Data Security and Chaos-based Data Security

Safwan El Assad

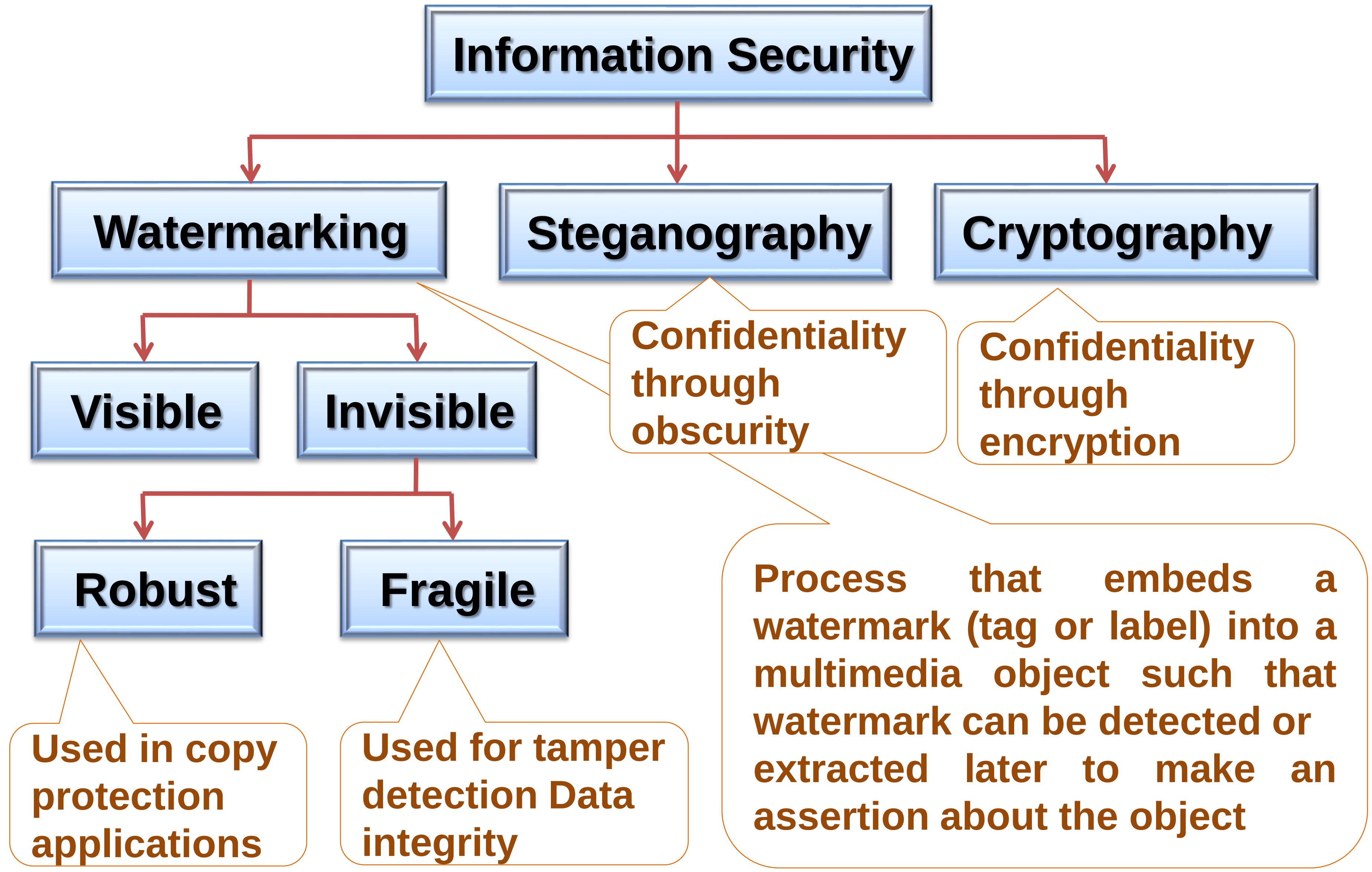
Polytech Nantes, school of engineering of the university of Nantes – France

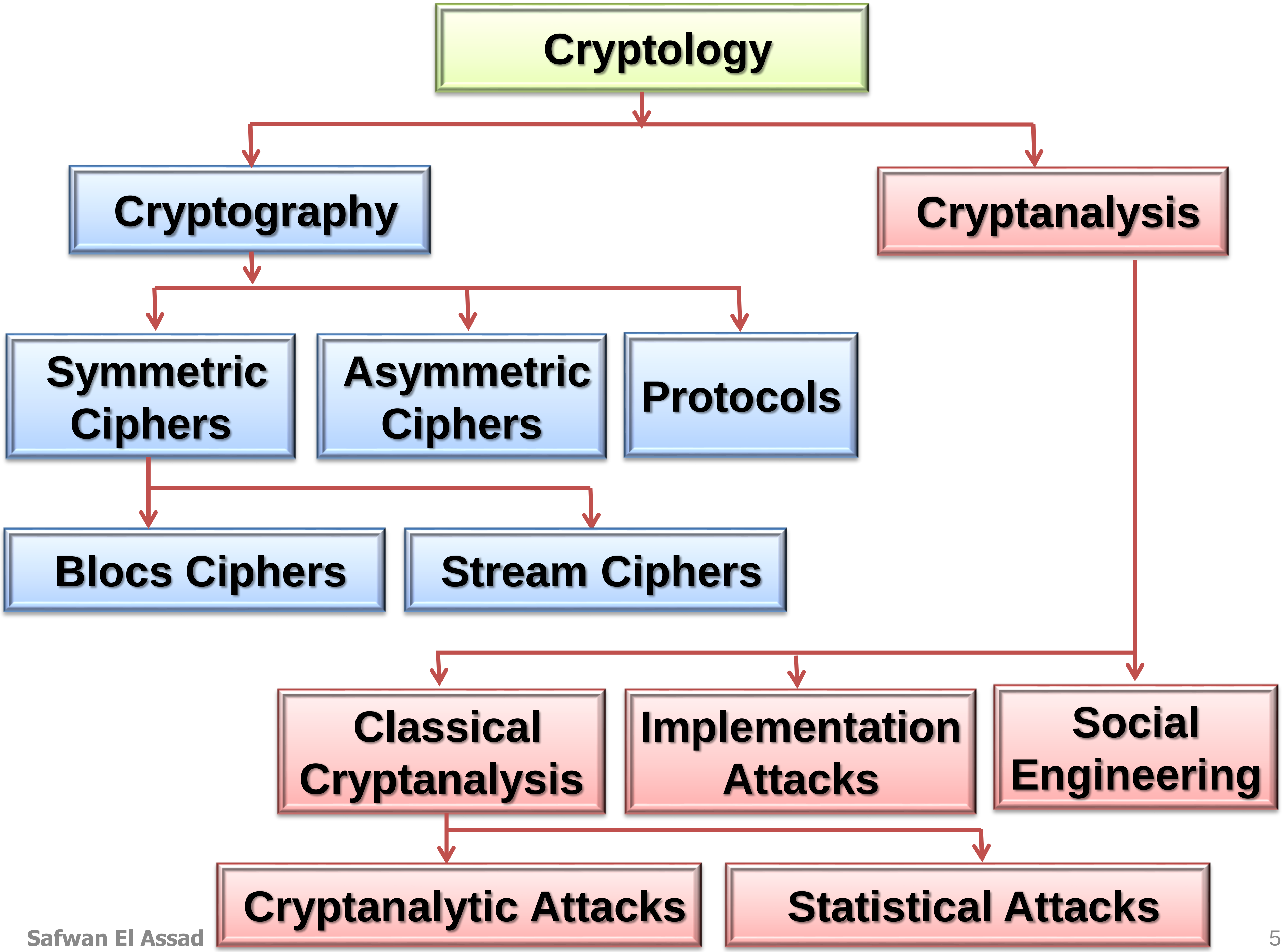
IETR Laboratory, UMR CNRS 6164; VAADER team - site of Nantes

SECURE DATA EXCHANGE
protect your digital data

Outline

Generalities


Information Security
Watermarking
Steganography
Cryptography
Visible
Invisible
Robust
Fragile
Confidentiality through obscurity
Confidentiality through encryption
Used in copy protection applications
Used for tamper detection Data integrity
Process that embeds a watermark (tag or label) into a multimedia object such that watermark can be detected or extracted later to make an assertion about the object

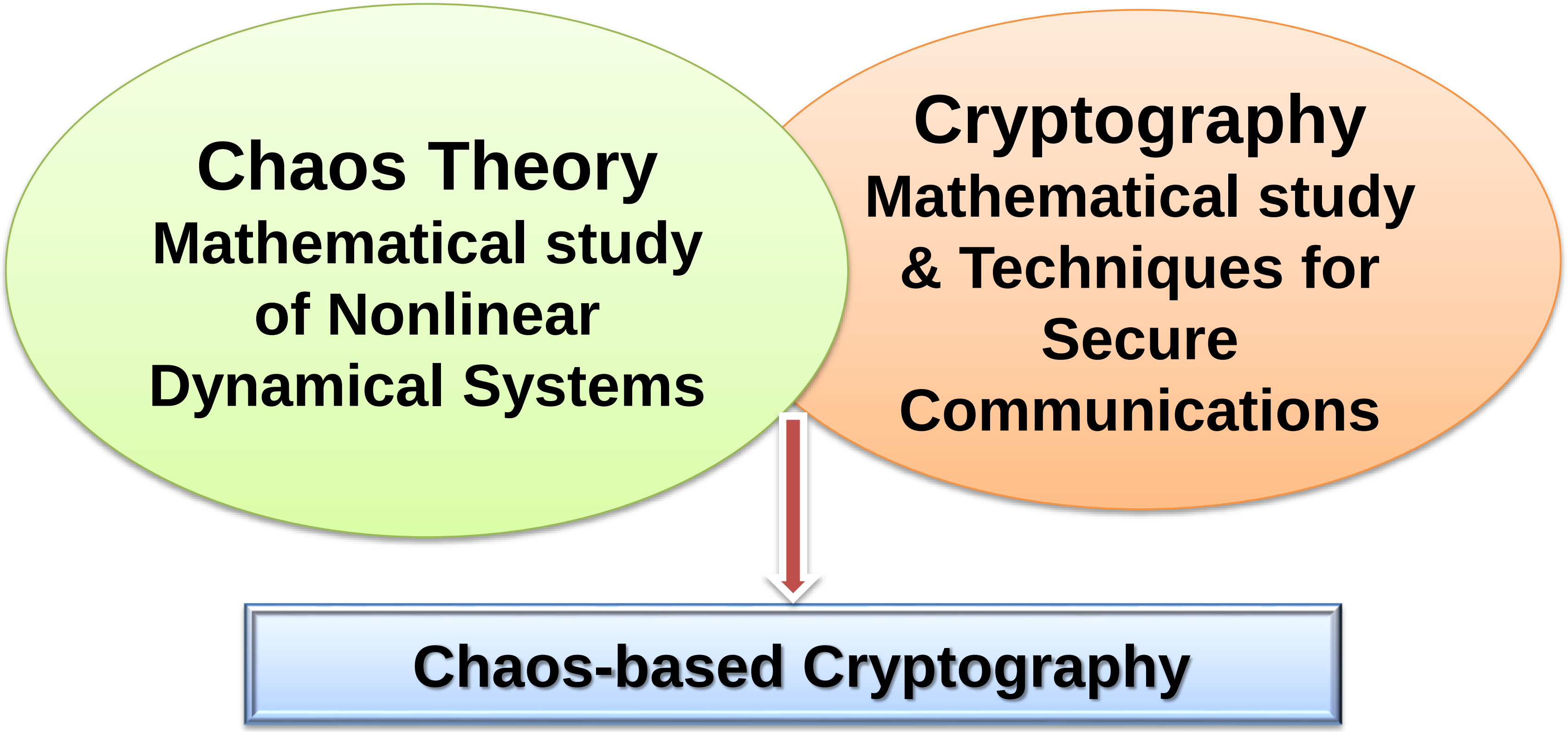

Cryptology
Cryptography
Cryptanalysis
Symmetric Ciphers
Asymmetric Ciphers
Protocols
Blocs Ciphers
Stream Ciphers
Classical Cryptanalysis
Implementation Attacks
Social Engineering
Cryptanalytic Attacks
Statistical Attacks

Chaos & Cryptography

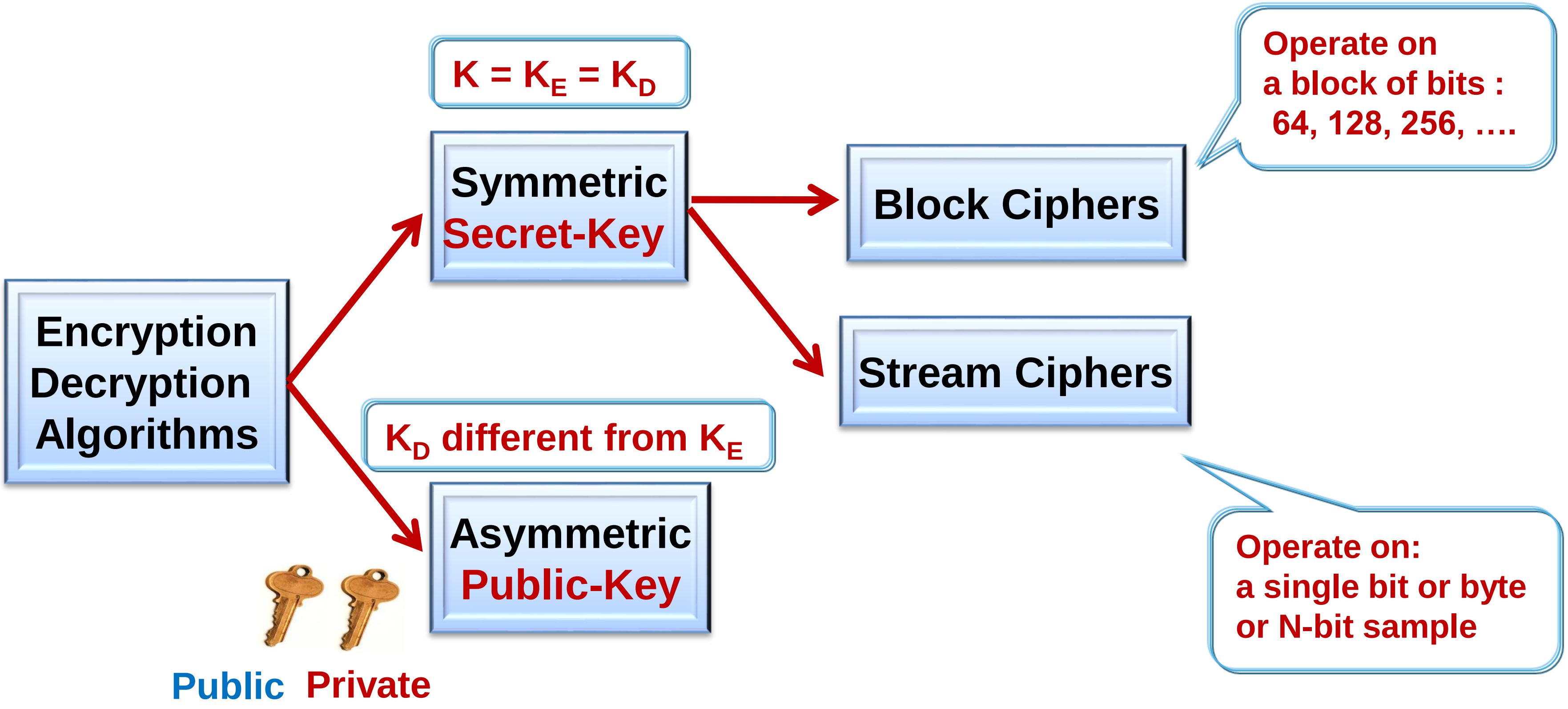
Both chaotic map and encryption system are deterministic

Both are unpredictable, if the secret key is not known

Both used iterative transformation


Chaos Theory
Mathematical study of Nonlinear Dynamical Systems
Cryptography
Mathematical study & Techniques for Secure Communications
Chaos-based Cryptography

Type of classical Encryption/Decryption algorithms


Encryption Decryption Algorithms
$K = K_E = K_D$
Symmetric Secret-Key
K_D different from K_E
Asymmetric Public-Key
Public
Private
Block Ciphers
Stream Ciphers
Operate on a block of bits : 64, 128, 256, ….
Operate on: a single bit or byte or N-bit sample

