

>

DI-FCT-UNL

Segurança de Redes e Sistemas de Computadores  
*Network and Computer Systems Security*

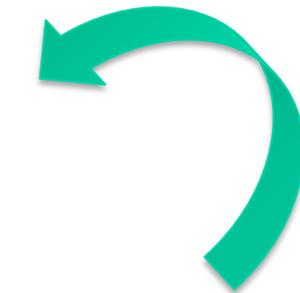
Mestrado Integrado em Engenharia Informática  
MSc Course: Informatics Engineering  
1<sup>st</sup> sem, 2019/2020

# Symmetric Cryptography

# X.800 Framework:

## Remember: Mappings for Symmetric Encryption

	Release of message contents	Traffic analysis	Masquerade	Replay	Modification of messages	Denial of service
Peer entity authentication			Y			
Data origin authentication			Y			
Access control			Y			
Confidentiality	Y					
Traffic flow confidentiality		Y				
Data integrity				Y	Y	
Non-repudiation			Y			
Availability						Y

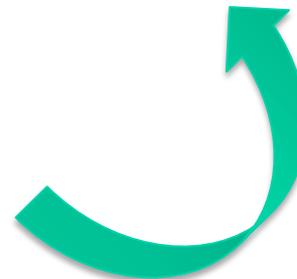


	Release of message contents	Traffic analysis	Masquerade	Replay	Modification of messages	Denial of service
Encipherment	Y					
Digital signature			Y	Y	Y	
Access control	Y	Y	Y	Y		Y
Data integrity				Y	Y	
Authentication exchange	Y		Y	Y		Y
Traffic padding		Y				
Routing control	Y	Y				Y
Notarization			Y	Y	Y	

Cryptography methods, Algorithms, models, techniques

**Symmetric Crypto: for Confidentiality**

Service	Mechanism							
	Encipherment	Digital signature	Access control	Data integrity	Authentication exchange	Traffic padding	Routing control	Notarization
Peer entity authentication	Y	Y			Y			
Data origin authentication	Y	Y						
Access control			Y					
Confidentiality	Y						Y	
Traffic flow confidentiality	Y					Y	Y	
Data integrity	Y	Y		Y				
Non-repudiation		Y		Y				Y
Availability				Y	Y			



# Symmetric Cryptography Scope (ref. w/ OSI X.800 Framework)

- Also known as: conventional encryption or shared-key cryptography or symmetric encryption

- **Primary use:** for Confidentiality (Encryption)

**Target: Data, Data-Storage or Message Confidentiality**

**(Security Properties for Secure Channels, - rem. X.800 Framework)**

- Connection-Oriented Confidentiality
  - Connectionless or Datagram Oriented Confidentiality
  - Selective Field Confidentiality
  - Traffic-Flow Confidentiality
- 
- ... **But also as a combined mechanism for:**
    - Peer-Authentication Services (given shared secrets, keys)
    - Data-Origin Authentication Services (using CMACs or Cryptographic Authentication Codes, based on shared secrets, keys)
    - Access-control Services (password/secret-key protection)
    - Enforced Traffic-flow confidentiality (ex., tunneling, ...)
    - Data-integrity Services (ex., as CMAC proofs)

# Outline

- **Symmetric cryptography**
  - Symmetric Encryption
    - Block Encryption Algorithms
  - Robustness of symmetric encryption algorithms
  - Cipher block modes of operation
  - Relevance of Padding
  - Stream Encryption Algorithms
  - Bit-Stream Ciphers using Block Ciphers

# Outline

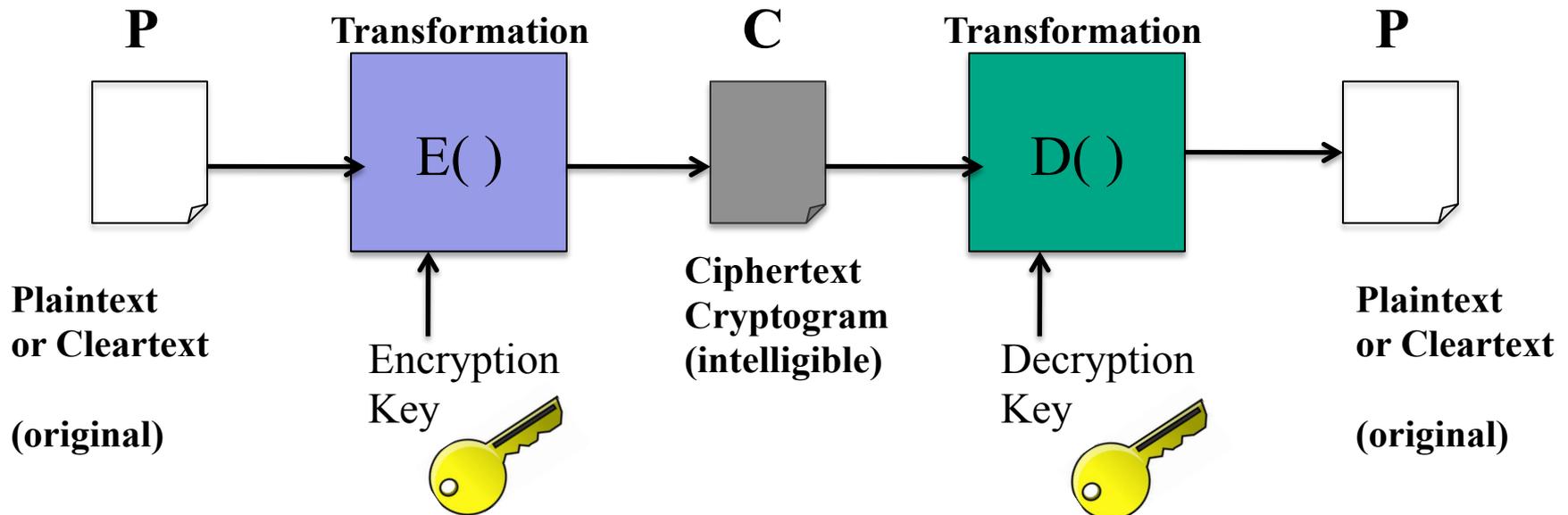
- **Symmetric cryptography**
  - Symmetric Encryption
    - Block Encryption Algorithms
  - Robustness of symmetric encryption algorithms
  - Cipher block modes of operation
  - Relevance of Padding
  - Stream Encryption Algorithms
  - Bit-Stream Ciphers using Block Ciphers

# Symmetric cyphers:

## Block Ciphers vs. Stream Ciphers

- **Block-Oriented (or block ciphers)**
  - Used (parameterized) with different **block modes of operation**, possible **Initialization vectors** and **padding processing** (as security association parameters)
  - Key sizes and block sizes defined (fixed) for each algorithm (you must know it ...)
    - ... Characteristics for each algorithm
- **Stream-oriented (or stream ciphers)**
  - Byte-stream-oriented or bit-stream-oriented operation
  - Variable key-sizes (algorithm dependent)
  - Fast to operate on stream-oriented inputs, ex., Real-Time Processing (bytes, bits)
    - Ex., real-time bit-streaming, iterative-traffic and/or low-latency communication requirements
  - Security issue: the security and period of the keystream generation

# Symmetric Encryption: Block Ciphers

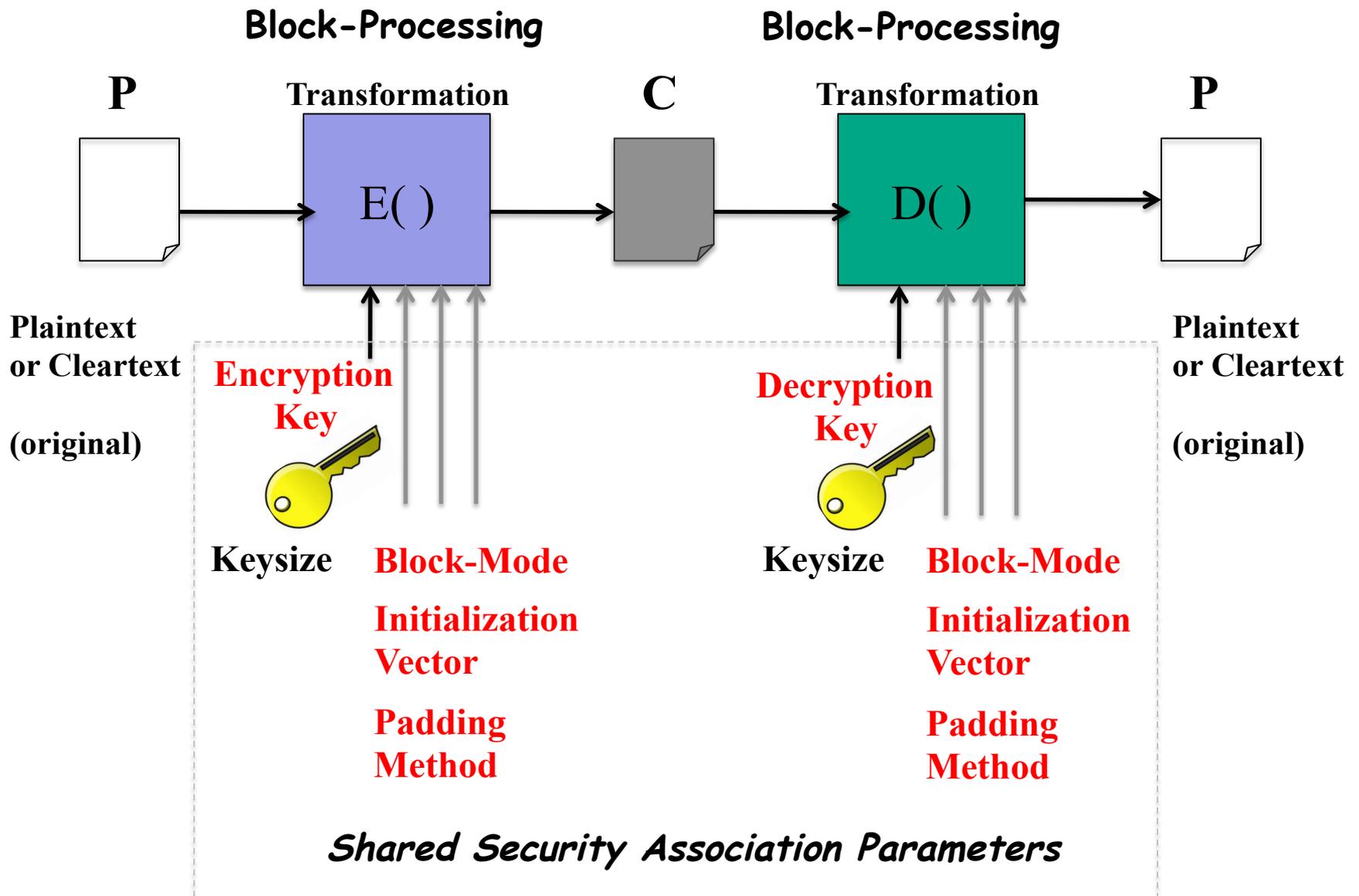


Notation:

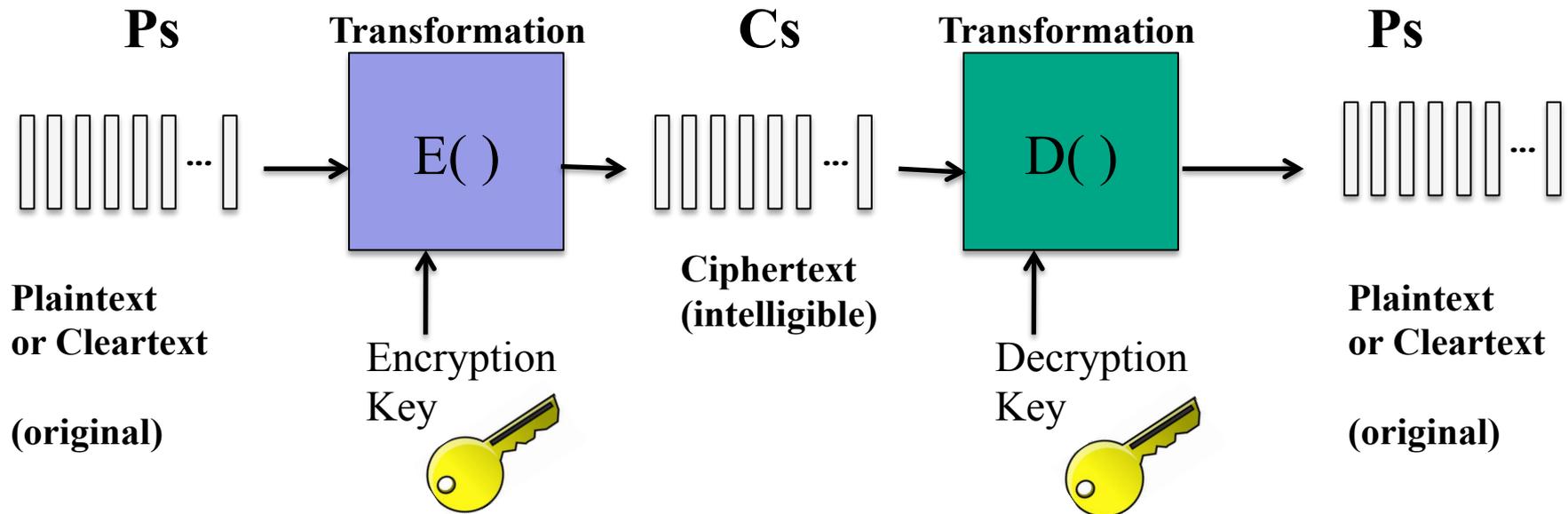
$C = \{P\}_K$  ;  $C = E(P, K)$  ;  $C = E_k(P)$  // M encrypted with key K

$P = \{C\}'_K$  ;  $P = D(C, K)$  ;  $P = D_k(C)$  // C decrypted with key K

# Symmetric Block Encryption and related Security Association Parameters



# Symmetric Encryption: Stream Ciphers



- Use for stream-encryption, ex., bit-streaming
- Interesting: real-time bit streaming (ex., radio-frequency communication)

# Requirements for security when using symmetric cryptography

- **Robustness** (security and correctness of the symmetric encryption algorithms on their trust-execution criteria)
  - Resistance against brute-force attacks and cryptanalysis attacks
- Need **security association parameters** for the intended purpose
- Need of **strong keys**: generated w/ randomness, distributed and maintained with security guarantees
  - TRNGs, PRNGs, PRFs for Key-Generation and other parameters
  - Ex., possible use of HSMs, Smartcards, Crypto-Tokens
- Need of **secure key-distribution and establishment services** (shared keys and related security association parameters)
- **Minimization or Avoidance of key-exposure**
- **Fast and secure "rekeying"** services with perfect future and past secrecy, with key-independence
  - Ex., Rekeying for temporary session keys, or keys used with OTP assumptions

# Symmetric Encryption Algorithms

## Symmetric Encryption (shared key cryptographic method)

### Two main families of Symmetric Encryption Algorithms:

- **Symmetric Block Encryption** : Block-Oriented Processing
  - Ex., DES, 3DES, RC5, Blowfish, Skipjack, Katsumi, ...  
Rijndael, AES, RC6, Twofish, MARS, Serpent, ...
- **Symmetric Stream Encryption**: Stream-Oriented Processing
  - Ex., ARC4, RC4, A5, A8, SEAL, WAKE, SALSA20, CryptMT, ...

See available algorithms in crypto provider libraries  
ex., JAVA/JCA-JCE, Remember Lab Exercises (lab 1)

---

[https://en.wikipedia.org/wiki/Advanced\\_Encryption\\_Standard\\_process](https://en.wikipedia.org/wiki/Advanced_Encryption_Standard_process)  
[https://en.wikipedia.org/wiki/Symmetric-key\\_algorithm](https://en.wikipedia.org/wiki/Symmetric-key_algorithm)

# Some Symmetric Block Algorithms

Alg.	Key size Block size	Rounds	Structure Operations	Ex. Applications (standardization)
DES	56 /64	16	XOR, S-boxes, P-Boxes EP Boxes, Rotations, ...	SET, Kerberos
3DES	112,168 /64	48	idem	Financial standards E-commerce, PGP S/MIME, SSL
IDEA	128 /128	8	XOR, addition mod $2^{16}$ , Multiplication mod $2^{16}$	PGP, SSL, TLS
TEA	128 /128	32	XOR + shift	
Blowfish	< 448 /128	16	XOR + Variable S-boxes	
RC5	< 2048 /128	< 255	Addition, subtraction XOR and rotation operations	SSL
Cast-128	40,128 /128	16	Addition, Subtraction, XOR, Rotation and S-boxes	PGP
AES	128, 192, 256 /128	10,12,14	S boxes, Matrix SR-MC, XOR	Many: "The Standard" after 2000 TLS, IPsec, etc ...



# Block Ciphers and Internal Structure

- A generic structure for block-encryption algorithms: a sequence of block transformation rounds (w/ the core algorithm) performing transformations, successively parameterized by secret subkeys derived from the initial key
  - Subkeys: derived for each new round from a key-scheduling algorithm
- For a particular instantiation (specific algorithm), we must define:
  - The blocksize,
  - The keysize
  - Sub-key generation (scheduling) algorithm
  - The core round function (and related transformations)
  - Number of rounds for the required confusion and diffusion (entropy), to materialize the effect of a "random-oracle" if the key is not known

## Taking into account:

- Fast encryption/decryption
- SW and HW implementation possibility
- Core round and sub-key generation algorithm (ease of analysis for cryptanalysis)
- Complexity analysis and soundness proofs

# Characteristics in symmetric block cyphers

## Security

- Security assessment criteria (brute-force and cryptanalysis)
- Confusion and Diffusion Criteria or Avalanche Effect
  - No distinguishability from "random" processing (not knowing the key)
  - Soundness of statistical studies and/or mathematical (arithmetical) proofs
- Requires complexity (complex transformations in each round): large number of rounds, large keysize, large blocks will be better

## But we must have security and fast operation (performance)

- Speed of the algorithm is a concern
- Simplicity (ex., HW implementations and embedded solutions, fast computation, low computational cost, energy-savings, ...)

## And Analysis and Verification: Ease for analysis and verification

- Simplicity in the internal reference structure
- "analyzable": formalisms and mathematical foundations, as well as, implementation correction

---

See for example the NIST requirements for the AES competition: ex.,  
<https://competitions.cr.yp.to/aes.html>

# Outline

- **Symmetric cryptography**
  - Symmetric Encryption
    - Block Encryption Algorithms
  - Robustness of symmetric encryption algorithms
  - Cipher block modes of operation
  - Relevance of Padding
  - Stream Encryption Algorithms
  - Bit-Stream Ciphers using Block Ciphers

# Security of Symmetric Block Ciphers

- **Brute force Attacks**

- Try all the possible combinations of:
  - Keys
  - Block Sizes
  - Encryption: Plain text Blocks > Found Ciphertext Blocks
  - Decryption: Ciphertext Blocks > Found Plaintext Blocks

- **Cryptanalysis Attacks (or Studies)**

- Find security vulnerabilities in the internal (core) structure of each algorithm
- Cryptanalysis types: Cryptanalyst (pr Attacker) perspective

<ul style="list-style-type: none"><li>• Ciphertext only</li><li>• Known plaintext</li><li>• Chosen plaintext</li><li>• Chosen ciphertext</li><li>• Chosen text</li></ul>	More difficult  Less difficult ? But More relevant ! Why ?
--	---

# Brute Force Crypto Attacks

**Brute Force Attack** (*Ataque por força bruta*): breaking the key by trying every possible keys (knowing the key size) to obtain the plaintext (on average, for  $N$  possible keys, try  $N/2$  keys, with probability = 0.5...  $N/3$ : 0,33,  $N/10$ : 0.1, ...  $N/10^6$ :  $10^{-6}$ )

- Rational for Brute Force: if there are no inherent mathematical weaknesses or processing vulnerabilities, a brute force attack is always possible
- Knowledge of the Algorithm
- Success only dependent from:
  - Key size and quality of (pseudo) random-based key generation (TRGs vs. PRGs)
  - Size of blocks
  - Processing time to complete (related with computational cost of the encryption/decryption functions) and the available processing power

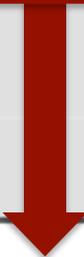
# Big numbers [Schneier 1996]

Bruce Schneier, Applied Cryptography, Wiley, 1996

<b>Physical Analogue</b>	<b>Number</b>
Odds of being killed by lightning (per day)	1 in 9 billion ( $2^{33}$ )
Odds of winning the top prize in a U.S. state lottery	1 in 4,000,000 ( $2^{22}$ )
Odds of winning the top prize in a U.S. state lottery and being killed by lightning in the same day	1 in $2^{55}$
Odds of drowning (in the U.S. per year)	1 in 59,000 ( $2^{16}$ )
Odds of being killed in an automobile accident(in the U.S. in 1993)	1 in 6100 ( $2^{13}$ )
Odds of being killed in an automobile accident(in the U.S. per lifetime)	1 in 88 ( $2^7$ )
Time until the next ice age	14,000 ( $2^{14}$ ) years
Time until the sun goes nova	$10^9$ ( $2^{30}$ ) years
Age of the planet	$10^9$ ( $2^{30}$ ) years
Age of the Universe	$10^{10}$ ( $2^{34}$ ) years
Number of atoms in the planet	$10^{51}$ ( $2^{170}$ )
Number of atoms in the sun	$10^{57}$ ( $2^{190}$ )
Number of atoms in the galaxy	$10^{67}$ ( $2^{223}$ )
Number of atoms in the Universe (dark matter excluded)	$10^{77}$ ( $2^{265}$ )
Volume of the Universe	$10^{84}$ ( $2^{280}$ ) $\text{cm}^3$
<b>If the Universe is Closed:</b>	
Total lifetime of the Universe	$10^{11}$ ( $2^{37}$ ) years $10^{18}$ ( $2^{61}$ ) seconds
<b>If the Universe is Open:</b>	
Time until low-mass stars cool off	$10^{14}$ ( $2^{47}$ ) years
Time until planets detach from stars	$10^{15}$ ( $2^{50}$ ) years
Time until stars detach from galaxies	$10^{19}$ ( $2^{64}$ ) years
Time until orbits decay by gravitational radiation	$10^{20}$ ( $2^{67}$ ) years
Time until black holes decay by the Hawking process	$10^{64}$ ( $2^{213}$ ) years
Time until all matter is liquid at zero temperature	$10^{65}$ ( $2^{216}$ ) years
Time until all matter decays to iron	$10^{10^{26}}$ years
Time until all matter collapses to black holes	$10^{10^{76}}$ years

# Exhaustive brute-force key-search

Key Size (bits)	Number of Alternative Keys	Time required at 1 encryption/ $\mu$ s	Time required at $10^6$ encryptions/ $\mu$ s
32	$2^{32} = 4.3 \times 10^9$	$2^{31} \mu s = 35.8$ minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55} \mu s = 1142$ years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127} \mu s = 5.4 \times 10^{24}$ years	$5.4 \times 10^{18}$ years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167} \mu s = 5.9 \times 10^{36}$ years	$5.9 \times 10^{30}$ years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26} \mu s = 6.4 \times 10^{12}$ years	$6.4 \times 10^6$ years



## Important factors:

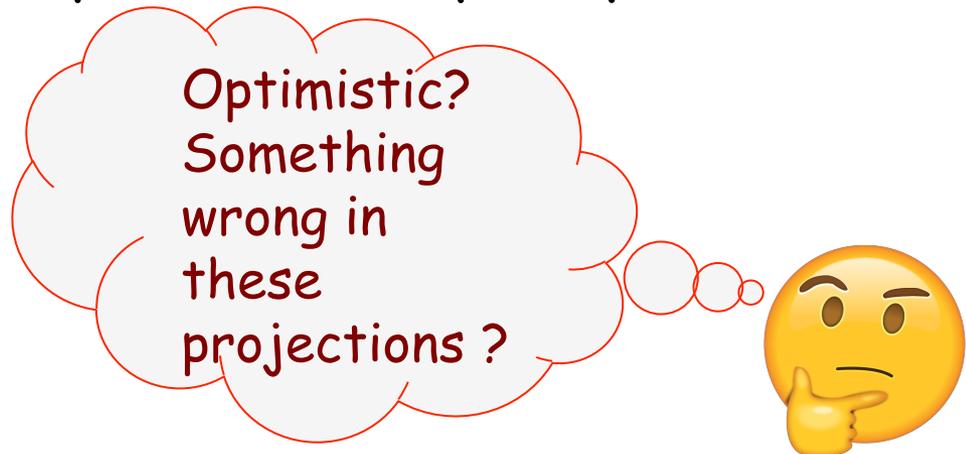
- Expectable computational effort and cost
- Key sizes (and also the block sizes)
- Degree of "plaintext intelligibility"
  - More difficult in some cases: ex., binary data, pre-processed plain txt transformation, compression, hashed-data, hashed data with mixed "secrets", etc...

