Chaos-based Cryptography Primitives for Data Security

Safwan El Assad

https://scholar.google.com/citations?user=69Jk1jQAAAAJ&hl=fr

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

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

Today, we all live today in a cyber world, and modern technologies involve fast communication links, potentially between billions of devices, via complex networks (satellites, mobile phones, the Internet, Internet of Things, etc.). Thus, the question of how we protect public communication networks and devices from passive and active attacks that could threaten public safety (sabotage, espionage, cyber terrorism) and personal privacy has become one of great importance.

Cryptography and Chaos-based Cryptography

Outline

❑ **Generalities**

❑ **Classical cryptography**

❑ **AES Algorithm**

❑ **Chaos-based data security**

  - ❑ **What is chaos? Why using chaos to secure information?**
  - ❑ **Some known chaotic maps used in chaos-based security**
  - ❑ **Design of efficient stream ciphers based on pseudo random number generators of chaotic sequences (PRNGs-CS) & performance evaluation**
  - ❑ **Design of efficient chaos-based cryptosystems (block ciphers) and performance evaluation**

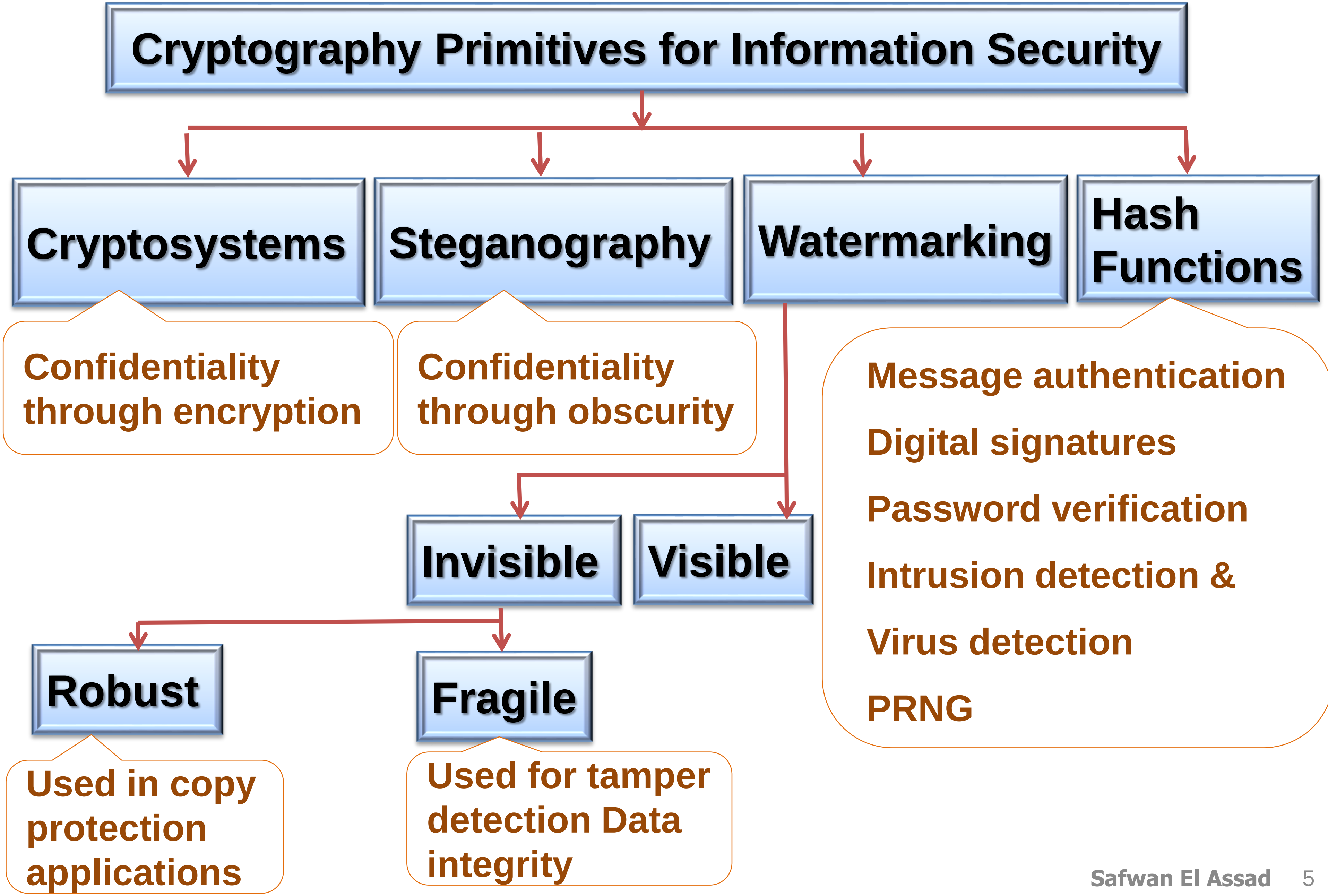
Outline

- ❑ Design of efficient chaos-based steganography systems

❑ **Appendix**

- ❑ Various block cipher modes: Symmetric key algorithms
- ❑ Error Propagation : summary of bit errors on decryption

