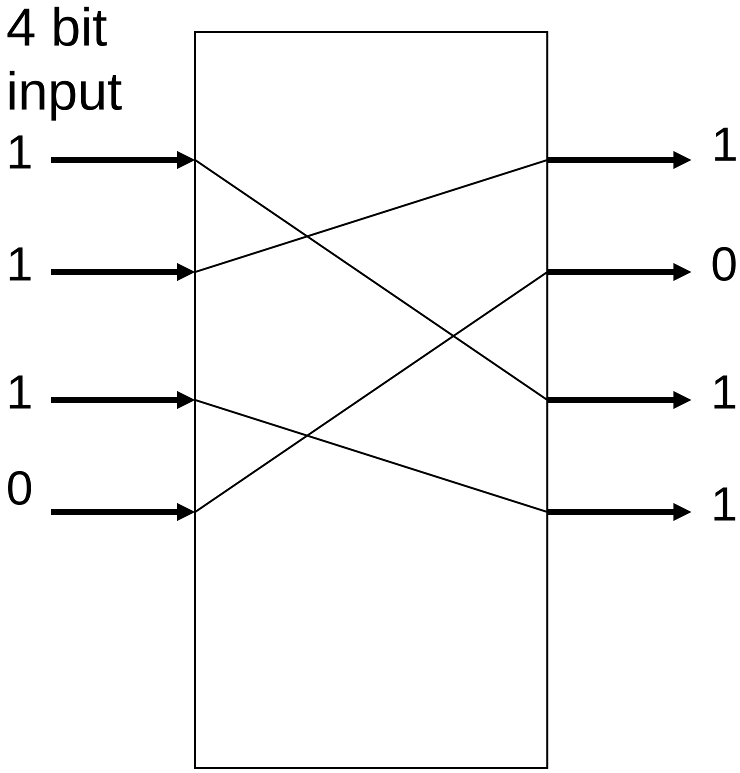
S-box (substitution)



Word size of 3 bits => mapping of $2^3 = 8$ values

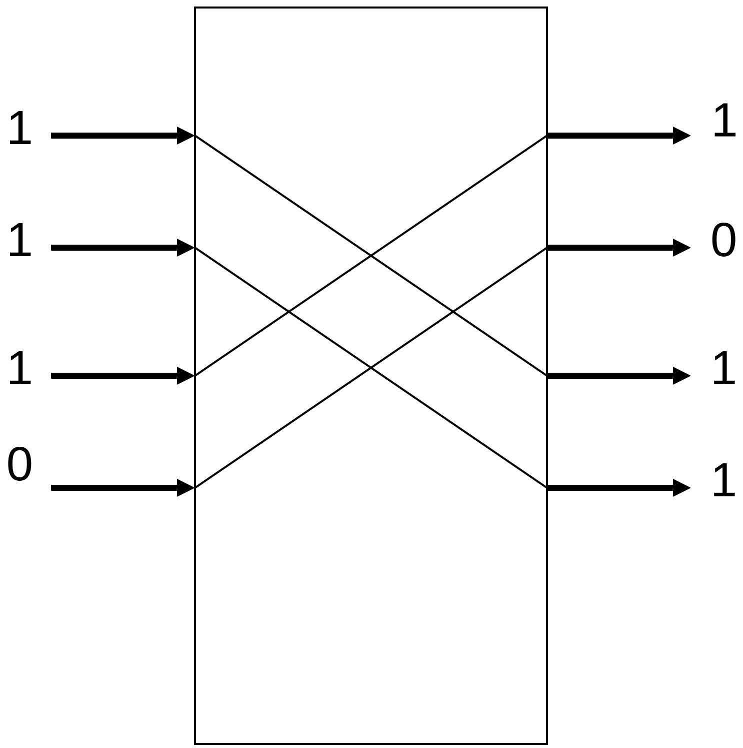
Note: mapping can be reversed

P-box (permutation)



Example 1

Note: reversible



Example 2 - swap two halves of input

DES: Data Encryption Standard

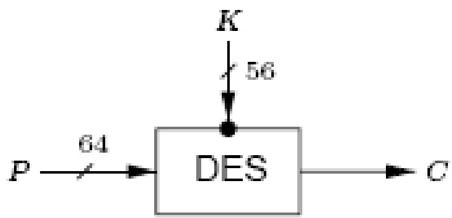
- 1972: National Bureau of Standards begins search
- 1975: DES: Lucifer by IBM, modified by NSA (key reduced from 128 to 56 bits)
- 1975: Approved by NBS (renewed periodically by NIST)

- Main characteristics:
  - Block size: 64 bits
  - Key size: 56 bits (4 weak keys)
  - Feistel cipher

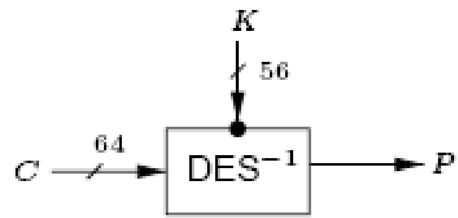
- Secure: hard to attack
- Easy to implement (in hardware, software)
- Easy to analyze

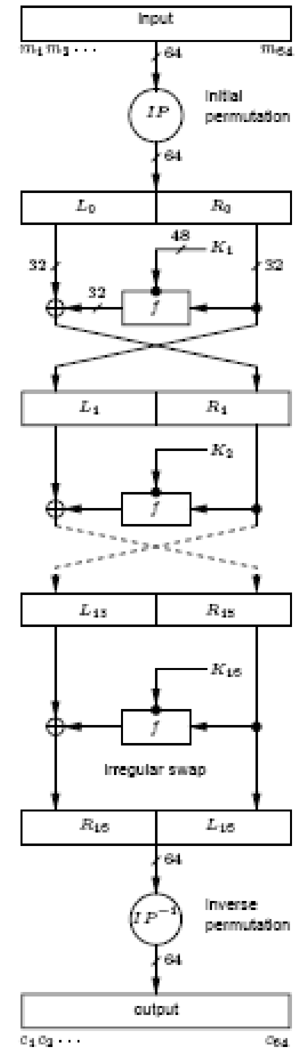
- Now considered obsolete due to the small key size (less than a day)

DES



plaintext P
ciphertext C
key K





- Block size: 64 bits
- Key size: 56 bits
- Initial permutation rearranges 64 bits (no cryptographic effect)
- Encoding is in 16 rounds

Linear Cryptanalysis

Linear cryptanalysis (Matsui, 1991):

- Look at algorithm structure: find places where, if you XOR plaintext and ciphertext bits together, you get key bits
- S-boxes not linear, but can approximate

- Need 2^{43} known pairs; best known attack
- DES apparently not optimized against this
- Still, not an easy-to-mount attack

Differential Cryptanalysis

- Biham & Shamir, 1993
- Against 8-round DES, attack requires:
  - 2^{14} = 16,384 chosen plaintexts, or
  - 2^{38} known plaintext-ciphertext pairs

- Against 16-round DES, attack requires:
  - 2^{47} chosen plaintexts, or
  - Roughly $2^{55.1}$ known plaintext-ciphertext pairs

- Differential cryptanalysis not effective
- Designers knew about it

DES security analysis

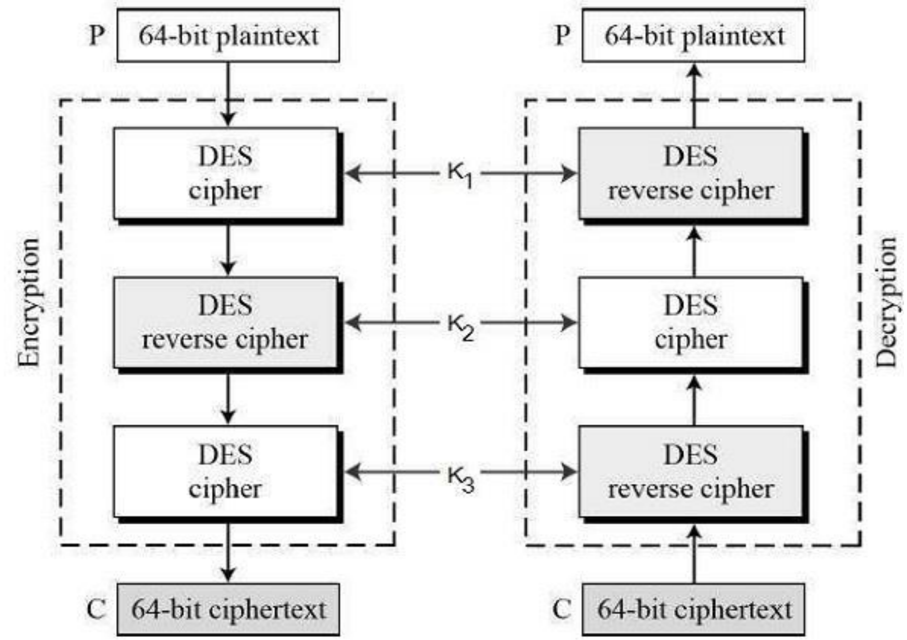
- “Weakest link” is size of key
- Attacks take advantage of encryption speed
- 1993: Weiner: $1M machine, 3.5 hours
- 1998: EFF’s Deep Crack: $250,000
  - 92 billion keys per second; 4 days on average

- 1999: distributed.net: 23 hours

- OK for some things (e.g., short time horizon)

- We need a solution!!!

Triple DES



- Several standards
- Run DES three times
- Main characteritics:

  Block size = 64 bits

  Key size (3 keys) = 168 bits (also, 1 and 2 keys)

  Security = 112 bits (there is an attack)
- It was a temporary solution.