# Exhaustive brute-force key-search

Key Size (bits)	Number of Alternative Keys	Time req. for $10^9$ decryptions/us	Time required at $10^{13}$ decryptions/us
32	$2^{32} = 4.3 \times 10^9$	$2^{31}$ ns , 3 minutes	~2 us
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55}$ ns , 1.125 years	1 hour
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127}$ ns , $5.3 \times 10^{21}$ years	$5.3 \times 10^{17}$ years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167}$ ns , $5.8 \times 10^{33}$ years	$5.8 \times 10^{29}$ years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26}$ ns , $6.3 \times 10^9$ years	$6.3 \times 10^6$ years
192	$2^{192} = 6,3 \times 10^{37}$	$2^{191}$ ns , $9.8 \times 10^{40}$ years	$9.8 \times 10^{36}$ years
256	$2^{256} = 1,2 \times 10^{77}$	$2^{255}$ ns , $1.8 \times 10^{60}$ years	$1.8 \times 10^{56}$ years

- What if we improve the computational capacity:
  - Ex.,  $10^9$  us or ...  $10^{13}$  us

Uhm ... How many Encryptions/Decryptions can you Do in your computer ? (Demo in Class)



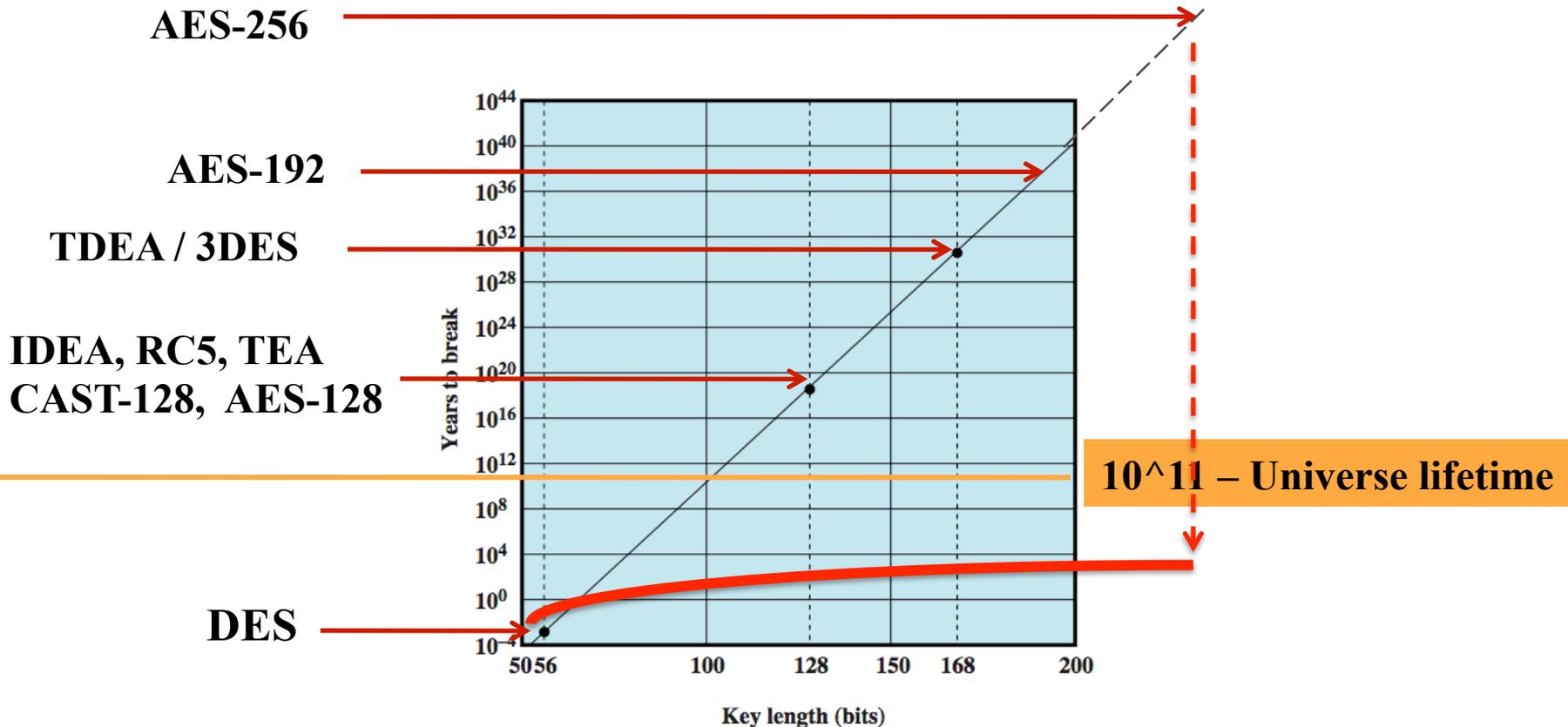
# ... Moore's Law and other predictions

[http://en.wikipedia.org/wiki/Moore's\\_law](http://en.wikipedia.org/wiki/Moore's_law)

- Processing power doubles, each 18 months (will continue ?)
- In 10 years, computers will be 100 times more powerful (in computational resources).
  - A desktop fits now into a smart phone ... next in a IoT device ?
  - Storage Capacity
  - Gigabit wireless connectivity everywhere
  - Global computation power: Networks connect our computing devices in the context of collective/global/shareable computing environments and datacenters ...
- Other aspects of the future (more difficult to predict)
  - Anyone can not predict what the emergent properties of "100 or 1000x computing power" will bring: new uses for computing, new paradigms of communication.
  - A "100x-1000x ... world" will be different, in ways that will be surprising, with possible disruptions (we don't know yet)

# Average time required for a brute-force attack

Examples: DES, IDEA, RC5, TEA, CAST-128, AES-128, AES-192, AES-256



- Time-to-Break, with computational power of  $10^6$  decryptions/ $\mu$ s
- 7 x faster than EFF announced in 1998:  
DES breaking experience in 3 days with USD\$250,000

# More brute-force attack estimations

- Moore Law (computing power):
  - double in each 18 months ...
- Cost estimation to break

Extrapolation using the Moore law

Key size	Year (attack)
80	2010
96	2034
112	2058
<b>128</b>	<b>2082</b>
144	2106
160	2130
176	2154
<b>192</b>	<b>2178</b>
208	2202
224	2226
240	2250
<b>256</b>	<b>2274</b>

Ex., Breaking with \$10 Millions in 2000:

Symmetric Alg. (key sizes)	Public-Key (ECC)	Public Key (RSA)	<i>Time to Break</i>	Storage needs
56	112	420	< 5 min	Trivial
80	160	760	600 months	4GB
96	192	1020	3x10 <sup>6</sup> years	170GB
128	256	1620	10 <sup>16</sup> years	120TB

*Robert Silverman, Technical Report RSA Laboratories, 2000*

# Two important considerations: Unconditionally vs. Computationally Security

- Unconditionally secure: generated cipher text does not contain enough information to determine uniquely the correspondent plaintext
  - Time vs. Space Independent Notion
  - Only theoretically achieved by OTP schemes
- Computationally security: security with two different evaluation criteria:
  - **Breaking cost** > value of encrypted information
    - Cost metrics: money, required resources
  - **Breaking time**: time required to break vs. useful lifetime of the encrypted information
    - Can we approach with “practical infinite time considerations” ?  
How ?
  - **Physics to break** (or possible physics impossibilities)
    - Can we have the required matter (ex., atoms) in the universe ?

# Robustness and computational security criteria in the study of cryptography

- **Robustness:** if it is impossible (**computationally impossible**) to decrypt a message on the basis of the ciphertext plus the knowledge of the encryption/decryption algorithm
- **Computational security criteria:**
  - **Infinite time to break with available computing power and resources (processing power MIPS, memory, storage,...)**
    - what is "infinite time" ?
      - Ex., More than the universe lifetime ...  $10^{11}$  years ?
      - Ex., More than the universe time in several orders of magnitude ?
      - **Finite time to break** ... but time exceeds the useful lifetime of the protected information
  - **Finite time to break but requires infinite or impossible computing power and information resources**
    - what is "impossible computing power" ?
      - Ex., A quantity of computers requiring more silicon or germanium atoms than the total quantity of atoms (estimated) available in our galaxy ?
      - Ex., Processing speed required will cause the fusion of the materials ?
      - **Available computing can break before the useful lifetime of the protected information** ... but the cost (money) of such computing resources exceeds the value of the protected information

# Brute-Force Attacks, Conclusions and Possible Enforcements w/ Symmetric Methods

- Even if managed to speed up by a factor of 1 trillion ( $10^{12}$ ) Brute force attacks on 128 bit keys require 1 million years
- So, in practice, today, 128 bit keys (generated randomly, discarding possible weak keys, keys securely established and maintained) and using rekeying strategies (for temporary session-keys) ... can be "unbreakable" in practice with brute-force attacks ...
- Can also use enforcements using Multiple-Encryption in Onion Encryption Constructions
  - Chains with the Same Encryption Algorithm, using different keys
  - An initial reference os Triple DES (or DES EDE)
    - Encryption - Decryption - Encryption
    - Double Key (112 bits) or Triple Key (168 Bits)
  - Improved security, Less performance ... Why ?

# Cryptanalysis Methods and Criteria

Some Cryptanalysis Criteria and other Practical Observations:

- **Coincidence tests:** Percentage of Similar Ciphertext Data in Overlapped Identical Cryptograms with Progressive Displacement
- **Kasisky tests:** distance measurements between identical blocks
- **Linear Cryptanalysis** (analysis of ciphertext observations related to input frequency)
- **Differential Cryptanalysis:** analysis of differences in output caused by differences in input to analyze "non-random behavior"
- **Related-Key Attacks** (Sub-Key Scheduling Generation Analysis)
- **Avalanche effects**
- **Other attacks:** Side-Channel Attacks, Timing Attacks, Power-Monitoring Attacks, Oracle-Attacks

# Cryptanalysis Methods and Criteria:

All consider the knowledge of the Algorithm

## Cryptanalysis Referential

### 1. Ciphertext-Only Hypothesis (Random-Oracle)

### 2. Known Plaintext Hypothesis

Known: Ciphertext and Some Plaintext-Ciphertext pairs

Plaintext not chosen by the cryptanalyst

### 3. Chosen Plaintext Hypothesis

Known: Ciphertext and Plaintext-Ciphertext pairs

Input: plaintext chosen or selected by the cryptanalyst

### 4. Chosen Ciphertext hypothesis

Known: Ciphertext and Plaintext-Ciphertext pairs

Input: Ciphertext chosen or selected by the cryptanalyst

### 5. Chosen Text (combining 1, 3, 4)

# Outline

- **Symmetric cryptography**
  - Symmetric Encryption
    - Block Encryption Algorithms
  - Robustness of symmetric encryption algorithms
  - Internal structure of symmetric algorithms
  - Cipher block modes of operation
  - Relevance of Padding
  - Stream Encryption Algorithms
  - Bit-Stream Ciphers using Block Ciphers

# Internal structure of symmetric block ciphers

## Characterized by:

- **Block size** (larger means more security)
- **Key size** (larger means more security)
- Possible initial block transformation (initial block)
- **Round transformation function** (greater complexity means more resistance)
- Sub-key generation from the key, used to parameterize the transformation in each round
- **Number of rounds** (multiple rounds increase security)
- Possible final block transformation (to obtain the ciphertext block)

# Internal Structure and Typical Core-Processing of Symmetric Algorithms

- Permutations, Substitutions, Expansion/Reductions, Shifts/Rotations, in successive steps of blocks' processing (called rounds), to achieve the necessary dispersion-entropy and the required avalanche effect
  - A sub-key derived from the previous one (starting from the input key) is used as a parameter of the successive block-transformations
- Boolean Operations: XOR
  
- Algebraic structures in  $GF_2$ : Modular Arithmetic (Modular Additions and Multiplications)
  - Matrix-Transformations
  - Polynomial Constructions
  
- Hybrid combinations of all classical and recent approaches above

Classic Approaches

More recent Approaches

# Some Symmetric Algorithms and Structure:

## The past and the emergence of strong algebraic constructions

<b>Alg.</b>	<b>Key size Block size</b>	<b>Rounds</b>	<b>Structure Operations</b>	<b>Ex. Applications (standardization)</b>
<b>DES</b>	<b>56 /64</b>	<b>16</b>	<b>XOR, S-boxes, P-Boxes EP Boxes, Rotations, ...</b>	<b>SET, Kerberos</b>
<b>3DES</b>	<b>112,168 /64</b>	<b>48</b>	<b>idem</b>	<b>Financial standards E-commerce, PGP S/MIME, SSL</b>
<b>IDEA</b>	<b>128 /128</b>	<b>8</b>	<b>XOR, addition mod <math>2^{16}</math> , Multiplication mod <math>2^{16}</math></b>	<b>PGP, SSL</b>
<b>TEA</b>	<b>128 /128</b>	<b>32</b>	<b>XOR + shift</b>	
<b>Blowfish</b>	<b>&lt; 448 /128</b>	<b>16</b>	<b>XOR + Variable S-boxes</b>	
<b>RC5</b>	<b>&lt; 2048 /128</b>	<b>&lt; 255</b>	<b>Addition, subtraction XOR and rotation operations</b>	<b>SSL</b>
<b>Cast-128</b>	<b>40,128 /128</b>	<b>16</b>	<b>Addition, Subtraction, XOR, Rotation and S-boxes</b>	<b>PGP</b>
<b>AES</b>	<b>128, 192, 256 /128</b>	<b>10,12,14</b>	<b>S boxes, Matrix SR-MC, XOR</b>	<b>Many: “The Standard” after 2000</b>

# Symmetric block cyphers: AES, DES

	DES	Triple DES	AES
Plaintext block size (bits)	64	64	128
Ciphertext block size (bits)	64	64	128
Key size (bits)	56	112 or 168	128, 192, or 256

DES = Data Encryption Standard

AES = Advanced Encryption Standard

## Figures in AES:

Key size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext block size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of rounds	10	12	14
Round key size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded key size (words/bytes)	44/176	52/208	60/240

Demo: Performance Comparisons in Class:  
(Ex., openssl speed aes des des-ede3)

# Internal Structure of DES and AES

- Complementary materials  
(in the end of these slides)

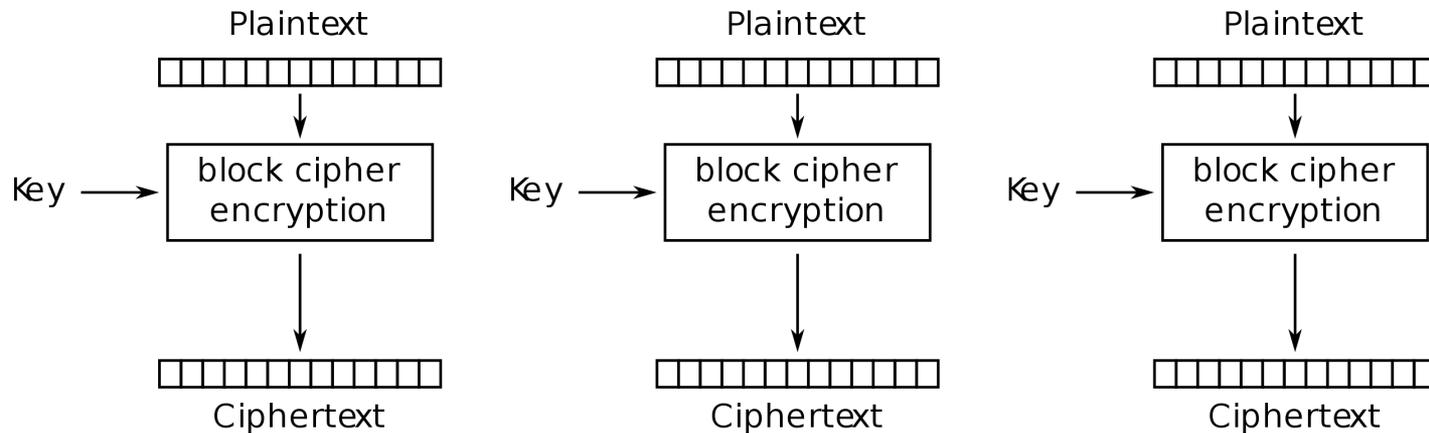
# Outline

- **Symmetric cryptography**
  - Symmetric Encryption
    - Block Encryption Algorithms
  - Robustness of symmetric encryption algorithms
  - Internal structure of symmetric algorithms
  - Cipher block modes of operation
  - Relevance of Padding
  - Stream Encryption Algorithms
  - Bit-Stream Ciphers using Block Ciphers

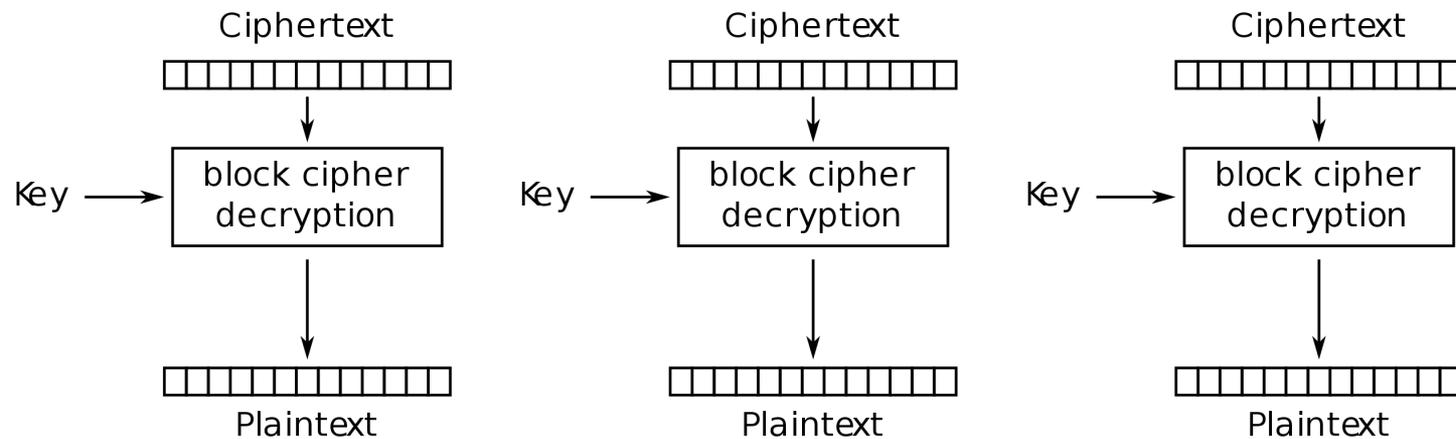
# Modes of Operation (Symmetric Block Ciphers)

- Block ciphers encrypt fixed size blocks
  - eg. DES encrypts 64-bit blocks with 56-bit key
- Need some way to encrypt/decrypt arbitrary amounts of data in practice (in a secure way and for the proper use)
- **ANSI X3.106-1983 Modes of Use** (now FIPS 81) defines 4 possible modes: **ECB, CBC, PCBC, CFB, OFB**
- Subsequently 5 more defined for AES & DES: **CTR, CTS**
- Today other modes are also relevant:
  - **XTS** (NIST SP 800-3E Specification)
  - **CCM, GCM, .... CWC, EAX, IAPM, OCB**
- Usable with any Symmetric Encryption Alg. (Block Ciphers or Block-Oriented Algorithms)
- Remember:
  - can have **block** and **stream** modes for block ciphers
  - Or ... stream-oriented encryption algorithms

# ECB Mode (Electronic Code Book)



Electronic Codebook (ECB) mode encryption



Electronic Codebook (ECB) mode decryption

# ECB Mode Characteristics

- Plaintext blocks encrypted in an independent way
- Fast (encryption and decryption are parallelizable)
- Random Read Access (block-indexing)