Symmetric encryption is # 1000

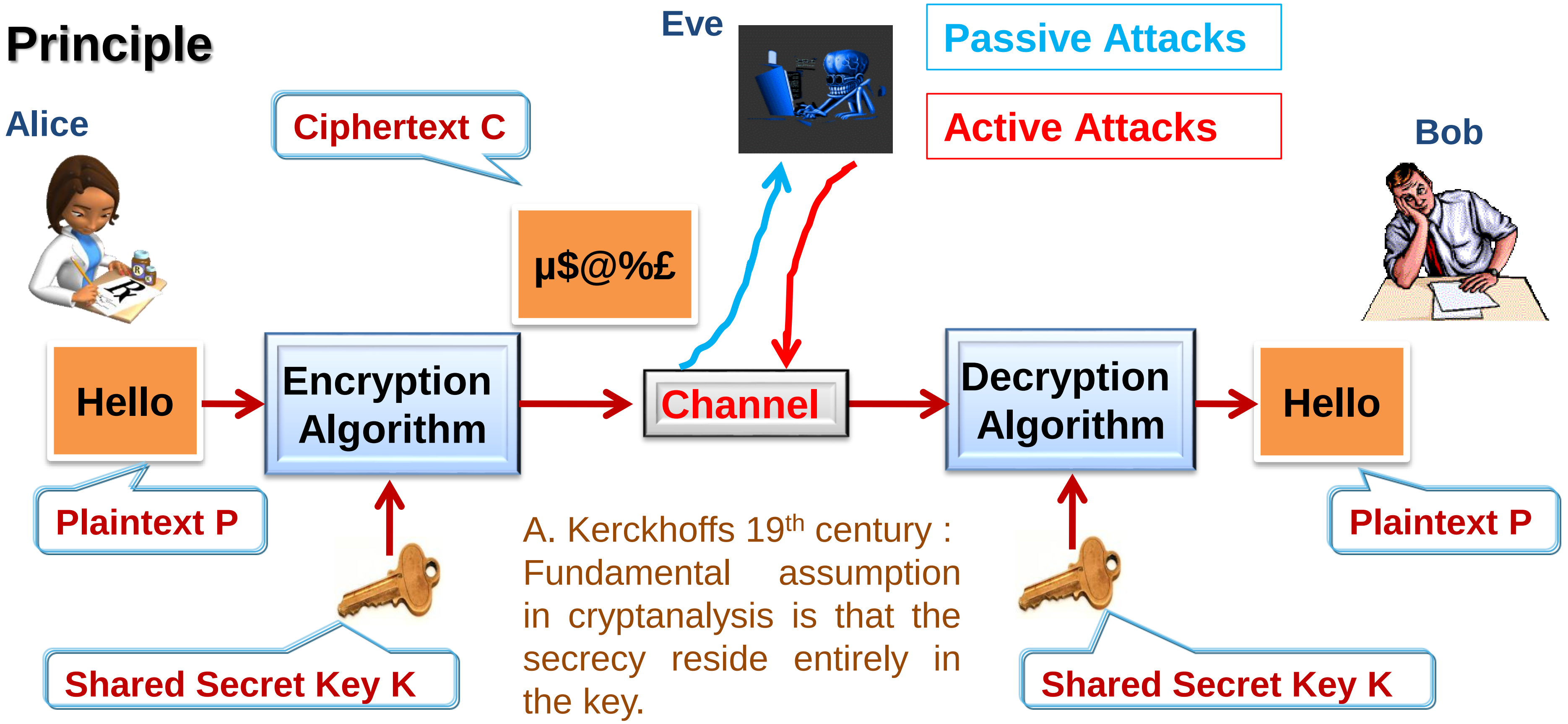
faster than asymmetric encryption

Classical cryptography

Encryption-Decryption

Symmetric key block cipher algorithms

Principle

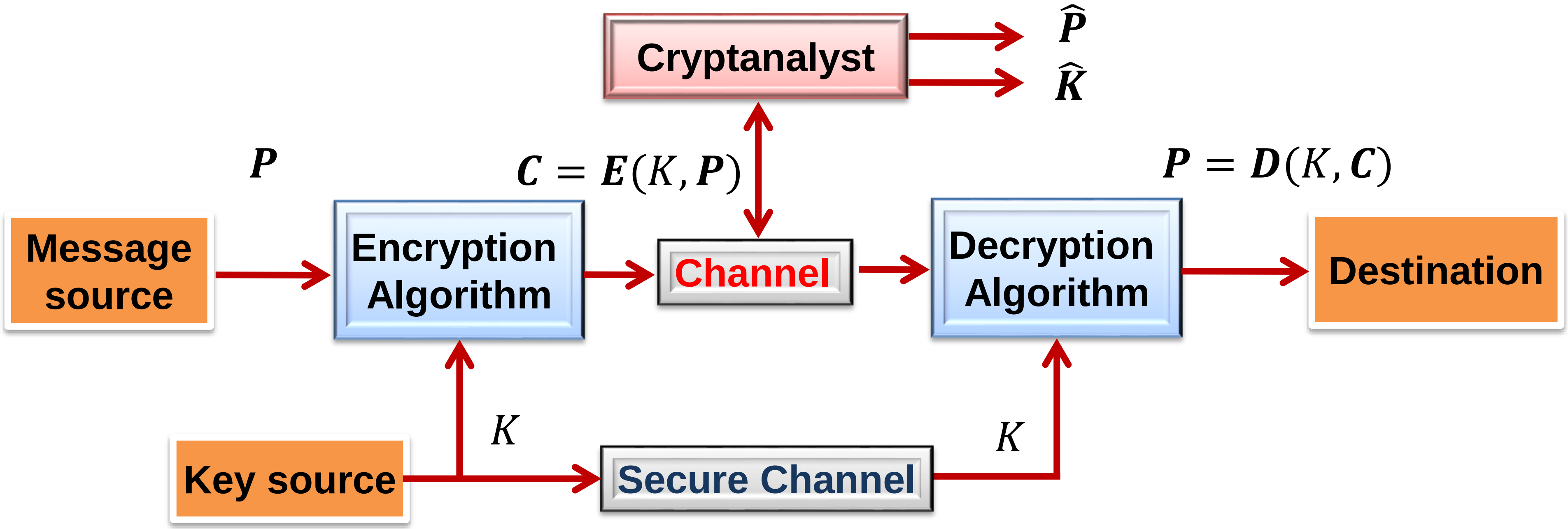


A. Kerckhoffs 19th century : Fundamental assumption in cryptanalysis is that the secrecy reside entirely in the key.

Passive attacks: Pb of Confidentiality

Active attacks: Pb of Data Integrity and Message Authentication

Model of Symmetric Cryptosystem



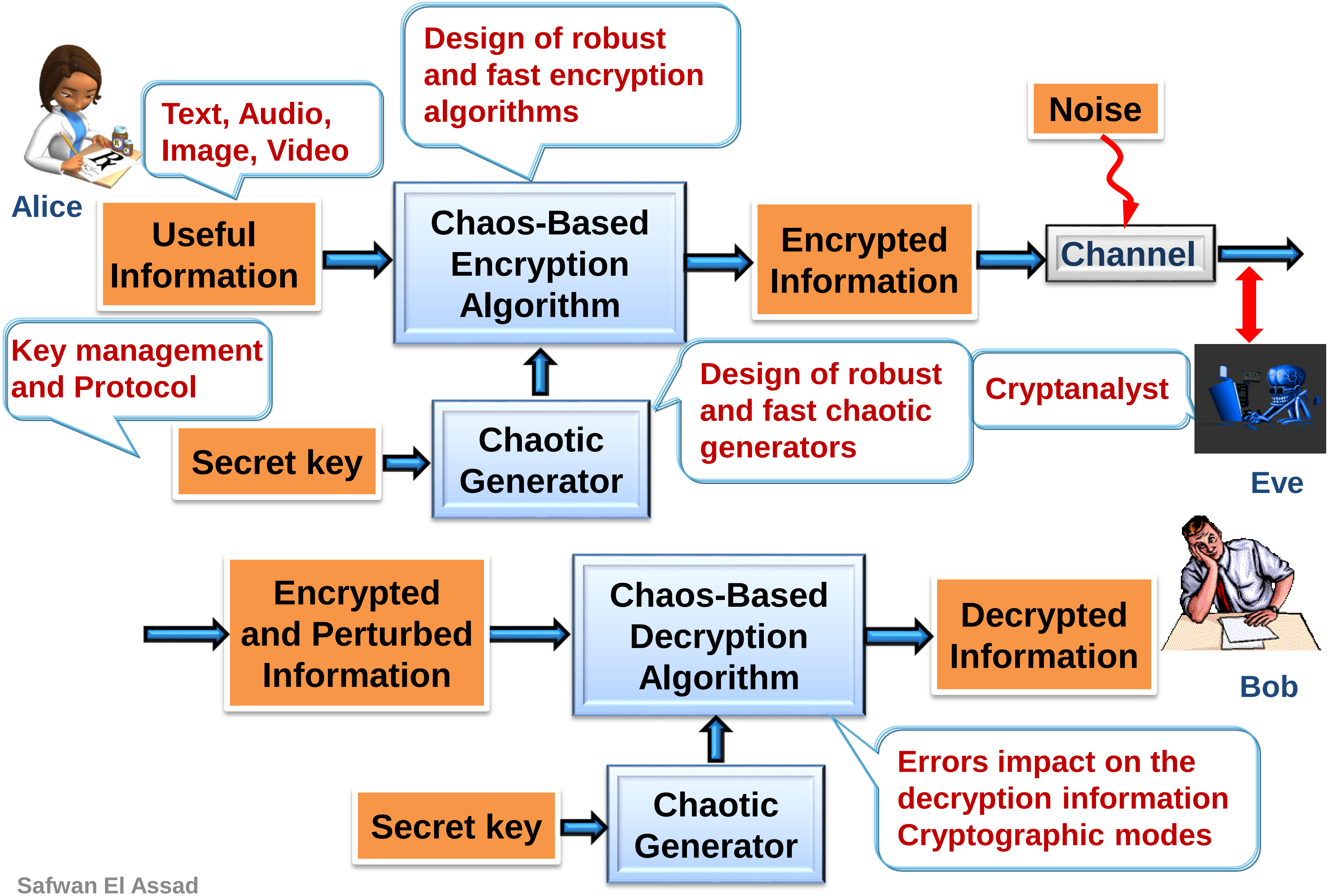
Definition: A cryptosystem is a six-tuple $\{\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{A}\}$, where the following conditions are satisfied:

1. $\mathcal{P}$ is a finite set of possile *plaintexts* (*message space*)
2. $\mathcal{C}$ is a finite set of possible *ciphertexts*
3. $\mathcal{K}$, the *key space*, is a finite set of possible *keys*
4. For each $K \in \mathcal{K}$, there is an *encryption rule* $E(K, P) \in \mathcal{E}$ and a corresponding *decryption rule* $D(K, C) \in \mathcal{D}$, such that:

$$D(K, E(K, P)) = P \in \mathcal{P}$$

With $P = \{p_1, p_2, \cdots, p_n\}$, $C = \{c_1, c_2, \cdots, c_n\}$; $E(K, p_i) = c_i$ *and* $D(K, c_i) = p_i \in \mathcal{A}$
$\mathcal{A}$ is a finite set (*alphabet of definition*). Example: $\mathcal{A} = \{0, 1\}$; $\mathcal{A} = \{0, 1, 2, \cdots, 255\}$
Clearly, $E(K, p_i)$ *is an injective function* (*i.e., one − to − one*).

Principle of chaos-based cryptosystems


Text, Audio, Image, Video
Design of robust and fast encryption algorithms
Noise
Alice
Useful Information
Chaos-Based Encryption Algorithm
Encrypted Information
Channel
Key management and Protocol
Secret key
Chaotic Generator
Design of robust and fast chaotic generators
Cryptanalyst
Eve
Encrypted and Perturbed Information
Chaos-Based Decryption Algorithm
Decrypted Information
Bob
Secret key
Chaotic Generator
Errors impact on the decryption information Cryptographic modes

Advanced Encryption Standard: AES

References:

Advanced Encryption Standard (AES), FIPS PUB 197, November 26, 2001.

Books:

Joan Daemen and Vincent Rijmen, "The design of Rijndael". Springer, 2010.

William Stallings, "Cryptography and Network Security, Principles and Practice". Sixth Edition, Pearson, 2014. Chapter 5.

Christof Paar and Jan Pelzl, "Understanding Cryptography". Springer, 2010. Chapter 4.

Douglas. R. Stinson, "Cryptography theory and Practice". Third edition, Taylor & Francis Group, LLC, 2006. Chapter 3.

Presentations Power Point and demo

AES-William_Stallings.ppt

Understanding_Cryptography_Chptr_4---AES.ppt

CrypTool project: www.cryptool.org by Enrique Zabala

Advanced Encryption Standard: AES

Learning Objectives: W. Stallings

- Present an overview of the general structure of AES
- Understand the transformations used in AES Encryption
- Byte Substitution layer
- Diffusion layer:

  Shift rows

  Mix columns
- Key Addition layer
- Explain the AES Key Expansion Algorithm.
- Understand the use of Polynomial Arithmetic in GF(2^8)
- Euclidian algorithm and Extended Euclidian algorithm
- Describe the Decryption process
- Practical Issues

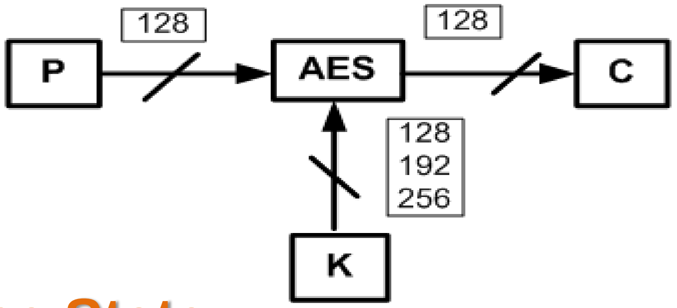
Overview of the AES Algorithm

AES origins: Lawrie Brown

- **Clear a replacement for DES (Data Encryption Standard) 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 (National Institute of Standards and Technology) issued call for ciphers in 1997**
- **15 candidates accepted in 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**

Overview of the AES Algorithm

The AES Cipher - Rijndael



Key size (bits/bytes/words)	Number of rounds Nr
128 / 16 / 4	10
192 / 24 / 6	12
256 / 32 / 8	14

The State

Input array

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

→

State array

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

→

Output array

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

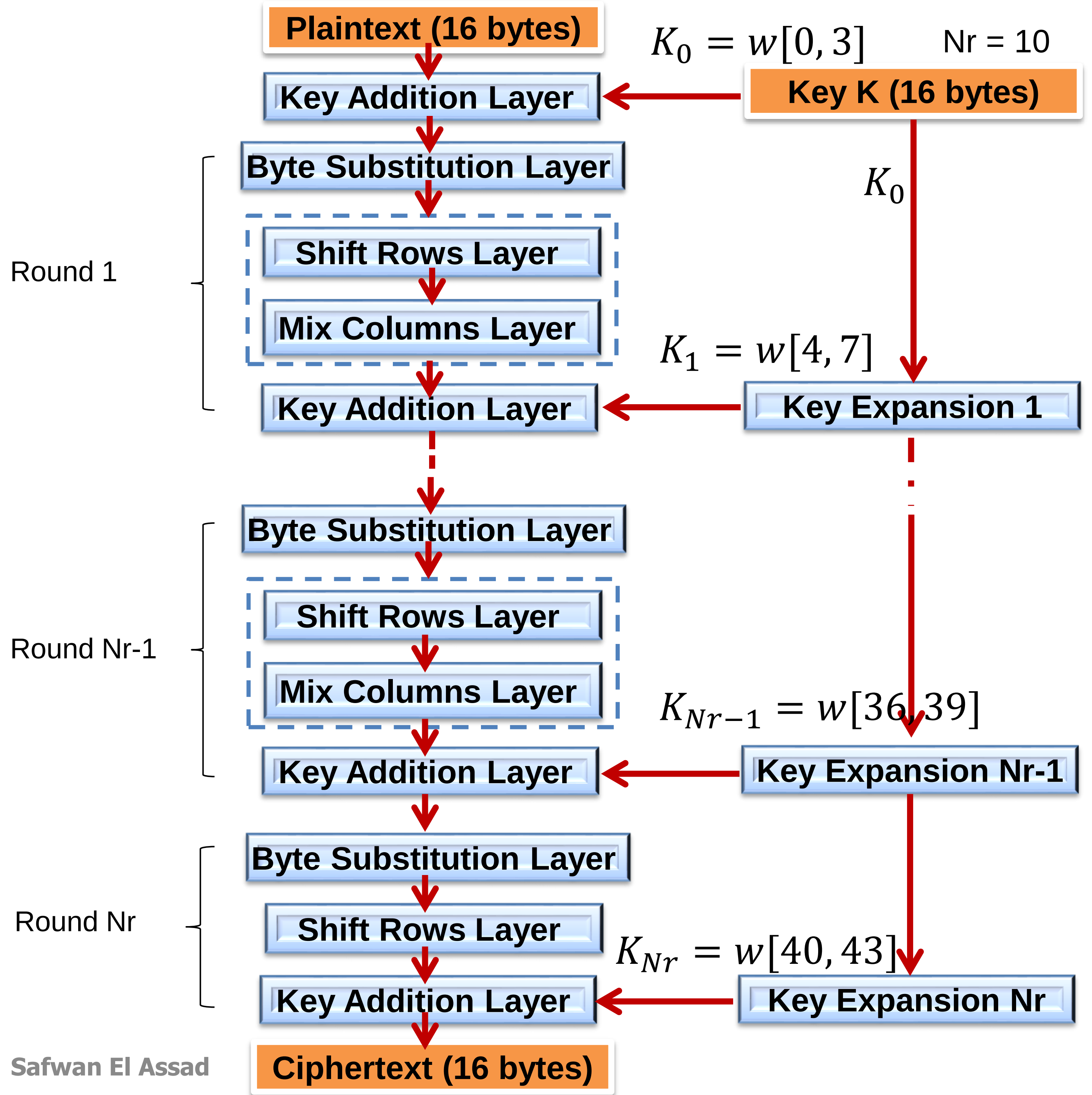
$s[r,c] = in[r+4c]$ **for** $0 \le r < 4$ **and** $0 \le c < 4$.

$out[r+4c] = s[r,c]$ **for** $0 \le r < 4$ **and** $0 \le c < 4$.