What about Double DES (?) – Meet in the middle attack

- Double-DES: $C_i = E_B(E_A(P_i))$
- Given P_1, C_1: Note that $D_B(C_1) = E_A(P_1)$
- Make a list of every $E_K(P_1)$.
- Try each L: if $D_L(C_1) = E_K(P_1)$, then maybe K = A, L = B. (2^{48} L's might work.)
- Test with P_2, C_2: if it checks, it was probably right.
- Time roughly 2^{56}. Memory very large.

- WE NEED A NEW SOLUTION

Advanced Encryption Standard

NIST
National Institute of Standards and Technology

- January 1997: NIST announces that AES competition
- September 1997: NIST issues call for algorithms;
- August 1998: First AES conference, 15 candidates from 12 countries;
- August 1998-March 1999: public debate
- August 1999: NIST announces 5 finalists:
  - MARS (IBM, US)
  - RC6 (Rivest et al, MIT and RSA, US)
  - Rijndael (Daemen and Rijmen, Belgium)
  - Serpent (Anderson, Biham, Knudsen)
  - Twofish (Schneier, Kelsey et al, Counterpane, US)
- September 2000: Rijndael selected
- November 2001: NIST FIPS 197

AES Shortlist

- Shortlist in Aug-99:

✓ MARS (IBM) - complex, fast, high security margin
✓ RC6 (USA) - v. simple, v. fast, low security margin
✓ Rijndael (Belgium) - clean, fast, good security margin
✓ Serpent (Euro) - slow, clean, v. high security margin
✓ Twofish (USA) - complex, v. fast, high security margin

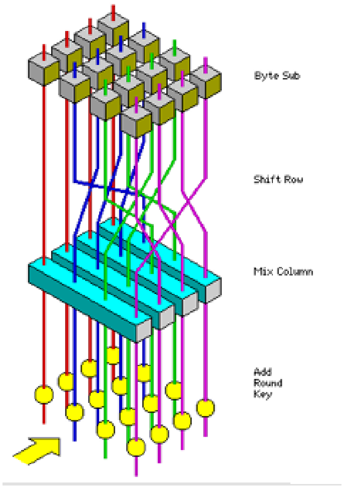
My name is AES (or Rijndael)

- <u>Designers:</u> Joan Daemen & Vincent Rijmen from the KULEUVEN- COSIC group)

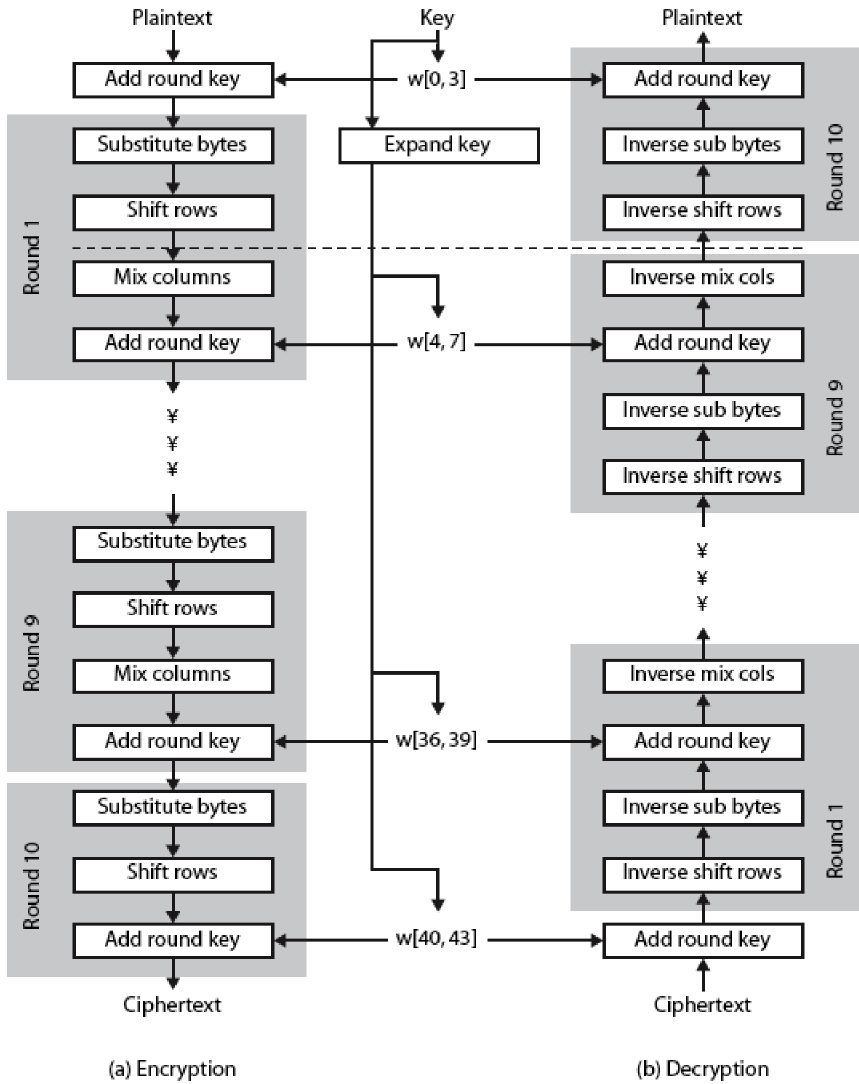
- An SPN block cipher
  - Standard: U.S. [FIPS](#) PUB 197 (FIPS 197)
- http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

- Block length:
  - 128 bits
  - 192
  - 256 bits

- Key size:
  - 128 bits (10 rounds)
  - 192 bits (12 rounds)
  - 256 bits (14 rounds)

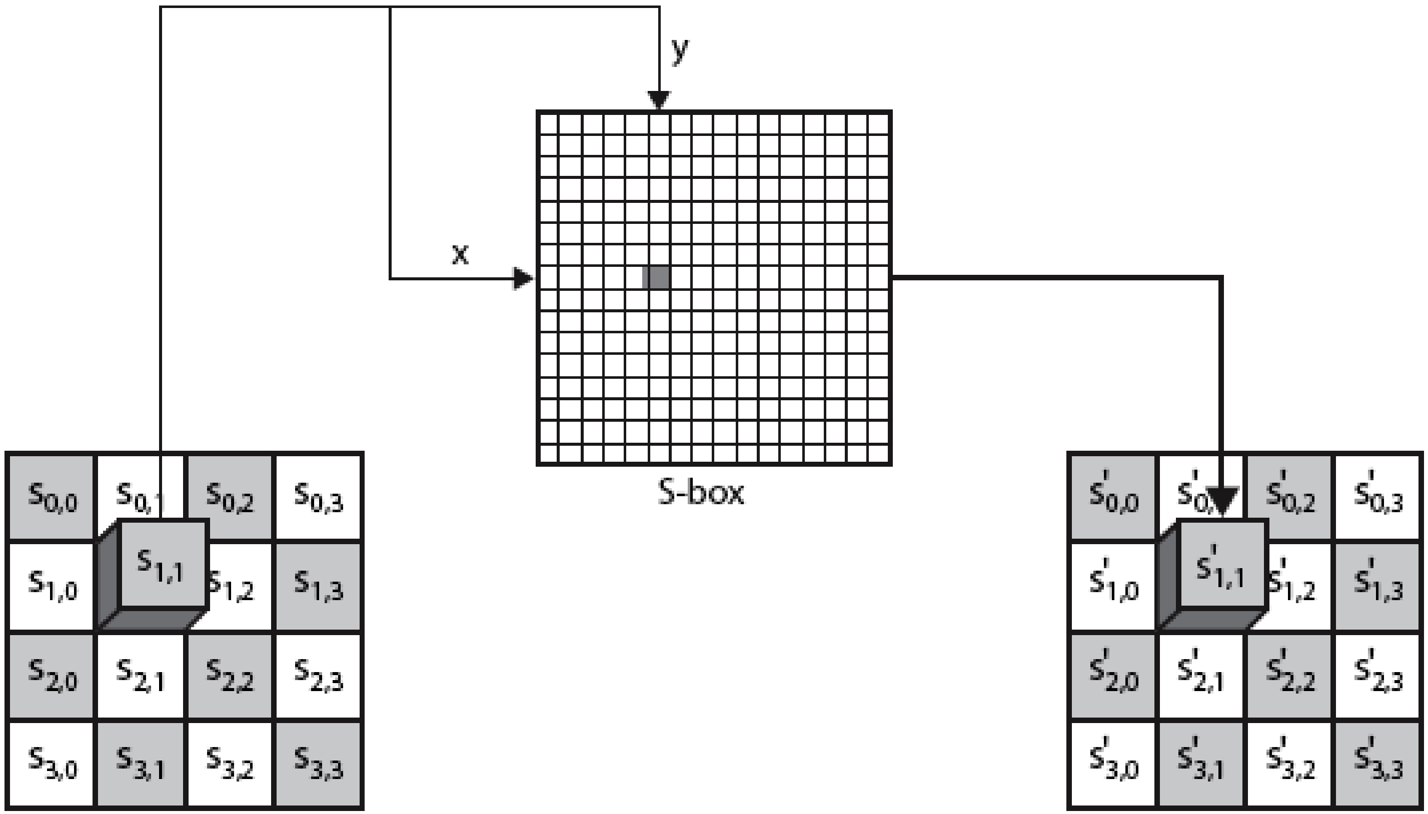


Rijndael



Plaintext
Key
Plaintext
Add round key
w[0, 3]
Add round key
Substitute bytes
Expand key
Inverse sub bytes
Shift rows
Inverse shift rows
Round 10
Round 1
Mix columns
Inverse mix cols
Add round key
w[4, 7]
Add round key
Inverse sub bytes
Inverse shift rows
Round 9
¥
¥
¥
Substitute bytes
Shift rows
Round 9
Mix columns
Inverse mix cols
Add round key
w[36, 39]
Add round key
Inverse sub bytes
Substitute bytes
Inverse shift rows
Round 1
Shift rows
Round 10
Add round key
w[40, 43]
Add round key
Ciphertext
Ciphertext
(a) Encryption
(b) Decryption