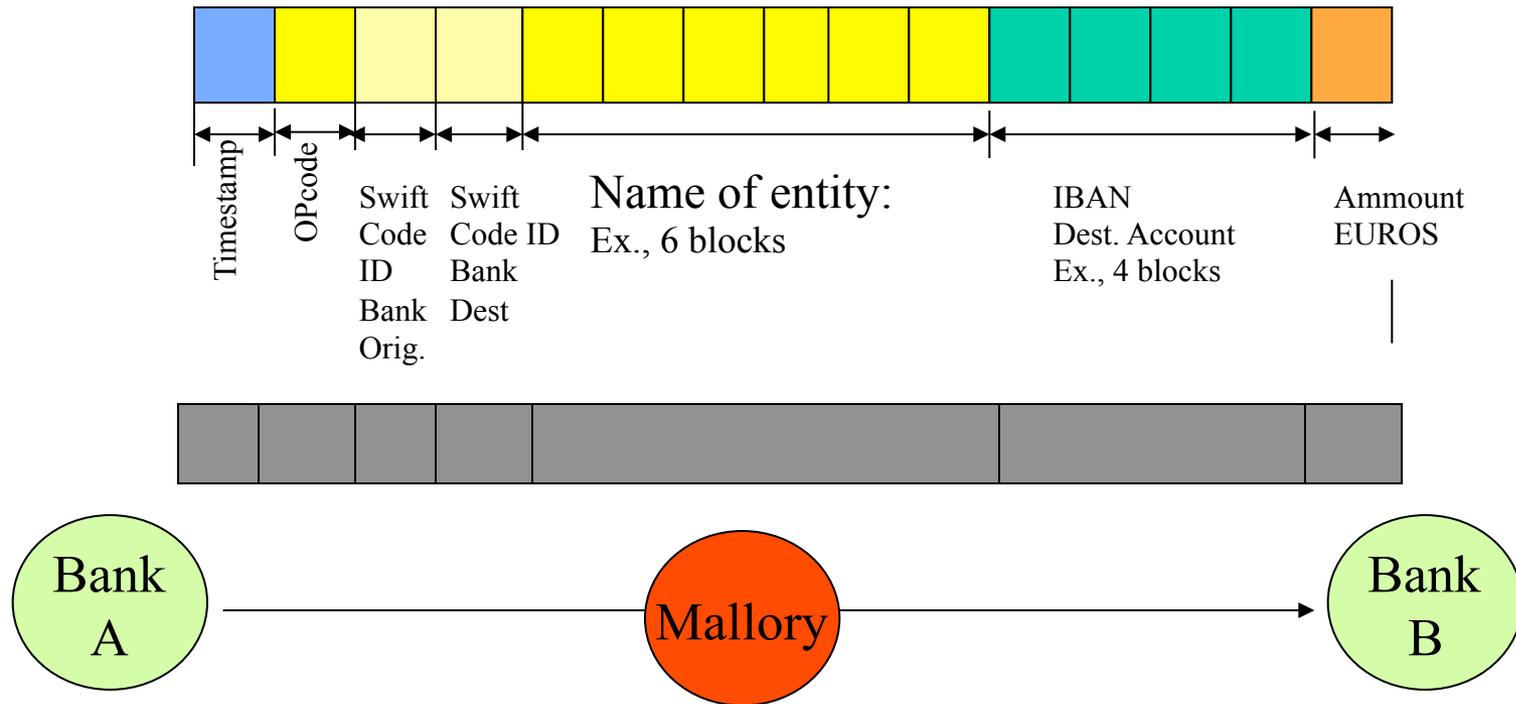
## Message Repetitions => Regularities show in ciphertext

- If aligned with message block
  - Particularly with data such graphics
  - or text-messages or messages that change very little, which become a code-book analysis problem
- 
- Weakness is due to the encrypted message blocks being independent: same plaintext blocks produces the same cyphertext blocks (using the same key)
  
  - Main use is sending a few blocks of data, encryption of small blocks (size small than the block size of a given algorithm, small-byte blocks, ex., passwords)

# ECB: manipulation of encrypted blocks

Ex., electronic banking transaction, ref. with DES

1 block = 64 bits



Eavesdropping + Message Replying +  
Message Tampering (on specific blocks)

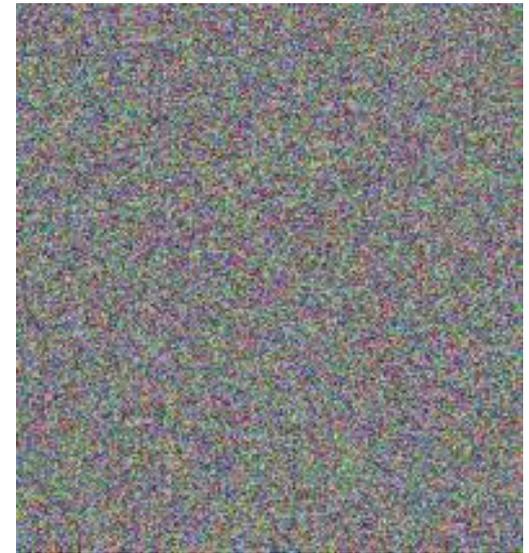
# ECB (graphics, revelation of patterns)

Plaintext and Ciphertext with ECB



Original de Larry Ewing's

What we want ...

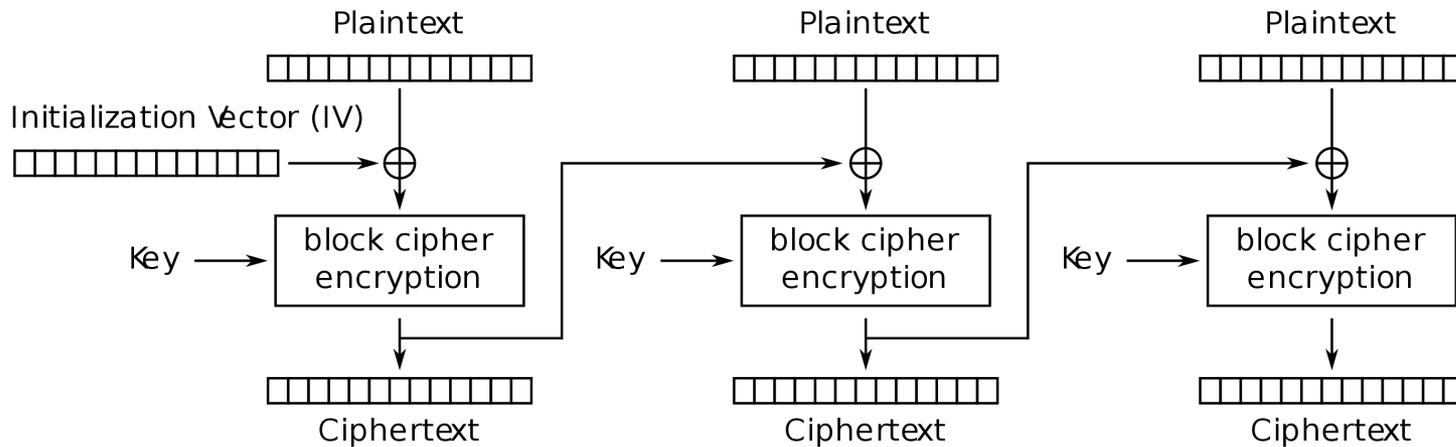


Demonstrated in Labs,  
for bmp/gif/jpeg files

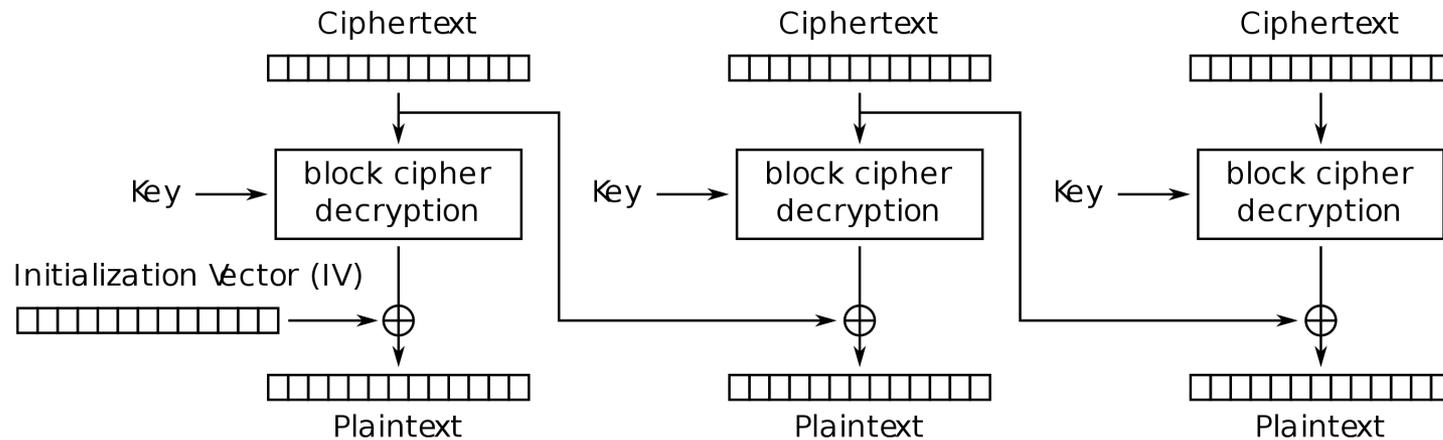
# ECB Advantages and Drawbacks

- Simple, Block-By-Block Independent Encryption
- Fast - Easily Parallelizable
  - The faster mode: faster than any other mode
- Easy to recover possible errors: simply retransmit the affected block
  - No Error Propagation
- Same plaintext blocks => Same ciphertext blocks if we use the same key
  - Regularities of plaintext blocks in a sequence of blocks are revealed
- But can be used when not-revealing block-patterns
  - If plaintext is first pre-processed in some way to hide possible regular patterns
  - Example: Compression before Encryption is usual
  - Or if we encrypt data with size lower than the block size (in a given algorithm)
    - Example: Encryption of Initialization Vectors, Counters, Variable Padded Passwords, Nonces/Responses, etc

# CBC Mode: Cipher Block Chain



Cipher Block Chaining (CBC) mode encryption

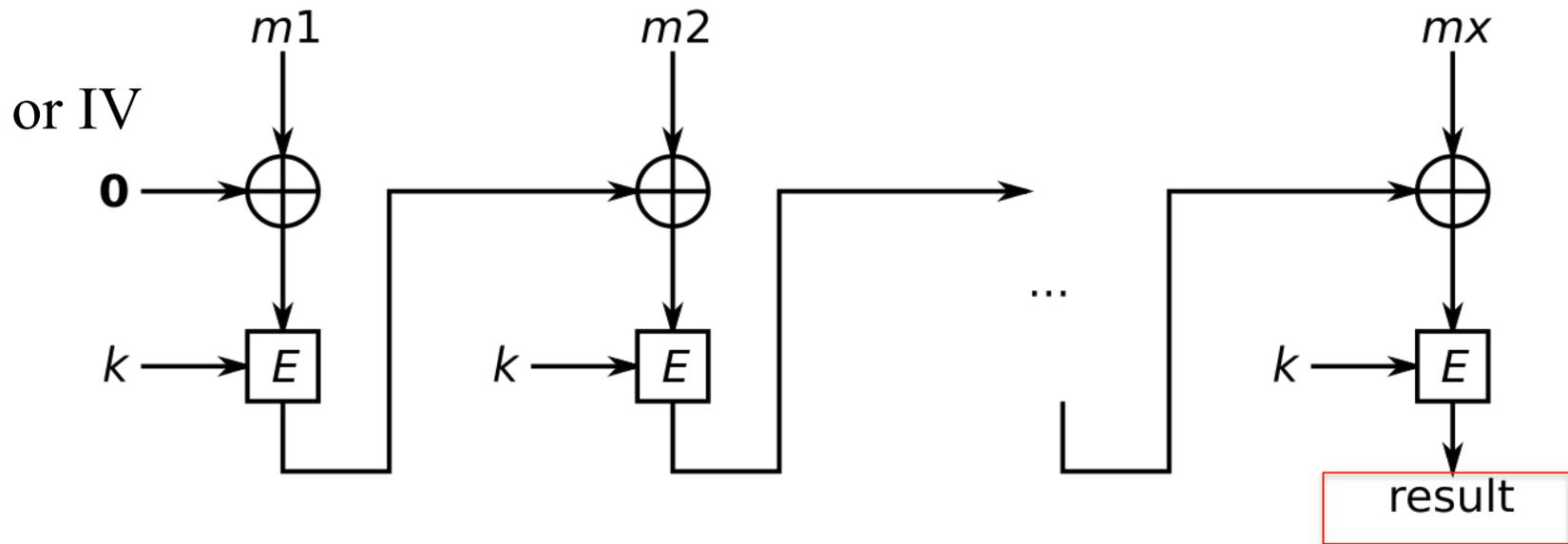


Cipher Block Chaining (CBC) mode decryption

# CBC Mode

- Cipher Block Chaining Mode (CBC)
  - The input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block.
  - Repeating pattern of 64-bits are not exposed
- Need an Initial Vector (IV) to start process
$$C_i = \text{DES}_{K_1}(P_i \text{ XOR } C_{i-1})$$
$$C_{-1} = \text{IV}$$
- Uses: for bulk data encryption for confidentiality, but also for authentication (example CBC-MAC schemes)
- Same plaintext blocks => Different cyphertext blocks, even when the same key is used (if use different IVs)

# CBC-MAC Scheme



Suppose you send ...

CBC-Obtained  
Message Authentication Code

$$E_{K_m}(M) \parallel \text{CBC-MAC}_{K_A}$$

# CBC Mode Characteristics (1)

## Security

- + Security: plaintext patterns concealed by XORing with previous ciphertext blocks - Input to blocks randomized by the previous cipher text block
- + Any change to a block affects all following ciphertext blocks
- +/- Plaintext somewhat difficult to manipulate  
Only 1<sup>st</sup> block and last block can be removed)  
Bits changed in the first block: repetition allows controlled changes
- + More than one message can be encrypted with the same key
- + A ciphertext block depends on **all** blocks before it

## Performance

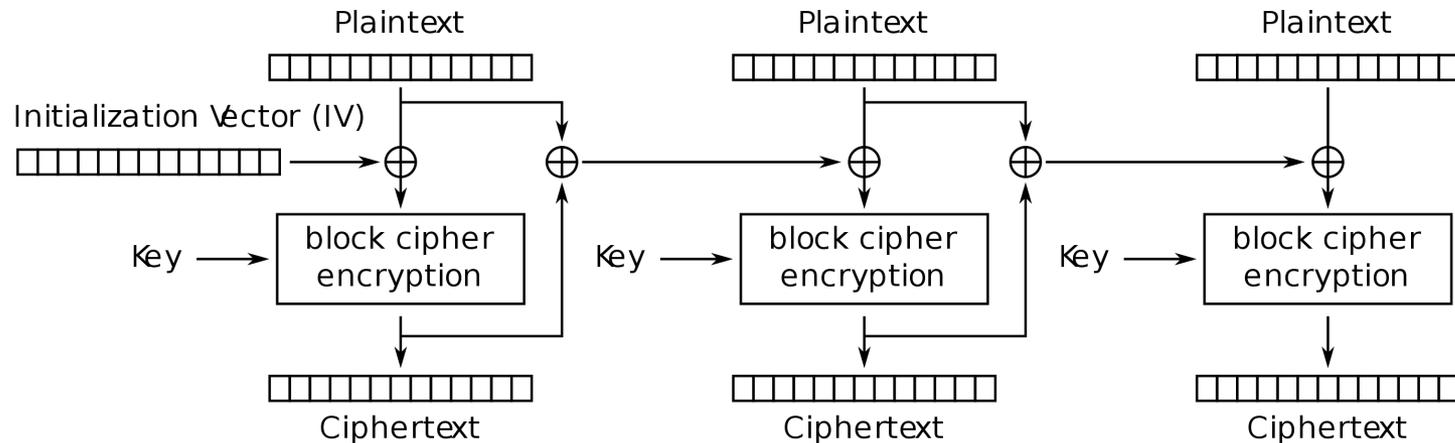
- + Speed in ~the same as block cipher
- - But ciphertext is up to one block longer than the plaintext
- - No pre-processing is possible
- -/+ Encryption is not parallelizable, but decryption is parallelizable and has a random-access property

# CBC Mode Characteristics (2)

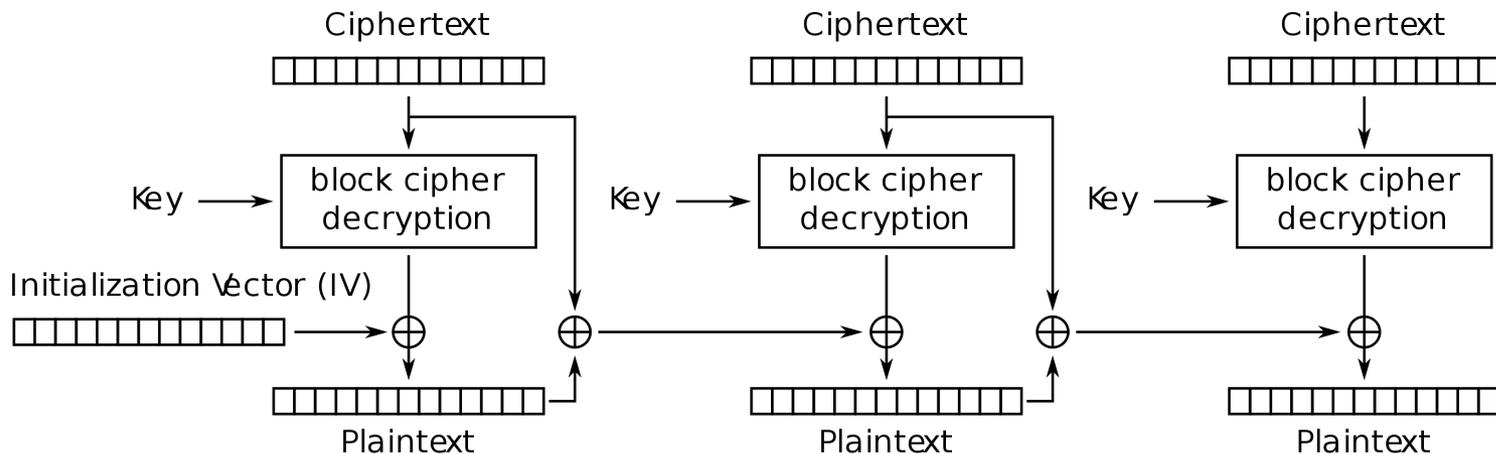
## Fault-Tolerance

- - Need **Initialization Vector (IV)**
  - which must be known to sender & receiver
    - Requires synchronization
  - if sent in clear, attacker can change bits of first block, and change IV to compensate
  - hence IV must either be a fixed value (as in EFTPOS)
  - or must be sent encrypted in ECB mode before rest of message
- A cipher text error affects one full plaintext block and the correspondent bit in the next block
- Synchronization errors unrecoverable

# PCBC Mode (Propagating CBC mode)



Propagating Cipher Block Chaining (PCBC) mode encryption



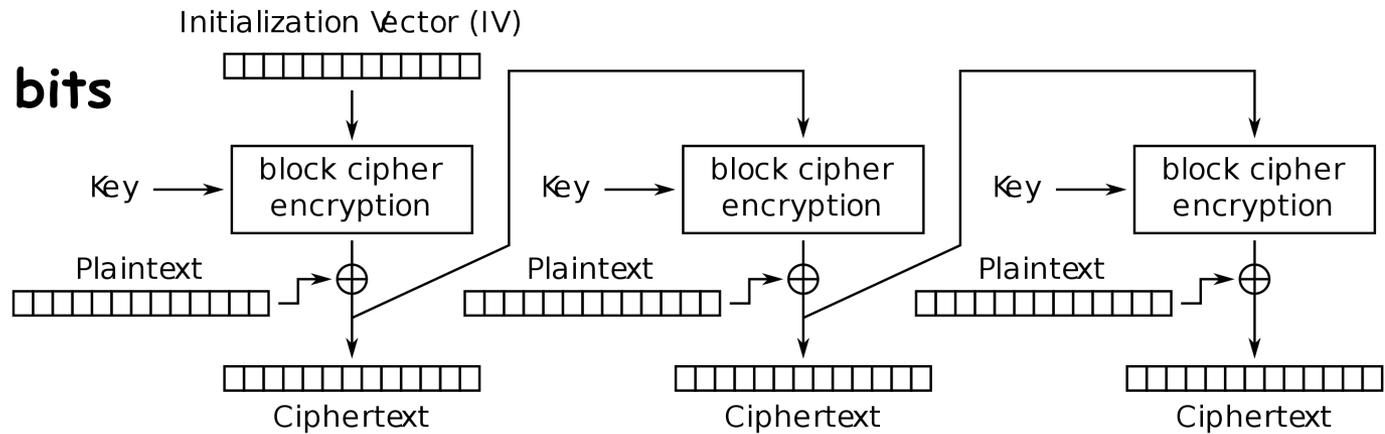
Propagating Cipher Block Chaining (PCBC) mode decryption

# PCBC Use

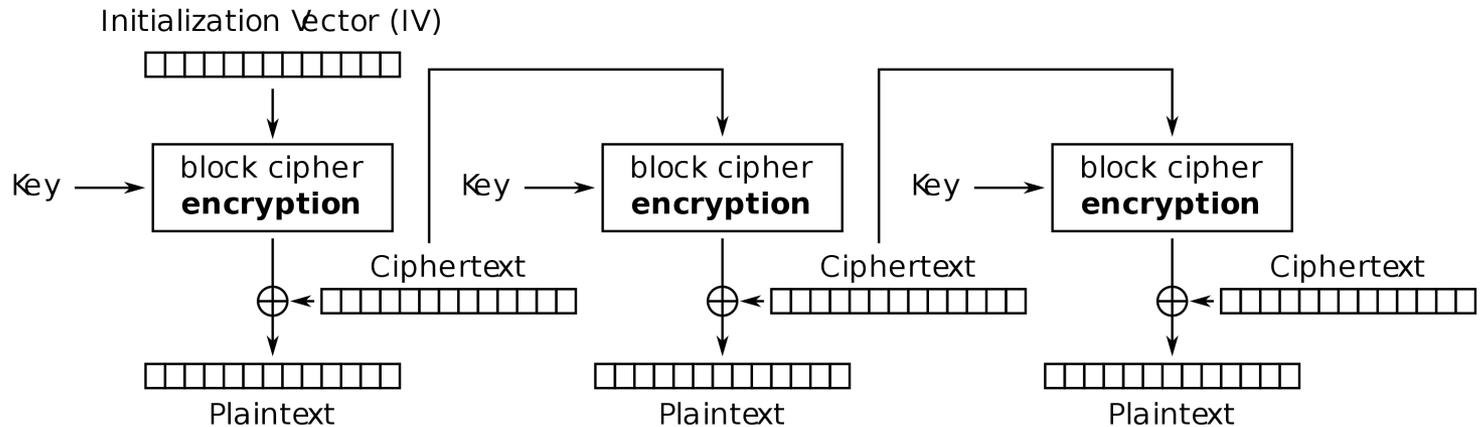
- Most notably standardized and used in some security protocols (ex., Kerberos V4)... but not so common
- Property that can be observed:
  - If two adjacent ciphertext blocks are exchanged this does not affect the decryption of the subsequent blocks, which may be an "insecurity" issue
  - Example: PCBC mode was abandoned in Kerberos V5

# CFB mode (Cipher Feedback Mode)

Can use w/  
8, 16, ... 128 bits



Cipher Feedback (CFB) mode encryption



Cipher Feedback (CFB) mode decryption

# CFB Mode Processing

## Cipher Feedback Mode (CFB)

Conversion of a block ciphers as stream cipher  
(byte-oriented block cipher)

- Message treated as a stream of bits (controlled size)
- Added to the output of the block cipher
- Result is feed back for next stage (hence name)
- Standard allows any number of bit (1,8, 64 or 128 etc) to be feed back
  - Denoted CFB-1, CFB-8, CFB-64, CFB-128 etc
- Most efficient to use all bits in block (64 or 128)  
$$C_i = P_i \text{ XOR } D_{K1}(C_{i-1})$$
$$C_{-1} = IV$$
- Uses: stream data encryption, authentication
- Note: Only need the Encryption Function Implementation !