$w_0 = s_{0,0}\ s_{1,0}\ s_{2,0}\ s_{3,0} \quad w_1 = s_{0,1}\ s_{1,1}\ s_{2,1}\ s_{3,1}$

$w_2 = s_{0,2}\ s_{1,2}\ s_{2,2}\ s_{3,2} \quad w_3 = s_{0,3}\ s_{1,3}\ s_{2,3}\ s_{3,3}$

AES Encryption Bloc Diagram



AES Encryption Round for rounds 1, 2,…, Nr-1

AES S-box, substitution values in hexadecimal notation for input byte (xy)

Hexadecimal notation: 9a = $\underbrace{1001}\ \underbrace{1010}$ (1 byte)

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

State

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

Sub Bytes

State

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

$S(9a)_{hex} = (b8)_{hex}$

- S-box is the only nonlinear element of the AES:

$$ByteSub(B_i) + ByteSub(B_j) \neq ByteSub(B_i + B_j)\,, for\ i, j = 0, \cdots, 15$$

- S-box is Bijective: one-to-one mapping of input and output bytes
- S-box is uniquely reversed

AES Encryption Round for rounds 1, 2,…, Nr-1

Shift Rows

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

No shift

One position left shift

Two positions left shift

Three positions left shift

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

Mix Columns

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

- 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 $P(x) = x^8 + x^4 + x^3 + x + 1$

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

X

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

=

04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c

AES Encryption Round for rounds 1, 2,…, Nr-1

Add round Key K_1 produced by the Key Expansion column by column

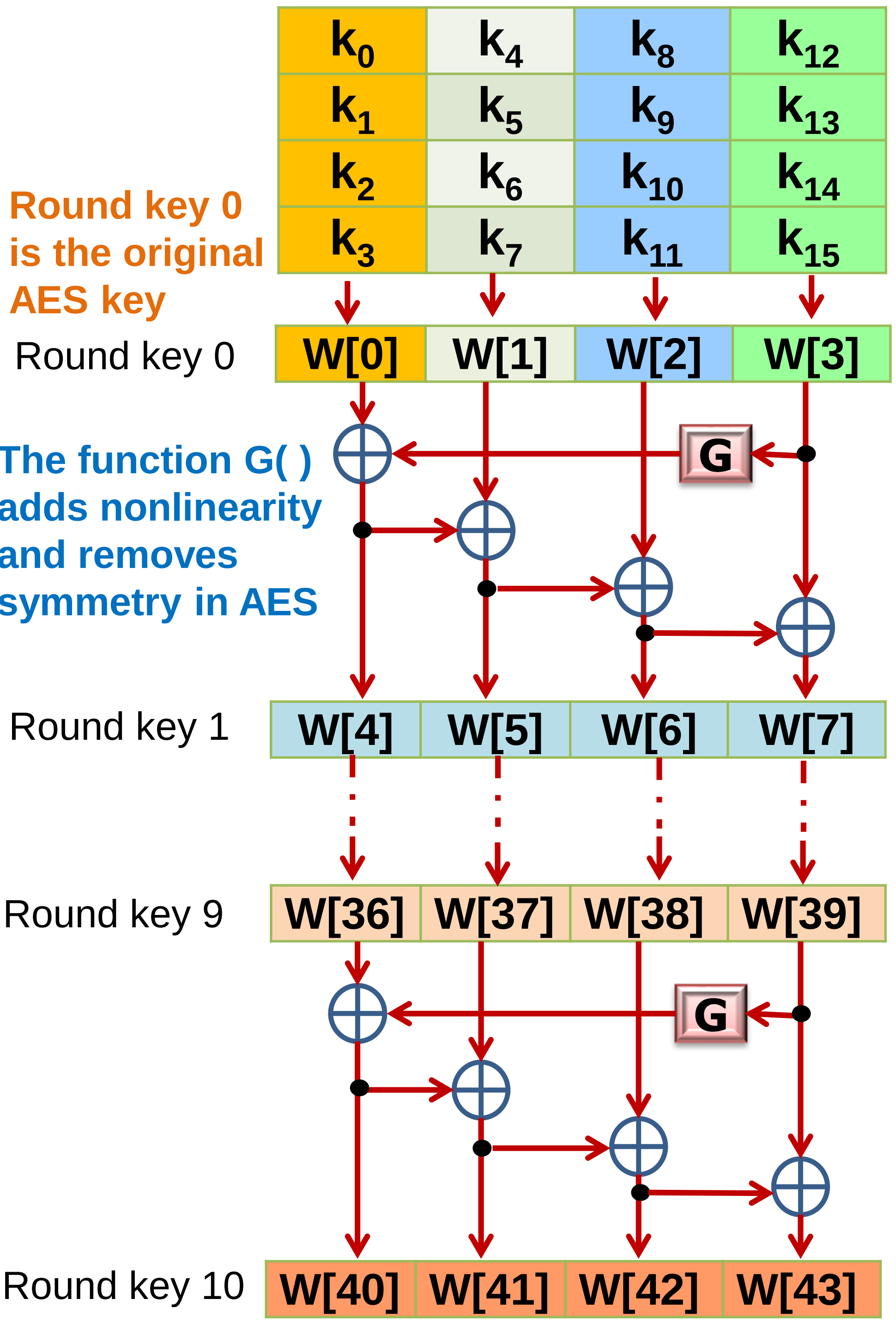
$$\begin{bmatrix} 04 & e0 & 48 & 28 \\ 66 & cb & f8 & 06 \\ 81 & 19 & d3 & 26 \\ e5 & 9a & 7a & 4c \end{bmatrix} \oplus \begin{bmatrix} a0 & 88 & 23 & 2a \\ fa & 54 & a3 & 6c \\ fe & 2c & 39 & 76 \\ 17 & b1 & 39 & 05 \end{bmatrix} = \begin{bmatrix} a4 & 68 & 6b & 02 \\ 9c & 9f & 5b & 6a \\ 7f & 35 & ea & 50 \\ f2 & 2b & 43 & 49 \end{bmatrix}$$

K_1=w[4, 7]

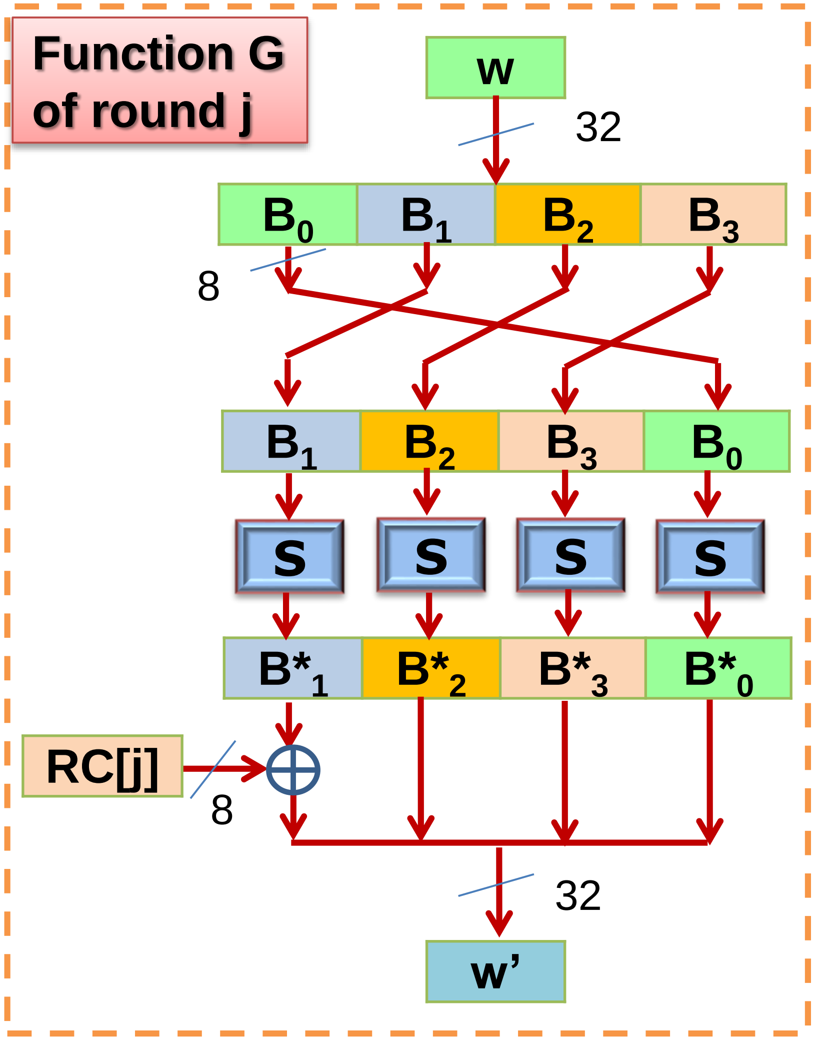
Key Expansion

Key size (bits/bytes/words)	Number of rounds Nr	Number of subkeys	Expanded Key size (bytes/words)
128 / 16 / 4	10	11	176/44
192 / 24 / 6	12	13	208/52
256 / 32 / 8	14	15	240/60

Key Expansion algorithm for 128-bit Key AES



k_0 k_4 k_8 k_{12}
k_1 k_5 k_9 k_{13}
k_2 k_6 k_{10} k_{14}
k_3 k_7 k_{11} k_{15}
Round key 0 is the original AES key
Round key 0
W[0] W[1] W[2] W[3]
G
The function G( ) adds nonlinearity and removes symmetry in AES
Round key 1
W[4] W[5] W[6] W[7]
Round key 9
W[36] W[37] W[38] W[39]
G
Round key 10
W[40] W[41] W[42] W[43]



AES Key expansion for 128-bit

Round j	RC[j]
1	{01}
2	{02}
3	{04}
4	{08}
5	{10}
6	{20}
7	{40}
8	{80}
9	{1b}
10	{36}

The round constant

is defined as:

Rcon[j] = (RC[j], 0, 0, 0) with RC[1] = 1,

RC[j] = 2 x RC[j-1] and with multiplication defined over $GF(2^8)$,

e.g, at round 9:

{02} x {80} = (000000010) x (10000000) = (00000000) $\oplus$ (00011011) =

(00011011) = {1b}

Key 0 ---> (w[0], w[1], w[2], w[3])

The other array elements are computed as:

The leftmost word of a round key w[4i], where

i = 1,…,10, is: w[4i] = w[4(i-1)]+G(w[4i-1]);

G() is a nonlinear function with a 4-byte input and output.

The remaining 3 words of a round key are computed recursively as:

w[4i+j] = w[4i+j-1] + w[4(i-1)+j], i=1,…,10; j=1, 2, 3

AES Arithmetic

Finite Field Arithmetic

- In AES all operations are performed on 8 bits bytes. The arithmetic operations of addition, subtraction, multiplication, division and inversion are performed over the Extension Finite Galois Field GF(2^8) of 256 elements [0, 1, …, 255], with the irreducible polynomial: $P(x) = x^8 + x^4 + x^3 + x + 1$
- Arithmetic on the coefficients is performed over GF(2) which is the smallest Prime Field. Addition modulo 2 is equivalent to XOR gate and multiplication is equivalent to the logical AND gate.

Remark:

- In the extension field GF(2^8) the order = 256 is not a Prime Number, then the addition and multiplication operation cannot be represented by addition and multiplication of integers modulo 2^8. For that:
- In the extension field GF(2^8) elements are not represented as integers but as polynomials with coefficients in GF(2). Computation in GF(2^8) is done by performing a certain type of polynomial arithmetic. The polynomials have a maximum degree of 7.

AES Arithmetic

- **Each element $A \in GF(2^8)$ is represented as:**

$$A(x) = a_7x^7 + a_6x^6 + \cdots + a_1x + a_0, \quad a_i \in GF(2) = [0, 1]$$

There are exactly $2^8 = 256$ such polynomials.

- **Every polynomial can simply be stored in digital form as an 8-bit word:**

$$A = (a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0)$$

We do not have to store the factor x^7, x^6, etc. It is clear from the bit positions to which power x^i each coefficient belongs.

AES Arithmetic

Addition and Subtraction in GF(2^8)

Let $A(x), B(x) \in GF(2^8)$.

The sum or difference of two elements is:

$$C(x) = A(x) + B(x) = A(x) - B(x) = \sum_{i=0}^{7} c_i x^i \,,$$

$$c_i = (a_i + b_i) \bmod 2 = (a_i - b_i) \bmod 2 = a_i \oplus b_i$$

Note that we perform modulo 2 addition (or subtraction) with the coefficients.

Example of addition modulo 2:

$$\begin{array}{llllll} A(x) = x^7 + & x^5 + & x^4 + & & & 1 \\ B(x) = & x^5 + & & x^2 + & & 1 \\ \hline C(x) = x^7 + & & x^4 + & x^2 & & \end{array}$$

In binary notation: (10110001) ⊕ (00100101) = (10010100)

In hexadecimal notation: {b1} ⊕ {25} = {94}

AES Arithmetic

Brief Reminder

Polynomial Arithmetic

- **Multiplication of two polynomials:**

$$A(x) = \sum_{i=0}^{m} a_i x^i, \qquad B(x) = \sum_{j=0}^{q} b_j x^j,$$

$$C(x) = A(x) \times B(x) = \sum_{i=0}^{m}\sum_{j=0}^{q} a_i b_j x^{i+j} = \sum_{n=0}^{m+q}\left[\sum_{i=0}^{m} a_i b_{n-i}\right] x^n, \qquad (n-i) \in [0, \cdots, q]$$

$$c_n = \sum_{i=0}^{m} a_i b_{n-i} \quad a_i, b_i, c_i \in GF(2) = \{0, 1\}$$

Is the discrete convolutional product of the coefficients of two polynomials

$$c_0 = a_0 b_0, \qquad c_1 = [a_0 b_1 + a_1 b_0], \qquad c_2 = [a_0 b_2 + a_1 b_1 + a_2 b_0]$$

$$c_{m+q-1} = [a_{m-1} b_q + a_m b_{q-1}], \qquad c_{m+q} = a_m b_q$$

Example of polynomials multiplication over GF(2)

$$A(x) = x^7 + x^5 + x^4 + 1, \qquad B(x) = x^5 + x^2 + 1$$

$$\begin{array}{rcccccccccl}
A(x) \times B(x) = & & & & x^7 + & & x^5 + & x^4 + & & 1 \\
 & & & & & \times \big(& x^5 + & & x^2 + & 1\big) \\
\hline
 & & & & x^7 + & & x^5 + & x^4 + & & 1 \\
 & & & x^9 + & x^7 + & x^6 + & & & x^2 & \\
 & x^{12} + & x^{10} + & x^9 + & & & x^5 & & & \\
\hline
 & x^{12} + & x^{10} + & & & x^6 & & + x^4 + & x^2 + & 1
\end{array}$$

Verification: ***m*=7, *q*=5**

$$c_0 = a_0 b_0 = 1, \qquad c_1 = [a_0 b_1 + a_1 b_0] = 0, \qquad c_2 = [a_0 b_2 + a_1 b_1 + a_2 b_0] = 1$$

$$c_{m+q-1} = \left[a_{m-1} b_q + a_m b_{q-1}\right] = 0, \qquad c_{m+q} = a_m = 1$$

AES Arithmetic

- **Polynomials division over GF(2)**

If we divide $C(x)$ by $\mathrm{D}(x)$, we get a quotient $\mathrm{Q}(x)$ and a remainder $\mathrm{R}(x)$ that obey the relationship:

$$C(x) = D(x)Q(x) + R(x)$$

With polynomial degrees:

Degrees of:

$$C(x) = n, \qquad D(x) = k, \qquad Q(x) = n - k\,, \qquad R(x) < k$$

In analogy with integer modular arithmetic, we can write:

$$R(x) = C(x)\ mod\ D(x)$$

If $R(x) = 0$, than we can say $D(x)$ divides $C(x)$ or $D(x)$ is a divisor of $C(x)$

AES Arithmetic

Example of polynomials division over GF(2)

$$\begin{array}{l|l} x^{12} + x^{10} + x^{6} + +x^{4} + x^{2} + 1 & x^{6} + x + 1 \\ \cline{2-2} x^{12} + x^{7} + x^{6} & x^{6} + x^{4} + x \\ x^{10} + x^{5} + x^{4} & \\ x^{7} + x^{2} + x & \\ \hline R(x) = x^{5} + x + 1 & \end{array}$$

AES Arithmetic

Modular Polynomial Arithmetic

Multiplication in GF(2^8)

Let $A(x), B(x) \in GF(2^8)$ and let $P(x) = x^8 + x^4 + x^3 + x + 1$ or {01} {1b} in hexadecimal notation, be the irreducible polynomial or prime polynomial

The multiplication of the two polynomials $A(x), B(x)$ is performed as:

$$C(x) = A(x) \times B(x) \; mod \; P(x), \qquad C(x) \in GF(2^8)$$

This means that if the degree of $C(x)$ is greater than 7, then $C(x)$ is reduced modulo $P(x)$ of degree 8. The remainder is expressed as: $R(x) = C(x) \; mod \; P(x)$

$$\begin{array}{l|l}
x^{12} + x^{10} + x^6 + x^4 + x^2 + 1 & x^8 + x^4 + x^3 + x + 1 \\ \hline
x^{12} + x^8 + x^7 + x^5 + x^4 & x^4 + x^2 + 1 \\
x^{10} + x^6 + x^5 + x^3 + x^2 & \\
x^8 + x^4 + x^3 + x + 1 & \\ \hline
R(x) = x^7 + x^4 + x &
\end{array}$$

AES Arithmetic

Remark:

There is no simple XOR operation that will accomplish multiplication in $GF(2^k)$.

However a straightforward implemented technique, based on the following observation is available:

$$x^k \bmod P(x) = [P(x) - x^k] \qquad \textit{in AES:} \qquad x^8 \bmod P(x) = x^4 + x^3 + x + 1 \quad (1)$$

Consider:

$$A(x) = a_7x^7 + a_6x^6 + \cdots + a_1x + a_0 \quad \in \ GF(2^8)$$

$$x \times A(x) = (a_7x^8 + a_6x^7 + \cdots + a_1x^2 + a_0x) \bmod P(x)$$

If $a_7 = 0$, then no need for reduction.

If $a_7 = 1$, then reduction modulo $P(x)$ is achieved using Eq (1):

$$x \times A(x) = (a_6x^7 + \cdots + a_1x^2 + a_0x) + x^4 + x^3 + x + 1$$

So, $$x \times A(x) = \begin{cases} (a_6, a_5, a_4, a_3, a_2, a_1, a_0, 0) & if \quad a_7 = 0 \\ (a_6, a_5, a_4, a_3, a_2, a_1, a_0, 0) \oplus (00011011) & if \quad a_7 = 1 \end{cases} \qquad (2)$$

It follows that multiplication by x (i.e., 00000010) can be implemented as a 1-bit left shift followed by a conditional bitwise XOR with (00011011).

AES Arithmetic

Example:

$$A(x) = x^7 + x^5 + x^4 + 1$$

$$x \times A(x) = \left(x^8 + x^6 + x^5 + x\right) \bmod P(x)$$

$$x \times A(x) = \left(x^6 + x^5 + x\right) + \left(x^4 + x^3 + x + 1\right) = x^6 + x^5 + x^4 + x^3 + 1$$

Indeed:

$$\begin{array}{rcccccc|l}
 & x^8 + & x^6 + x^5 + & & & x & x^8 + x^4 + x^3 + x + 1 \\
 & x^8 + & & x^4 + x^3 + & & x + 1 & \\
\cline{8-8}
 & & & & & & 1 \\
\cline{1-7}
R(x) = & & x^6 + x^5 + & x^4 + x^3 + & & 1 &
\end{array}$$

Multiplication by a higher power of x can be achieved by repeated Eq (2). By adding intermediate results, multiplication by any constant in $GF(2^8)$ can be achieved.