Generalities



Cryptography Primitives for Information Security

Cryptosystems

Confidentiality through encryption

Steganography

Confidentiality through obscurity

Watermarking

Invisible

Robust

Used in copy protection applications

Fragile

Used for tamper detection Data integrity

Visible

Hash Functions

Message authentication

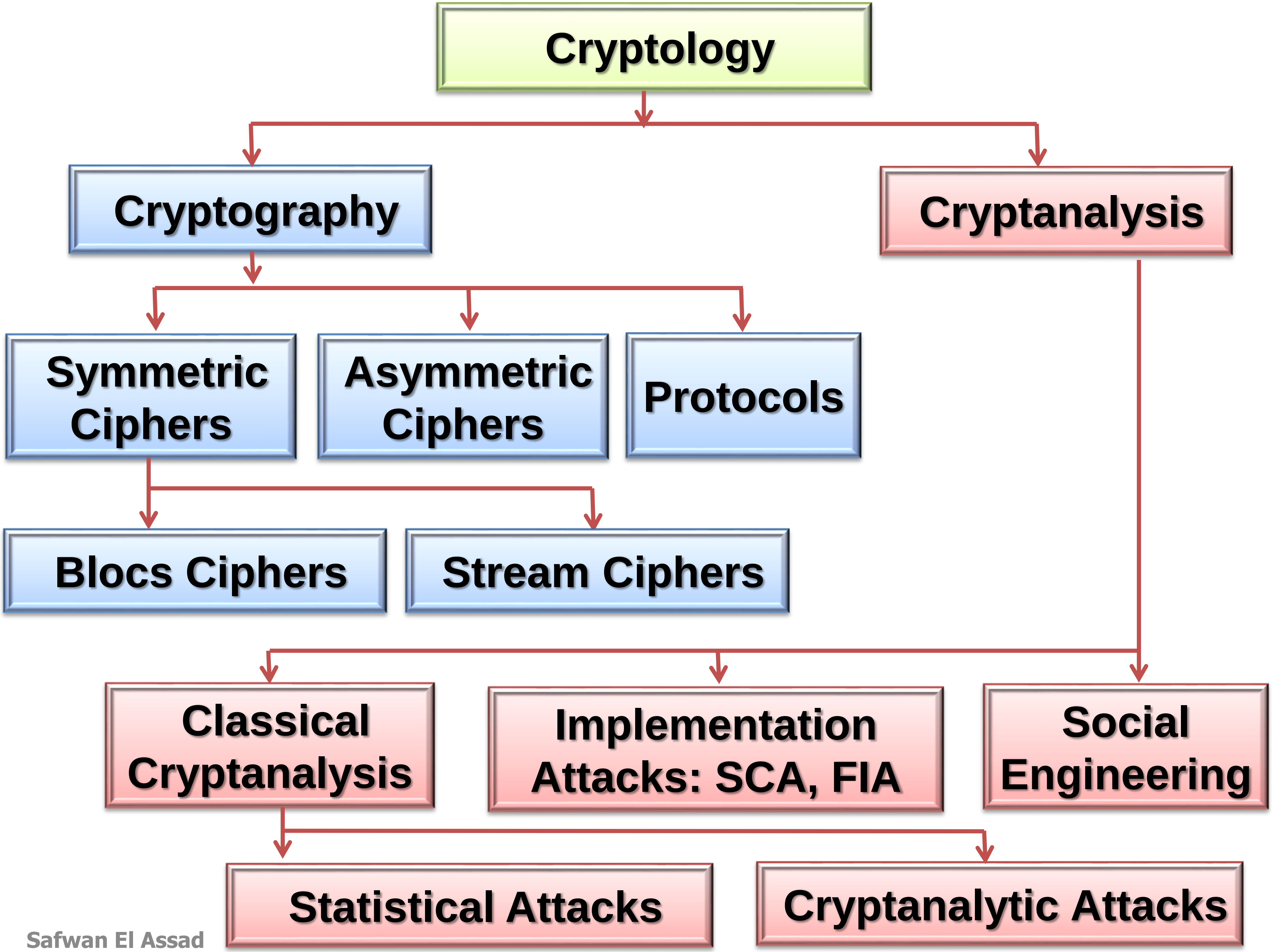
Digital signatures

Password verification

Intrusion detection &

Virus detection

PRNG



Cryptology

Cryptography

Cryptanalysis

Symmetric Ciphers

Asymmetric Ciphers

Protocols

Blocs Ciphers