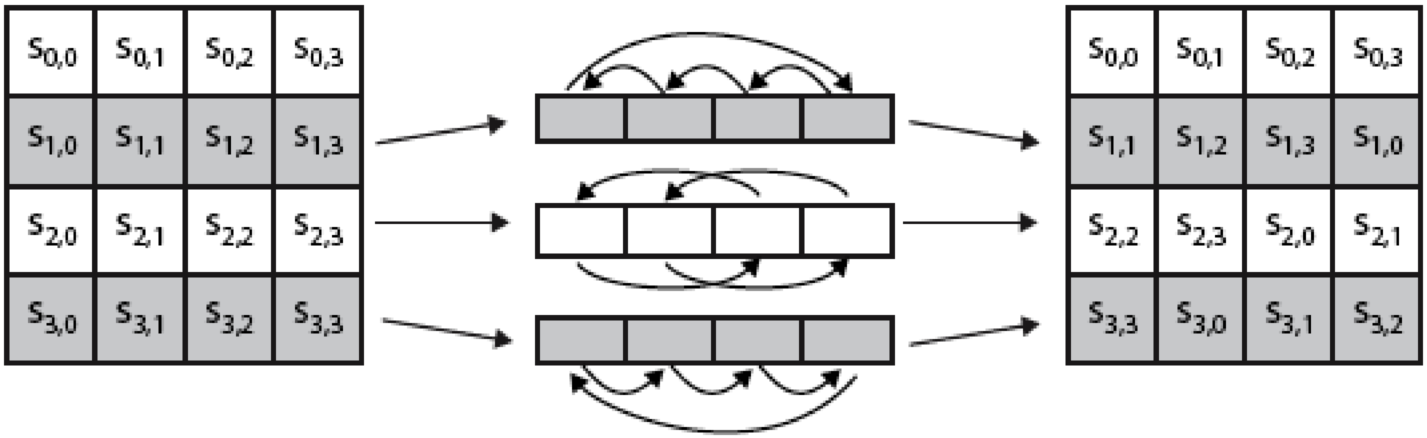
Byte Substitution



Byte Substitution

- a simple substitution of each byte
- uses one S-box of 16x16 bytes containing a permutation of all 256 8-bit values
- each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits)
  - eg. byte {95} is replaced by byte in row 9 column 5
  - which has value {2A}
- S-box constructed using defined transformation of values in GF(256)
- S-box constructed using a simple math formula using a non-linear function :   1/x.
- Construction of S-Box (on board)

Shift Rows

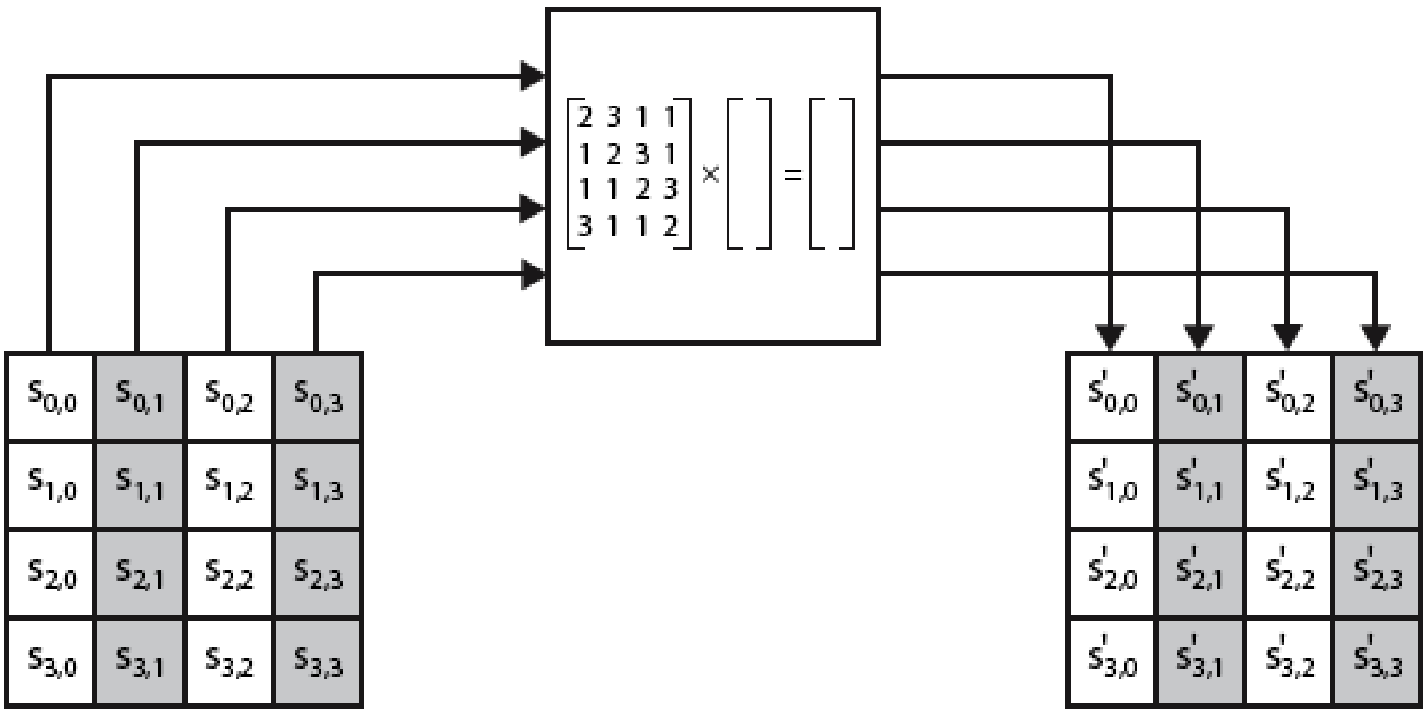


$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$ $s_{3,2}$

Shift Rows

- a circular byte shift in each each
  - 1st row is unchanged
  - 2nd row does 1 byte circular shift to left
  - 3rd row does 2 byte circular shift to left
  - 4th row does 3 byte circular shift to left
- decrypt inverts using shifts to right
- since state is processed by columns, this step permutes bytes between the columns

Mix Columns



2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2
×
=
s0,0 s0,1 s0,2 s0,3
s1,0 s1,1 s1,2 s1,3
s2,0 s2,1 s2,2 s2,3
s3,0 s3,1 s3,2 s3,3
s'0,0 s'0,1 s'0,2 s'0,3
s'1,0 s'1,1 s'1,2 s'1,3
s'2,0 s'2,1 s'2,2 s'2,3
s'3,0 s'3,1 s'3,2 s'3,3

Mix Columns

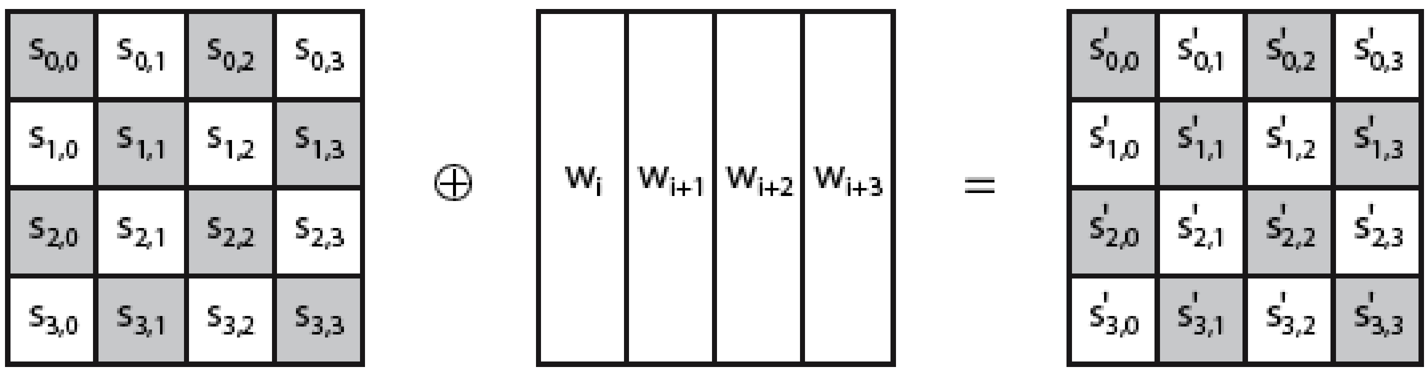
- each column is processed separately
- each byte is replaced by a value dependent on all 4 bytes in the column
- effectively a matrix multiplication in GF(2^8) using prime poly $m(x) = x^8+x^4+x^3+x+1$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Mix Columns