AES Arithmetic

Inversion in $GF(2^8)$

By using the **Extended Euclidean Algorithm**, the inverse A^{-1} of a nonzero element $A \in GF(2^8)$ is defined by:

$$A^{-1}(x) \times A(x) = 1 \; mod \; P(x)$$

The element “0” of the field doesn't have an inverse, however in the AES S-box, the input value ‘0’ is mapped to the output value ‘0’ .

For small fields (order or cardinality of a field is < 2^{16} elements, Lookup tables which contain the precomputed inverses of all field are often used. The following table shows the values of the multiplication inverse in $GF(2^8)$ for bytes (xy).

Note that the table below doesn’t contain the S-box of AES.

Indeed, the S-box does not have any fixed points, i.e., there are not any input values A_i such that $S(A_i) = A_i$, even for the input value ‘0’.

AES Arithmetic

Inversion in $GF(2^8)$

Multiplication inverse table in $GF(2^8)$ for bytes {xy}

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

Example: $A(x) = x^7 + x^5 + x^4 + 1 = (10110001) = \{b1\} = \{xy\}$

The inverse $A^{-1}(x)$ is $\{e0\} = (11100000) = x^7 + x^6 + x^5$. This can be verified by:

$$(x^7+x^5 + x^4 + 1) \times (x^7+x^6 + x^5) = 1 \; mod \; P(x)$$

Mathematical description of the AES S-Box

AES S-Box is built by applying two mathematical transformation.

1. **Map each byte $A \in GF(2^8)$ to its multiplicative inverse $B = A^{-1}$.**
2. **Apply the affine transformation to each bit of each byte B**

$$d_i = b_i \oplus b_{(i+4)\ mod\ 8} \oplus b_{(i+5)\ mod\ 8} \oplus b_{(i+6)\ mod\ 8} \oplus b_{(i+7)\ mod\ 8} \oplus c_i$$

Where c_i is the *i*th bit of byte $C = (01100011) = \{63\}$

$$\begin{tikzcd} {} \arrow[r, "A"] & \boxed{\text{Multiplicative inverse in } GF(2^8)} \arrow[r, "B = A^{-1}"] & \boxed{\text{Affine mapping}} \arrow[r, "D = S(A)"] & {} \end{tikzcd}$$

The AES standard depict the affine transformation in matrix form as follows:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Example:

$$A = (10110001) = \{b1\} = \{xy\}$$

From multiplicative inverse:

$$B = A^{-1} = \{e0\}$$

From affine mapping:

$$D = S(A) = \{c8\}$$

For $A = (00000000) = \{00\} = \{xy\}$

$$D = S(A) = \{63\}$$

AES S-Box

Remark:

The Multiplicative inverse operation in $GF(2^8)$ is highly nonlinear, this provides optimum protection against known cryptanalytic attacks.

The affine mapping destroys the algebraic structure of the Galois field, this allows to prevent attacks that would exploit the finite field inversion.

AES Mix Columns transformation

Mix Columns layer is defined by the following matrixes multiplication on state

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

Mix Column transformation operates on each column j of state individually and can be expressed as:

$$s'_{0,j} = (\{02\} \times \{s_{0,j}\}) \oplus (\{03\} \times \{s_{1,j}\}) \oplus (\{01\} \times \{s_{2,j}\}) \oplus (\{01\} \times \{s_{3,j}\})$$
$$s'_{1,j} = (\{01\} \times \{s_{0,j}\}) \oplus (\{02\} \times \{s_{1,j}\}) \oplus (\{03\} \times \{s_{2,j}\}) \oplus (\{01\} \times \{s_{3,j}\})$$
$$s'_{2,j} = (\{01\} \times \{s_{0,j}\}) \oplus (\{01\} \times \{s_{1,j}\}) \oplus (\{02\} \times \{s_{2,j}\}) \oplus (\{03\} \times \{s_{3,j}\})$$
$$s'_{3,j} = (\{03\} \times \{s_{0,j}\}) \oplus (\{01\} \times \{s_{1,j}\}) \oplus (\{01\} \times \{s_{2,j}\}) \oplus (\{02\} \times \{s_{3,j}\})$$

The additions and multiplications are performed in $GF(2^8)$.

Mix Columns is the major diffusion element. Indeed, every input byte influences 4 output bytes. The combination of the Shift Rows and Mix Columns layer makes it possible that after only three rounds every byte of the state matrix depends on all 16 plaintext bytes.

In AES, encryption is more important than decryption for 2 reasons:

1. **For the CTR, OFB and CFB cipher modes, only Encryption is used.**
2. **AES can be used to construct a message authentication code, and for this, only encryption is used.**

AES Mix Columns transformation

Example of Mix Columns for the first column:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} d4 \\ bf \\ 5d \\ 30 \end{bmatrix} = \begin{bmatrix} 04 \\ 66 \\ 81 \\ e5 \end{bmatrix}$$

The constants {01}, {02} or {03} are chosen for their efficient polynomial multiplication, for e.g. Multiplication by {02} is achieved by a left shift by one bit, and a modular reduction with $P(x)$

To verify the Mix Columns operation on the first column, we need to show that:

$$(\{02\}\times\{d4\})\oplus(\{03\}\times\{bf\})\oplus(\{01\}\times\{5d\})\oplus(\{01\}\times\{30\}) = \{04\}$$
$$(\{01\}\times\{d4\})\oplus(\{02\}\times\{bf\})\oplus(\{03\}\times\{5d\})\oplus(\{01\}\times\{30\}) = \{66\}$$
$$(\{01\}\times\{d4\})\oplus(\{01\}\times\{bf\})\oplus(\{02\}\times\{5d\})\oplus(\{03\}\times\{30\}) = \{81\}$$
$$(\{03\}\times\{d4\})\oplus(\{01\}\times\{bf\})\oplus(\{01\}\times\{5d\})\oplus(\{02\}\times\{30\}) = \{e5\}$$

Recall that, in $GF(2^8)$ polynomial:

{01} = {00000001} = 1; {02} = {00000010} = x; {03} = {00000011} = $(x+1)$

$$x \times A(x) = \begin{cases} (a_6, a_5, a_4, a_3, a_2, a_1, a_0, 0) & if \quad a_7 = 0 \\ (a_6, a_5, a_4, a_3, a_2, a_1, a_0, 0)\oplus(00011011) & if \quad a_7 = 1 \end{cases}$$

$$(x+1)\times A(x) = x \times A(x) \oplus A(x)$$

{02} x {d4} = (00000010) x (11010100) = (10101000) ⊕ (00011011) = (10110011)

{03} x {bf} = (00000011) x (10111111) = (01111110) ⊕ (00011011)⊕(10111111)
= (11011010)

{01} x {5d} = (00000001) x (01011101) = (01011101)

{01} x {30} = (00000001) x (00110000) = (00110000)

So: (10110011) ⊕ (11011010) ⊕ (01011101) ⊕ (00110000) = (00000100) = {04}

Euclidian algorithm and Extended Euclidean algorithm

Mathematical reminder

Modular Arithmetic

Modulo operation

Let $a, r, m \epsilon \mathbb{Z}$ and $m > 0. We\ can\ write$:

$$a \bmod m = a - \left\lfloor \frac{a}{m} \right\rfloor \times m = r \Leftrightarrow a = q \times m + r \Leftrightarrow a \equiv r \bmod m$$

$$with\ 0 \leq r < m; q = \left\lfloor \frac{a}{m} \right\rfloor$$

Where: m, r, q are called the modulus, the reminder, the quotient and $\lfloor z \rfloor$ is the largest integer less than or equal to z (the floor function).

Example: $42 \bmod 9 = 42 - \left\lfloor \frac{42}{9} \right\rfloor \times 9 = 42 - 4 \times 9 = 6 \Rightarrow 42 \equiv 6 \bmod 9$

Multiplication Inverse

Let $a \epsilon \mathbb{Z}$, the inverse a^{-1} (if exist) is defined such that:

$$a \times a^{-1} = 1 \bmod m$$

An element **$a \epsilon \mathbb{Z}$ has a multiplicative inverse a^{-1}** if and only if

$$gcd(a, m) = 1$$

Where *gcd* is the greatest common divisor, i.e, the largest integer that divides both a and m. Then a and m are said to be **relatively prime** or **coprime**

Finding the Greatest Common Divisor by the Euclidean algorithm

The *gcd* of two positive integers $\boldsymbol{r_0}$ and $\boldsymbol{r_1}$ $\boldsymbol{gcd(r_0, r_1)}$ **with $\boldsymbol{r_0 > r_1}$**

can be calculated for small numbers, by factoring both numbers and finding the highest common factor. Example:

Let $\boldsymbol{r_0 = 84 = 2 \times 2 \times 3 \times 7}$; $\boldsymbol{r_1 = 30 = 2 \times 3 \times 5}$

The gcd is the product of all common prime factors: $\boldsymbol{gcd(84, 30) = 2 \times 3 = 6}$

For large numbers (bit length from 1024 to 3076 as used in public-key algorithms), factoring often is not efficient and then it is necessary to use an efficient algorithm such the **Euclidean algorithm** which is based on the following observation:

$$\boldsymbol{gcd(r_0, r_1) = gcd\big((r_0 - r_1), r_1\big)} \qquad (3)$$

Indeed, let $\boldsymbol{gcd(r_0, r_1) = g}$. Since, $\boldsymbol{g}$ divides both $\boldsymbol{r_0}$ and $\boldsymbol{r_1}$, we can write:

$\boldsymbol{r_0 = g \times x}$ and $\boldsymbol{r_1 = g \times y}$, where $\boldsymbol{x > y}$, and $\boldsymbol{x}$ and $\boldsymbol{y}$ are coprime integers, i.e, they do not have common factors, also $(x - y)$ and $\boldsymbol{y}$ are coprime integers:

$$\boldsymbol{gcd(r_0, r_1) = gcd\big((r_0 - r_1), r_1\big) = gcd(g \times (x - y), g \times y) = g}$$

$$\boldsymbol{gcd(x, y) = gcd\big((x - y), y\big) = 1}$$

Finding the Greatest Common Divisor by the Euclidean algorithm

Let verify this property with the numbers from the previous example: $\boldsymbol{r_0 = 84,\ r_1 = 30}$

$$\boldsymbol{r_0 - r_1 = 54 = 2 \times 3 \times 3 \times 3; \quad r_1 = 30 = 2 \times 3 \times 5}$$

$$\boldsymbol{\Longrightarrow gcd(54, 30) = 2 \times 3 = 6 = gcd(84, 30)}$$

Also, as: $\boldsymbol{r_0 = 6 \times 14,\ r_1 = 6 \times 5}$, then $\boldsymbol{gcd(14, 5) = gcd(9, 5) = 1}$

It is follows immediately that, equation (3) can be applied iteratively:

$$\boldsymbol{gcd(r_0, r_1) = gcd\big((r_0 - r_1), r_1\big) = gcd\big((r_0 - 2r_1), r_1\big) = \cdots = gcd\big((r_0 - qr_1), r_1\big)}$$

As long as $\boldsymbol{(r_0 - qr_1) > 0}$. Then:

$$\boldsymbol{gcd(r_0, r_1) = gcd\big((r_0 - qr_1), r_1\big) = gcd(r_0 \bmod r_1, r_1) = gcd(r_1, r_0 \bmod r_1)} \quad (4)$$

Because $\boldsymbol{r_0 \bmod r_1 < r_1}$

Equation (4) is applied recursively until we obtain finally $\boldsymbol{gcd(r_n, 0) = r_n}$.

Since each iteration preserves the $\boldsymbol{gcd}$ of the previous iteration step, it turns out that this final $\boldsymbol{gcd}$ is the $\boldsymbol{gcd}$ of the original problem, i.e:

$$\boldsymbol{gcd(r_0, r_1) = \cdots = gcd(r_n, 0) = r_n} \qquad (5)$$

Finding the Greatest Common Divisor by the Euclidean algorithm

Let first show the system of equations calculating the $gcd(r_0, r_1)$ of two given positive integers r_0 and r_1 with $r_0 > r_1$.

$$r_{i-2} \, mod \, r_{i-1} = r_{i-2} - \left\lfloor \frac{r_{i-2}}{r_{i-1}} \right\rfloor \times r_{i-1} = r_i \Rightarrow r_{i-2} = q_{i-1} \times r_{i-1} + r_i$$

With $0 \leq r_i < r_{i-1}$ and $q_{i-1} = \left\lfloor \frac{r_{i-2}}{r_{i-1}} \right\rfloor$ Example:

i	$r_{i-2} = q_{i-1} \times r_{i-1} + r_i$ $\quad 0 \leq r_i < r_{i-1}$	$gcd(r_0, r_1) = gcd(973, 301)$
2	$r_0 = q_1 \times r_1 + r_2$ $\quad 0 < r_2 < r_1$	$973 = 3 \times 301 + 70$ $\quad 0 < 70 < 301$
3	$r_1 = q_2 \times r_2 + r_3$ $\quad 0 < r_3 < r_2$	$301 = 4 \times 70 + 21$ $\quad 0 < 21 < 70$
4	$r_2 = q_3 \times r_3 + r_4$ $\quad 0 < r_4 < r_3$	$70 = 3 \times 21 + 7$ $\quad 0 < 7 < 21$
⋮	⋮ $\quad$ ⋮	$21 = 3 \times 7 + 0$
n	$r_{n-2} = q_{n-1} \times r_{n-1} + r_n$ $\quad 0 < r_n < r_{n-1}$	$gcd(973, 301) = 7$
n+1	$r_{n-1} = q_n \times r_n + 0$	
	$gcd(r_0, r_1) = r_n$	$gcd(973, 301) = gcd(301, 70)$
		$gcd(301, 70) = gcd(70, 21)$
		$gcd(70, 21) = gcd(21, 7)$
		$gcd(21, 7) = gcd(7, 0) = 7$

Euclid’s algorithm

Euclidean Algorithm

```
Input: positive integers $r_0$ and $r_1$ with $r_0 > r_1$
Output: $\boldsymbol{gcd(r_0, r_1)}$
Initialization: $i = 1$
Algorithm:
DO
    $i = i + 1$
    $\boldsymbol{r_i = r_{i-2} \; mod \; r_{i-1}}$
WHILE $r_i \neq 0$
RETURN
    $gcd(r_0, r_1) = r_{i-1}$
```

Note that the algorithm terminates if a remainder with the value $\boldsymbol{r_i = 0}$ is computed.

The number of needed iterations is close to the number of digits of the input operands. That means, for instance, that the number of iterations of a $\boldsymbol{gcd}$ involving 1024-bit numbers is 1024.

Extended Euclidean algorithm

The extended Euclidean algorithm allows us to compute **modular inverses**, which is of major importance in asymmetric and symmetric encryption. It not only calculate the ***gcd*** but also two additional integers ***s*** and ***t*** that verify the following equation:

$$gcd(r_0, r_1) = s \times r_0 + t \times r_1 \qquad (6)$$

The idea is to use the Euclidean algorithm, but we express the current remainder $\boldsymbol{r_i}$ in every iteration as a linear combination of the form:

$$r_i = s_i \times r_0 + t_i \times r_1 \qquad (7)$$

In the last iteration we obtain:

$$r_n = gcd(r_0, r_1) = s_n \times r_0 + t_n \times r_1 = s \times r_0 + t \times r_1 \qquad (8)$$

This means that the last coefficients $\boldsymbol{s_n}$ and $\boldsymbol{t_n}$ are the coefficients $\boldsymbol{s}$ and $\boldsymbol{t}$ of Eq (6)

Let consider the extended Euclidean algorithm with the same values as in the previous example, $r_0 = 973$ and $r_1 = 301$.

In the following table, in every iteration, on the left-hand side we compute the Euclidean algorithm and the integer quotient q_{i-1} and on the right-hand side we compute the coefficients s_i and t_i, verifying Eq (7).

Extended Euclidean algorithm

i	$r_{i-2} = q_{i-1} \times r_{i-1} + r_i \quad 0 \le r_i < r_{i-1}$	$r_i = [s_i] \times r_0 + [t_i] \times r_1$
2	$r_0 = q_1 \times r_1 + r_2 \quad 0 < r_2 < r_1$	$r_2 = [s_2] \times r_0 + [t_2] \times r_1$
3	$r_1 = q_2 \times r_2 + r_3 \quad 0 < r_3 < r_2$	$r_3 = [s_3] \times r_0 + [t_3] \times r_1$
4	$r_2 = q_3 \times r_3 + r_4 \quad 0 < r_4 < r_3$	$r_4 = [s_4] \times r_0 + [t_4] \times r_1$
⋮	⋮ ⋮	⋮
n	$r_{n-2} = q_{n-1} \times r_{n-1} + r_n \quad 0 < r_n < r_{n-1}$	$r_n = [s_n] \times r_0 + [t_n] \times r_1$
n+1	$r_{n-1} = q_n \times r_n + 0$	

We will now derive recursive formulae for computing $[s_i]$ and $[t_i]$ in every iteration.

In the iteration i we first compute q_{i-1} and the new reminder r_i from r_{i-1} and r_{i-2}.

$$r_i = r_{i-2} - q_{i-1} \times r_{i-1} \qquad (9)$$

In the previous iterations $(i-2)$ and $(i-1)$ we computed the values:

$$r_{i-2} = [s_{i-2}] \times r_0 + [t_{i-2}] \times r_1$$
$$r_{i-1} = [s_{i-1}] \times r_0 + [t_{i-1}] \times r_1$$

In order to compute r_i as a linear combination of r_0 and r_1, we substitute the previous values r_{i-2} and r_{i-1} in Eq (9), we obtain:

$$r_i = \{[s_{i-2}] \times r_0 + [t_{i-2}] \times r_1\} - q_{i-1} \times \{[s_{i-1}] \times r_0 + [t_{i-1}] \times r_1\}$$
$$r_i = \{[s_{i-2}] - q_{i-1} \times [s_{i-1}]\} \times r_0 + \{[t_{i-2}] - q_{i-1} \times [t_{i-1}]\} \times r_1 = [s_i] \times r_0 + [t_i] \times r_1$$

Extended Euclidean algorithm

From the later equation we deduce the recursive equations:

$$[s_i] = [s_{i-2}] - q_{i-1} \times [s_{i-1}] \qquad (10)$$

$$[t_i] = [t_{i-2}] - q_{i-1} \times [t_{i-1}] \qquad (11)$$

These equations are valid for $i \geq 2$ and the initial values are:

$s_0 = 1$, $s_1 = 0$, $t_0 = 0$, $t_1 = 1$.