Stream Ciphers

Classical Cryptanalysis

Implementation Attacks: SCA, FIA

Social Engineering

Statistical Attacks

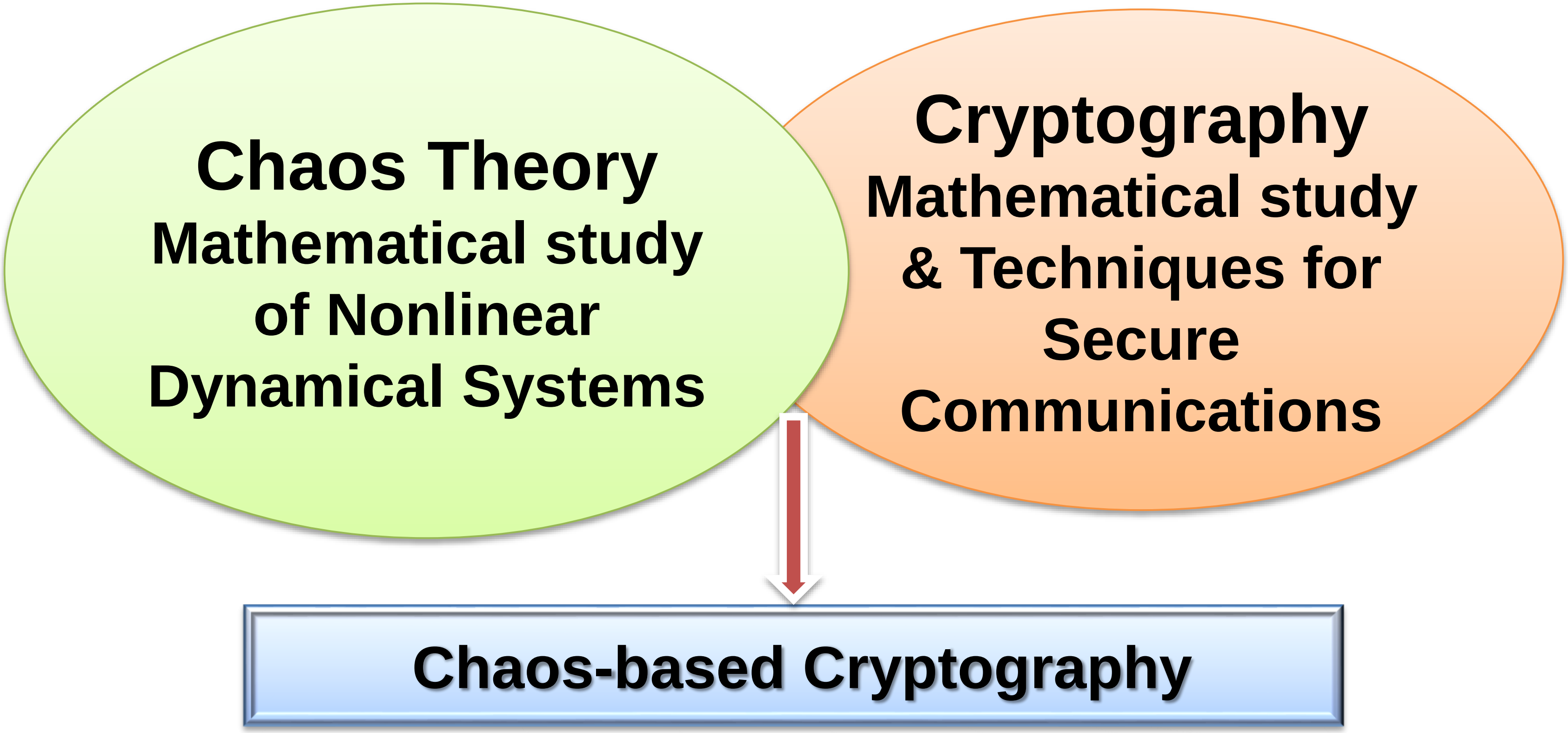
Cryptanalytic Attacks

Chaos & Cryptography

Both chaotic map and encryption system are deterministic

Both are unpredictable, if the secret key is not known

Both used iterative transformation

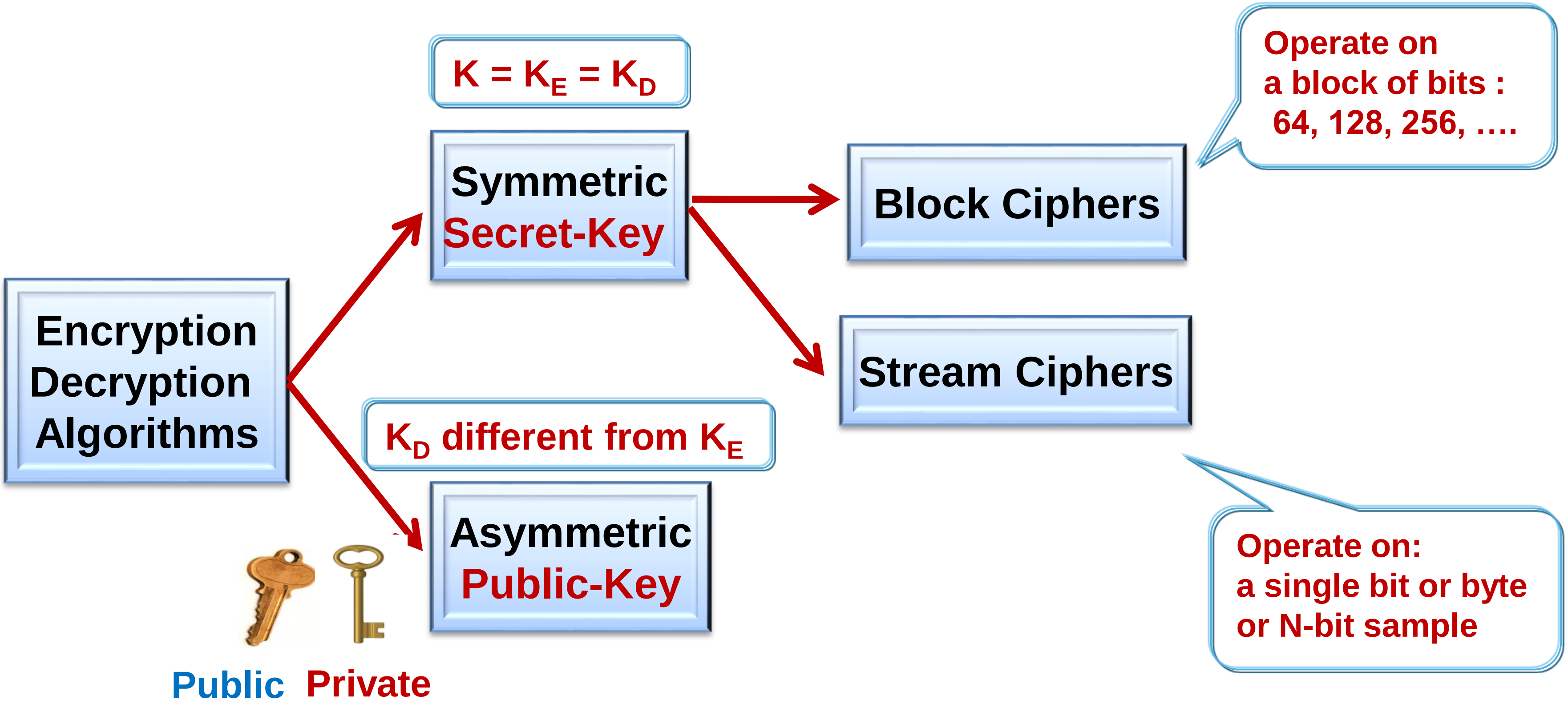


Chaos Theory
Mathematical study of Nonlinear Dynamical Systems

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

Block Ciphers

Operate on a block of bits : 64, 128, 256, ….