# CFB Mode: Characteristics (1)

## Security

- + Appropriate when data arrives in bits/bytes, most common stream mode (without counting)
- + Plaintext concealed, input to block cipher is randomized
- + More than one message can be encrypted with the same key (changing the IV)
- +/- Plaintext somewhat difficult to manipulate
  - Only 1<sup>st</sup> block and last block can be removed
  - Bits changed in the first block: repetition allows controlled changes

## Performance

- + Speed is ~the same as the block cipher
- + Cipher text with the same size than plaintext (not counting the IV)
- +/- Encryption not parallelizable, decryption parallelizable and has a random access property
- + Some pre-processing is possible (previous cipher text can be encrypted (before a new plaintext block) - Note that the block cipher is used in **encryption** mode at **both**

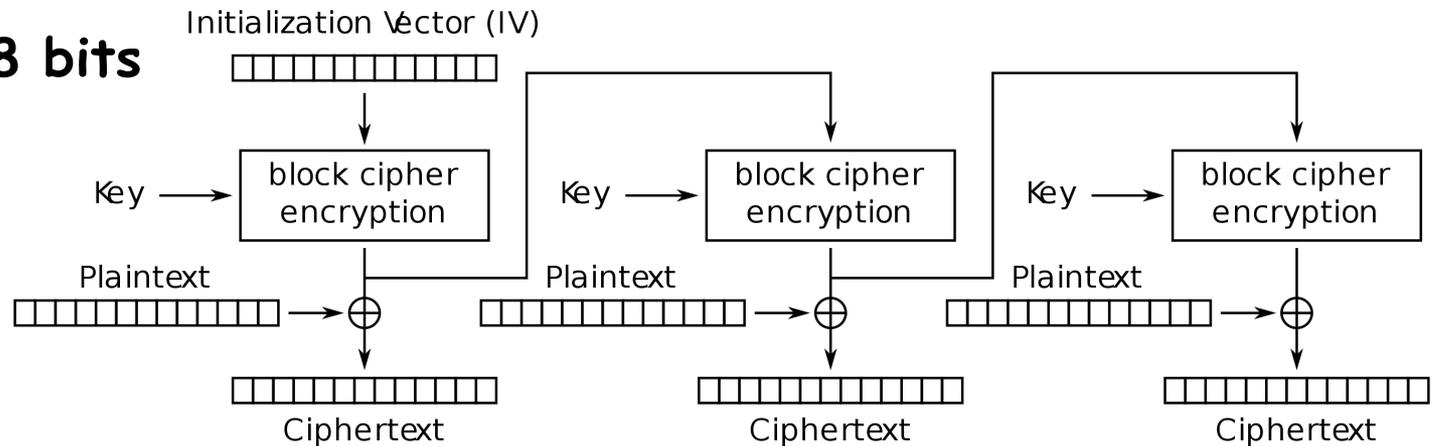
# CFB Mode: Characteristics (2)

## Fault-tolerance

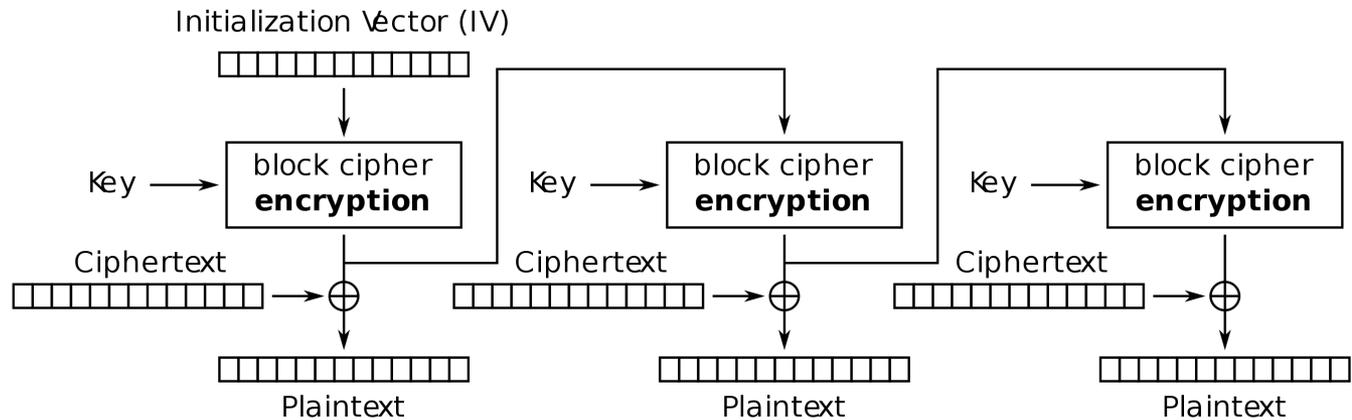
- Limitation is need to stall while do block encryption after every n-bits
- Errors propagate for several blocks after the error: affecting correspondent bits of plaintext and the next full block
- Synchronization errors of full block sizes are recoverable. A 1-bit CFB can recover from the addition or loss of single bits

# OFB Mode: Output FeedBack Mode

Can use w/  
8, 16, ... 128 bits



Output Feedback (OFB) mode encryption



Output Feedback (OFB) mode decryption

# OFB Mode Processing

- Message is treated as a stream of bits (allowing sizes of 8, 16, 32, 64, 128 bits)
  - Output of cipher is added to message
  - Output is then feed back (hence name)
  - Feedback is independent of message
- 
- Uses: stream encryption on noisy channels

# OFB Mode: Characteristics (1)

## Security

- + Plaintext patterns concealed
- + Input to the block cipher is randomized
- + More than one message can be encrypted with the same key (provided that a different IV is used)
- - Plaintext is very easy to manipulate; any change in cipher text directly affects the plaintext - more vulnerable to message stream modification
  - A variation of a Vernam cipher
    - hence must **never** reuse the same sequence (key+IV)
- originally specified with m-bit feedback
  - (but subsequent research has shown that only **full block feedback** ie CFB-64 or CFB-128) should ever be used for high security)

## Performance

- + Speed is ~the same as block cipher
- - Cipher text is the same size as the plaintext, not counting the IV
- + Pre-Processing possible before the message is seen
- -/+ not parallelizable

# OFB Mode: Characteristics (2)

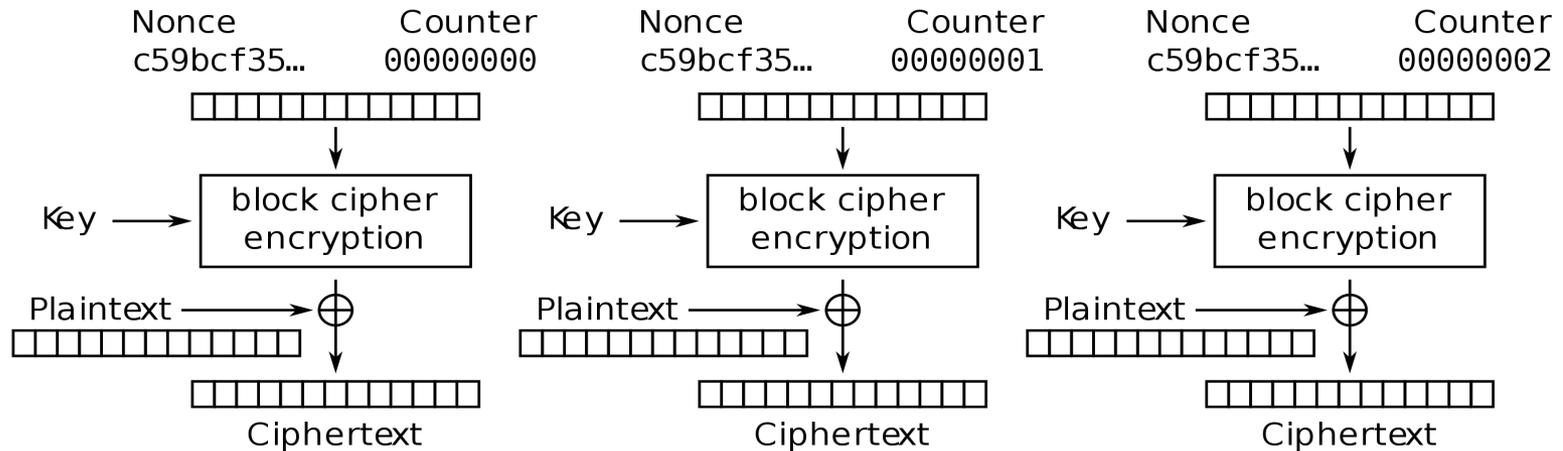
## Fault-tolerance

- + Bit errors do not propagate - Cipher text error affects only the corresponding bit of plaintext
- - But sender & receiver must remain in sync: synchronization error not recoverable

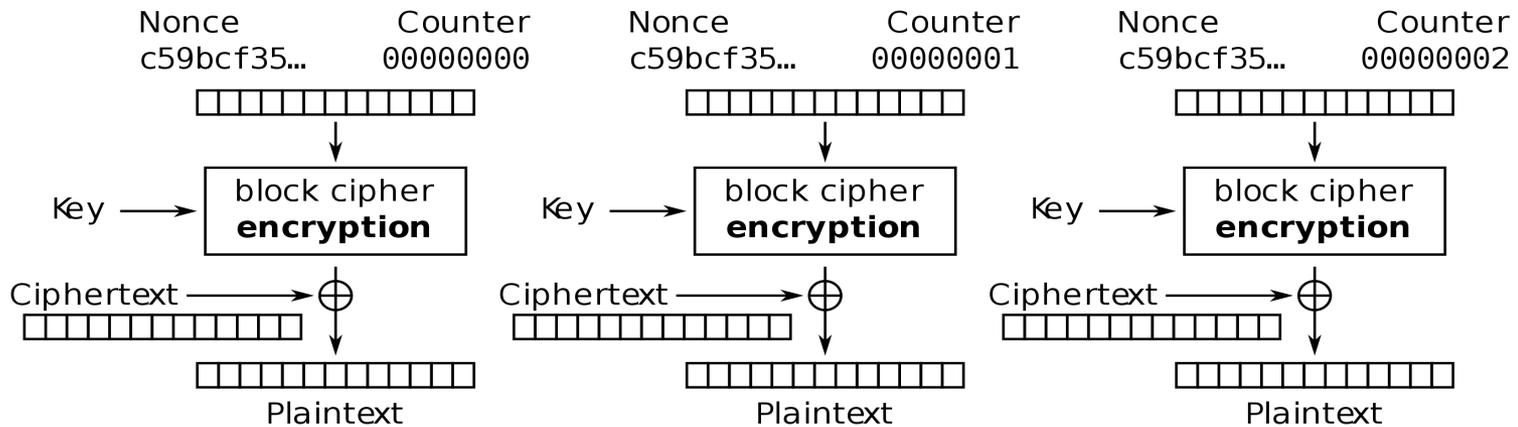
# CTR Mode: CounTeR Mode

- A “new” mode, though proposed early on
- Similar to OFB but encrypts counter value rather than any feedback value
- Must have a different key & counter value for every plaintext block (never reused)
  - $C_i = P_i \text{ XOR } O_i$
  - $O_i = \text{DES}_{K_1}(i)$
- Uses: high-speed network encryptions

# CTR Mode



Counter (CTR) mode encryption



Counter (CTR) mode decryption

# CTR: Characteristics (1)

## Security

- Same criteria as OFB
  - Random access to encrypted data blocks
  - Provable security (good as other modes)
  - But must ensure never reuse key/counter values, otherwise could break (cf OFB)

## Performance

- Efficiency
  - + ~The same Better than OFB
    - Parallelizable: can do parallel encryptions in h/w or s/w
    - can pre-process in advance of need
    - good for bursty high speed links

## Performance

as OFB

# Summary of Block-Cipher Modes of Operation: characteristics vs. tradeoffs

## Security

- CBC (entire blocks), CFB (mainly with  $m$ -bit blocks, more secure => larger  $m$ )
- ECB, OFB and Counter: plaintext easy to manipulate

## Performance

- ECB, CTR (parallelizable)
- CFB (only partial parallelization in the decryption)
- CBC same but no pre-processing
- OFB no parallelization, no pre-processing

## Fault-Tolerance

- CFB - synchronization errors of full block sizes are recoverable, ... but cipher text error affect the correspondent bit of plaintext + the next full block), OFB, CTR (ciphertext errors only affect the correspondent bit in plaintext)
- ECB, CBC, OFB, CTR - Sync. Errors unrecoverable,

# Symmetric Modes of Operation: Summary of Representation and Formulas (1)

Summary of modes

Mode	Formulas	Ciphertext
Electronic Codebook (ECB)	$Y_i = F(\text{PlainText}_i, \text{Key})$	$Y_i$
Cipher Block Chaining (CBC)	$Y_i = \text{PlainText}_i \text{ XOR Ciphertext}_{i-1}$	$F(Y, \text{key}); \text{Ciphertext}_0 = \text{IV}$
Propagating CBC (PCBC)	$Y_i = \text{PlainText}_i \text{ XOR } (\text{Ciphertext}_{i-1} \text{ XOR PlainText}_{i-1})$	$F(Y, \text{key}); \text{Ciphertext}_0 = \text{IV}$
Cipher Feedback (CFB)	$Y_i = \text{Ciphertext}_{i-1}$	$\text{Plaintext XOR } F(Y, \text{key}); \text{Ciphertext}_0 = \text{IV}$
Output Feedback (OFB)	$Y_i = F(\text{Key}, Y_{i-1}); Y_0 = \text{IV}$	$\text{Plaintext XOR } Y_i$
Counter (CTR)	$Y_i = F(\text{Key}, \text{IV} + g(i)); \text{IV} = \text{token}();$	$\text{Plaintext XOR } Y_i$

# Symmetric Modes of Operation: Summary of Representation and Formulas (2)

**ECB:**  $C_i = E_k(P_i)$ ;  $P_i = D_k(C_i)$

**CBC:**  $C_i = E_k(P_i \text{ XOR } C_{i-1})$   
 $P_i = D_k(C_i) \text{ XOR } C_{i-1}$

$C_0 = IV$

**PCBC:**  $C_i = E_k(C_{i-1} \text{ XOR } P_i)$   
 $P_i = D_k(C_{i-1}) \text{ XOR } C_i$

$C_0 = IV$

**CBF:**  $C_i = \text{head}(E_k(S_{i-1}), x) \text{ XOR } P_i$   
 $P_i = \text{head}(E_k(S_{i-1}), x) \text{ XOR } C_i$

$S_i = ((S_{i-1} \ll x) + C_i) \bmod 2^n$

$S_0 = IV$

**OFB:**  $C_j = P_j \text{ XOR } O_j$   
 $P_j = C_j \text{ XOR } O_j$

$O_j = E_k(I_j)$ ,  $I_j = O_{j-1}$ ,  $I_0 = IV$

**CTR:**  $C_i = E_k(n\text{-ctr}_i) \text{ XOR } P_i$   
 $P_i = E_k(n\text{-ctr}_i) \text{ XOR } C_i$

$n\text{-ctr}_{i+1} = F_c(n\text{-ctr}_i)$ ,  $n\text{-ctr}_0 = IV$

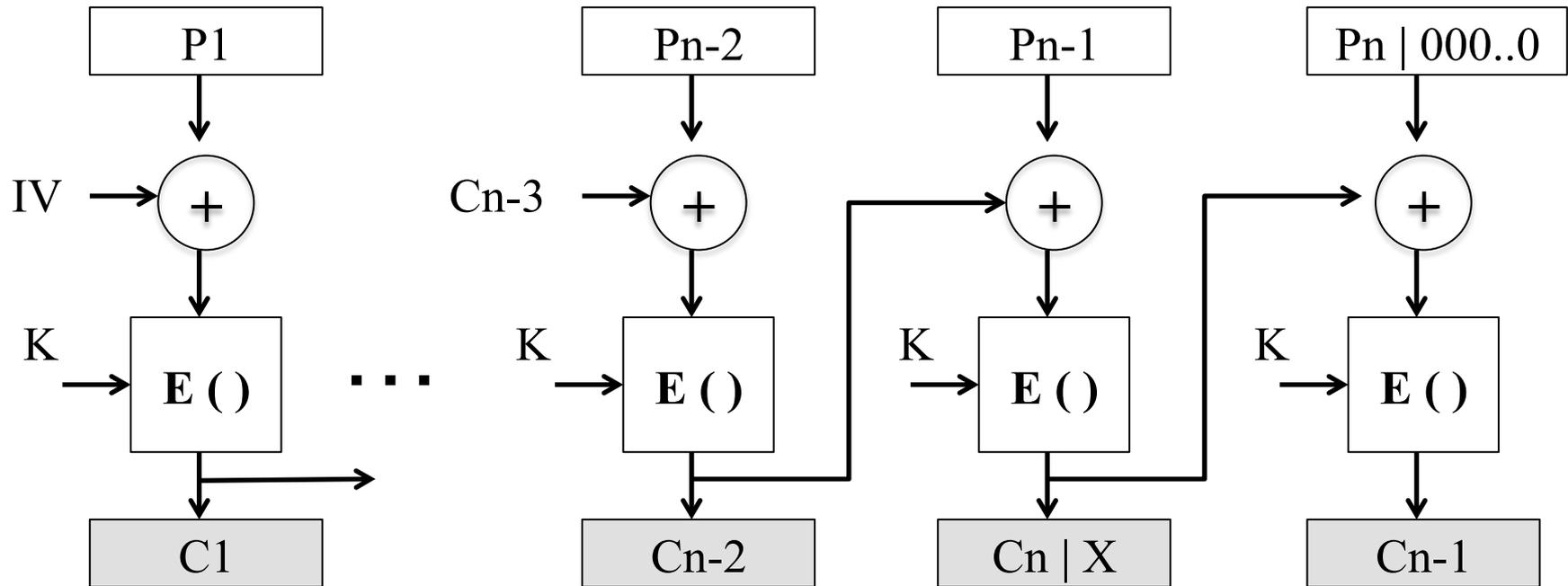
# Other Cipher Modes of Operation

- Interleaving technique
  - Ex., CBC Interleaving
- Multiplex/Demultiplex with Interleaving
  - Single message
  - Multiple messages
- CTS (Ciphertext Stealing Mode)
- CCM, GCM: Modes w/ Implicit Authentication (CMAC)
  - AES-CCM :
    - IETF RFC 6665 (used in TLS Cryptosuites)
    - IETF RFC 4309 (used in IPsec Ciphersuites)
  - GCM

---

See more here (papers ref.): [https://en.wikipedia.org/wiki/CCM\\_mod](https://en.wikipedia.org/wiki/CCM_mod)  
[https://en.wikipedia.org/wiki/Galois/Counter\\_Mode](https://en.wikipedia.org/wiki/Galois/Counter_Mode)