- can express each col of the new state as 4 equations
  - One equation to derive each new byte in col
- decryption requires use of inverse matrix
  - with larger coefficients, hence a little harder
- have an alternate characterization
  - each column a 4-term polynomial
  - with coefficients in $GF(2^8)$
  - and polynomials multiplied modulo (x^4+1)

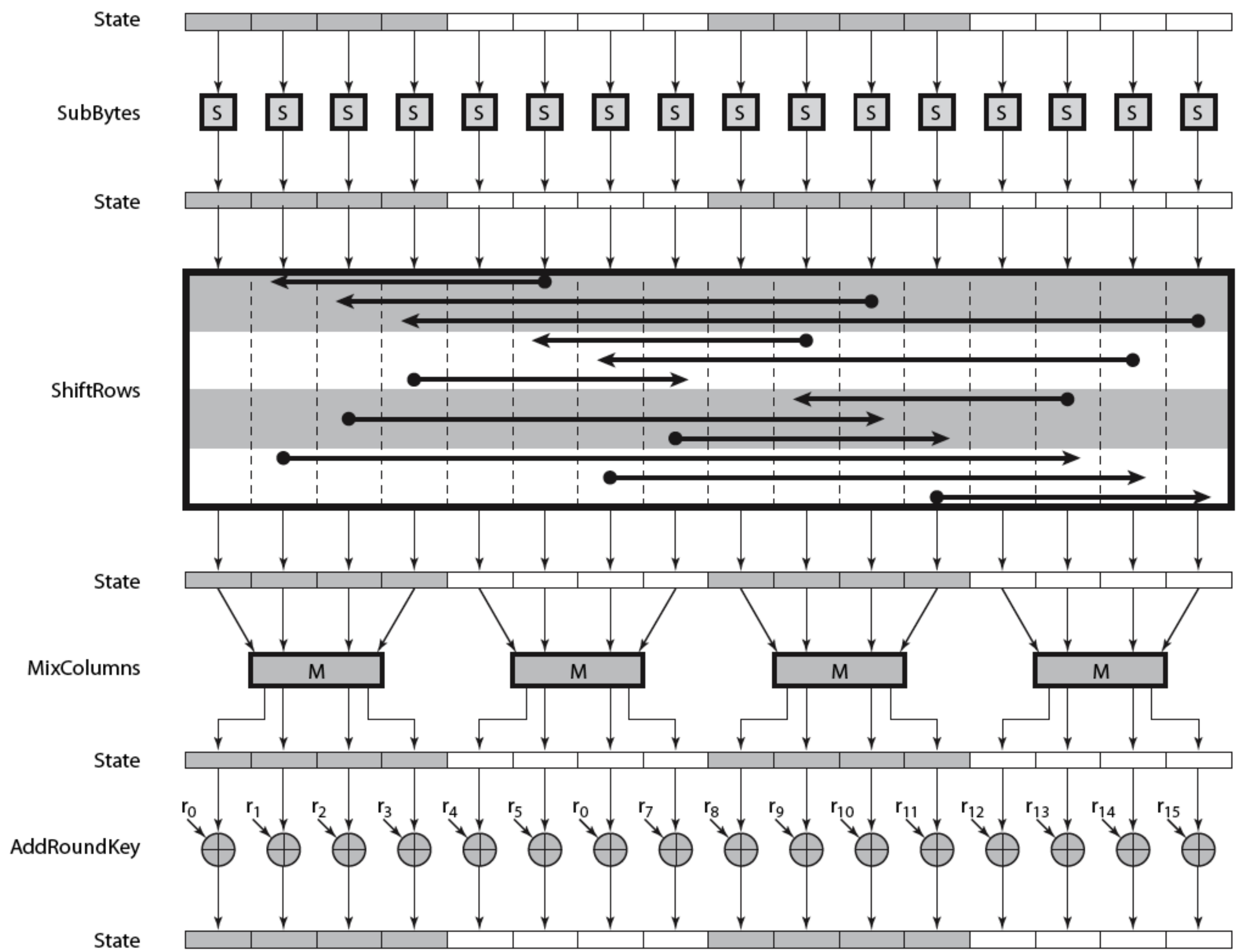
Add Round Key



Add Round Key

- XOR state with 128-bits of the round key
- again processed by column (though effectively a series of byte operations)
- inverse for decryption identical
  - since XOR own inverse, with reversed keys
- designed to be as simple as possible

AES Round



State
SubBytes
S S S S S S S S S S S S S S S S
State
ShiftRows
State
MixColumns
M M M M
State
AddRoundKey
r0 r1 r2 r3 r4 r5 r0 r7 r8 r9 r10 r11 r12 r13 r14 r15
State

AES Key Scheduling

- takes 128-bit (16-byte) key and expands into array of 44 32-bit words

- AES (Rijndeal)
  - Standard: U.S. FIPS PUB 197 (FIPS 197)
    - Serpent, Mars, RC6, Twofish (the AES finalists
  - http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

Security Analysis

During the competition

- Algebraic Attacks
- Boomerang
- Square attack
- High order differential attack
- ....

New Attacks

- Related-key attacks on the full versions of AES-192 and AES-256 which are faster than exhaustive search, but have impractical complexities.
- Related-key attacks requiring practical time complexity of 2^{45} on AES-256 with up to 10 rounds,
- Related key attacks requiring time complexity of 2^{70} on AES-256 with 11 rounds.
- Related key attacks requiring time complexity of $2^{99.5}$ on AES-256 and and $2^{99.5}$ data complexity (4-related keys.
- AES-128 with $2^{126,2}$ encryption operations and 2^{88} chosen plaintexts (bi-clique). Similar numbers for the other two key sizes.

- **No efficient attack against the full AES**

AES performance

- AES performed well on a wide variety of hardware, from 8-bit smart cards to high-performance computers.
- On a Pentium Pro, AES encryption requires 18 clock cycles per byte, equivalent to a throughput of about 11 MB/s for a 200 MHz processor.
- On a 1.7 GHz Pentium M throughput is about 60 MB/s.

AES instruction set

- Intel AES New Instructions (AES-NI)
- extension to the x86 instruction set architecture for microprocessors from Intel and AMD proposed by Intel in March 2008

Instruction	Description[2]
AESENC	Perform one round of an AES encryption flow
AESENCLAST	Perform the last round of an AES encryption flow
AESDEC	Perform one round of an AES decryption flow
AESDECLAST	Perform the last round of an AES decryption flow
AESKEYGENASSIST	Assist in AES round key generation
AESIMC	Assist in AES Inverse Mix Columns
PCLMULQDQ	Carryless multiply (CLMUL)[3]

Source Wikipedia

AES performance – NI enabled

From AES-NI Performance Analyzed, Patrick Schmid and Achim Roos

- Crypto++ security library
- Increase in throughput from approximately 28.0 cycles per byte to 3.5 cycles per byte with AES/GCM versus a Pentium 4 with no acceleration
- On Intel Core i3/i5/i7 and AMD Ryzen CPUs supporting AES-NI instruction set extensions, throughput can be multiple GB/s (even over 10 GB/s)

State of the art

Primitive	Classification: Legacy	Classification: Future
AES	✓	✓
Camellia	✓	✓
Three-Key-3DES	✓	✗
Two-Key-3DES	✓	✗
Kasumi	✓	✗
Blowfish$^{\geq 80\text{-bit keys}}$	✓	✗
DES	✗	✗

- ***Algorithms, key size and parameters report. ENISA– Nov. 2014***

Other Block ciphers

- Camellia
  - One of the possible cipher suites in TLS
  - Feistel cipher design
  - Block length of 128 bits
  - Supports 3 key lengths: 128, 192 and 256 bits (33% slower than 128-bit key)
  - No effective attacks are known.

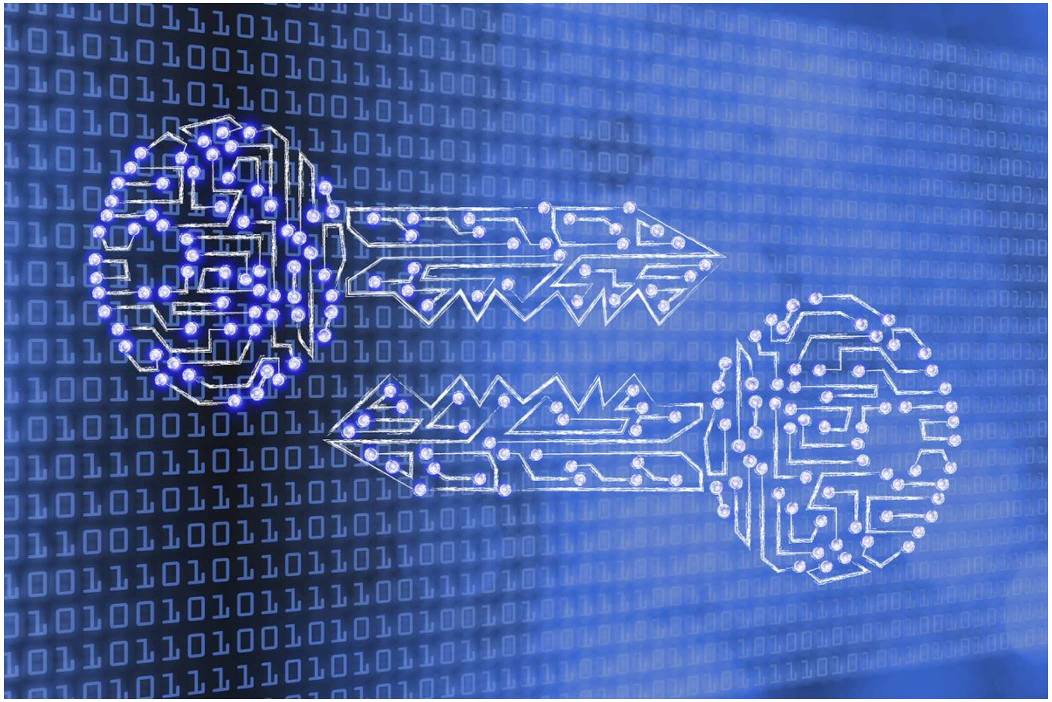
Legacy Block Ciphers

- Blowfish

➢ 64-bit block size (too small)
➢ Key size ranging from 32- to 448-bits
➢ Used in some IPsec configurations.
➢ A number of attacks on reduced round versions.