AES Decryption

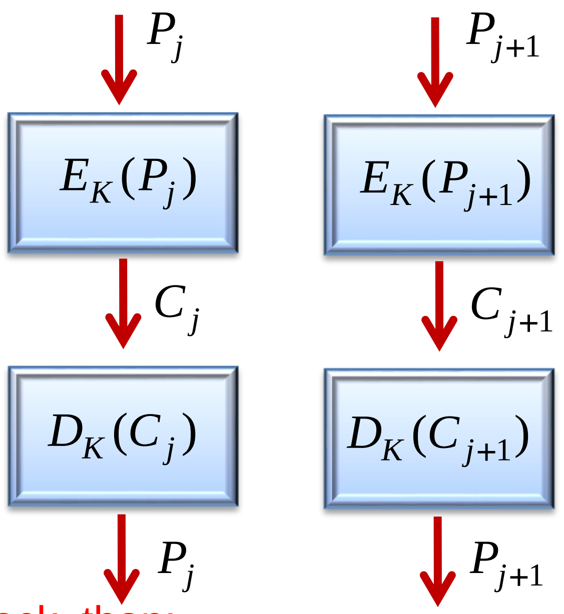
Various Cipher Block Modes : Symmetric key algorithms

A cryptographic mode combines the basic cipher, some sort of feedback, and some simple operations.
References:

Five confidentiality modes of operation can provide cryptographic protection:

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

ECB (Electronic Code Book) mode



Advantages :

As each plain block is encrypted independently, than :

- we can encrypt/decrypt records accessed randomly like a data base
- we can do parallel processing

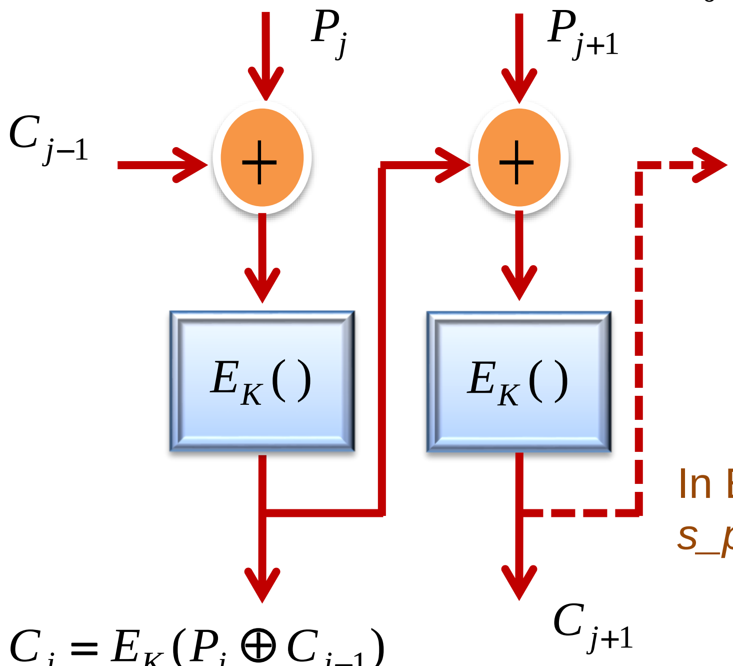
Disadvantages :

Since the same plain block always encrypts to the same cipher block, than:

- It is possible to create a Code Book of plaintexts and corresponding ciphertexts. However, if the block size is 128 bits, the code book will have 2^{128} entries, too much to pre-compute and store.
- Block Replay : a cryptanalyst could modify encrypted messages without knowing the key or the algorithm.
- It is possible to mount statistical attacks, because messages may be highly redundant
- It is evident to mount Known-plaintext and Chosen-plaintext attacks (the cryptanalyst has complete knowledge of the used encryption algorithm). Suppose P_j = “5e081bc5” is encrypted to C_j = “7ae593A4”, than, the cryptanalyst can decrypt C_j whenever it appears in another message.

CBC (Cipher Block Chaining) mode

Encryption process



$C_j = E_K(P_j \oplus C_{j-1})$

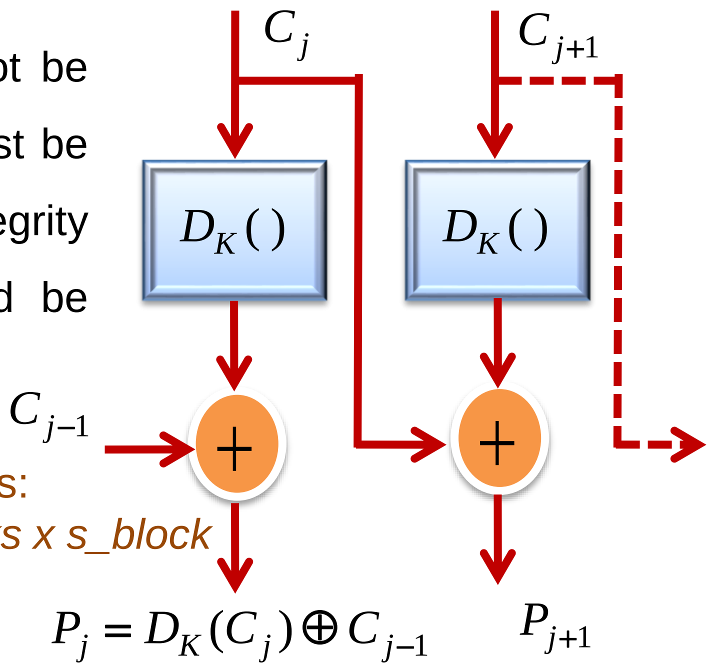
$j = 1, \cdots, n\_blocks$

$C_0 = IV : Initialization\ Vector$

The *IV* need not be secret, but it must be random. The integrity of the *IV* should be protected.

In ECB & CBC modes:
s_plaintext = n_blocks x s_block

Decryption process



$P_j = D_K(C_j) \oplus C_{j-1}$

$j = 1, \cdots, n\_blocks$

Advantages :

Identical plaintext messages encrypt to different ciphertext messages.

Thus, it is impossible to attempt Block Replay and to build a Code Book.

Disadvantages :

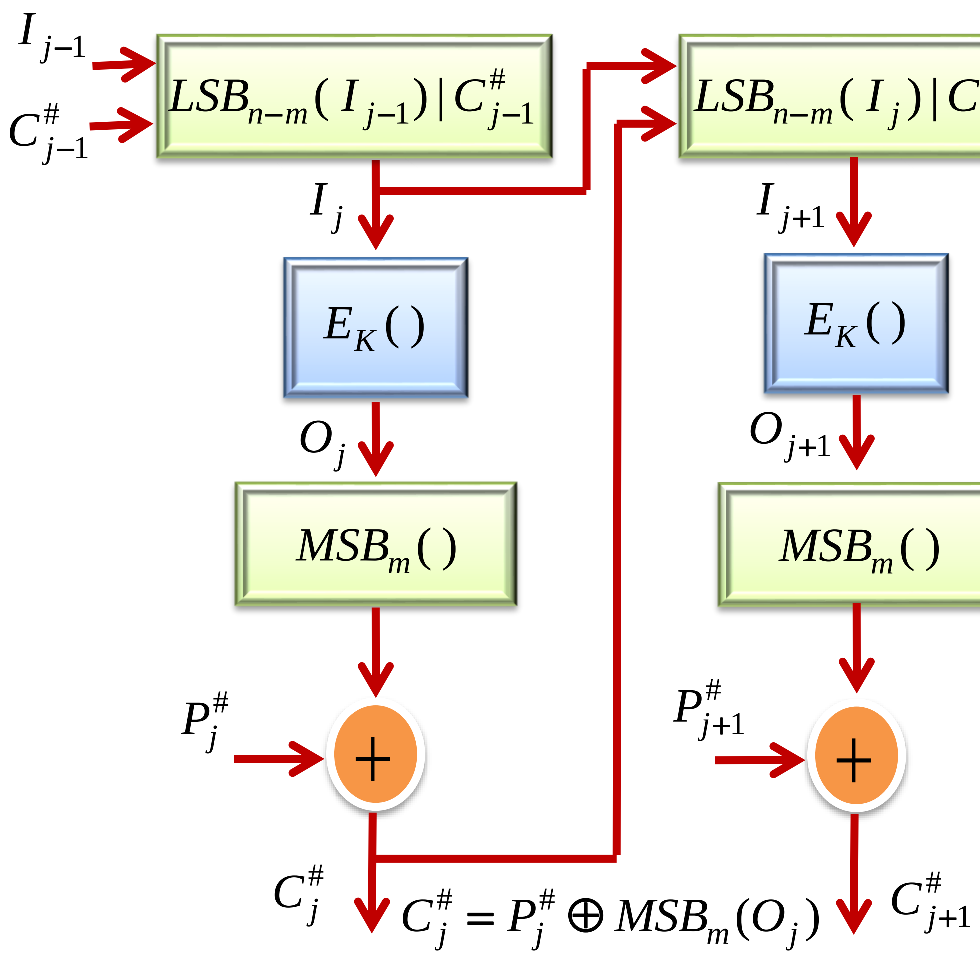
- Error Propagation: a single bit error in the ciphertext affects one block and one bit of the recovred plaintext.
- Encryption of blocks can't be performed in parallel, but decryption can be performed in parallel.
- Block structure must remains intact : if a bit is added or lost from the ciphertext stream, then decryption will generate garbage indefinitely.

CFB (Cipher Feedback) mode

Encryption process

$$I_j = \begin{cases} LSB_{n-m}(I_{j-1}) \mid C^{\#}_{j-1},\ j = 1, 2, \cdots, n\_blocks \\ I_0 = IV,\ C^{\#}_0 = MSB_m(I_0) \end{cases}$$

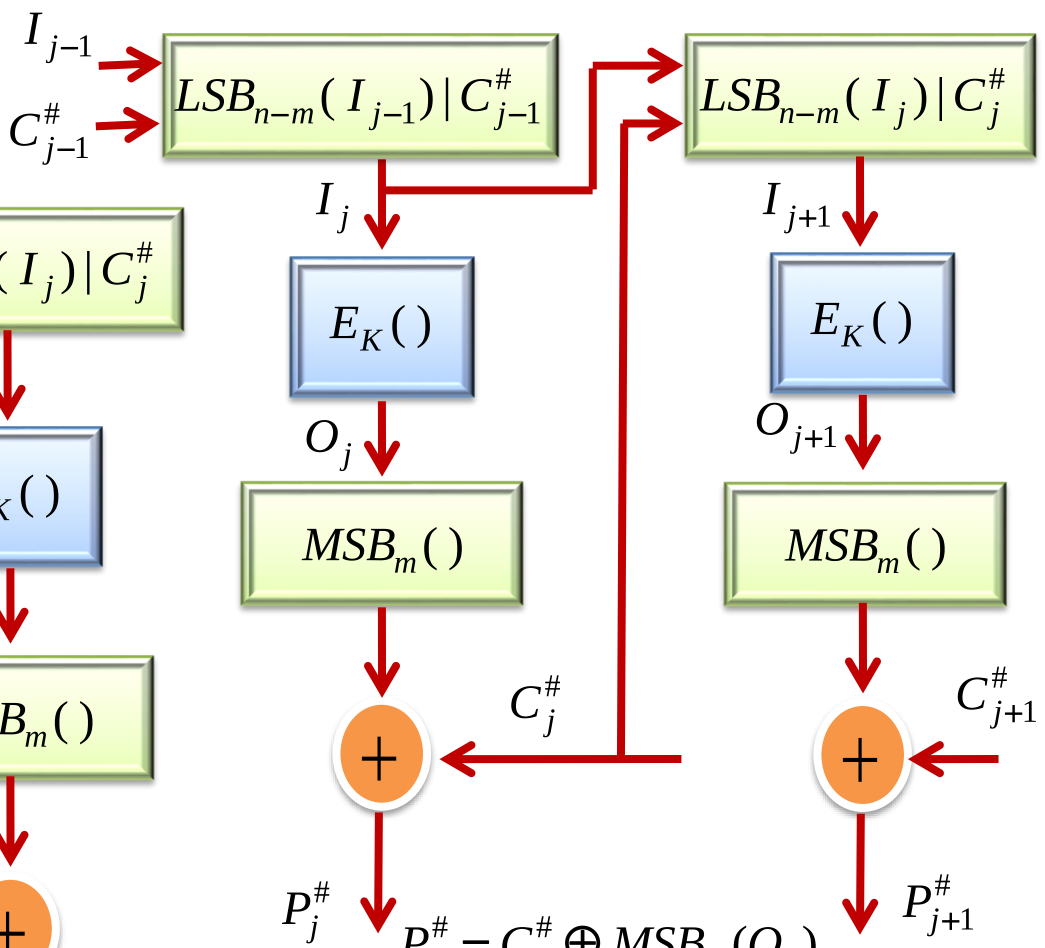
Size of I_j, O_j is *s_block* Size of $P_j^{\#}$, $C_j^{\#}$ *is m*



$$C^{\#}_j = P^{\#}_j \oplus MSB_m(O_j)$$

Decryption process

$$O_j = E_K(I_j)$$



$$P^{\#}_j = C^{\#}_j \oplus MSB_m(O_j)$$

In CFB mode:
s_plaintext = n_blocks x m $1 \le m \le s\_block$

CFB (Cipher Feedback) mode

The *IV* need not be secret, but it must be random and must be changed with every message. The integrity of the *IV* should be protected.

Advantages :

- Identical plaintext messages encrypt to different ciphertext messages.

Thus, it is impossible to attempt Block Replay and to build a Code Book.

- Unlike CBC mode, in CFB mode, data can be encrypted in units *m* bits smaller than the block size *s-block*. This mean that it is not necessary to receive a complete block of data to begin the encryption process.
- It can be implemented as a self-synchronization stream cipher.

Disadvantages :

- Error Propagation: a single bit error in the ciphertext affects the current and the following *s-blocs*/*m* -1 blocks.
- Encryption of blocks can't be performed in parallel, but decryption can be performed in parallel if the input blocks are first constructed, in series, from the IV and the ciphertext.
- Block structure must remains intact : if a bit is added or lost from the ciphertext stream, then decryption will generate garbage indefinitely.

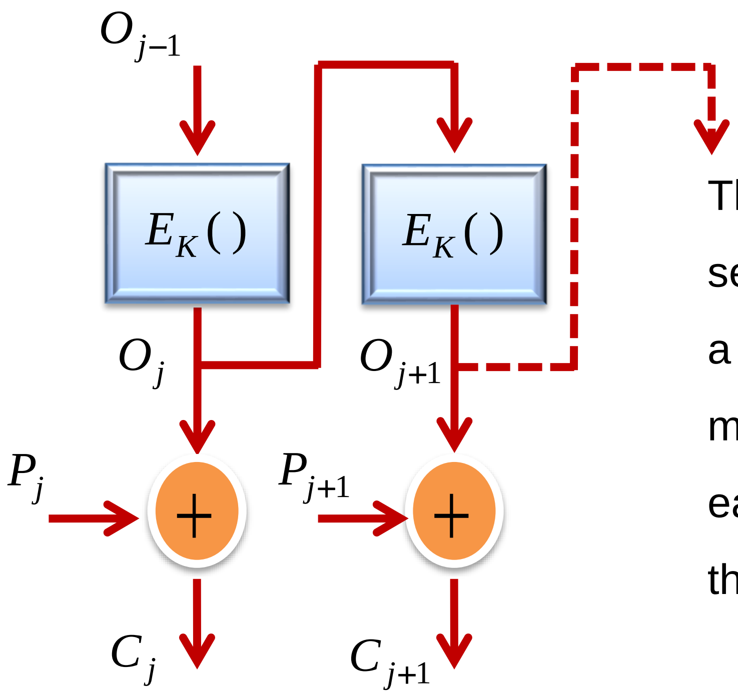
OFB (Output Feedback) mode

Encryption process

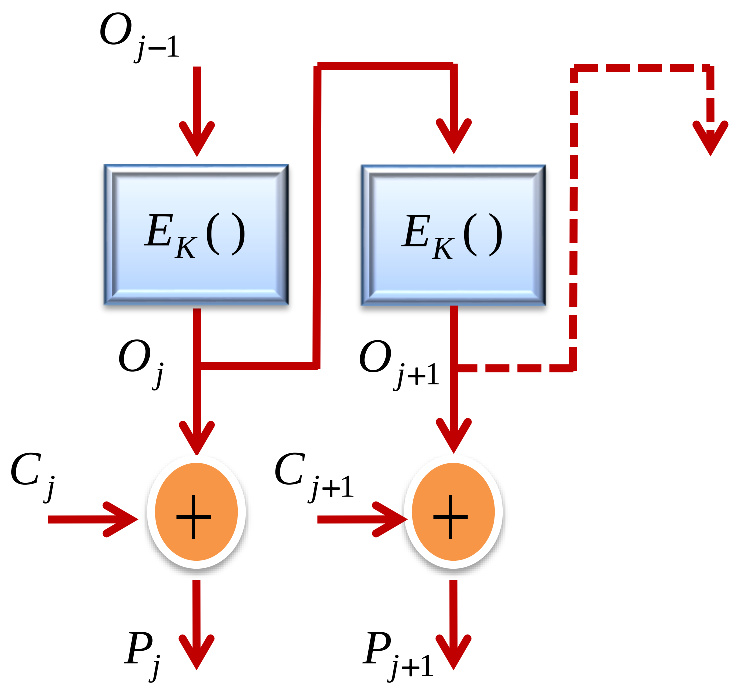
Decryption process

In OFB & CTR modes:
s_plaintext = (n_blocks - 1) x s_block + u

$1 \le u \le s\_block$



The *IV* need not be secret, but it must be a nonce, i.e., the *IV* must be unique for each execution under the given key.