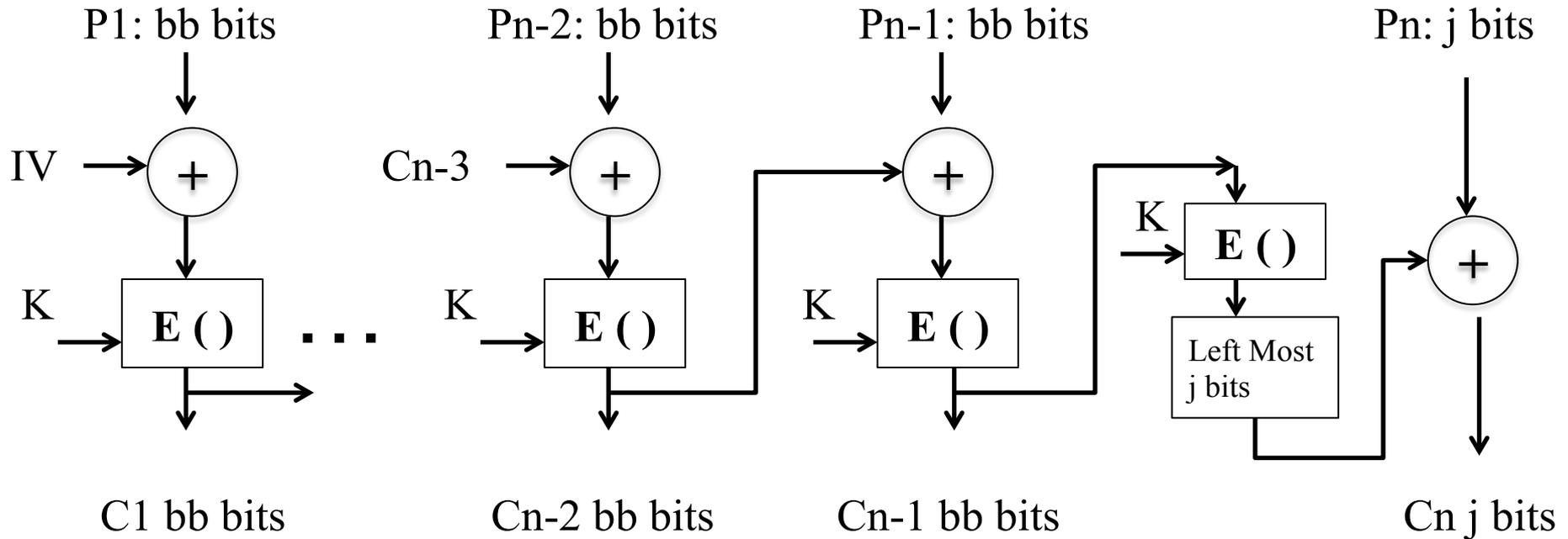
# CTS Mode - Ciphertext Stealing



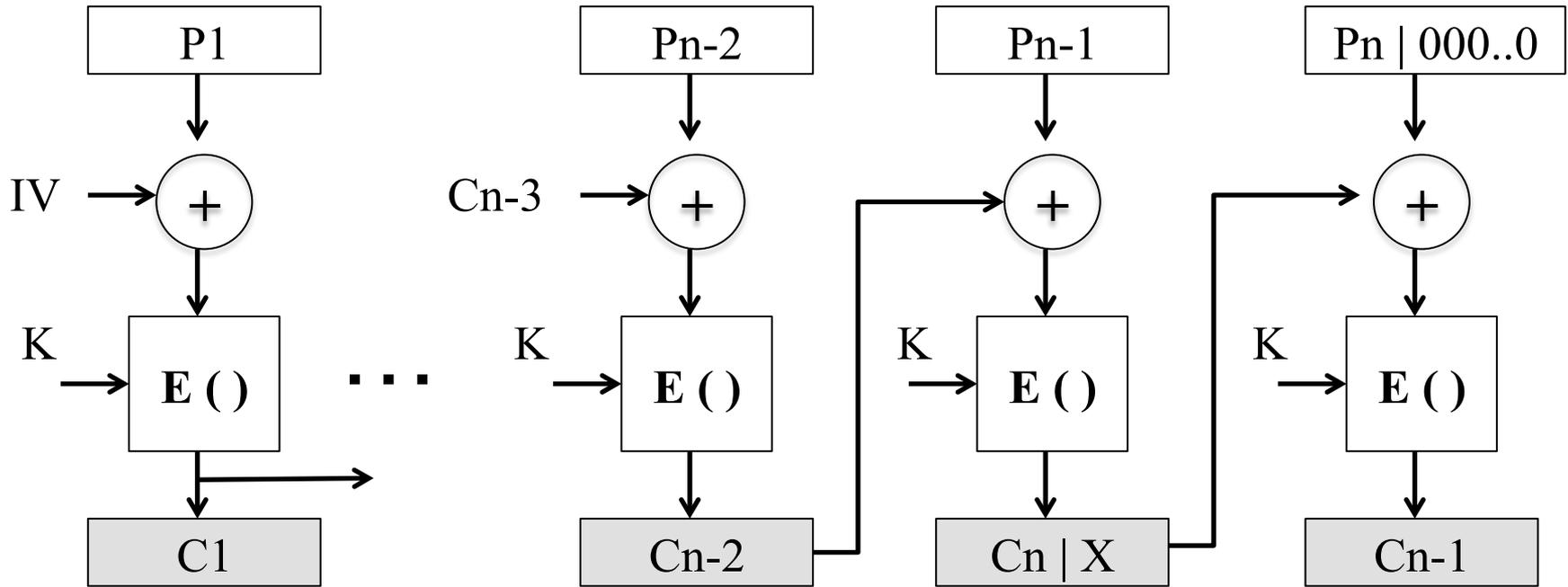
Similar to CBC Considerations but ciphertext will have the Same size as plaintext

(Think if you want to store encrypted data in the same memry buffer initially storing plaintext data)

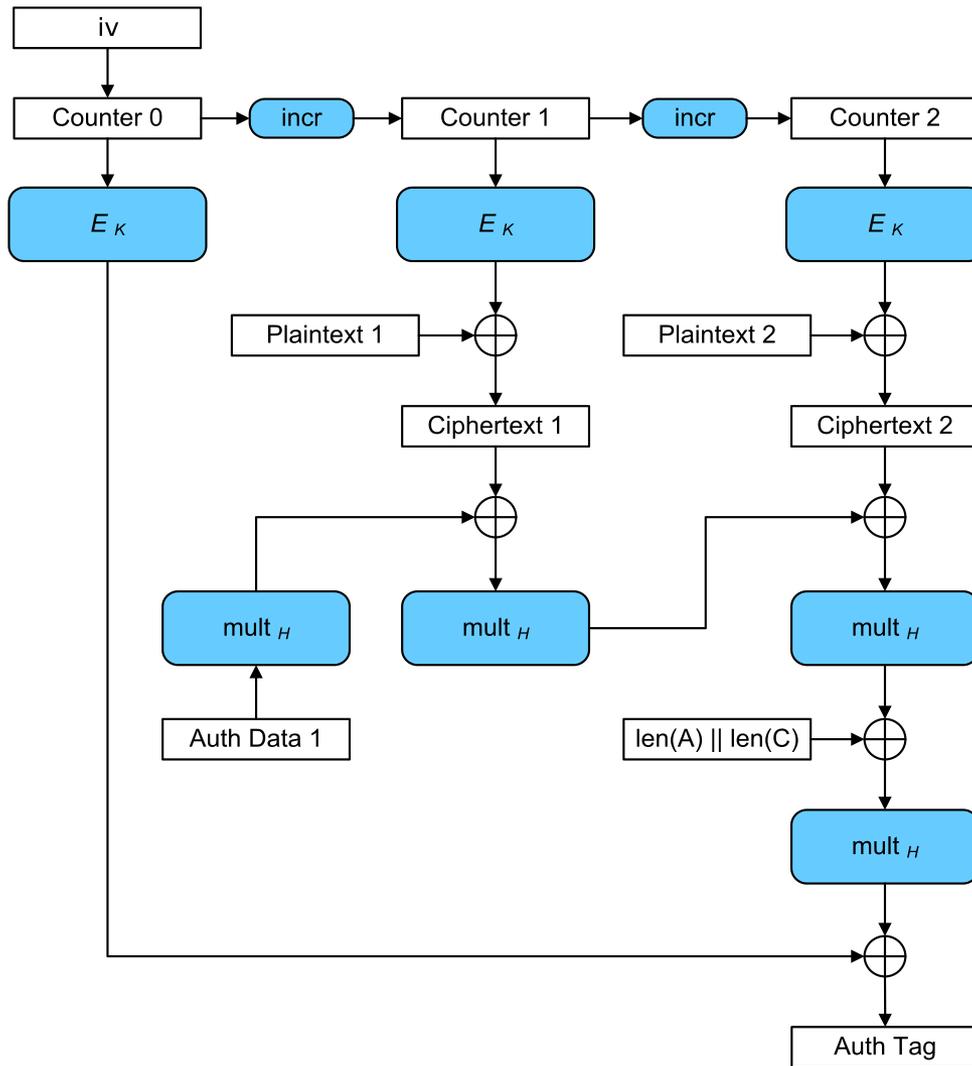
# CTS Mode (alternative)



# CTS Mode - Ciphertext Stealing



# GCM (Galois Counter Mode)



$$\text{GF}(2^{128}) = x^{128} + x^7 + x^2 + x + 1$$

$$\text{GHASH}(H, A, C) = X_{m+n+1}$$

# Outline

- **Symmetric cryptography**
  - Symmetric Encryption
    - Block Encryption Algorithms
  - Robustness of symmetric encryption algorithms
  - Internal structure of symmetric algorithms
  - Cipher block modes of operation
  - Relevance of Padding
  - Stream Encryption Algorithms
  - Bit-Stream Ciphers using Block Ciphers

# Padding

- At end of message must handle a possible last short block
  - Which is not as large as blocksize of cipher
  - PAD either with known non-data value (eg nulls)
  - or PAD last block following a standardized padding scheme
  - When using Padding, it must be required an extra entire block over those in message (when multiples of blocks)
- Padding standardization (next slide):
  - See practical classes / labs - examples
- There are other, more esoteric modes, which avoid the need for an extra block
- Easy to adopt secure padding for specific designed protocols
- See this in more detail in LABs

# Some Standardized Padding Schemes (1)

ZERO BIT PADDING, RFC 1321, ISO/IEC 9797-1 (called Padding Method 2):

... | 1011 1001 1101 0100 0010 0111 0000 0000 |

ZERO PADDING

... | DD DD DD DD DD DD DD DD | DD DD DD DD 00 00 00 00 |

ANSI X9.23

... | DD DD DD DD DD DD DD DD | DD DD DD DD 00 00 00 04 |

ISO 10126

... | DD DD DD DD DD DD DD DD | DD DD DD DD 81 A6 23 04 |

PKCS#5, PKCS#7, RFC 5652

... | DD DD DD DD DD DD DD DD | DD DD DD DD 04 04 04 04 |

# Some Standardized Padding Schemes (2)

ISO/IEC 7816-4

... | DD DD DD DD DD DD DD DD | DD DD DD DD 80 00 00 00 |  
... | DD DD DD DD DD DD DD DD | DD DD DD DD DD DD DD 80 |

What about a more secure padding (for improved paranoia) ?

Ex:

... | DD DD DD DD DD DD DD DD | DD DD 06 XX XX XX XX XX |

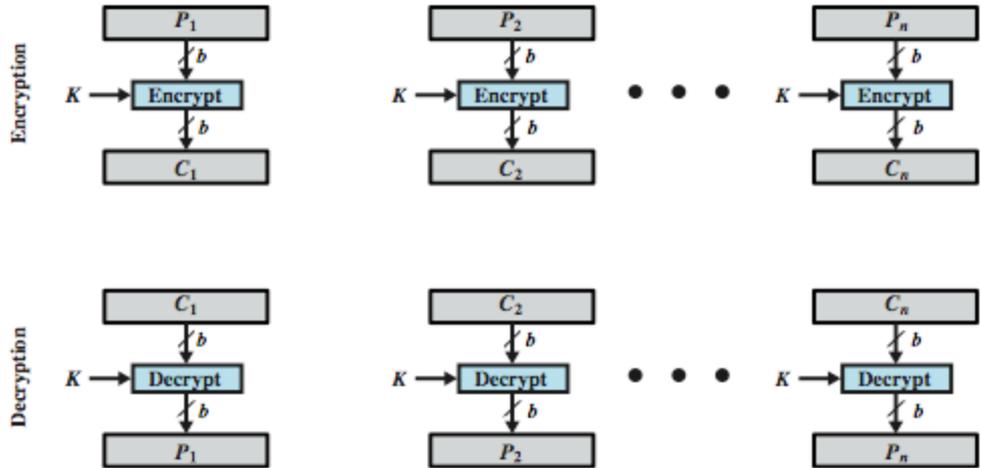
$\text{Msb}_{30} ( \text{SecureHash} (m) )$

# Outline

- **Symmetric cryptography**
  - Symmetric Encryption
    - Block Encryption Algorithms
  - Robustness of symmetric encryption algorithms
  - Internal structure of symmetric algorithms
  - Cipher block modes of operation
  - Relevance of Padding
  - Stream Encryption Algorithms
  - Bit-Stream Ciphers using Block Ciphers

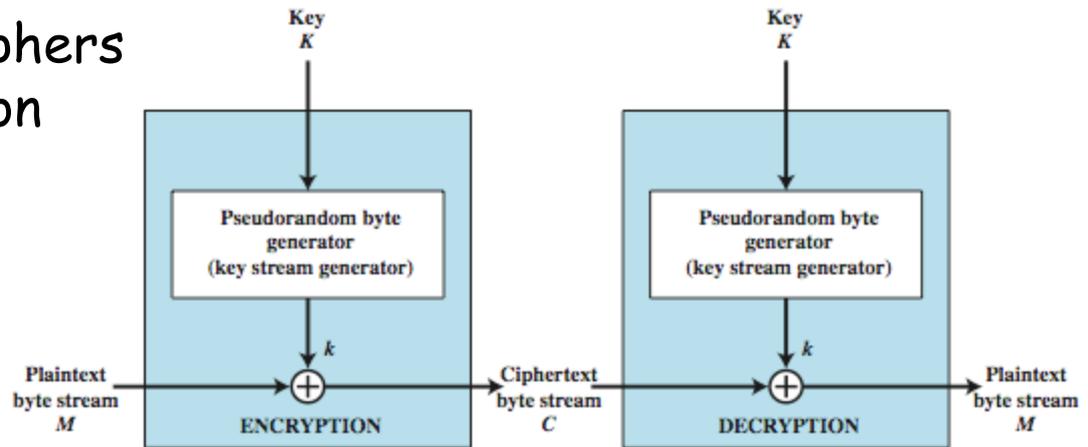
# Stream ciphers

- Block cipher (with ECB mode of operation)



(a) Block cipher encryption (electronic codebook mode)

- Structure for stream ciphers
  - Possible implementation with block ciphers.
  - How ?



(b) Stream encryption

# Stream ciphers

- Process message bit by bit (as a stream)
  - This is different than the use of byte-oriented block-encryption, like CTR, OFB or CFB
  - Symmetric Block-Encryption (using those modes) is not a stream cypher algorithm
- Have a pseudo random **keystream**
- Combined (XOR) with plaintext bit by bit
- Randomness of **stream key** completely destroys statistically properties in message
  - $C_i = M_i \text{ XOR StreamKey}_i$
- But must never reuse stream key: Problem of the Period in Stream Cipher Algorithms
  - otherwise can recover messages (cf book cipher)
- Some design considerations are:
  - long period with no repetitions, Statistically random
  - depends on large enough key
  - large linear complexity
- Properly designed, can be as secure as a block cipher with same size key, but usually simpler & faster

# RC4 Algorithm

- Proprietary cipher owned by RSA DSI
  - Code revealed, known-algorithm
- Another Ron Rivest design, simple but effective
- Variable key size, byte-oriented stream cipher
- Widely used (web SSL/TLS, wireless WEP)
- Key forms random permutation of all 8-bit values
- Uses that permutation to scramble input info processed a byte at a time

# RC4 Key Schedule

- Starts with an array  $S$  of numbers:  $0..255$
- Use key to well and truly shuffle
- $S$  forms **internal state** of the cipher

```
// Initialization, S and T
```

```
for i = 0 to 255 do
```

```
    S[i] = i
```

```
    T[i] = K[i mod keylen])
```

```
// Initial Permutation of S
```

```
j = 0
```

```
for i = 0 to 255 do
```

```
    j = (j + S[i] + T[i]) (mod 256)
```

```
    swap (S[i], S[j])
```

# RC4 Encryption

- Encryption continues shuffling array values
- Sum of shuffled pair selects "stream key" value from permutation
- XOR  $S[t]$  with next byte of message to en/decrypt

$i = j = 0$

for each message byte  $M_i$

$i = (i + 1) \pmod{256}$

$j = (j + S[i]) \pmod{256}$

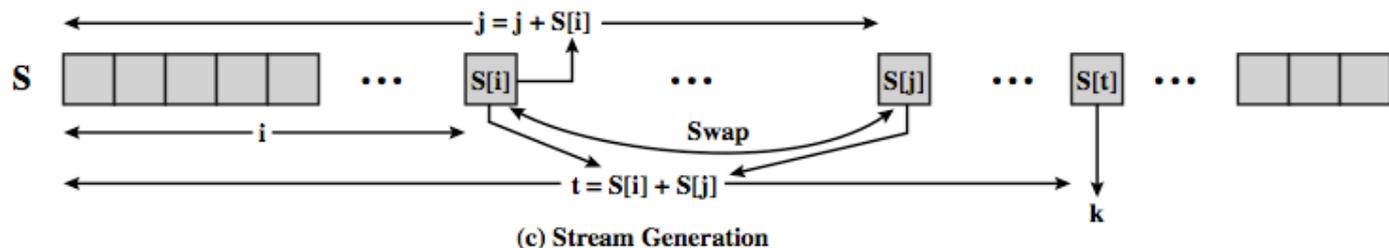
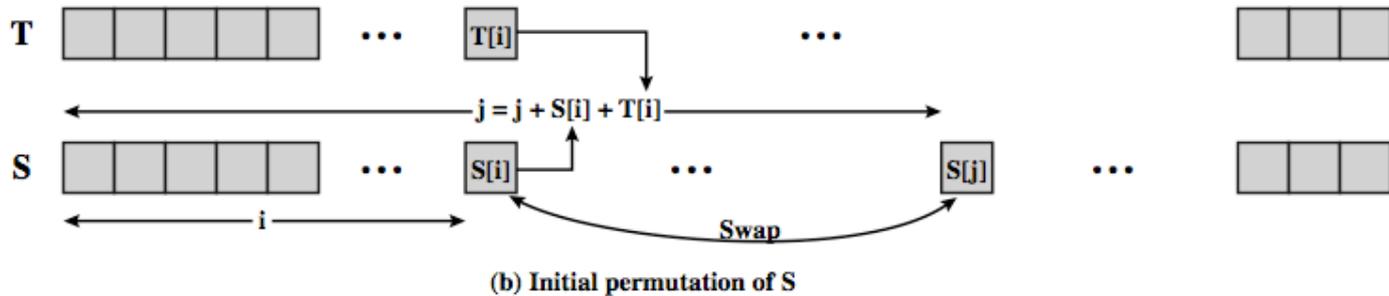
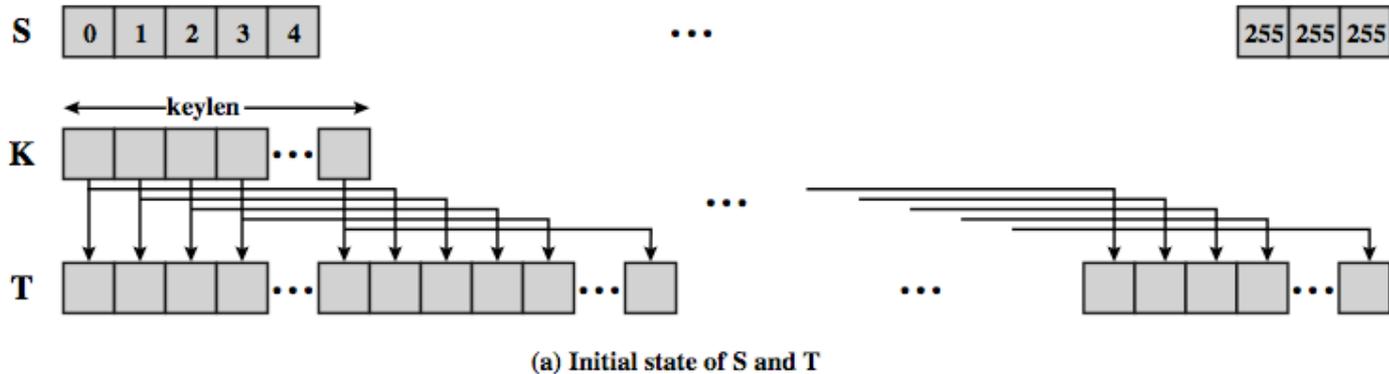
swap( $S[i]$ ,  $S[j]$ )

$t = (S[i] + S[j]) \pmod{256}$

$C_i = M_i \text{ XOR } S[t]$

# RC4 encryption model

- Algorithm explained in Stallings, Network Security Essentials (see section 2.4, Chapter 2)



# RC4 Security

- Claimed secure against known attacks
  - have some analyses, none practical
- Result is very non-linear
- Since RC4 is a stream cipher, must **never reuse a key**
- Have a concern with WEP, but **due to key handling and/or pattern-repetitions** rather than RC4 itself

# Other stream ciphers

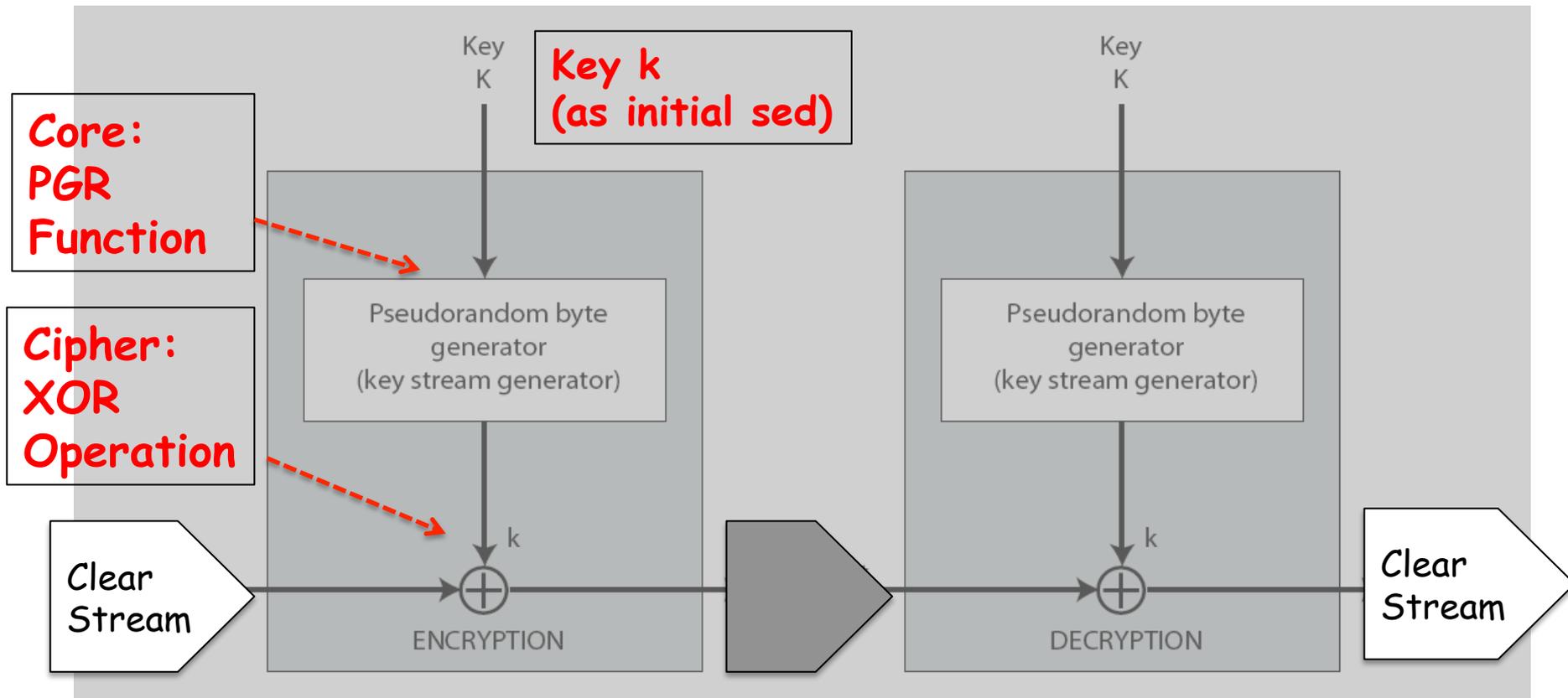
- SEAL
- "A" symmetric stream ciphers series (ex., A5, A8, in Industry/GSM)
- ... Many others ...
  - See more: [https://en.wikipedia.org/wiki/Stream\\_cipher](https://en.wikipedia.org/wiki/Stream_cipher)