- Kasumi

➢ Used in 3GPP (MISTY-1), UIA1 in UMTS and A5/3 in GSM
➢ 128-bit key
➢ 64-bit block size.
➢ Related key attack is given which requires 2^{32} time and 2^{26} plaintext/ciphertext pairs.
➢ These attacks *do not affect* the practical use of Kasumi in applications such as 3GPP,



SYMMETRIC ENCRYPTION SCHEMES

Symmetric encryption schemes

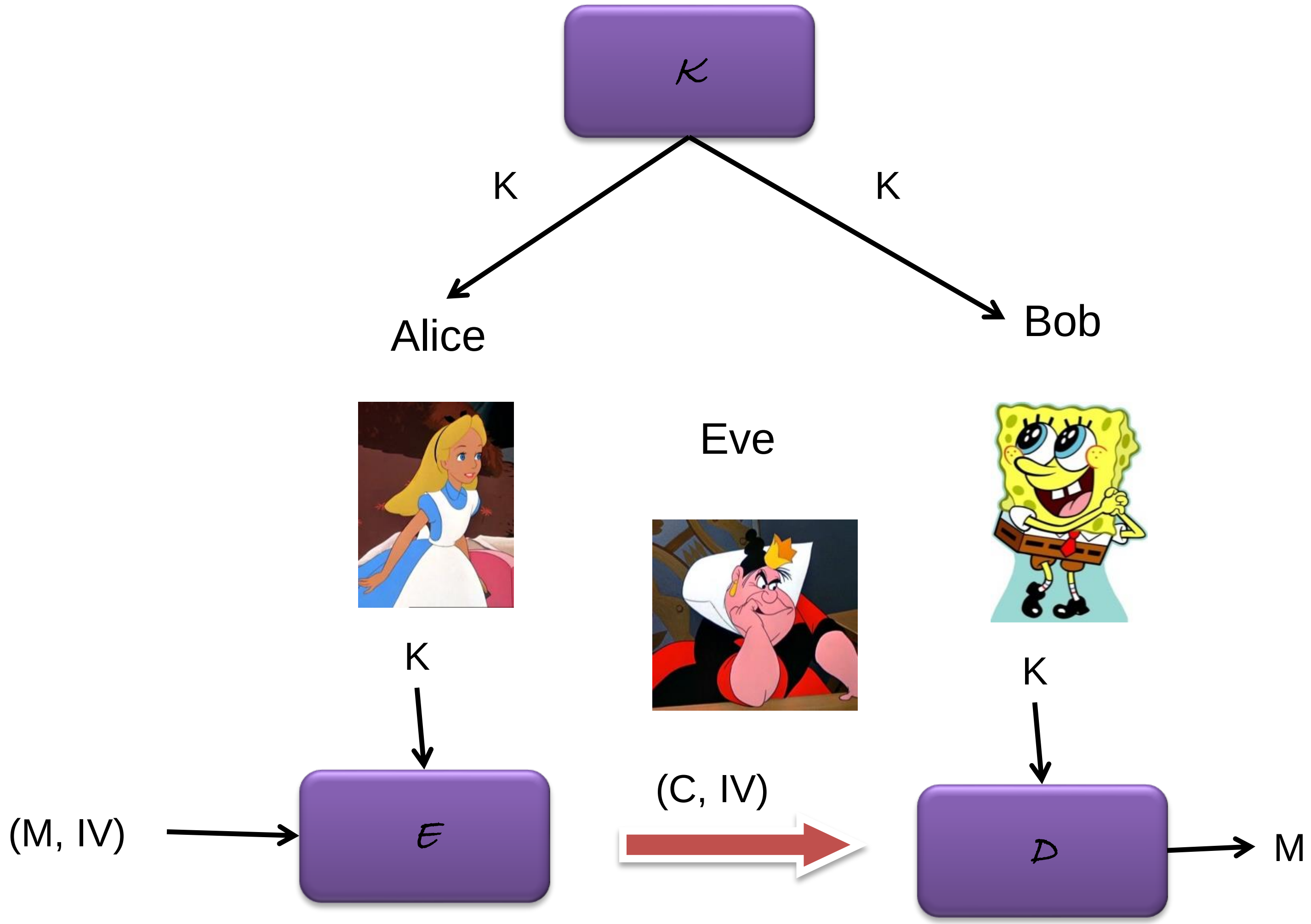
- Symmetric key scheme
  - Bob and Alice share the same secret key

- Security goal: Message/data confidentiality (ONLY)

- Consists of 3 algorithms:

1. Key generation
2. Encryption algorithm
3. Decryption algorithm

Abstract model



Symmetric encryption schemes

Key generation

- It takes no inputs (only the security parameter)
- It flips coins internally and uses these to select a key *K*.
- It is assumed the two entities are in possession of *K*

Encryption algorithm

- Usually non-deterministic
- Randomized
- Stateful (state update, usually a counter)
- Stateless

Decryption Algorithm

- Deterministic
- Correct

Abstract model

- ✓ IV (Initialization Vector) can be
  - ➢ Static (predefined)
  - ➢ Random and new per encryption
  - ➢ Modified using a counter logic (called nonce)
  - ➢ Always publicly known!!!

- ✓ Plaintext space size $|M|$
- ✓ Ciphertext space size $|C|$
- ✓ Key space size $|K|$

- Must be sufficiently big (for a given security parameter)

Attack types and design choices

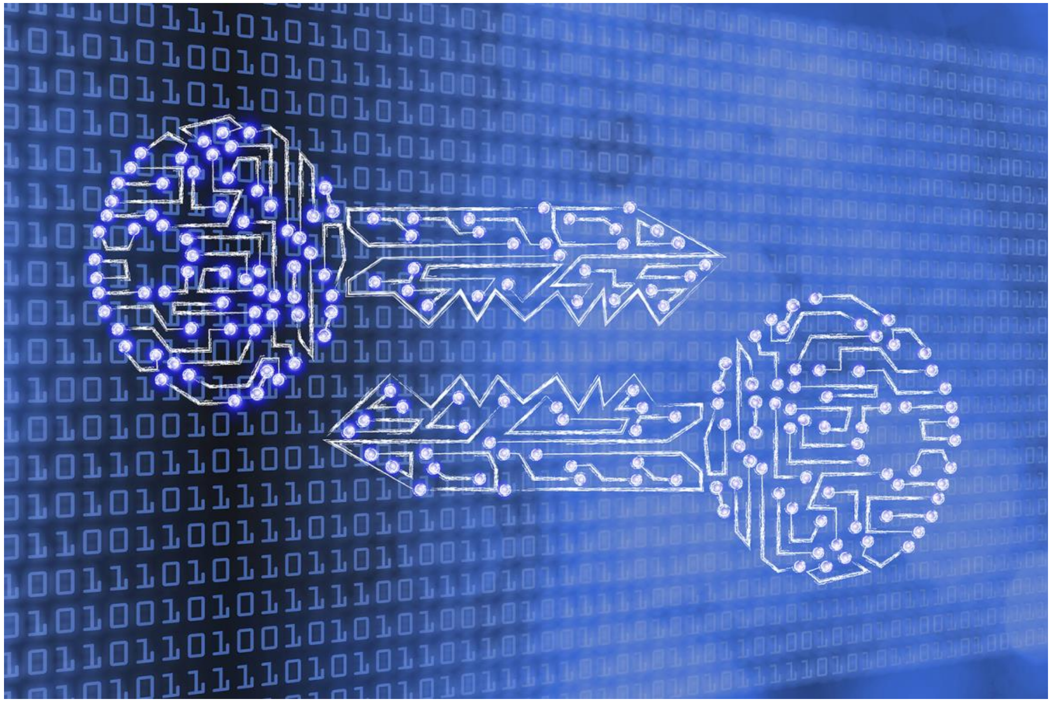
- Recover the secret key
  - Guess the key, i.e. complexity $2^{|K|}$
- Create a valid ciphertext for a given plaintext
  - Guess the ciphertext, i.e. complexity $2^{|C|}$
- Recover the plaintext from the ciphertext
  - Guess the plaintext, i.e. complexity $2^{|M|}$

Types of SES

- Depending on the size of the plaintext we distinguish two main types:

- ❑ Stream ciphers. Traditionally every bit is processed separately

- ❑ Block cipher modes. Encryption per block (64, 128, 256 bits)