$C_j = P_j \oplus O_j$

$j = 1, \cdots, n\_blocks - 1$

$O_j = E_K(O_{j-1})$

$O_0 = IV$

$P_j = C_j \oplus O_j$

$j = 1, \cdots, n\_blocks - 1$

$C^{¤}_{n\_blocks} = P^{¤}_{n\_blocks} \oplus MSB_u(O_{n\_blocks})$

$P^{¤}_{n\_blocks} = C^{¤}_{n\_blocks} \oplus MSB_u(O_{n\_blocks})$

OFB (Output Feedback) mode

Advantages :

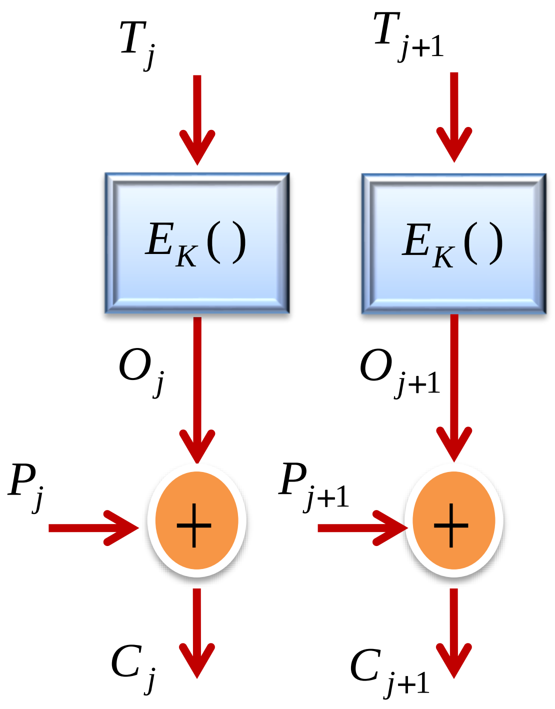
- Outputs O_j can be generated offline, before the plaintext or ciphertext data exists.
- It can be implemented as a synchronous stream cipher.
- Error Propagation : OFB mode has no error extension. A single bit error in the ciphertext causes a single bit in the recovered plaintext. This is useful for digital communication

Disadvantages :

- If the same IV is used for the encryption of more than one message, then the confidentiality of those messages may be compromised.
- Confidentiality is compromised if any of the input blocks O_j for the encryption of a message is designated as the *IV* for the encryption of another message under the given key.
- Both Encryption and decryption processes can not be performed in parallel.

CTR (Counter) mode

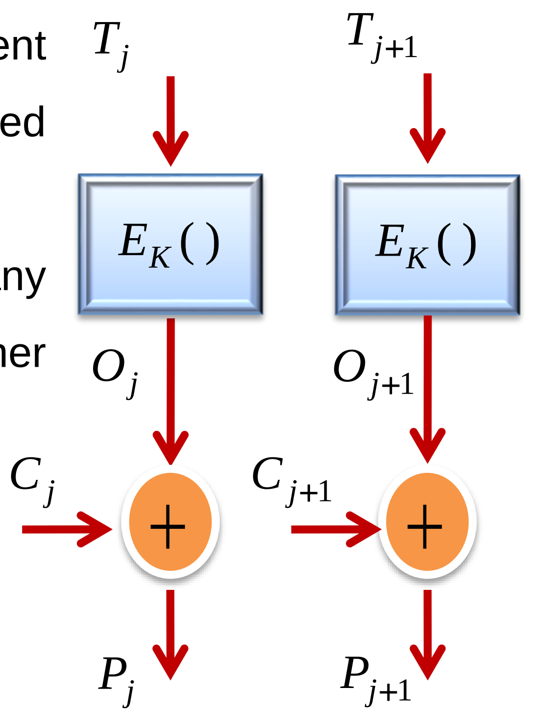
Encryption process



The sequence T_i must be different from T_j for all messages encrypted under the same key.

Sequences can be generated by any random-sequence generators, whether cryptographically secure or not.

Decryption process



$$C_j = P_j \oplus O_j \qquad O_j = E_K(T_j) \qquad P_j = C_j \oplus O_j$$

$$j = 1, \cdots, n\_blocks - 1 \qquad j = 1, \cdots, n\_blocks \qquad j = 1, \cdots, n\_blocks - 1$$

$$C^{*}_{n\_blocks} = P^{*}_{n\_blocks} \oplus MSB_u(O_{n\_blocks}) \qquad P^{*}_{n\_blocks} = C^{*}_{n\_blocks} \oplus MSB_u(O_{n\_blocks})$$

CTR (Output Feedback) mode

Advantages :

- Encryption and decryption processes can be performed in parallel.
- Outputs O_j can be generated offline, before the plaintext or ciphertext data exists.
- Error Propagation : CTR mode (as OFB mode) has no error extension. A single bit error in the ciphertext causes a single bit in the recovered plaintext. This is useful for digital communication.

Error Propagation : summary of bit errors on decryption

Block $C_j = (c_{1,j}, c_{2,j}, \cdots, c_{s\_block,j})$ Segment $C_j^{\#} = (c_{1,j}^{\#}, c_{2,j}^{\#}, \cdots, c_{m,j}^{\#})$ Decrypted plaintext

$P_j = (p_{1,j}, p_{2,j}, \cdots, p_{s\_block,j})$ $P_j^{\#} = (p_{1,j}^{\#}, p_{2,j}^{\#}, \cdots, p_{m,j}^{\#})$ $P_j, P_j^{\#}$

Mode	Effect of Bit Errors in C_j or $C_j^{\#}$	Effect of Bit Errors in *IV*
ECB	RBE in P_j	Not applicable
CBC	RBE in P_j SBE in P_{j+1}	SBE in P_j
CFB	SBE in $P_j^{\#}$ RBE in $P_{j+1}^{\#}, P_{j+2}^{\#}, \ldots, P_{j+s\_block/m}^{\#}$	RBE in $P_1^{\#}, P_2^{\#}, \cdots, P_j^{\#}$ $for\ some\ 1 \le j \le s\_block / m$
OFB	SBE in P_j	RBE in $P_1, P_2, \cdots, P_{n\_blocks}$
CTR	SBE in P_j	Not applicable

SBE: (specific bit errors) an error bit $c_{i,j}$ or $c_{i,j}^{\#}$ produces an error bit $p_{i,j}$ or $p_{i,j}^{\#}$

RBE: (random bit errors) an error bit $c_{i,j}$ or $c_{i,j}^{\#}$ affects randomly all bits in the P_j block or in segment $P_j^{\#}$. In this case each bit in P_j or $P_j^{\#}$ is incorrect with probability $P_{inv} = 1/2$

Error Propagation

Binary Symmetric channel

$P_{e,c}$: the bit error probability in the channel or in the cryptogram : C_j or $C_j^{\#}$

$P_{e,d}$: the bit error probability in the decrypted plaintext: P_j or $P_j^{\#}$

$P(k)$: the probability that there are *k* error bits out of *n* received bits

$$P(k) = C_k^{s\_block}\ P_{e,c}^{k}\ (1 - P_{e,c})^{s\_block-k} \qquad P_{e,c} \in [0, 1/2]$$

So :

$P_0 = (1 - P_{e,c})^{s\_block}$: Probability that *s_block* bits are correct,

or the correct block probability.

$Q_0 = 1 - P_0$: Probability that at least one bit is incorrect,

or the incorrect block probability.

Error Propagation

ECB mode : $P_j = D_k(C_j)$

The bit $p_{i,j}$ is incorrect if the block C_j is incorrect

and simultaneously the bit $p_{i,j}$ is inverted

$$\Rightarrow P_{e,d} = Q_0 \; P_{inv} = \frac{1}{2}\left[1-(1-P_{e,c})^{s\_block}\right]$$

CBC mode : $P_j = D_k(C_j) \oplus C_{j-1} = U_j \oplus C_{j-1}$

The bit $p_{i,j}$ is incorrect in the following cases :

a) The bit $c_{i,j-1}$ is incorrect and the block C_j is correct: $P_{e,c} \; P_0$

b) The bit $c_{i,j-1}$ is incorrect, the block C_j is incorrect and the bit $u_{i,j}$ is not inverted. $\Big\}\; P_{e,c} \; Q_0 \; (1-P_{inv})$

c) The bit $c_{i,j-1}$ is correct and the block C_j is incorrect and the bit $u_{i,j}$ is inverted. $\Big\}\; \left[1-P_{e,c}\right] Q_0 \; P_{inv}$

$$\Rightarrow P_{e,d} = P_{e,c} \; P_0 + Q_0 \; P_{inv}$$

Error Propagation

CFB mode : $P_j^{\#} = C_j^{\#} \oplus MSB_m(O_j); O_j = E_k(I_j), I_j = LSB_{n-m}(I_{j-1}) \; | C_{j-1}^{\#}$

The bit $p_{i,j}$ is incorrect in the following cases:

a) The bit $c_{i,j}^{\#}$ is incorrect and the block I_j is correct: $P_{e,c} \, P_0$

b) The bit $c_{i,j}^{\#}$ is incorrect, the block I_j is incorrect and the bit $o_{i,j}$ is not inverted. $\Big\}$ $P_{e,c} \, Q_0 \left[1 - P_{inv}\right]$

c) The bit $c_{i,j}^{\#}$ is correct, the block I_j is incorrect and the bit $o_{i,j}$ is inverted. $\Big\}$ $\left[1 - P_{e,c}\right] Q_0 \, P_{inv}$

$\Rightarrow P_{e,d} = P_{e,c} \, P_0 + Q_0 \, P_{inv}$ not depend on the length *m* of segments.

The CFB and CFB modes are equivalent from the viewpoint of error propagation.

OFB and CTR modes:

Only the SBE type of error propagation can occur.

So, each error bit $c_{i,j}$ of the cryptogram causes only one incorrect bit $p_{i,j}$ of the plaintext $\Rightarrow P_{e,d} = P_{e,c}$

Error Propagation

- ECB mode $$P_{e,d} = \frac{1}{2}\left[1-\left(1-P_{e,c}\right)^{s\_block}\right]$$
- CBC and CFB modes $$P_{e,d} = P_{e,c}\left(1-P_{e,c}\right)^{s\_block} + \frac{1}{2}\left[1-\left(1-P_{e,c}\right)^{s\_block}\right]$$
- OFB and CTR modes $$P_{e,d} = P_{e,c}$$

Chaos-based Data Security

What is chaos?

- Chaos is the art of forming complex from simple
- Chaos can be generated by a non-linear dynamical system
- Edward Lorenz a meteorologist trying to predict the weather
- **Butterfly Effect (1960)**: If a butterfly flaps its wings in Paris, it could change the weather in New York.

- **Lorenz map (1963): 3-D chaotic map**

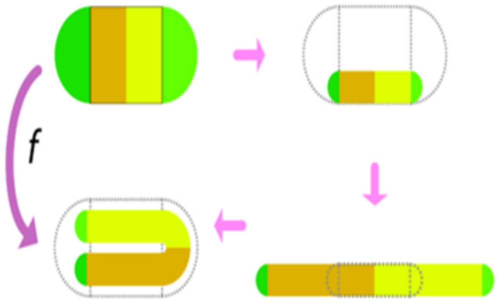
Dynamical non-linear systems can generate chaos

- Discrete-time dynamical system: $X(n) = F\left[X(n-1)\right]$

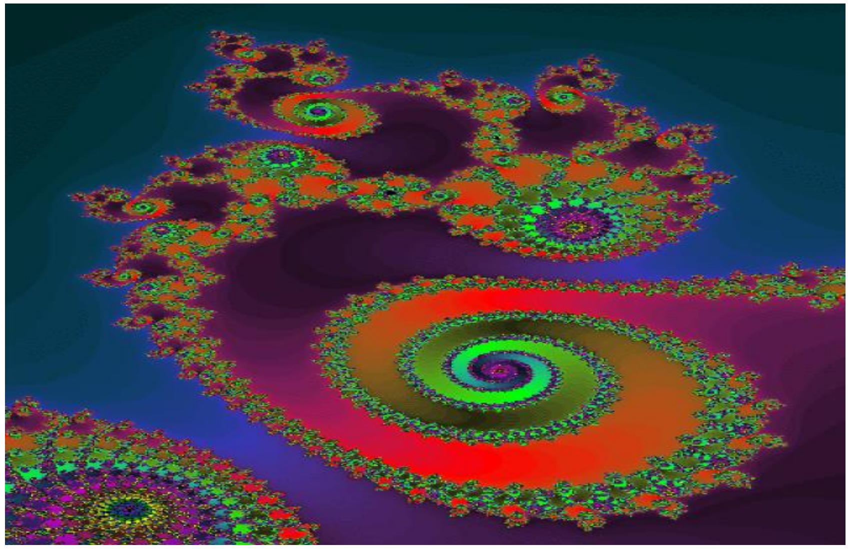
  Recursion relations, iterated maps or simply maps

- Continuous-time dynamical system: $\dot{X}(t) = F\left[X(t)\right]$

Flow: continuous evolution of field lines in the phase space



Application: S. Smale horseshoe map
Horseshoe map is a class of chaotic maps, it is defined geometrically by:
- squishing the square,
- stretching the result into a long strip,
- folding the strip into the shape of a horseshoe



Attractor: Signature & Beauty of dynamical chaos

Chaotic dynamical System

- A chaotic dynamical system is:
  - Deterministic, not random and unpredictable

    Means that the system has no random or noisy inputs. The irregular behaviour arises from the system’s nonlinearity.
  - Aperiodic long term behaviour for continuous-time dynamical system

    Means that there should be trajectories which do not settle down to fixed points, periodic orbits or quasi-periodic orbits as $t \rightarrow \infty$.
  - Periodic behaviour for discrete-time dynamical system
  - Sensitive to initial conditions and initial parameters (Secret Key)

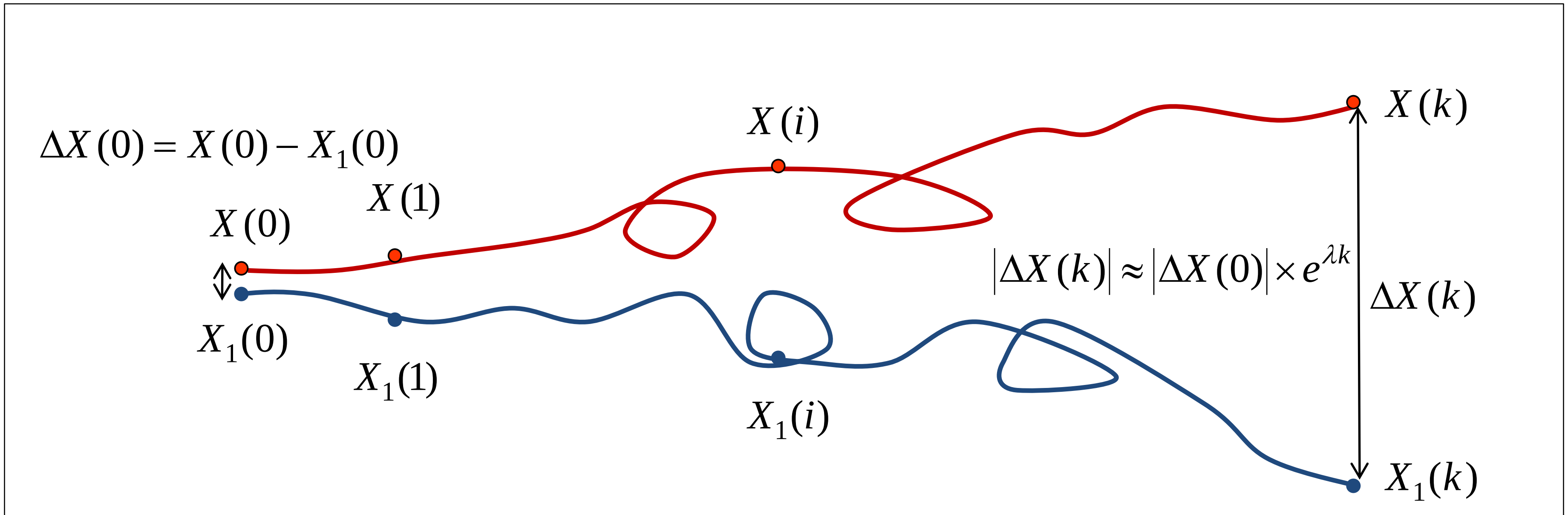
    Means that nearby trajectories separate exponentially fast, which means the system has positive Lyapunov exponent.

Chaotic dynamical System

Low-dimensional chaotic dynamical system $\boldsymbol{X(n) = F[X(n-1)]}$ is capable of complex and unpredictable behavior

The set of points: $\{\boldsymbol{X(0), X(1) = F[X(0)], \cdots, X(k) = F[X(k-1)]}\}$

is called a **trajectory (or orbit)**



$\Delta X(0) = X(0) - X_1(0)$
$X(0)$
$X(1)$
$X(i)$
$X(k)$
$|\Delta X(k)| \approx |\Delta X(0)| \times e^{\lambda k}$
$\Delta X(k)$
$X_1(0)$
$X_1(1)$
$X_1(i)$
$X_1(k)$

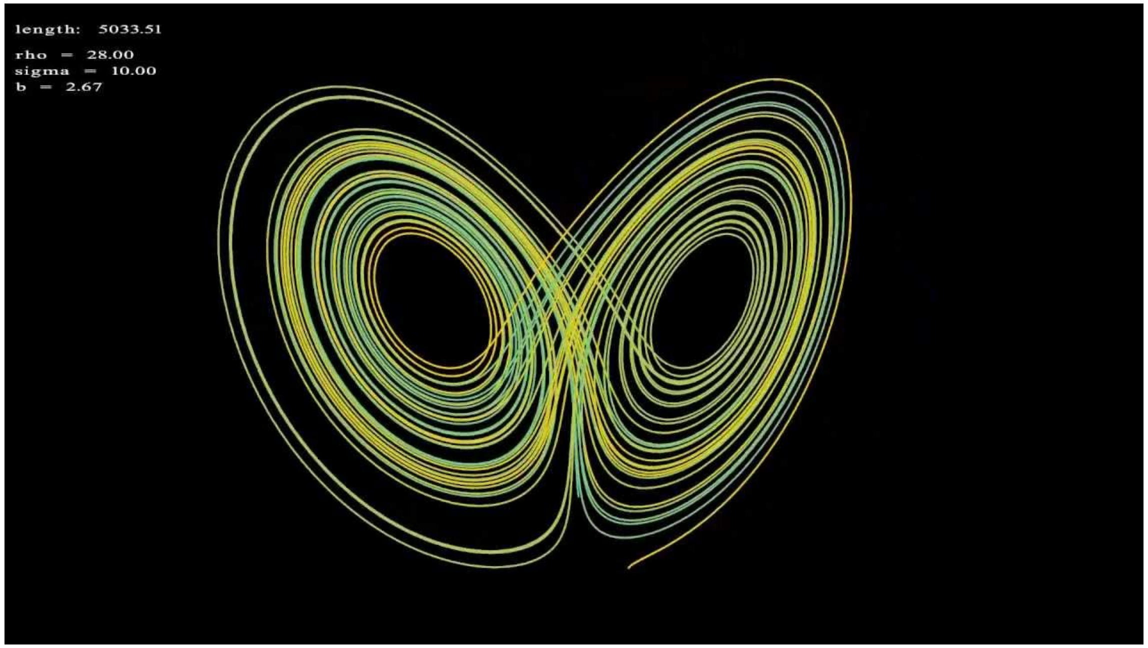
Lyapunov exponent λ measure the divergence rate between orbits

Chaotic dynamical System

- Imperfect knowledge of present, so (practically) no prediction of future
- Dense

  Infinite number of trajectories in finite region of phase space

- Attractor*:* set of orbits to which the system approaches from any initial state (within the attractor basin)



Lorenz Attractor

Why using chaos to secure information?