# Outline

- **Symmetric cryptography**
  - Symmetric Encryption
    - Block Encryption Algorithms
  - Robustness of symmetric encryption algorithms
  - Internal structure of symmetric algorithms
  - Cipher block modes of operation
  - Relevance of Padding
  - Stream Encryption Algorithms
  - Bit-Stream Ciphers using Block Ciphers

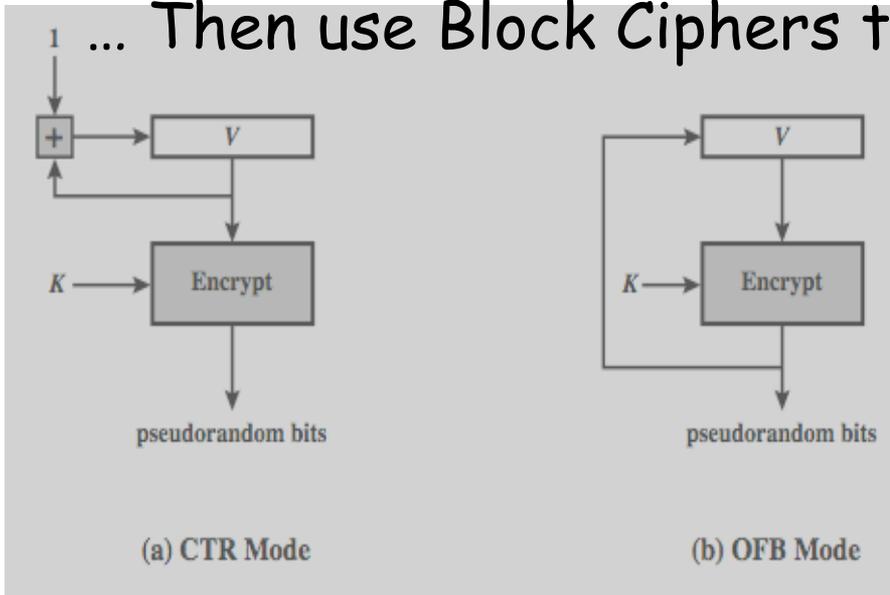
# Bit-Stream Encryption using Block Ciphers

- Can have real-time bit streaming using only block-ciphers ?
- Yes, look to the generic stream cipher architecture ...



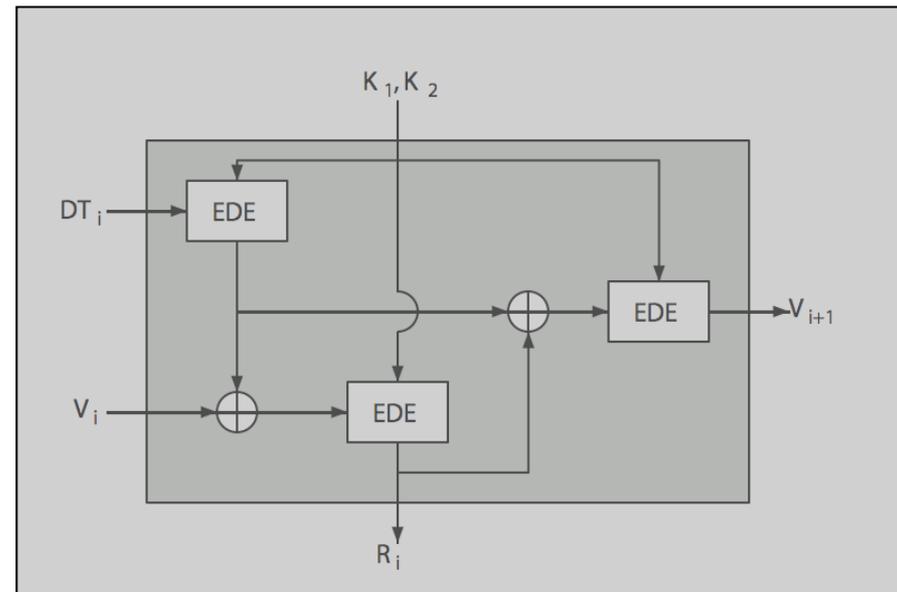
# Bit Streaming using Block Ciphers

... Then use Block Ciphers to implement PRG Boxes



Example of the  
ANSI X9.17 PRG Specification  
Using 3DES

*Keys  $K_1, K_2$  (+ opt  $K_3$ )*



*$DT_i$   
Initial Timestamp  
(or counter-state)*

*Seed Value*

Questions ?

# Recommended Reading (with the slides)

## Symmetric Encryption and Message Confidentiality



Reading:

W. Stallings, Network Security Essentials - Applications and Standards, Part I - Cryptography, Chap.2 - Symmetric Encryption and Message Confidentiality

# Additional Materials

## Complementary Materials on:

- Internal Structure of Symmetric Crypto Algorithms
- Feistel Structure and DES Algorithm
- AES Algorithm

# Ref. Feistel Structure

Horst Feistel,  
IBM, 1973

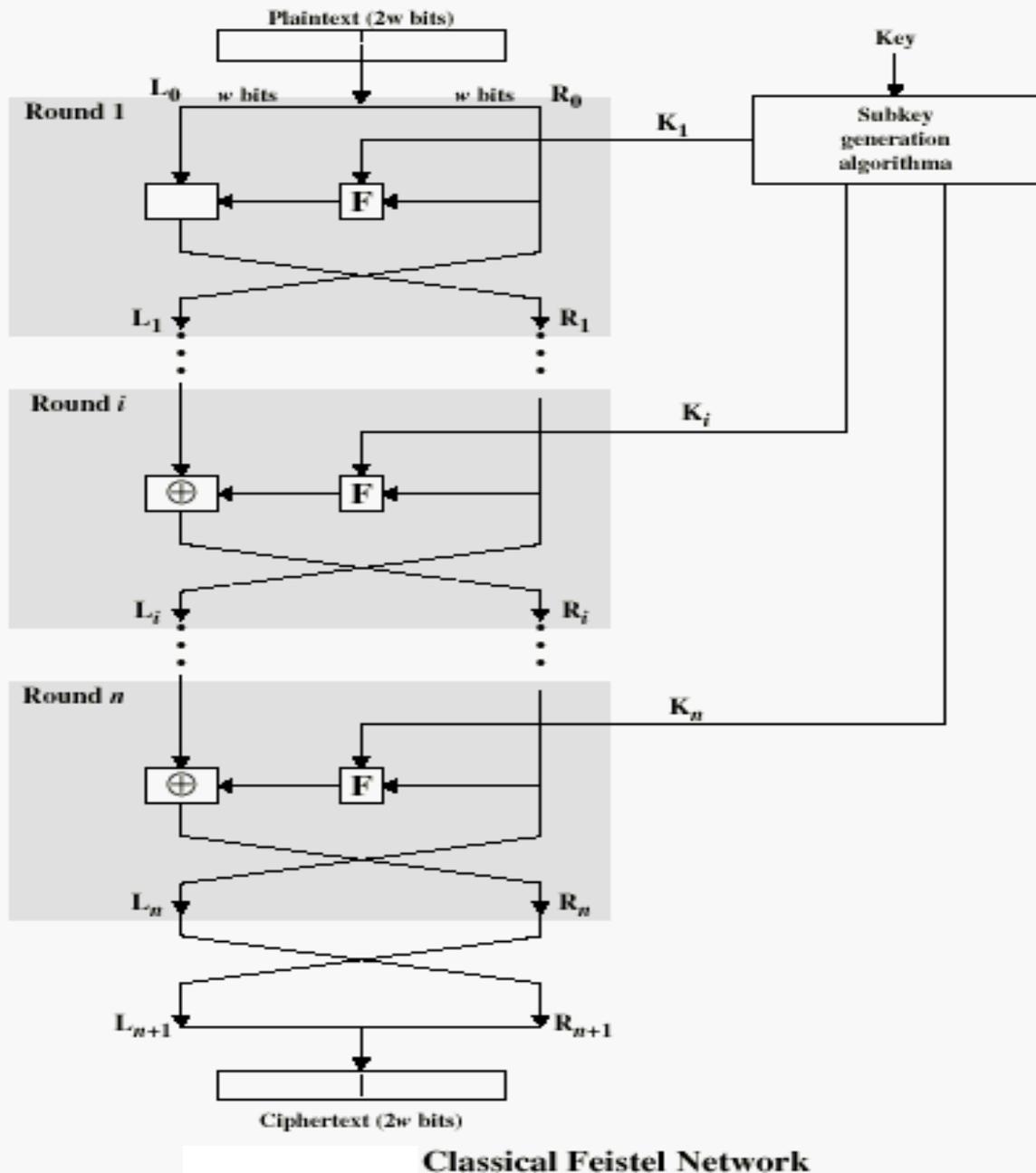
Base structure  
for many symmetric  
algorithms  
(ex., DES)

Decryption  
implemented as  
Encryption  
(input: ciphertext,  
and use of sub-keys  
in the reverse order)

In each round:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \text{ xor } F(R_{i-1}, K_i)$$



# Feistel Structure: summary to retain

- A generic structure for block-encryption algorithms, as a sequence of rounds (of core algorithm) performing substitutions and permutations, parameterized by a secret subkey
- Subkeys: derived in each round from a key-scheduling algorithm
- For a particular instantiation (specific algorithm), we must define:
  - The blocksize,
  - The keysize
  - Sub-key generation (scheduling) algorithm
  - The core round function (and related transformations)
  - Number of rounds for required confusion and diffusion

## Taking into account:

- Fast encryption/decryption
- SW and HW implementation possibility
- Core round and sub-key generation algorithm (ease of analysis for cryptanalysis)
- Complexity analysis and soundness proofs

# DES Algorithm

DES

## Data Encryption Standard (DES)

The most widely used encryption scheme, until 2000

Also referred as Data Encryption Algorithm (DEA)

DES is a block cipher

Input (plaintext) Blocks 64-bit blocks

The key: 56-bits in length



# Initial and final permutations

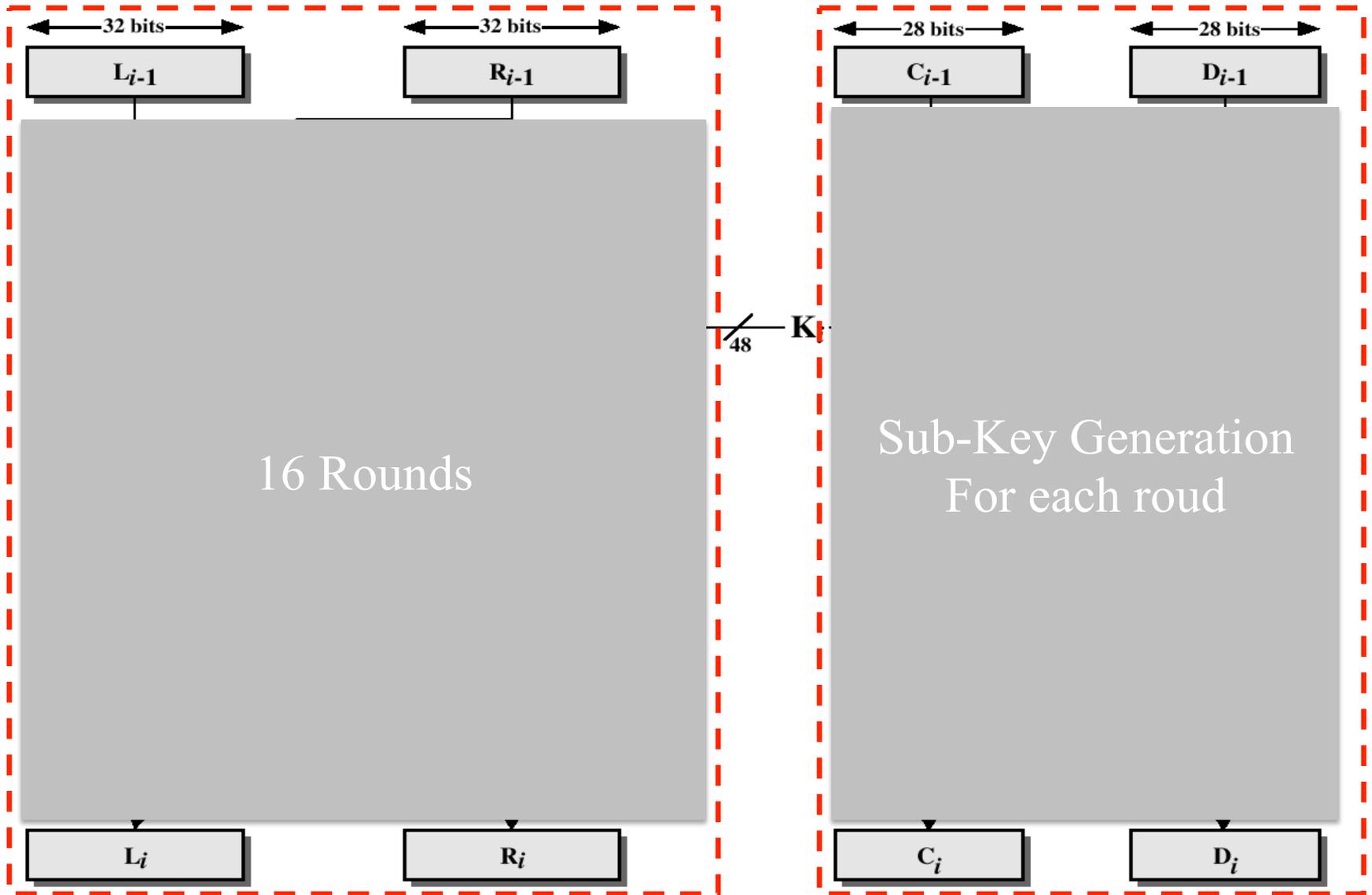
(a) Initial Permutation (IP)

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

(b) Inverse Initial Permutation ( $IP^{-1}$ )

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

Ex: Init Permut(675a6967 5e5a6b5a) = (ffb2194d 004df6fb)  
0110011101011010011010010110011101011110010110100110101101011010  
111111110110010...etc

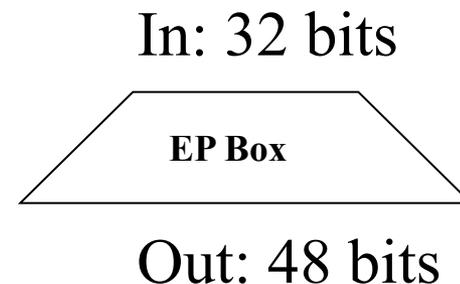


**Single Round of DES Algorithm**

# Expansion/Permutation (EP box) and Pbox

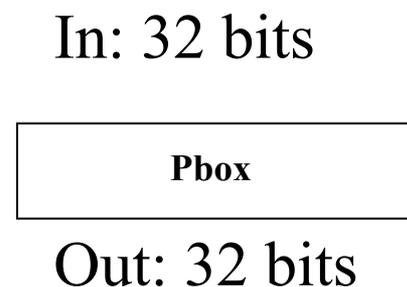
(c) Expansion Permutation (E)

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

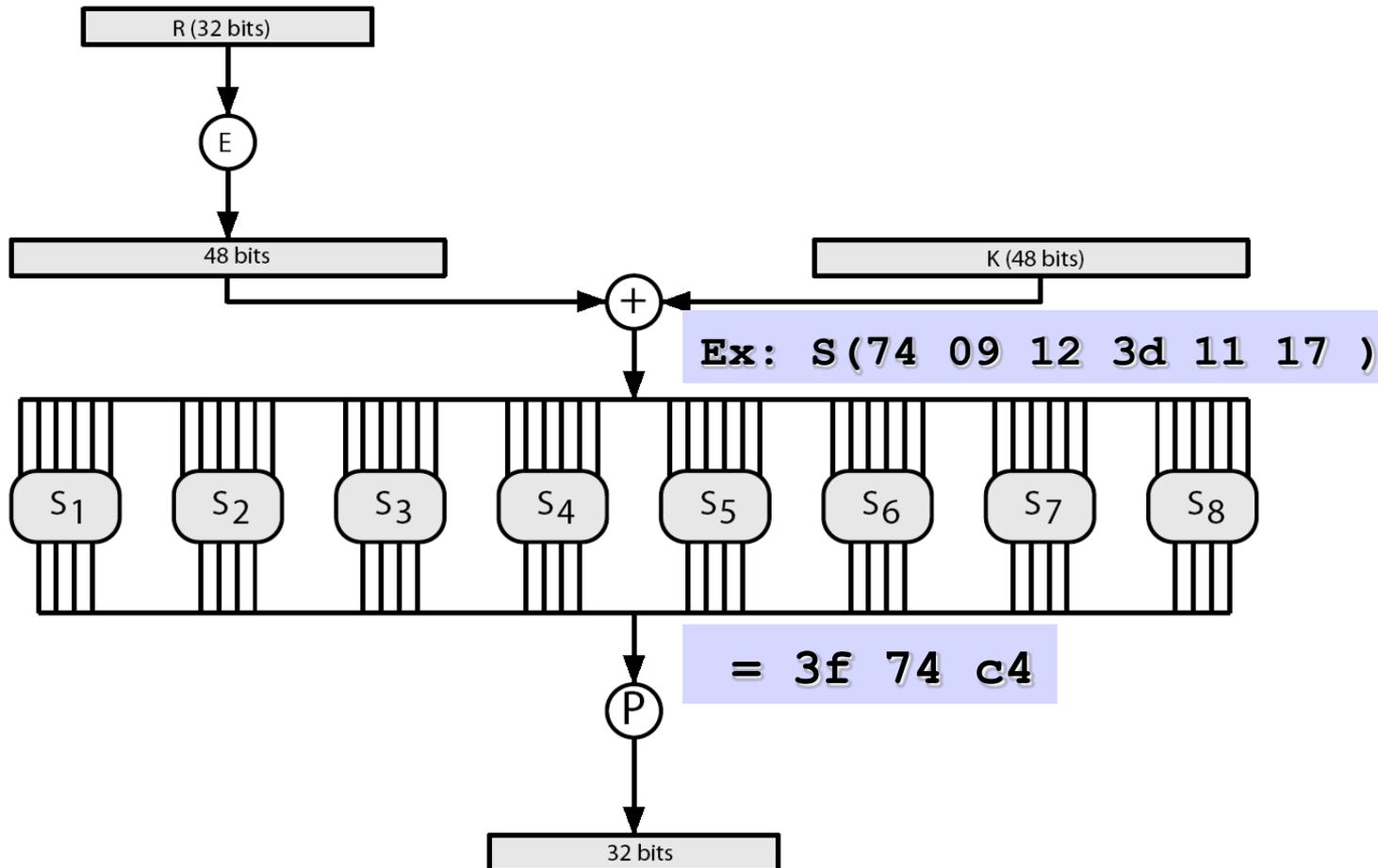


(d) Permutation Function (P)

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



# Substitution Boxes (for 16 rounds)



# S boxes (S1 to S4)

Valores de entrada: **0 (000000)** a **63 (111111)**

S<sub>1</sub>

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S<sub>2</sub>

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S<sub>3</sub>

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S<sub>4</sub>

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Ex:  
011101  
↓  
0011

Note: values in each sBOX: 0 to 15 (0000 a 1111)

Ex: 8º sexteto MS **011101** → **01** (row 1) **1110** (column 14)