SYMMETRIC ENCRYPTION SCHEMES

STREAM CIPHERS

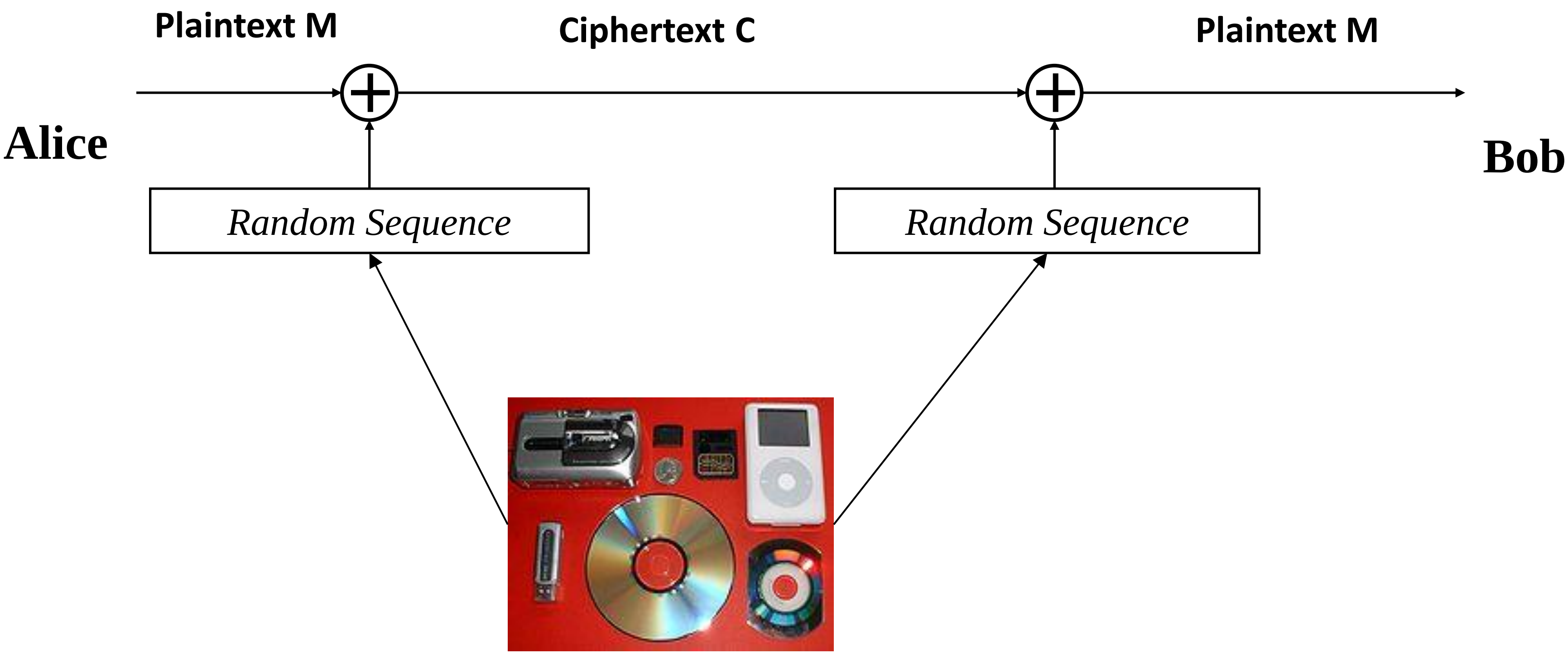
ONE-TIME PAD

- **Known as Vernam cipher (1920)**

Gilbert Sandford Vernam (3 April 1890 – 7 February 1960) was an AT&T Bell Labs engineer

- **Provably secure**
- **Unconditionally secure!!**

ONE-TIME PAD


Plaintext M
Ciphertext C
Plaintext M
Alice
Bob
Random Sequence
Random Sequence

HOW IT WORKS

- **Alice**

M: 0 1 1 0 1 1 1

RS: 1 0 1 1 0 1 0 $\oplus$

C: 1 1 0 1 1 0 1

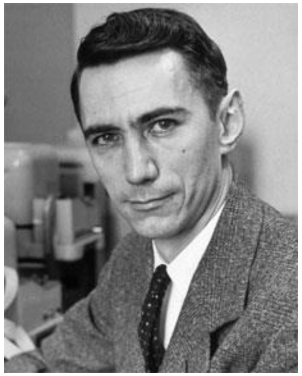
Bob

C: 1 1 0 1 1 0 1

RS: 1 0 1 1 0 1 0 $\oplus$

M: 0 1 1 0 1 1 1

PERFECT SECRECY



- ➢ Claude Shannon has proven that the One time
- pad offers **Information Theoretic Security** or
- **perfect secrecy**.

- ➢ It is unconditionally secure!

- **But…to good to be practical!**
- ➢ Perfect secrecy implies that size of the key must be greater or equal to the message.
- ➢ We can not use more than once the same random sequence (one time pad…). Otherwise, there is an attack…

- A Stream Cipher is the solution

STREAM CIPHERS

- **One time pad is not practical**

How can share all this randomness

- **Solution:**

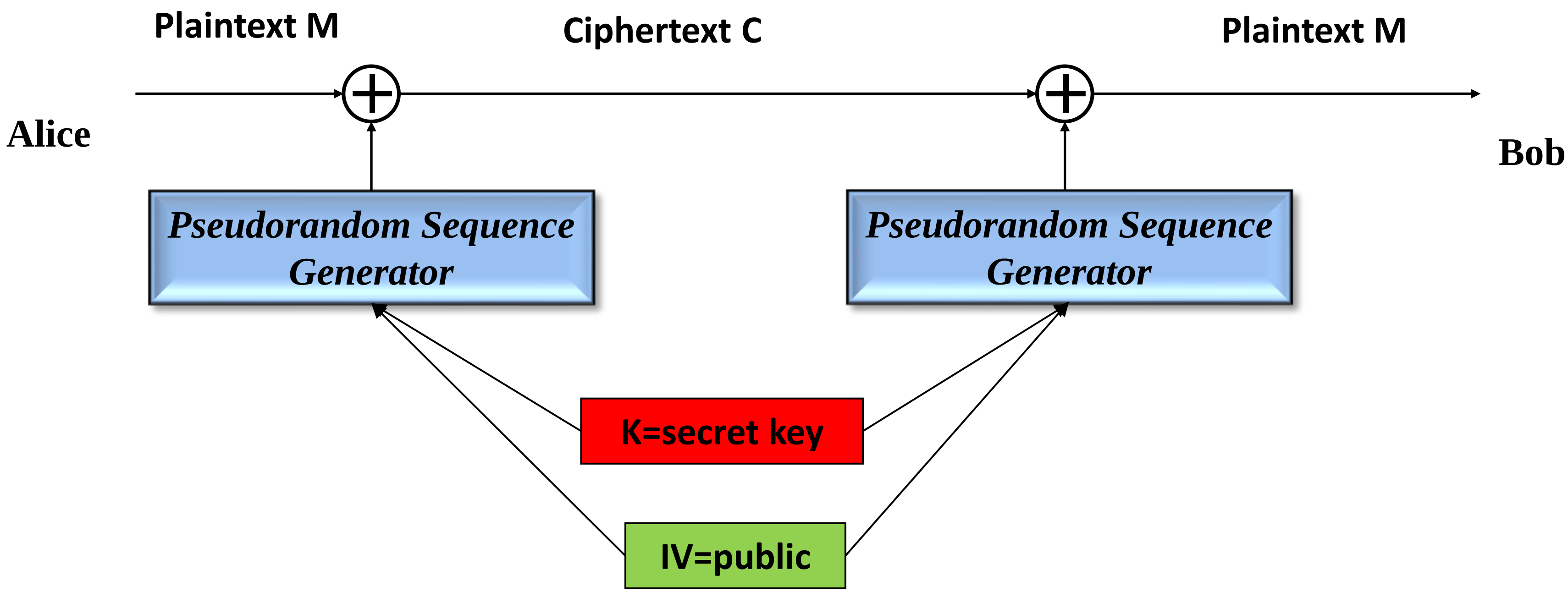
Replace the random source with a pseudorandom bit generator

Part of the seed is kept secret and used as the key!

Remember: ONLY confidentiality!!!

No Integrity protection!!!

STREAM CIPHERS (SYNCHRONOUS)



Types

- Synchronous stream cipher
- ✓ Sender and receiver must be in-synch
- ✓ Lost bit garbles all subsequent bits unless synch up
- ✓ Can precompute key stream

- Self-synchronizing stream ciphers
- ✓ Use n previous ciphertext bits to compute keystream
- ✓ Lost bit: synch up after n bits
- ✓ Can’t precompute keystream

Designs

- Two main types:

1. Ad hoc
2. Provably secure
   - Block cipher based
   - Public key based

- Most famous ad hoc stream ciphers:
- A5/1, A5/2 (GSM)
- E0 (Bluetooth)
- RC4 (SSL/TLS, WEP, Microsoft)
- SNOW (3G)

- Two international competitions (no standardization):
- NESSIE
- ESTREAM

NESSIE Stream Cipher portfolio

- None recommended!!!!

- BMGL – too slow, small internal state – time/memory tradeoff attack
- Leviathan - distinguishing attack
- LILI-128 – attack $O(2^{71})$
- SNOW – distinguishing attack
- SOBER-t16 – distinguishing attack
- SOBER-t32 – distinguishing attack
- Both Sober algorithms thought to be subject to side channel analysis

ECRYPT's eStream Contest

- *ECRYPT: European Network of Excellence for Cryptology*
  - From November 2004 to 2008
- http://www.ecrypt.eu.org/stream/

- Categories

✓ key length of 128 bits and an IV length of 64 and/or 128 bits

✓ key length of 80 bits and an IV length of 32 and/or 64 bits

➢ Separate software and hardware categories

➢ Free of licensing requirements …

➢ Committee was only collecting submissions.