Stream Ciphers

Operate on: a single bit or byte or N-bit sample

K_D different from K_E

Asymmetric Public-Key

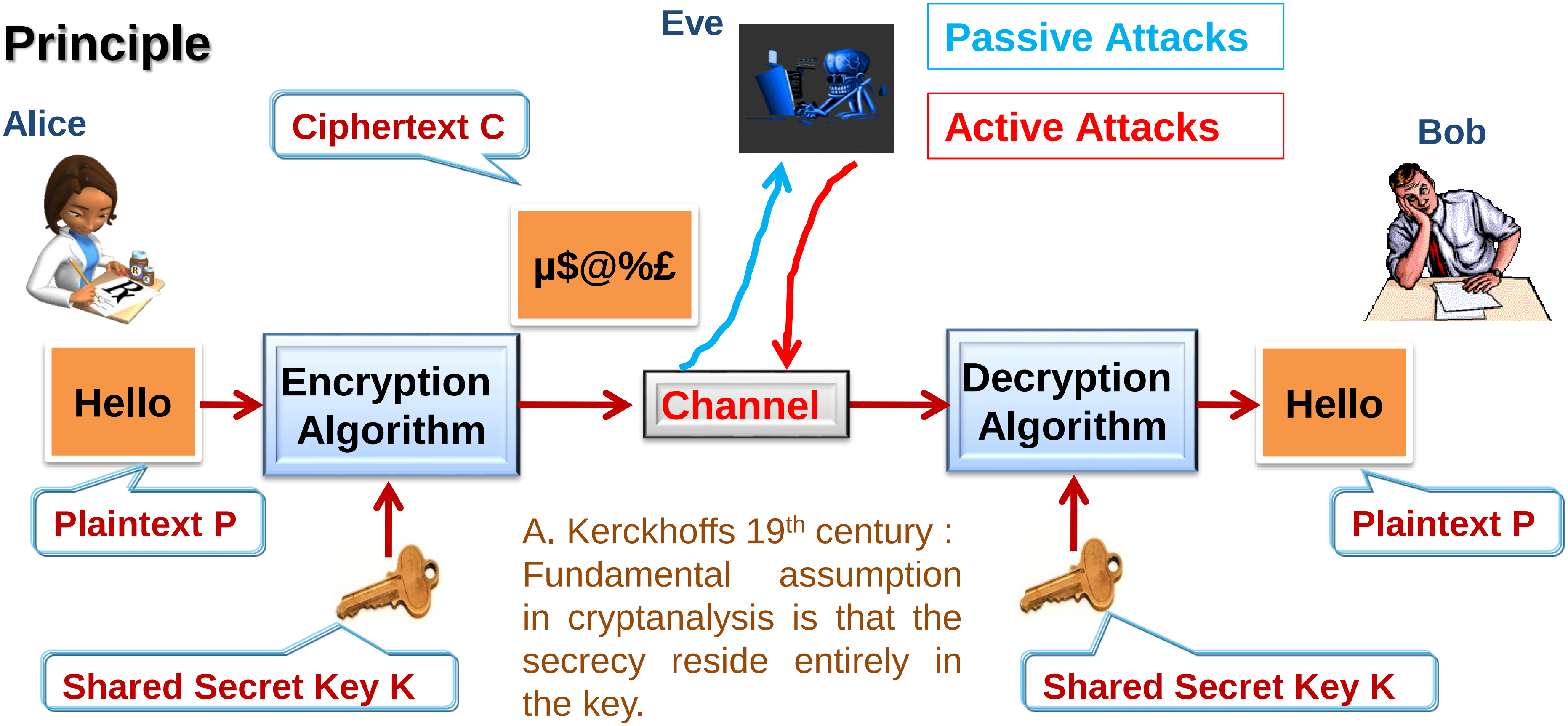
Public Private

Symmetric encryption is # 1000

faster than asymmetric encryption

Classical cryptosystems
Symmetric key algorithms

Principle

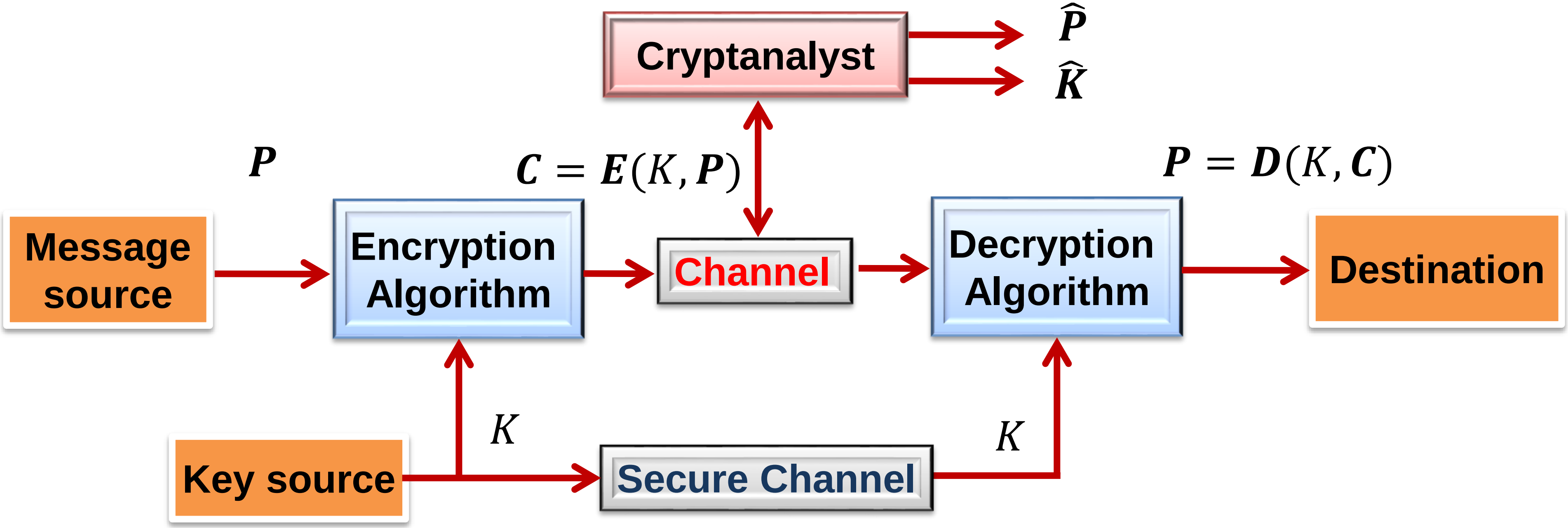


A. Kerckhoffs 19th century : Fundamental assumption in cryptanalysis is that the secrecy reside entirely in the key.

Passive attacks: Pb of Confidentiality

Active attacks: Pb of Data Integrity and Message Authentication

Model of Symmetric Cryptosystem



Cryptanalyst

$\widehat{\boldsymbol{P}}$

$\widehat{\boldsymbol{K}}$

$\boldsymbol{P}$

$\boldsymbol{C} = \boldsymbol{E}(K, \boldsymbol{P})$

$\boldsymbol{P} = \boldsymbol{D}(K, \boldsymbol{C})$

Message source

Encryption Algorithm

Channel

Decryption Algorithm

Destination

K

K

Key source

Secure Channel

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:

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

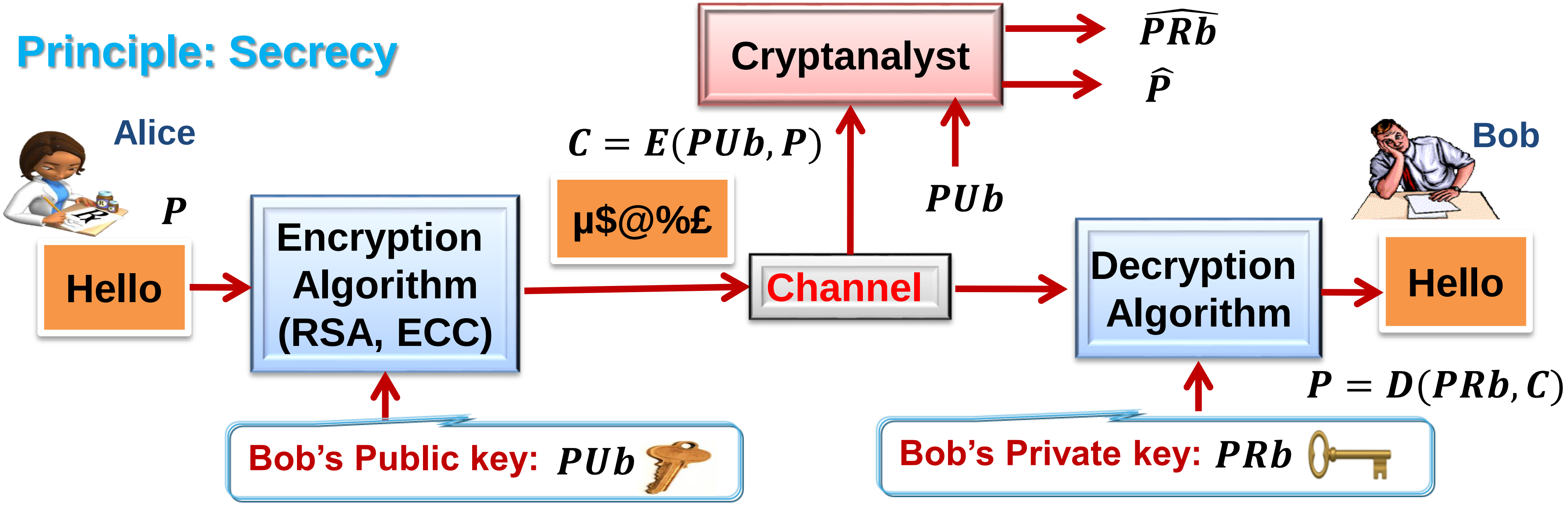
With $P = \{p_1, p_2, \cdots, p_n\}$, $C = \{c_1, c_2, \cdots, c_n\}$; $E(K, p_i) = c_i$ *and* $D(K, c_i) = p_i \in \mathcal{A}$

$\mathcal{A}$ 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*).

Public-Key cryptosystems: Asymmetric algorithms

Principle: Secrecy



Cryptanalyst → $\widehat{PRb}$, $\hat{P}$

Alice

Bob

P

Hello

Encryption Algorithm (RSA, ECC)

$C = E(PUb, P)$

µ$@%£

Channel

PUb

Decryption Algorithm

Hello

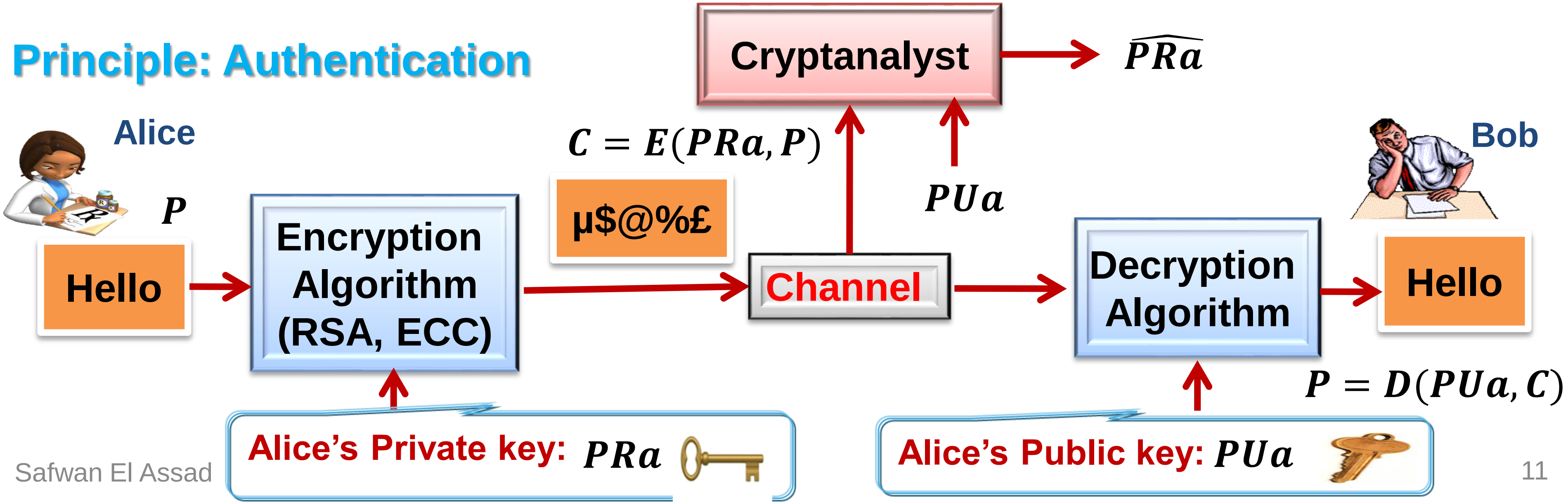
$P = D(PRb, C)$

Bob's Public key: PUb

Bob's Private key: PRb

Exhaustive attacks: an optical computer is # 1,000 times faster than a classical computer