Then: **011101** → 3, then output is **0011**

# S boxes (S5 a S8)

$S_5$

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

$S_6$

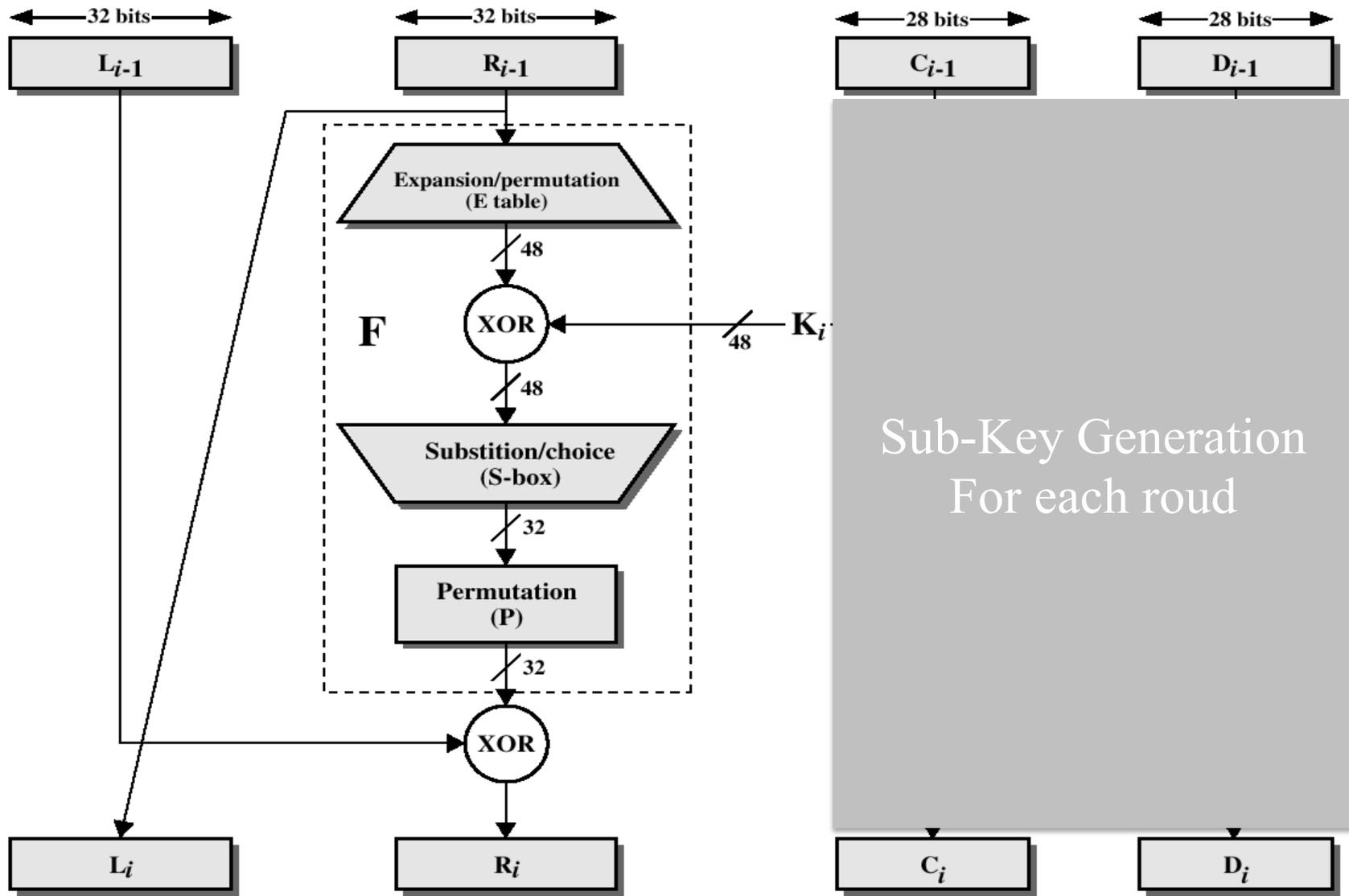
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

$S_7$

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

$S_8$

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11



Single Round of DES Algorithm

# Sub-key generation

- A sub-key is generated in each round
- PC1: two halves (28 bits) obtained by permutation
- In each round, each half is rotated for the next round

## Input Key (56 bits)

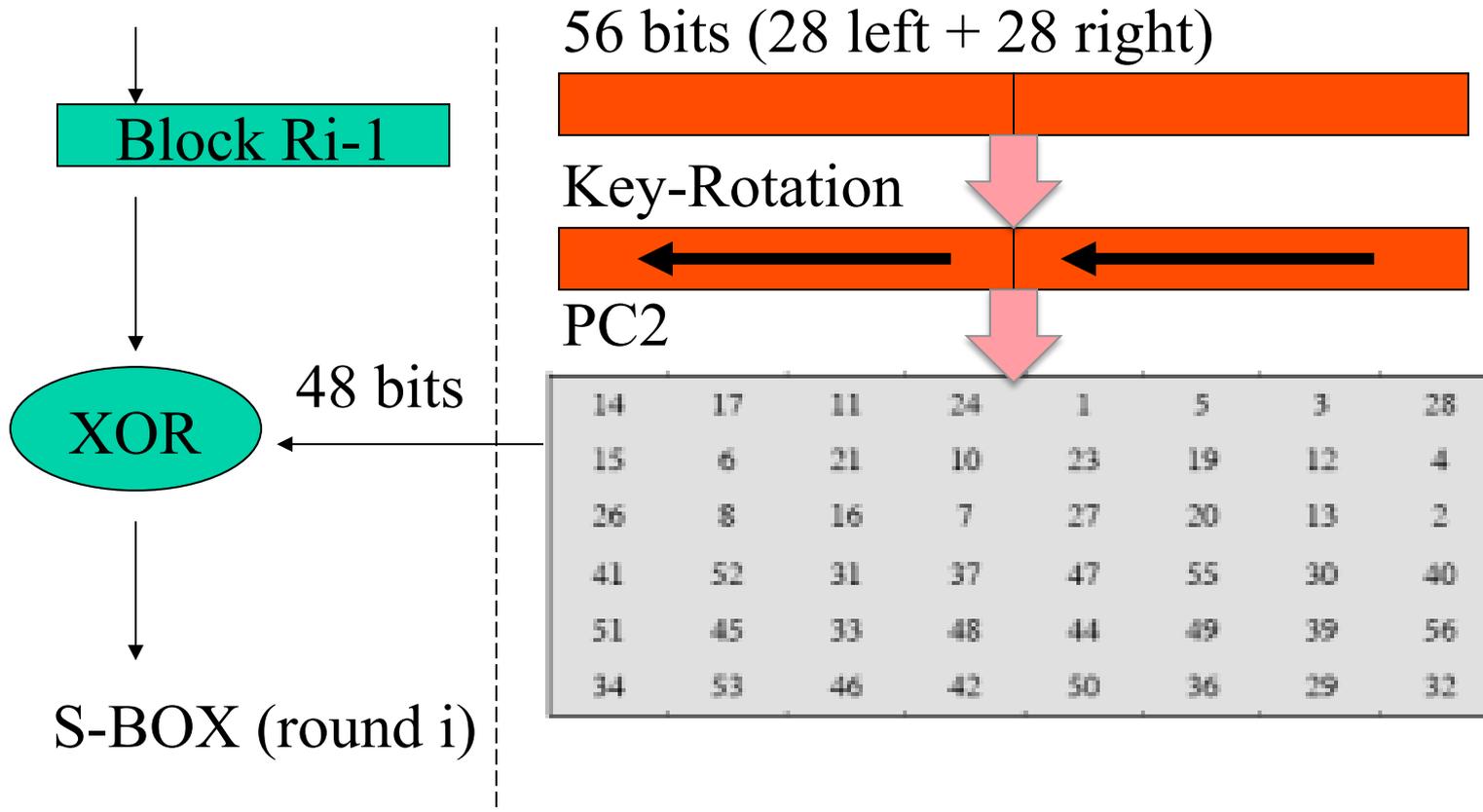
1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

## 2 Initial Sub-Keys (28 bits) (Initial Permutation PC1)

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

- 16 rotations (one per round): a specific rotation is defined for each round
- PC2: In each round, a permutation with reduction (56 > 48 bits)

# PC2 sub-key rescheduling



## Key-Rotation (for the next round)

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1



# DES Strengths and Security Analysis

- DES Standard (NIST DEA, FIPS46, 1977)
- From the algorithm viewpoint: probably one of the more studied algorithms in cryptanalysis - no one so far succeeded in discovering fatal weaknesses in DES (at least no one has credible publicly published acknowledged such a discovery)
- Form the keysize and blocksize viewpoints: 56 bits is to short !
  - Only possible  $7.2 \times 10^{16}$  keys
  - Brute-force attacks using one single computer (single PC):
    - 1 encryption-decryption / nanosecond  $\Rightarrow 2^{55}$  nanoseconds, 1125 years to break
  - But using  $10^4$  (parallelized) x PC computing power  $\Rightarrow$  1 hour max. to break it for sure
  - EFF DES Cracking Contest (Special cracker machine) in 1998: US\$ 250 K, 3 Days !
  - Off-the-Shelf Processors / Processor Cards
    - $10^9$  DES operations/second currently in Multicore HW Processors
    - $10^{13}$  DES operations is reasonably expectable in the short term !

# Symmetric Block Encryption Algorithms

Multiple Encryption and Onion-Encryption Constructions

# DES and Multiple Encryption Methods

- Clear a replacement for DES was needed (short key size)
  - Theoretical attacks that can break it
  - Demonstrated exhaustive key search attacks
  - Cryptanalysis: demonstrated vulnerabilities
  - Weak, Semi-Weak and Potentially Weak keys
- AES Context: AES as the new cipher alternative
  - The “standard” symmetric encryption for the XXI century
  - Promoted by different organizations: NIST, ANSI, etc...
- Prior to new alternatives people start using multiple encryption with DES implementations
- Triple-DES is the chosen form
  - Compatibility, during the progressive adoption of AES

# Double-DES ? No thanks...

- Could use 2 DES encrypts on each block

$$C = [ E_{K_2} ( E_{K_1}(P) ) ]$$

- Issue of reduction to single stage with single key (1992)
- ... and have "meet-in-the-middle" attack
  - Works whenever use a cipher twice
  - Since  $X = E_{K_1}(P) = D_{K_2}(C)$
  - Attack by encrypting  $P$  with all keys and store
  - Then decrypt  $C$  with keys and match  $X$  value
  - Can show takes  $O(2^{56})$  steps

# Triple-DES with Two-Keys (DES-EDE-2)

- hence must use 3 encryptions
  - would seem to need 3 distinct keys
- but can use 2 keys with E-D-E sequence
$$C = E_{K_1}(D_{K_2}(E_{K_1}(P)))$$
  - Encrypt & decrypt equivalent in security
  - If  $K_1=K_2$  then can work with single DES (compatibility)
- standardized in ANSI X9.17 & ISO8732
- Effective key length of 112 bits
- no current known practical attacks
  - But the use of 3 keys is better (improvement of the key-length effect)

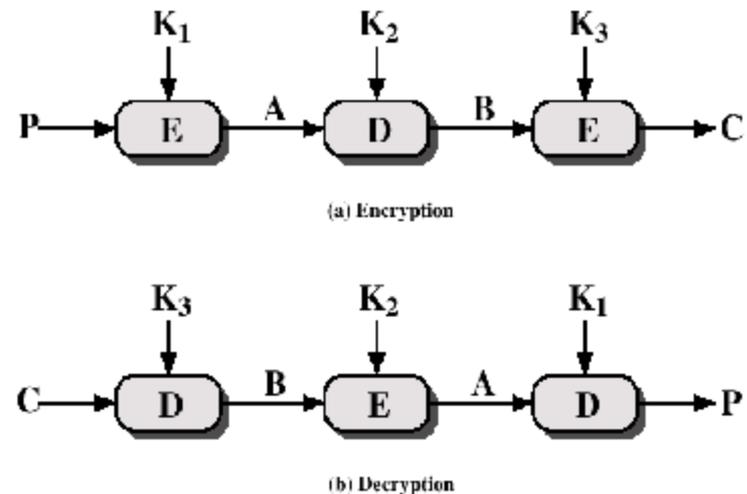
# Triple DES with 3 keys (DES-EDE-3)

- Although there are no practical attacks on two-key TripleDES, there are some indications
- can use Triple-DES with Three-Keys to avoid even these

$$C = [ E_{K_3} [ D_{K_2} ( E_{K_1}(P) ) ] ]$$

- has been adopted by some Internet applications, eg PGP, S/MIME
- Effective key length: 168 bits

Paranoid? Use N times DEA with N different keys  
Problem?



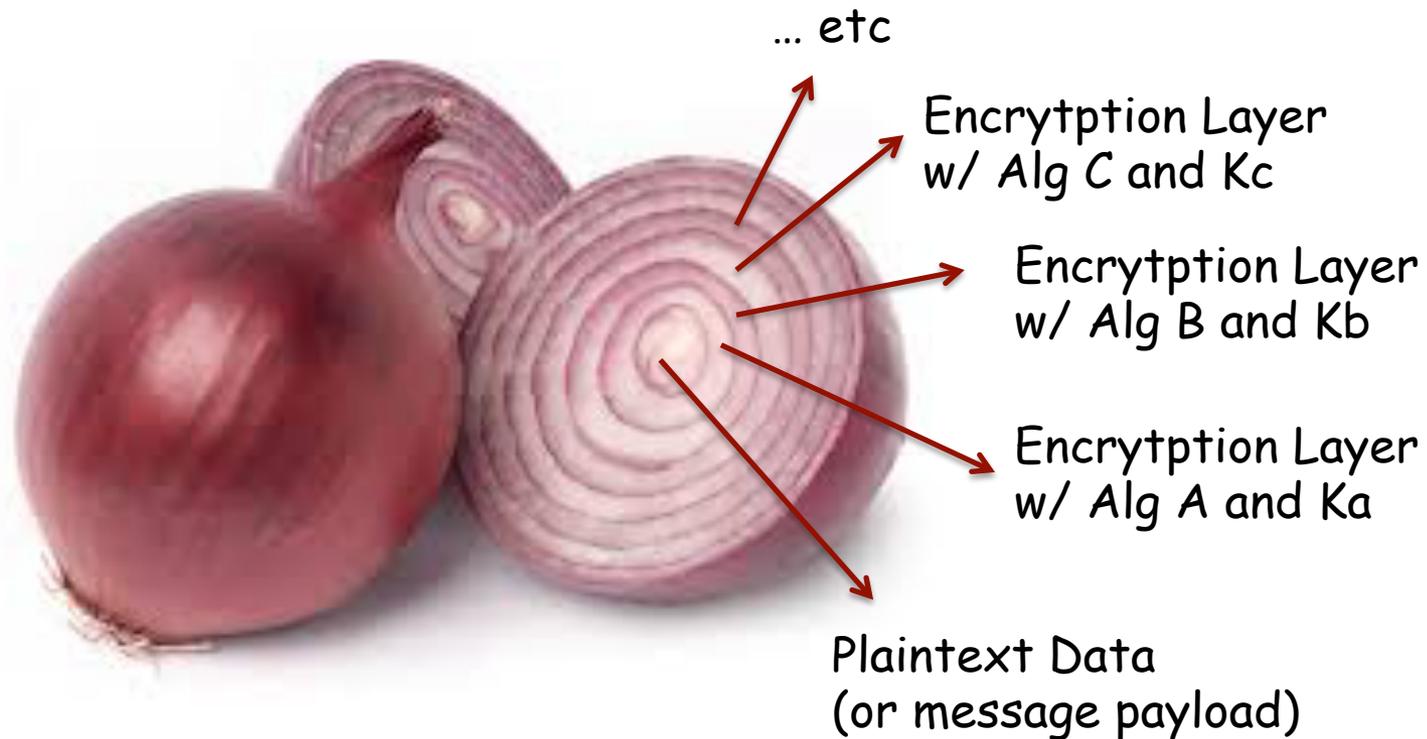
Note:  
Triple or N-encryption-decryption can be applied to any block cipher algorithm

# Other multiple encryption schemes can be used in general for any Sym. Block Algorithm

- TEMK - Triple Encryption with Minimum Key  
Constants  $T_1, T_2$  and  $T_3$  and Two initial Keys:  $K_1, K_2$  to derive three keys:  $KX_1, KX_2, KX_3$  for triple encryption  
 $KX_i = D(K_2, E(K_1, T_i))$
- Use of Triple encryption modes (ex., Inner CBC, Outer CBC)
- Triple encryption with prefix-pads (pad size < block size)
- Doubling Block Length Space
- Ntuple Encryption: ex.,  $n=5, 3$  keys,  
 $C = E_{K_1}(D_{K_2}(E_{K_3}(D_{K_4}(E_{K_5}(P))))))$
- Whitening technique:  $C = K_3 \text{ XOR } E(K_2, (P \text{ XOR } K_1))$
- Cascading multiple block ciphers: it is at least as hard to break as any of its component ciphers
- Combination technique
  - Random bit-string  $R$  with the same size as  $M$
  - $C = E(K_1, R) \text{ XOR } E(K_2, (M \text{ XOR } R))$

# Onion-Encryption Constructions or Schemes

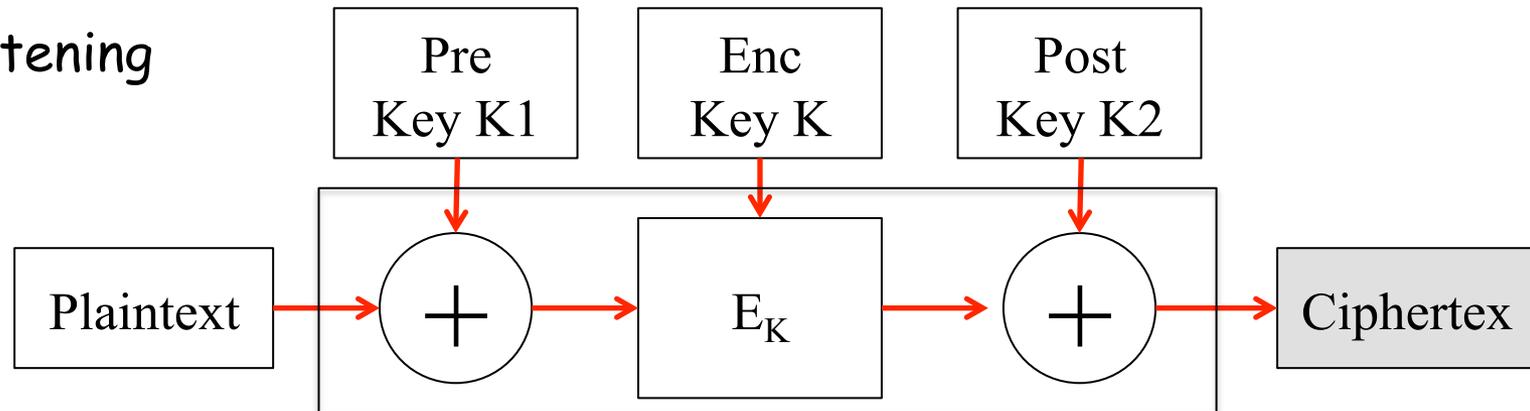
- More robustness combining different encryption algorithms in a multiple-encryption processing approach



Problems ?

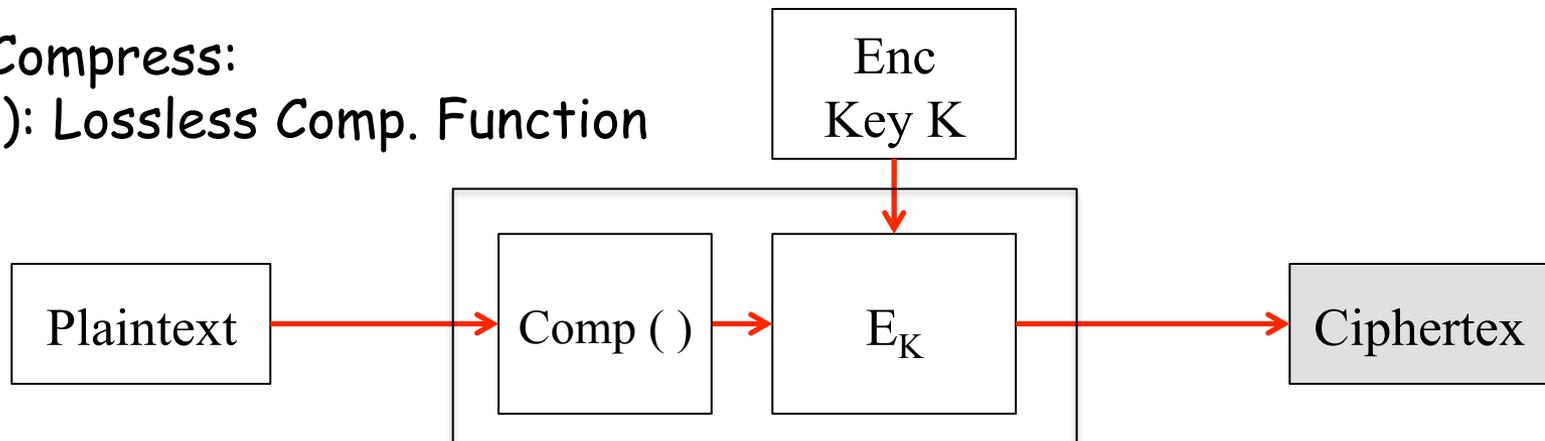
# Other usual techniques: Whitening or Pre-Compressing

Whitening



$$C_i = K_2 \text{ XOR } E_k (P_i \text{ XOR } K_1)$$

Pre-Compress:  
Com(): Lossless Comp. Function



$$C_i = E_k ( \text{Comp}(P_i) )$$

# Symmetric Block Encryption Algorithms

Symmetric Block Encryption Algorithms

# Other Symmetric Block Ciphers

- **International Data Encryption Algorithm (IDEA)**
  - 128-bit key
  - Used in PGP
- **Blowfish**
  - Easy to implement
  - High execution speed , Run in less than 5K of memory
- **Cast-128**
  - Key size from 40 to 128 bits
  - The round function differs from round to round
- **RC5**
  - Suitable for hardware and software
  - Fast, simple, Low memory requirement
  - Adaptable to processors of different word lengths
  - Variable number of rounds, Variable-length key
  - High security with Data-dependent rotations

# Symmetric block cyphers and characteristics

<b>Alg.</b>	<b>Key size</b>	<b>Rounds</b>	<b>Structure Operations</b>	<b>Applications (standadization)</b>
<b>DES</b>	<b>56</b>	<b>16</b>	<b>XOR, S-boxes, P-Boxes EP Boxes, Rotations, ...</b>	<b>SET, Kerberos</b>
<b>3DES</b>	<b>112/168</b>	<b>48</b>	<b>idem</b>	<b>Financial standards E-commerce, PGP S/MIME, SSL</b>
<b>IDEA</b>	<b>128</b>	<b>8</b>	<b>XOR, addition mod <math>2^{16}</math> , Multiplication mod <math>2^{16}</math></b>	<b>PGP, SSL</b>
<b>TEA</b>	<b>128</b>	<b>32</b>	<b>XOR + shift</b>	
<b>Blowfish</b>	<b>&lt; 448</b>	<b>16</b>	<b>XOR + Variable S-boxes</b>	
<b>RC5</b>	<b>&lt; 2048</b>	<b>&lt; 255</b>	<b>Addition, subtraction XOR and rotation operations</b>	<b>SSL</b>
<b>Cast-128</b>	<b>40-128</b>	<b>16</b>	<b>Addition, Subtraction, XOR, Rotation and S-boxes</b>	<b>PGP</b>
<b>AES</b>	<b>128, 192, 256</b>	<b>10,12,14</b>	<b>S boxes, Matrix SR-MC, XOR</b>	<b>Many: promoted as a standard after 2000</b>