➢ Evaluations were done by the general cryptographic community.

eStream Evaluation

- Security Criteria
- ✓ Any key-recovery attack should be at least as difficult as exhaustive search.
- ✓ Distinguishing attacks
    - ➢ Interest to the cryptographic community
    - ➢ Relative importance of high complexity distinguishing attacks is an issue for wider discussion
- ✓ Clarity of design

- Implementation Criteria
- ✓ Software and hardware efficiency
- ✓ Execution code and memory sizes
- ✓ Performance
- ✓ Flexibility of use

eSTREAM Winners

Profile 1 (SW)	Profile 2 (HW)
HC (HC-128 and HC-256)	**F-FCSR** (F-FCSR-H v2 and F-FCSR-16)
Rabbit	**Grain** (Grain v1 and Grain-128)
Salsa20	**MICKEY** (MICKEY 2.0 and MICKEY-12 2.0)
SOSEMANUK	**Trivium**

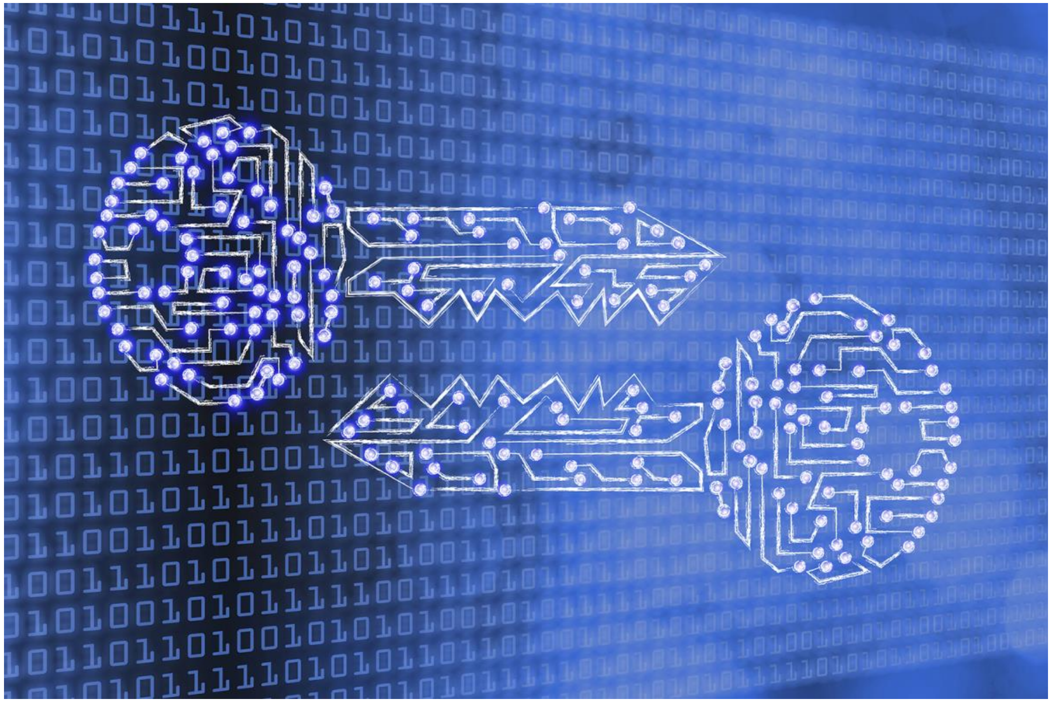
Stream Cipher Summary

Primitive	Classification Legacy	Classification Future
HC-128	✓	✓
Salsa20/20	✓	✓
ChaCha	✓	✓
SNOW 2.0	✓	✓
SNOW 3G	✓	✓
SOSEMANUK	✓	✓
Grain	✓	✗
Mickey 2.0	✓	✗
Trivium	✓	✗
Rabbit	✓	✗
A5/1	✗	✗
A5/2	✗	✗
E0	✗	✗
RC4	✗	✗

- ***Algorithms, key size and parameters report. ENISA– Nov. 2014***

More details

Cipher	Key size	IV size	
HC-128	128-bits key	128-bits	eSTREAM SW
Salsa20/20 and ChaCha	128-bits key-256 bits	128-bits	eSTREAM SW web browser Chrome
SNOW 2.0	128-bits key-256 bits	128-bits	ISO/IEC 18033-4
SNOW 3G	128-bits key	128-bits	core of the algorithms UEA2 and UIA2 of the 3GPP UMTS system (128-EIA1 and 128-EEA1 in LTE).
SOSEMANUK	128-bits key-256 bits	128-bits	eSTREAM SW



SYMMETRIC ENCRYPTION SCHEMES

BLOCK CIPHER MODES

Modes of operation

- How do I use a block cipher for encryption (confidentiality)
- There are several modes of operation
  - Electronic Codebook (ECB)
  - Cipher-block Chaining (CBC)
  - Cipher Feedback (CFB)
  - Output Feedback (OFB)
  - Counter mode (CTR)
  - XEX Tweakable Block Cipher with Ciphertext Stealing (XTS)
  - ECB-mask-ECB (EME)

- Of special interest authenticated encryption modes (next session)

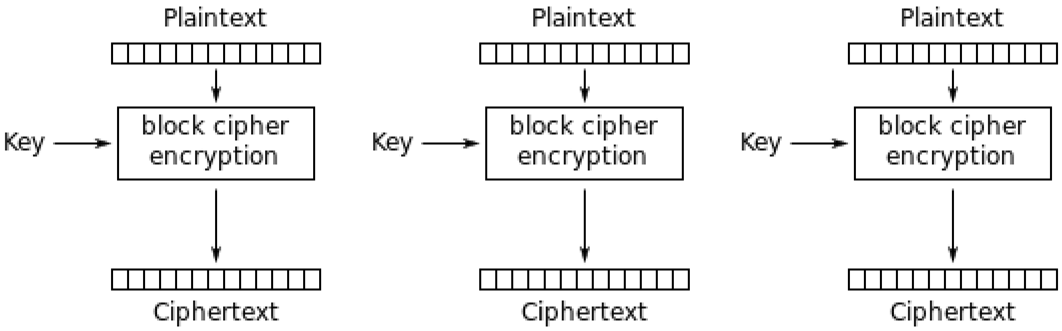
Modes of operation

- How do I use a block cipher for encryption (confidentiality)
- There are several modes of operation
  - **Electronic Codebook (ECB)**
  - **Cipher-block Chaining (CBC)**
  - Cipher Feedback (CFB)
  - Output Feedback (OFB)
  - **Counter mode (CTR)**
  - XEX Tweakable Block Cipher with Ciphertext Stealing (XTS)
    - IEEE P1619 Standard and NIST SP 800-38E Recommendation
  - ECB-mask-ECB (EME)

- Of special interest authenticated encryption modes (next session)

ECB Mode

- Encryption



Electronic Codebook (ECB) mode encryption

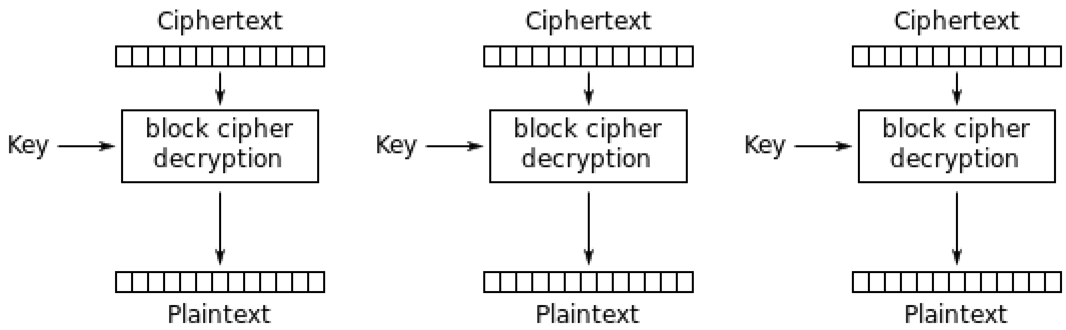
•**Identical plaintext blocks produce identical ciphertext block:** pattern detection

•Patterns not likely in normal text – newspaper, book – due to need to align on block boundary

•Patterns likely in structured text – log files

ECB Mode

- Decryption



Electronic Codebook (ECB) mode decryption

•**Identical plaintext blocks produce identical ciphertext block:** pattern detection
•Patterns not likely in normal text – newspaper, book – due to need to align on block boundary
•Patterns likely in structured text – log files