Useful properties of chaos in secure information

- **Easy to generate: simple discrete-time dynamical system is capable to generate a complex and random like behavior sequences :** $X(n) = F\left[X(n-1)\right]$
- **Chaotic signal is deterministic, not random (we can regenerate it) and it has a broadband spectrum**
- **Chaotic signal is extremely difficult to predict because of the high sensitivity to the secret key**
- **Very big number of orbits in finite region of phase space**

Why are we interested in chaotic signals?
Random sequences for CDMA

- Binary m-sequences

  Good auto-correlation, bad cross-correlation, few codes

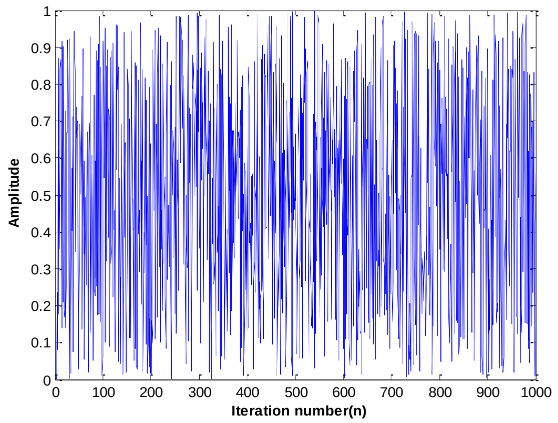
- Binary Gold and Kasami sequences

  Good auto-correlation, acceptable cross-correlation, few Gold codes

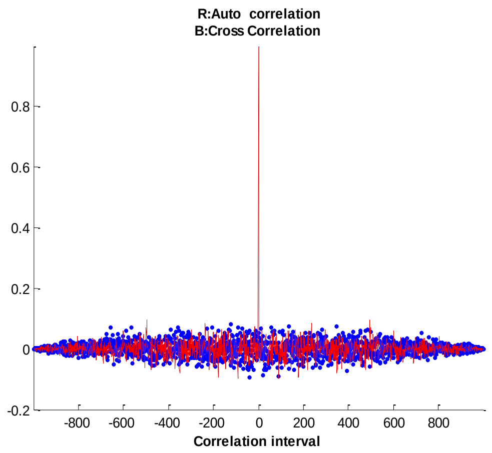
  many Kasami codes

- Binary chaotic sequences

  Very good auto/cross-correlation, very large number of codes



Discrete chaotic sequence



Auto and cross correlations

Examples of systems exhibiting chaos

- Biological Systems
    - Prey-predator models: Logistic map

  Models describing the interaction between predators and their prey to investigate species population year on year.

    - Human physiology

  - Brain: normal brain activity is thought to be chaotic.

  - Heart: normal heart activity is more or less periodic but has variability thought to be chaotic. Fibrillation (loss of stability of the heart muscle) is thought to be chaotic

- Laser instabilities

- Weather systems

Models of the weather including convection, viscous effects and temperature can produce chaotic results. First shown by Edward Lorenz in 1963.

Long term prediction is impossible since the initial state is not known exactly.

- Turbulence

Experiments and modeling show that turbulence in fluid systems is a chaotic phenomenon

Some known chaotic maps used in chaos-based security

- **Chaotic maps used as PRNG:**

  1-D: Logistic, PWLCM, Skew Tent

  3-D: Lorenz

- **Chaotic maps used as permutation layer :**

  2-D : Cat, Standard, and Baker maps

- **Chaotic map used as nonlinear substitution layer :**

  1-D : Skew Tent

- **General scheme of a Stream Cipher & Structure of the proposed Pseudo Chaotic Number Generators: PCNGs**

Presentation of some 1-D chaotic generators

- **Logistic Map:**

Logistic map is a prey-predator model for predicting the population of a species year on year. Also used in many secure communication systems

Population from generation *n-1* to generation *n* is given by:

$$x(n) = f[x(n-1)] = r \times x(n-1) \times [1 - x(n-1)] \; with \begin{cases} 0 < r \leq 4 \\ 0 < x(n-1) < 1 \end{cases}$$

Fixed points: $x(n) = f[x(n-1)] = x(n-1) = \left[1 - \frac{1}{r}\right]$

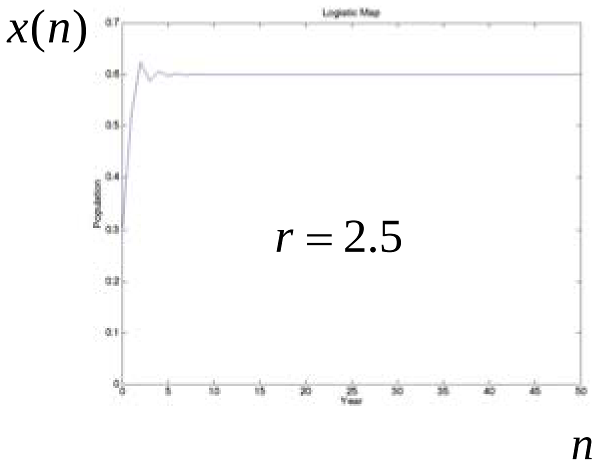
- **Discrete Logistic map: quantized on N-bit**

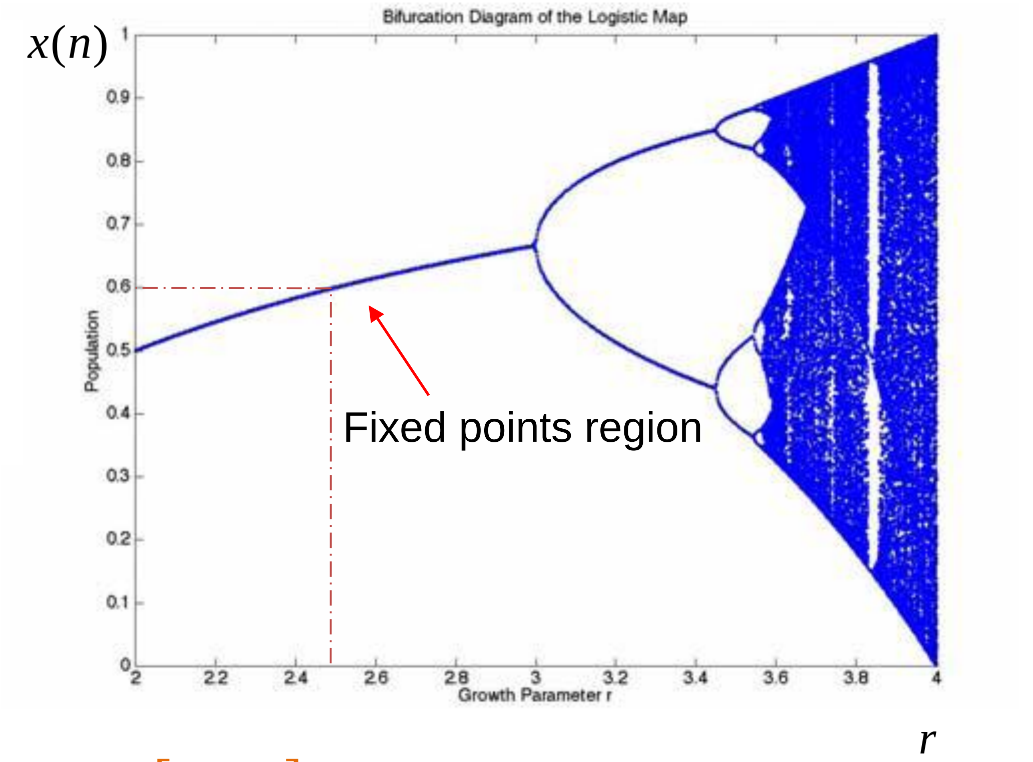
$$X(n) = \begin{cases} \left\lfloor \frac{X(n-1)[2^N - X(n-1)]}{2^{N-2}} \right\rfloor & if \; X(n-1) \neq [3 \times 2^{N-2}, \quad 2^N] \\ 2^N - 1 & if \; X(n-1) = [3 \times 2^{N-2}, \quad 2^N] \end{cases}$$

With: $r = 4 \; and \; 0 < X(n-1) < 2^N$, $\lfloor Z \rfloor$ means *floor (Z), biggest integer no bigger than Z*

r: ***control or growth parameter;*** $x(n), X(n)$: $dynamical \; variables$

Logistic map



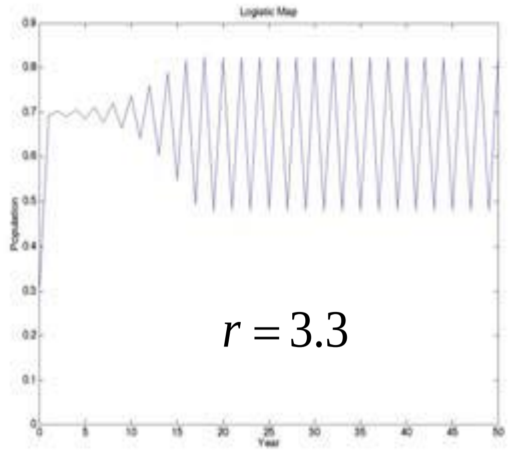


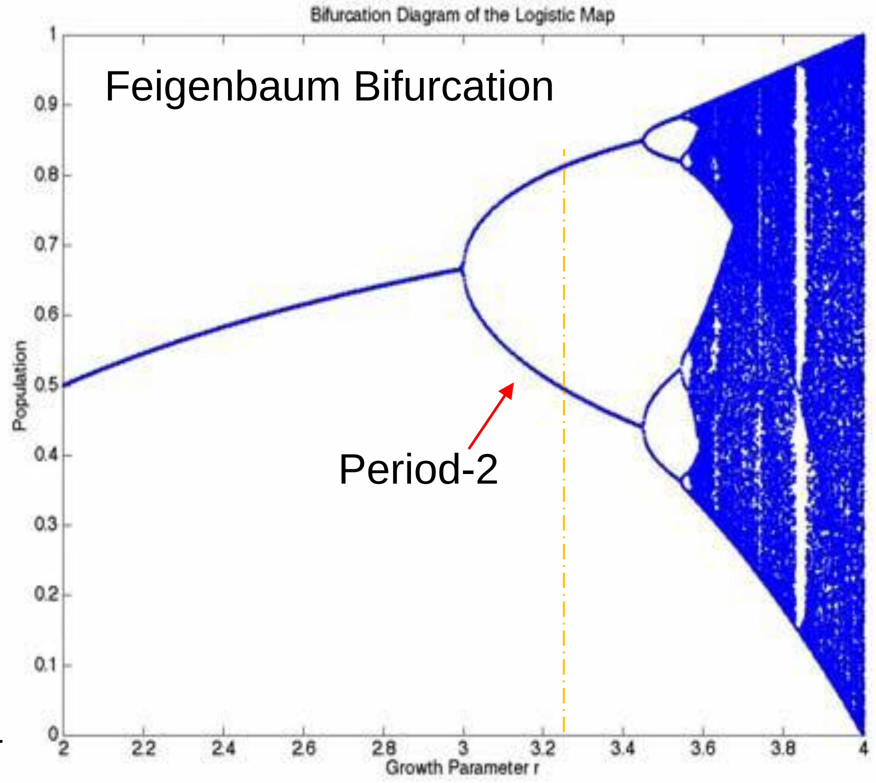
Three fixed points:

$$\boldsymbol{x(n) = f[x(n-1)] = x(n-1) = \left[1 - \frac{1}{r}\right]}$$

$$\boldsymbol{x_f(n) = \begin{cases} \left[1 - \frac{1}{r}\right] = \left[1 - \frac{1}{2.5}\right] = 0.6 \\ 0 \; and \; 1 \end{cases}}$$

Logistic map

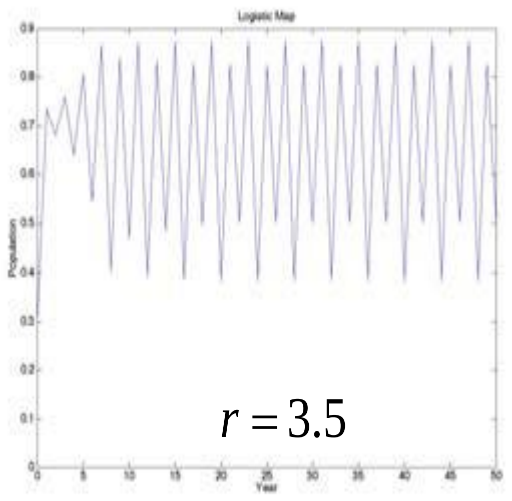


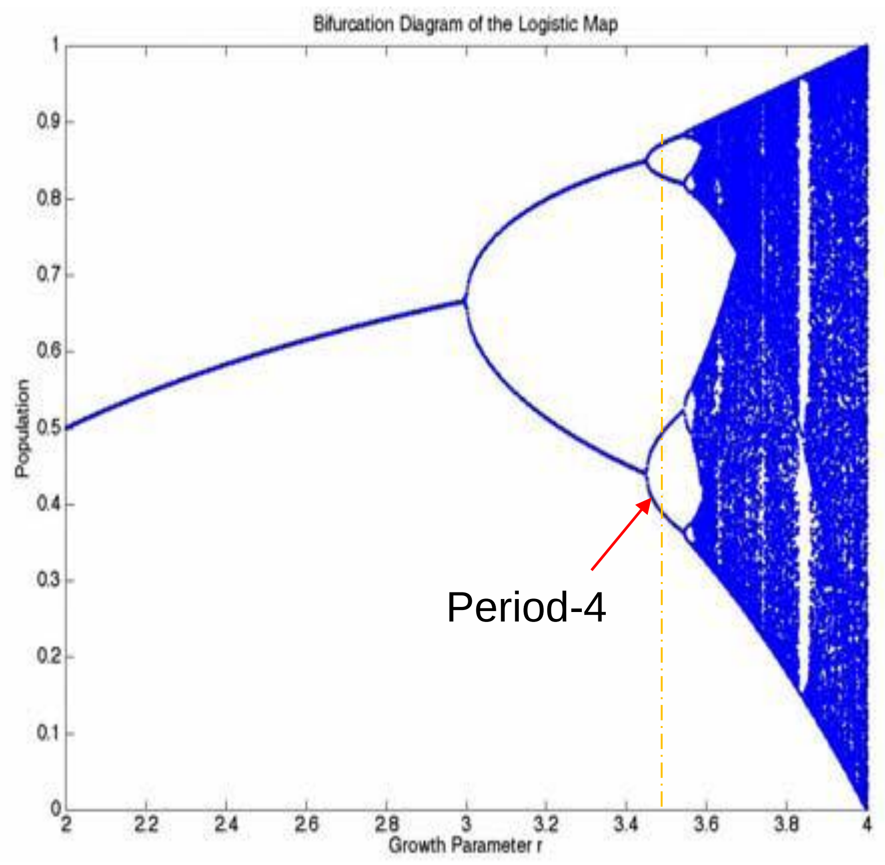


If initial condition is changed, the sequence always converge to the same cycle of period-2, but with a different rate

Bifurcations mark the transition from order into chaos

Logistic map





3.544090 – period of 8

3.564407 – period of 16

3.568759 – period of 32

3.569692 – period of 64

3.569946 – period doubling ends

The attractor branches into two, then four, then eight and so on

$r \geq r_c = 3.56996 \ \rightarrow$ **Chaos emerges**

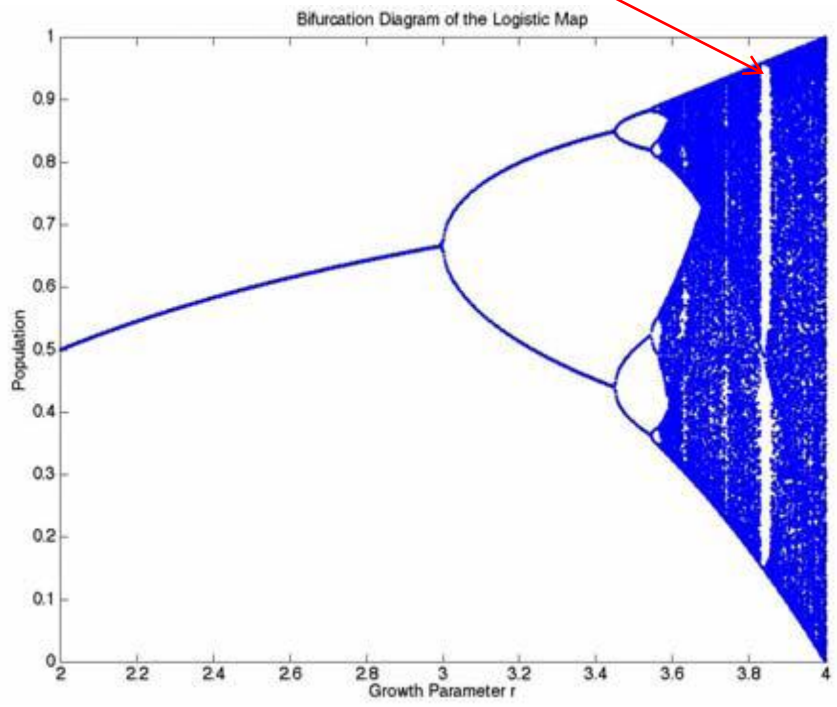
Logistic map

~ 3.569946 – period doubling region ends and chaos begins

3.828427 – small period tripling window opens up

~ 3.855 – period tripling cascade ends and chaos resumes

~ 4.0 chaos reigns



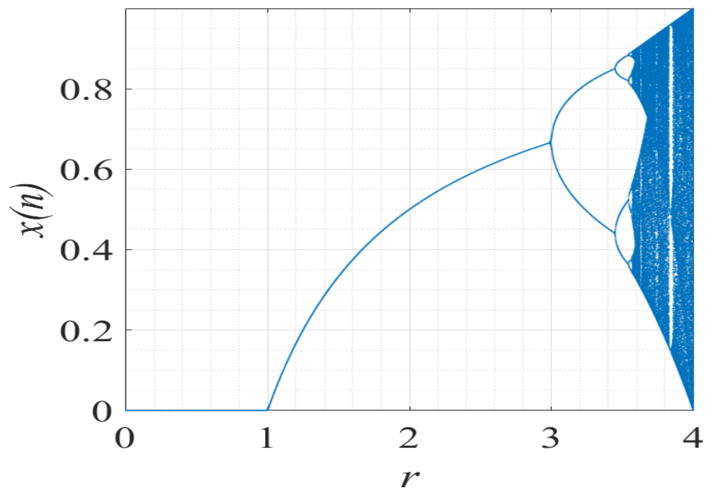
The sequence follows a geometric progression, but soon looks like a mess.