# AES (Advanced Encryption Standard)

## Origins and characteristics

- clear a replacement for DES was needed
  - have theoretical attacks that can break it
  - have demonstrated exhaustive key search attacks
- can use Triple-DES - but slow, has small blocks
- US NIST issued call for ciphers in 1997
- 15 candidates accepted (Jun/98), 5 were shortlisted in Aug/99
- Rijndael was selected as the AES in Oct-2000
- issued as FIPS PUB 197 standard in Nov-2001
- designed by Rijmen-Daemen in Belgium
- an **iterative** rather than **feistel** cipher
  - processes data as block of 4 columns of 4 bytes
  - operates on entire data block in every round
- designed to be:
  - resistant against known attacks
  - speed and code compactness on many CPUs
  - Simple to analyze

# AES Requirements

## NIST

Established the following reference criteria for proposals:

- Secret key / symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- stronger & faster than Triple-DES
- active life of 20-30 years (+ archival use)
- provide full specification & design details
- both C & Java implementations
- NIST have released all submissions & unclassified analyses

# AES structure

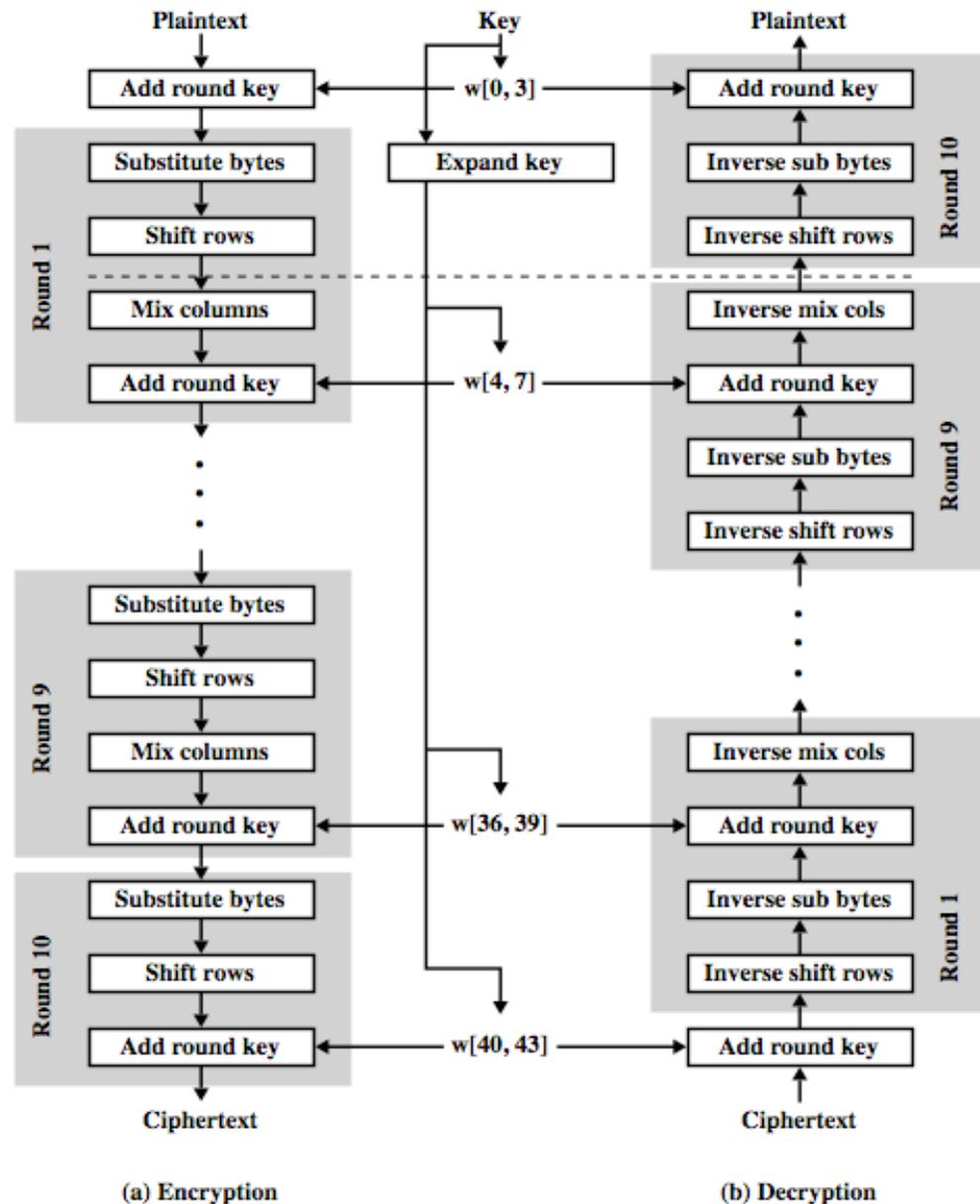
Not a Feistel structure ...

- Data block of 4 columns of 4 bytes is state (128 bits)
- Key is expanded to array of words (44 words of 32 bits)
  
- Has 9/11/13 rounds in which state undergoes:
  - **I** : **byte substitution** (1 S-box used on every byte)
  - **II** : **shift rows** (permute bytes between groups/columns)
  - **III** : **mix columns** (subs using matrix multiply of groups)
  - **IV** : **add round key** (XOR state with key material)
  - view as alternating XOR key & scramble data bytes
  
- Initial XOR key material & incomplete last round
- With fast XOR & table lookup implementation

# Advanced Encryption Algorithm (AES)

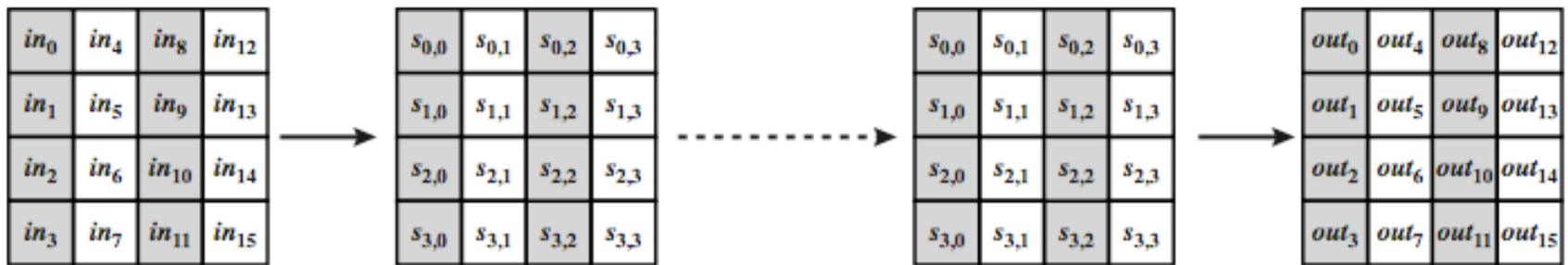
- Internal structure
- Ex., 10 rounds for 128 bit keys
- Successive processing of block-state arrays, combined with the expanded key

128 bit sub-key  
in each round

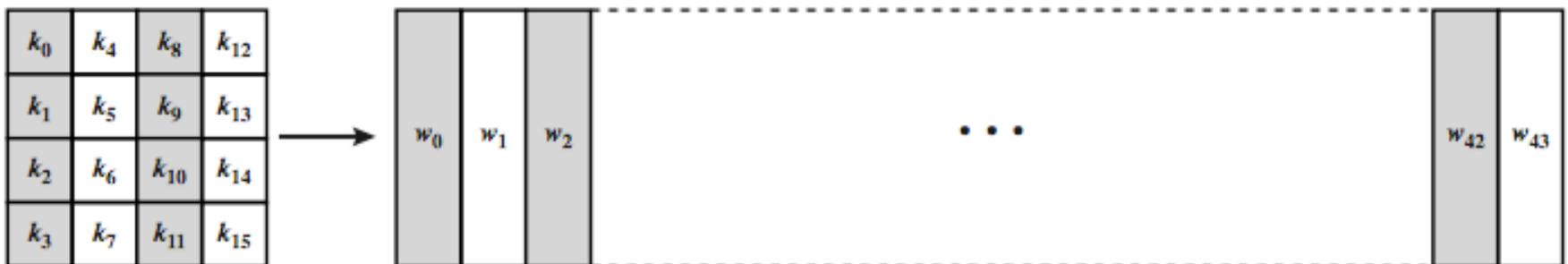


# Input/Output State Array and Key Expansion

16 rounds of successive transformation of a 128 bit blocks (state arrays), organized as a square matrix of bytes (16x16 bits, ordered by column)

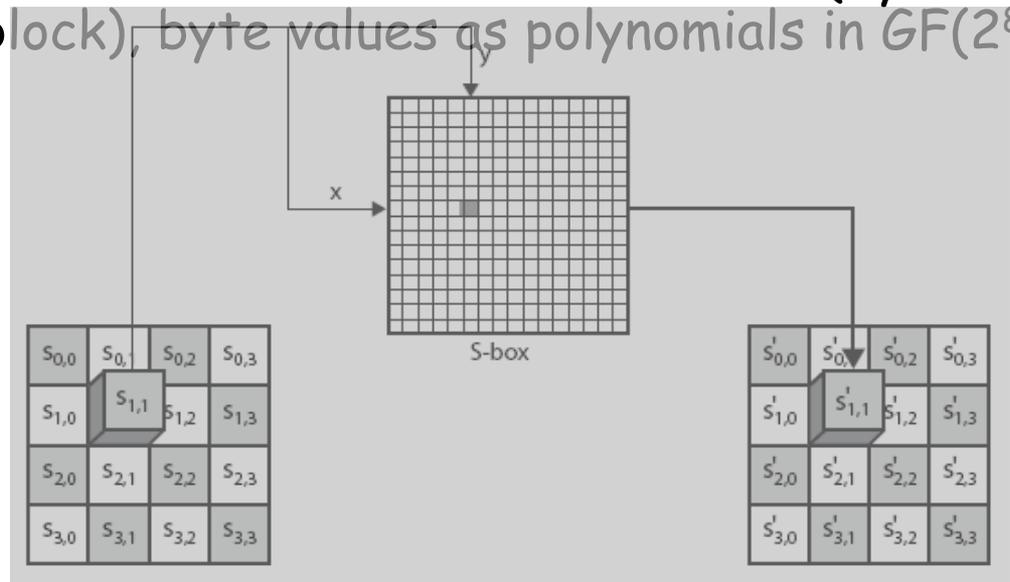


Key expanded from initial size (as a square matrix of bytes) to 44 columns (44 words, 4 bytes each)



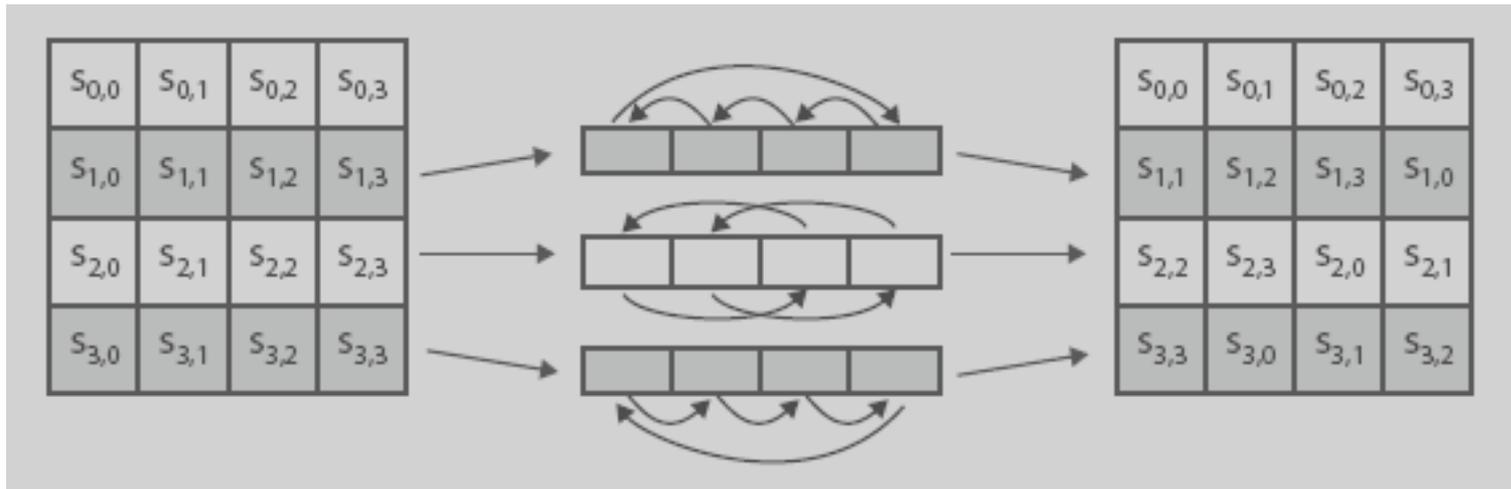
# Byte Substitution

- A simple substitution of each byte (defined)
- Uses one table 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 (byte-to-byte substitution of the block), byte values as polynomials in  $GF(2^8)$
- designed to be resistant to all known attacks



# Shift Rows

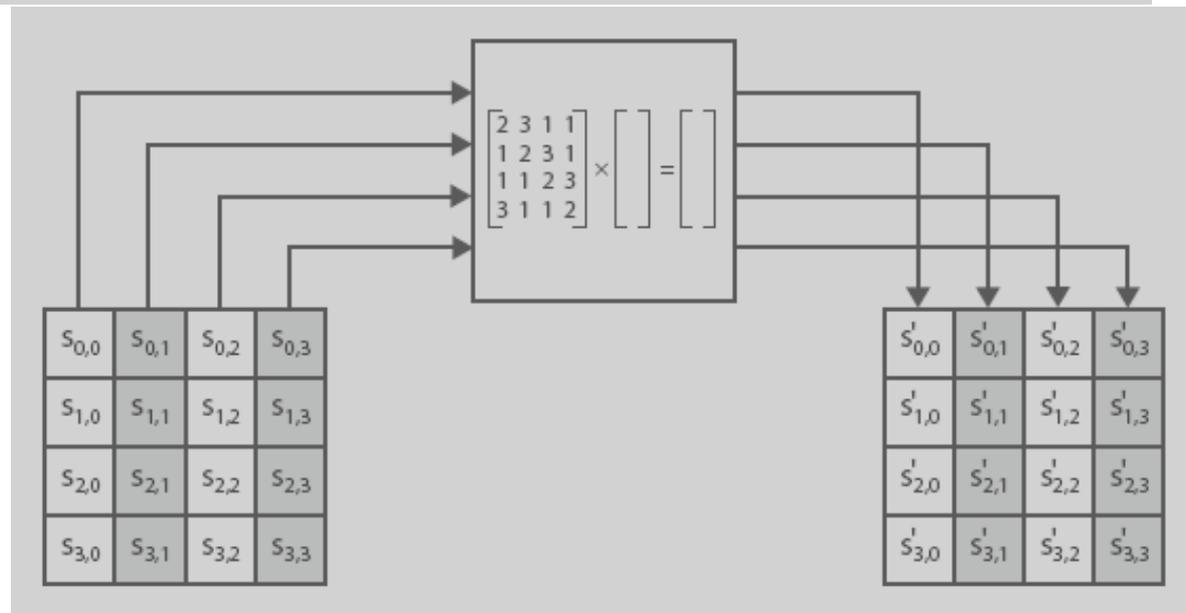
- a circular byte shift in each each
  - 1<sup>st</sup> row is unchanged
  - 2<sup>nd</sup> row does 1 byte circular shift to left
  - 3<sup>rd</sup> row does 2 byte circular shift to left
  - 4<sup>th</sup> row does 3 byte circular shift to left
- decrypt inverts using shifts to right
- since state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



# Mix Columns

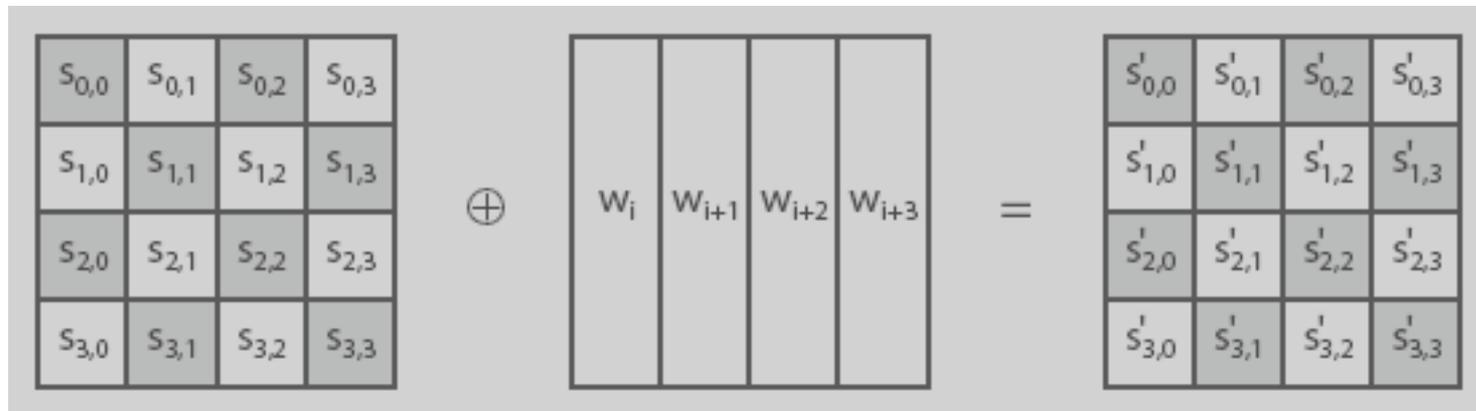
- each column is processed separately / each byte is replaced by a value dependent on all 4 bytes in the column
- effectively a matrix multiplication in  $GF(2^8)$  using prime poly  $m(x) = x^8 + x^4 + x^3 + x + 1$

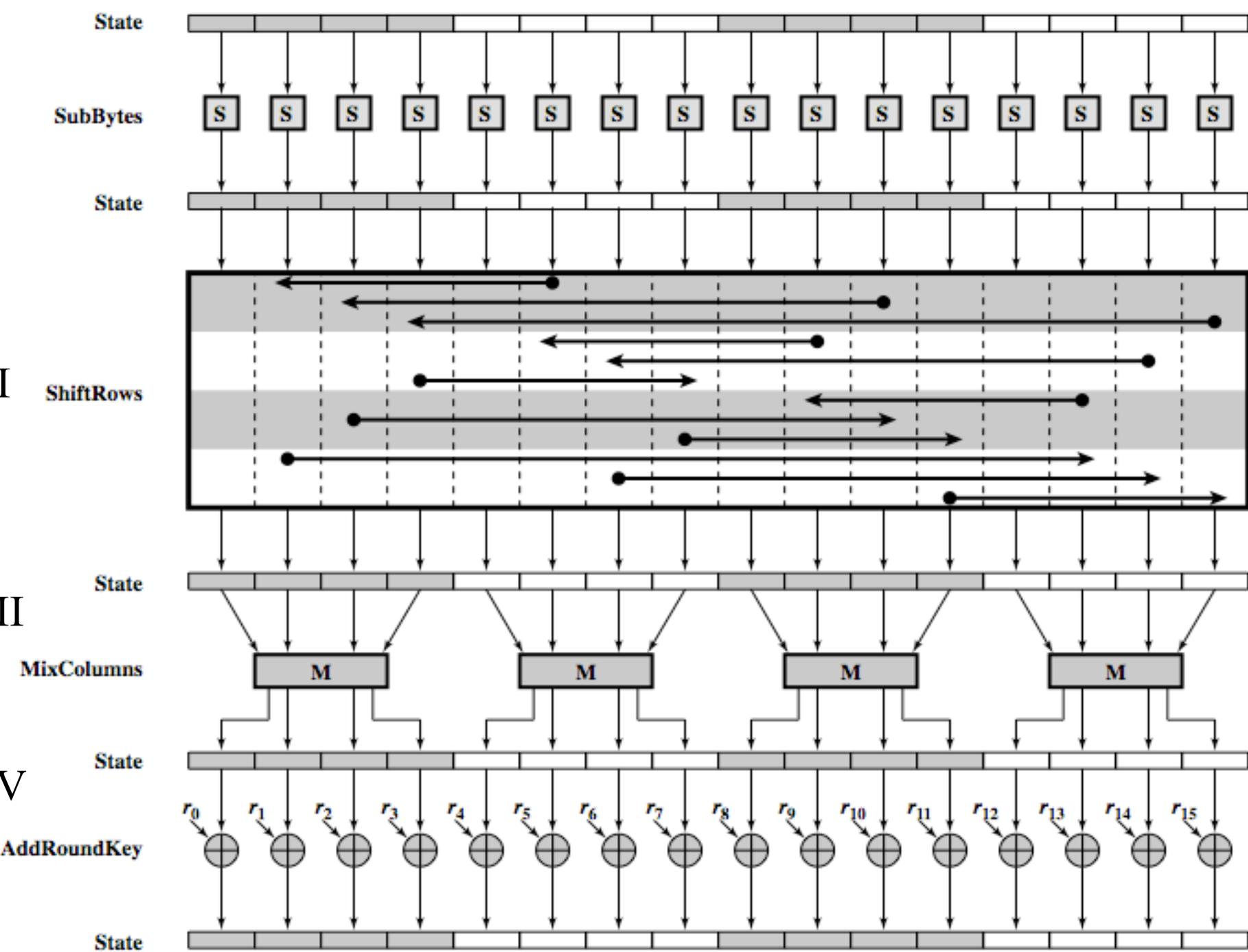
$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}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



# 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
  - a form of Vernam cipher on expanded key
  - requires other stages for complexity / security





# Outline

- **Symmetric cryptography**
  - Symmetric Encryption
    - Block Encryption Algorithms
  - Robustness of symmetric encryption algorithms
  - Internal structure of symmetric algorithms
  - Cipher block modes of operation
  - Relevance of Padding
  - Stream Encryption Algorithms
  - Bit-Stream Ciphers using Block Ciphers