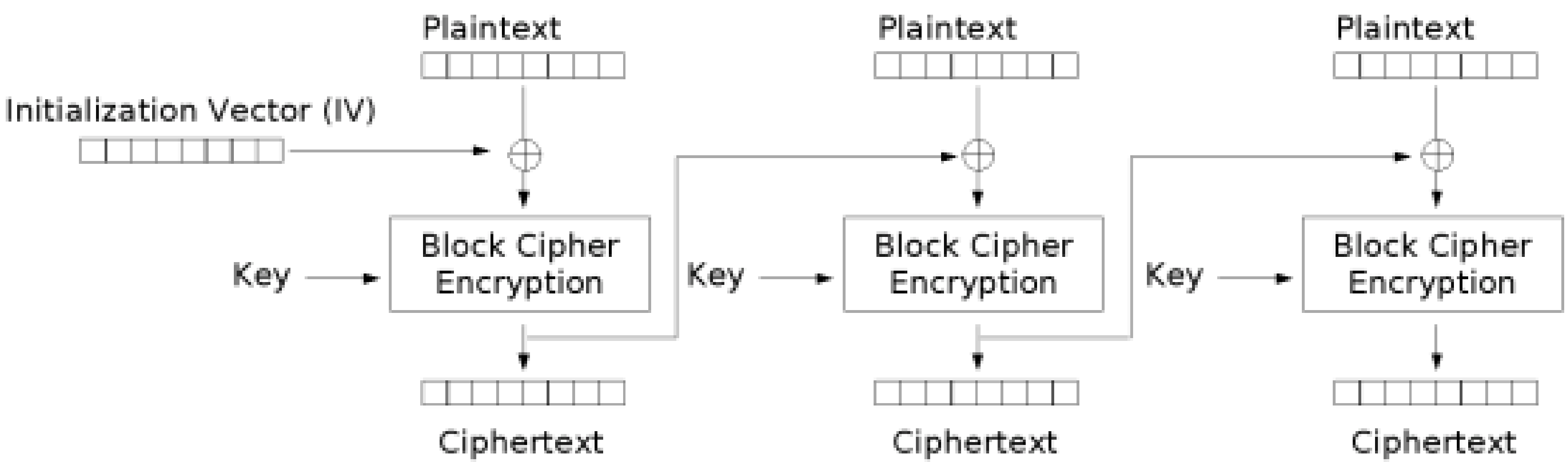
ECB Mode

- Identical plaintext blocks produce identical ciphertext block
- pattern detection

- Generally not secure
- Should be used with care.
    - only to encrypt messages with length at most that of the underlying block size,
    - only for keys which are used in a one-time manner

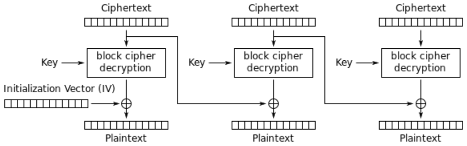
CBC Mode

Encryption



CBC Mode

Decryption



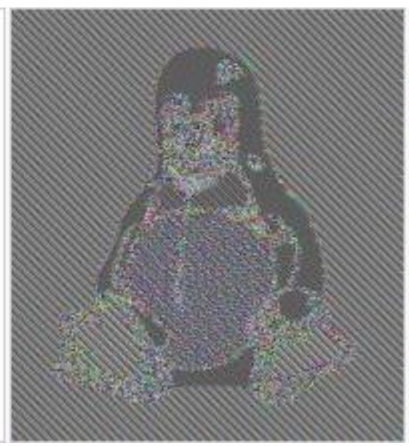
Cipher Block Chaining (CBC) mode decryption

CBC Mode

- the most widely used mode of operation
- an independent and random IV *must* be used for each message
- With a non-random or predictable IV, CBC mode is insecure
- Cannot perform parallel processing

ECB insecurity

Original Image

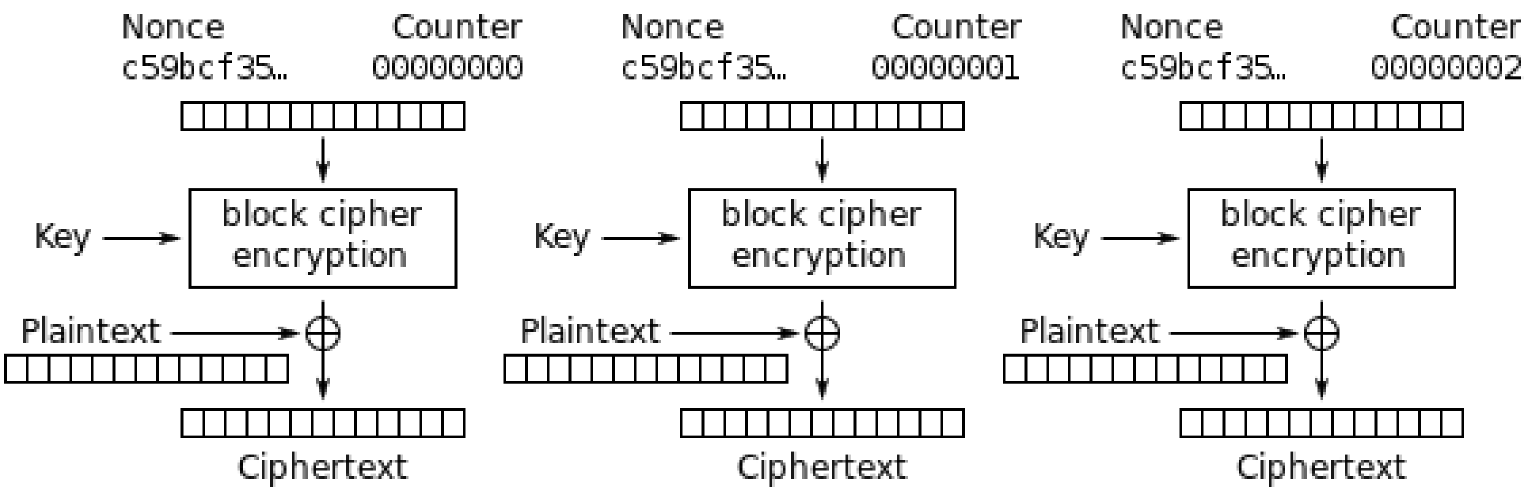


ECB encrypted

CBC encrypted

CTR Mode

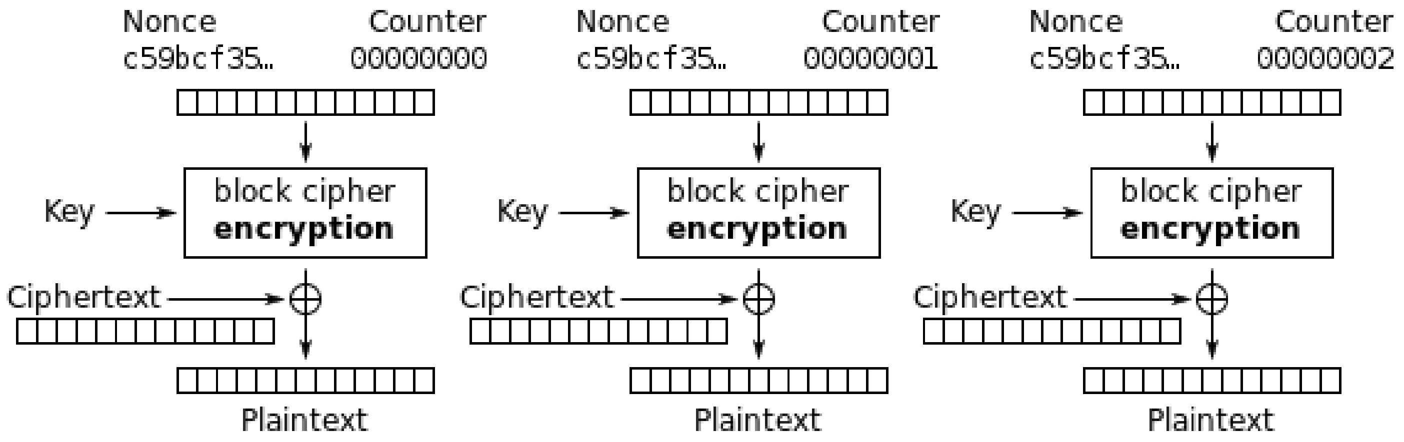
Encryption



Counter (CTR) mode encryption

CTR Mode

- Decryption



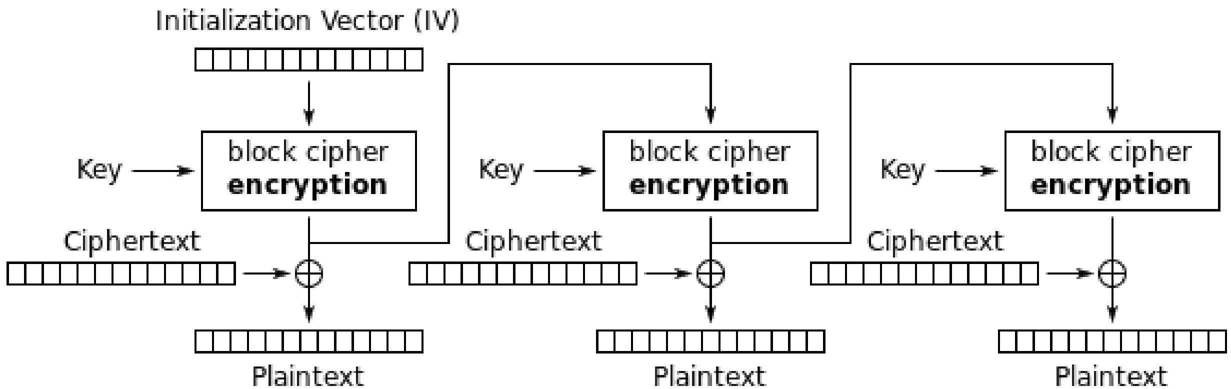
Counter (CTR) mode decryption

CTR Mode

- It is a stream cipher
- Both encryption & decryption are parallelizable
- Identical messages: changing nonce results in different ciphertext
- No chaining dependencies
- No padding is needed
- Counter (IV) should be nonce. Must not repeated (one time pad…)

OFB Mode

Encryption



Output Feedback (OFB) mode decryption

- Stream Cipher
- IV must be random (if nonce, then insecure)

Standard

- NIST Special Publication 800-38A Recommendation for Block, 2001 Edition
- "Recommendation for Block Cipher Modes of Operation Methods and Techniques"

- http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf


Questions?