Messy regions are cyclically interspersed with clear "windows".

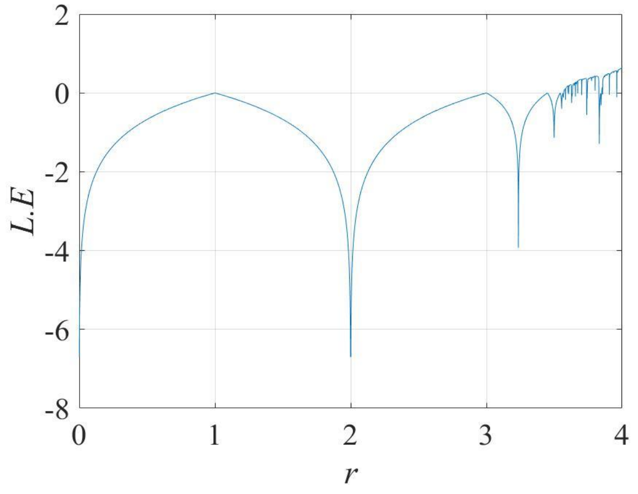
Existence of period-3 windows implies chaos

Chaos does not necessarily imply disorder

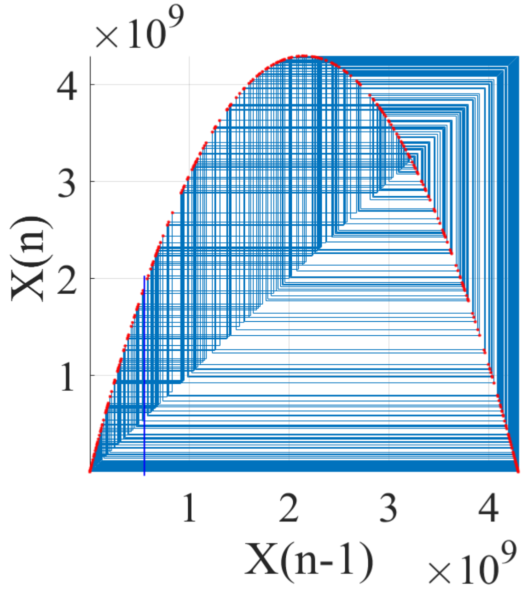
Chaos is the "randomness" in predicting the next iteration



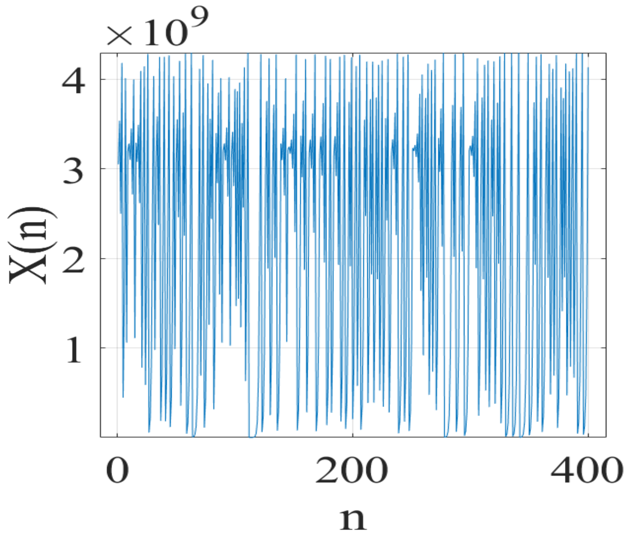
Bifurcation Diagram



Lyapunov Exponent



Strange Attractor: cobweb trajectory



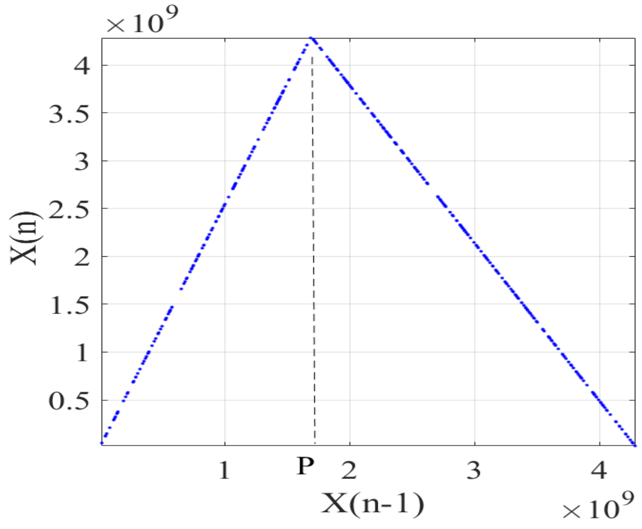
Discrete Variation

Logistic Map

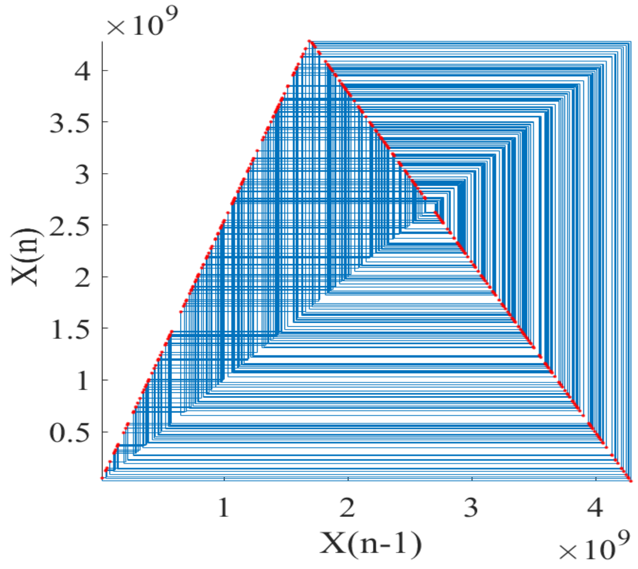
Discrete Skew Tent Map

$$X[n] = F[X(n-1), P] = \begin{cases} \left\lfloor 2^N \times \dfrac{X(n-1)}{P} \right\rfloor & \text{if } 0 < X(n-1) < P \\ \left\lfloor 2^N \times \dfrac{[2^N - X(n-1)]}{2^N - P} \right\rfloor & \text{if } P < X(n-1) < 2^N \\ 2^N - 1 & \text{otherwise} \end{cases}$$

$0 < P < 2^N$: Control parameter



Mapping



Attractor

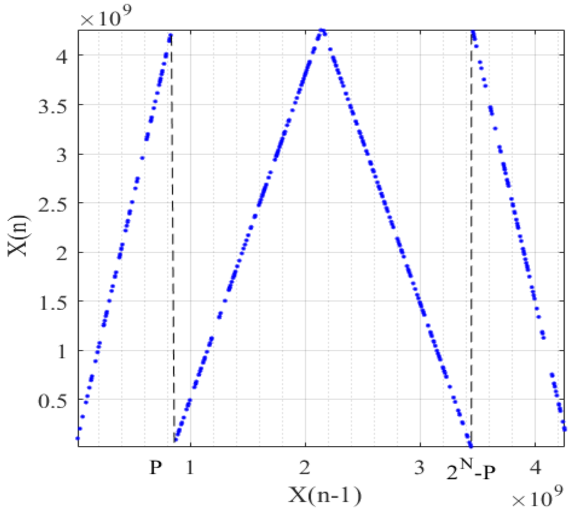
Better cryptographic performances than the Logistic map

Histogram is more uniform. Antagonist characteristics with the PWLCM

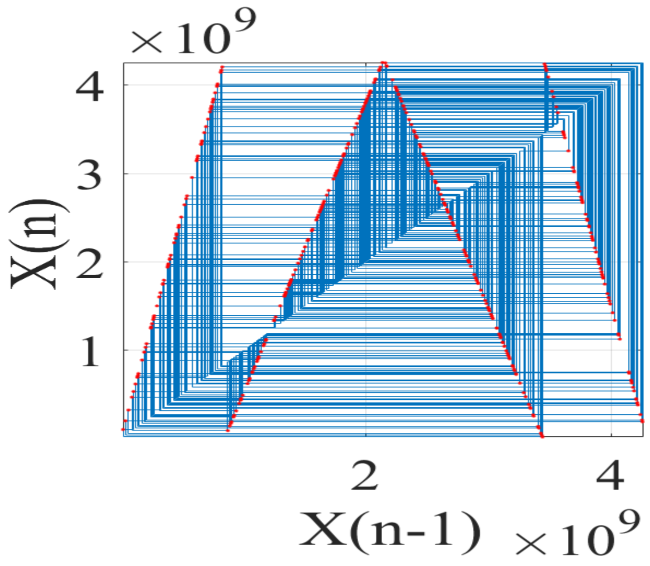
Discrete PWLCM Map

$$X[n] = \begin{cases} \left\lfloor 2^N \times \dfrac{X(n-1)}{P} \right\rfloor & \text{if } 0 < X(n-1) < P \\ \left\lfloor 2^N \times \dfrac{[X(n-1) - P]}{2^{N-1} - P} \right\rfloor & \text{if } P < X(n-1) < 2^{N-1} \\ \left\lfloor 2^N \times \dfrac{[2^N - P - X(n-1)]}{2^{N-1} - P} \right\rfloor & \text{if } 2^{N-1} < X(n-1) < 2^N - P \\ \left\lfloor 2^N \times \dfrac{[2^N - X(n-1)]}{P} \right\rfloor & \text{if } 2^N - P < X(n-1) < 2^N \\ 2^N - 1 & \text{otherwise} \end{cases}$$

$0 < P < 2^{N-1}$: Control parameter

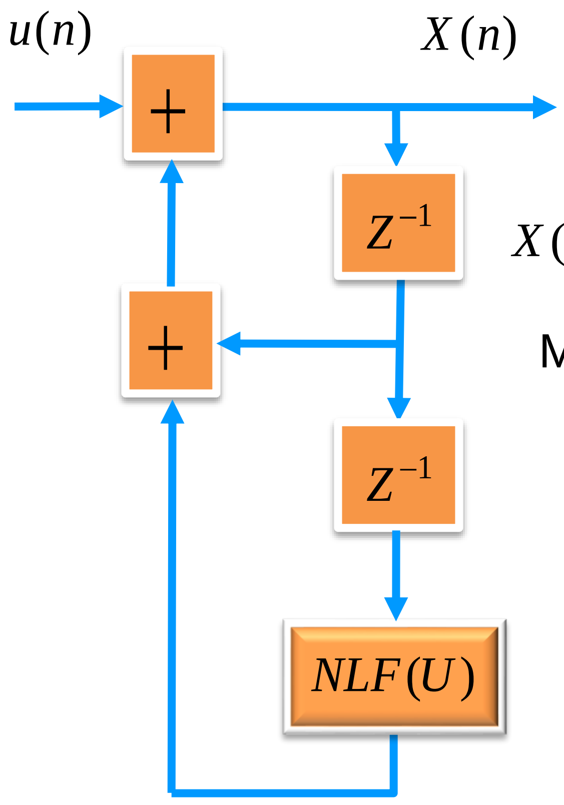


Mapping



Attractor

Frey generator



$$NLF(U) = LCIRC\left[X(n-2)\right] = \mathrm{mod}\left[2\times X(n-2),\, 2^N\right]$$

$$X(n) = \mathrm{mod}\left\{\left[u(n) + X(n-1) + 2\times X(n-2)\right],\, 2^N\right\} \quad 0 \le X(n) \le 2^N - 1$$

Maximum length cycle: $P_{\max}(X) = \left(2^N\right)^2 - 1$

```
function [z] = Frey_generator(N,X1,X2)
% Fery Generator
% Initial states: X1= X(n-1); X2=X(n-2)
% N is finite precision
clc;
N=4;
ns=30;
X1=1;
X2=2;
u=0;
X=zeros(1,ns);
 z=[];
for i=1:ns
    X=mod(u+X1+2*X2, 2^N)
   z=[z X];
    X2=X1;
    X1=X;
end
```

Example: N = 4 bits, *u*(n) = 0

$$X(n) = \mathrm{mod}\left\{\left[X1 + 2\times X2\right], 16\right\} \quad 0 \le X(n) \le 16$$

X2=2, *X1*=1: orbit = 2, 1, 5, 7, 1, 15, 1, 15

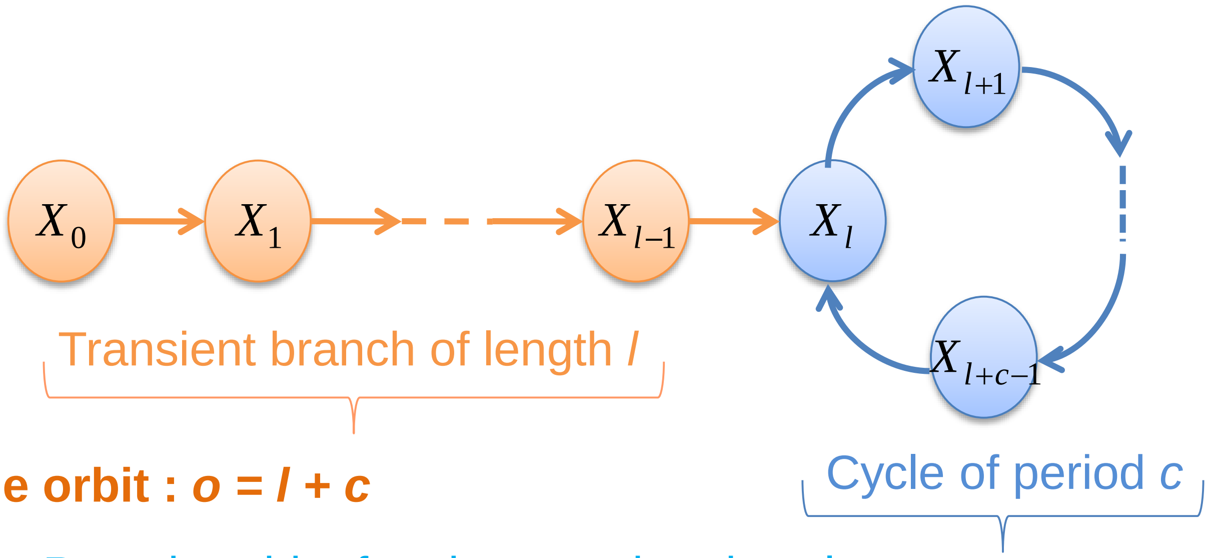
X2=1, *X1*=0: orbit = 1, 0, 2, 2, 6, 10, 6, 10

Effects of the finite precision N

- **In finite precision *N* bits with 1-D chaotic map**

$$X(n) = F[X(n-1), P], \quad X(n) \in [1, 2^N - 1], \;\; \mathrm{n} = 1, 2, \cdots$$

$$X_n = X(n)$$



Length of the orbit : *o* = *l* + *c*

Pseudo-orbit of an integer chaotic value

Maximum length of the orbit : $o_{max} = 2^N - 1$**, extremely rare to obtain**

Analytical rule of the Average length of the orbit is: $\Delta_{nom} \cong \left[\left(2^N\right)^{1/2}\right]^d = 2^{\frac{N}{2}\times d}$

Were d is the number of delays of the recursive structure, if exist

Effects of the finite precision N

Example : N = 4 bits, and two I.Cs

- Situation a : 2 different I.Cs give 2 different cycles

1° I.C : l = 3, c = 5

$$\begin{tikzcd}
& & & & 11 \arrow[r] & 4 \arrow[d] \\
7 \arrow[r] & 13 \arrow[r] & 6 \arrow[r] & 9 \arrow[ur] & 5 \arrow[l] & 14 \arrow[l]
\end{tikzcd}$$

2° I.C : l = 3, c = 2

$$\begin{tikzcd}
3 \arrow[r] & 1 \arrow[r] & 10 \arrow[r] & 8 \arrow[d, bend left] \\
& & & 15 \arrow[u, bend left]
\end{tikzcd}$$

- Situation b : 2 different I.Cs give the same cycle c

1° I.C : l = 3, c = 5

2° I.C : l = 2, c = 5

$$\begin{tikzcd}
& & & & 11 \arrow[r] & 4 \arrow[d] \\
7 \arrow[r] & 13 \arrow[r] & 6 \arrow[r] & 9 \arrow[ur] & 5 \arrow[l] & 14 \arrow[l] \\
& 2 \arrow[ur] & & & &
\end{tikzcd}$$

- Situation c l = 1, c = 1

$$\begin{tikzcd}
12 \arrow[r] & 0 \arrow[loop right]
\end{tikzcd}$$

Fixed point