Principle: Authentication



Cryptanalyst → $\widehat{PRa}$

Alice

Bob

P

Hello

Encryption Algorithm (RSA, ECC)

$C = E(PRa, P)$

µ$@%£

Channel

PUa

Decryption Algorithm

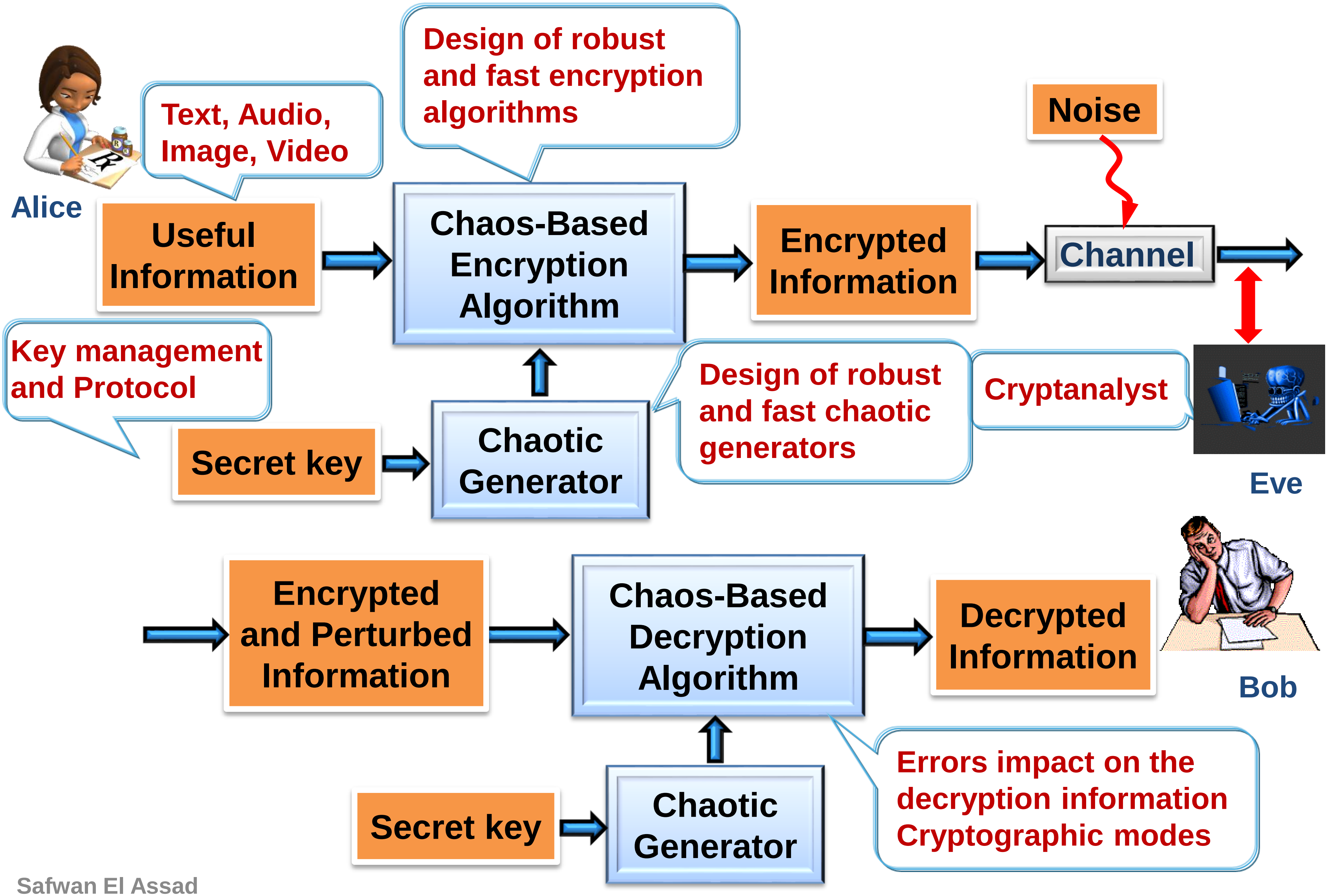
Hello

$P = D(PUa, C)$

Alice's Private key: PRa

Alice's Public key: PUa

Principle of chaos-based cryptosystems



Alice

Text, Audio, Image, Video

Useful Information

Design of robust and fast encryption algorithms

Chaos-Based Encryption Algorithm

Encrypted Information

Noise

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

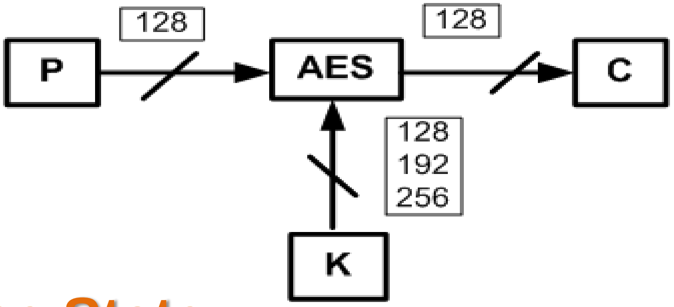
Overview of the AES Algorithm

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}

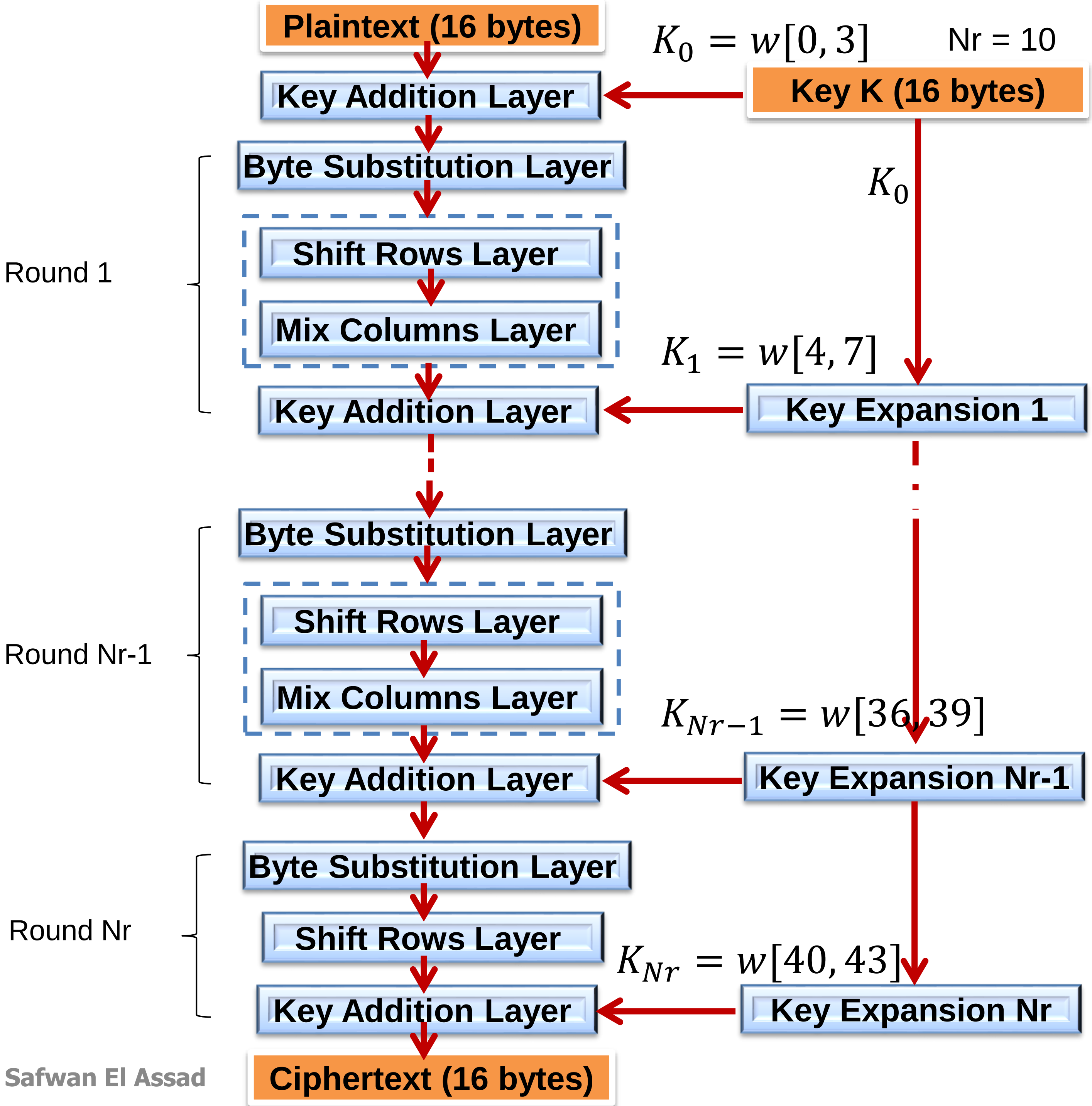
$s[r,c] = in[r+4c]$ **for** $0 \le r < 4$ **and** $0 \le c < 4$.

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

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

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

AES Encryption Bloc Diagram



Plaintext (16 bytes)

$K_0 = w[0,3]$

Nr = 10

Key K (16 bytes)

Key Addition Layer

K_0

Round 1

Byte Substitution Layer

Shift Rows Layer

Mix Columns Layer

$K_1 = w[4,7]$

Key Addition Layer

Key Expansion 1

Round Nr-1

Byte Substitution Layer

Shift Rows Layer

Mix Columns Layer

$K_{Nr-1} = w[36,39]$

Key Addition Layer

Key Expansion Nr-1

Round Nr

Byte Substitution Layer

Shift Rows Layer

$K_{Nr} = w[40,43]$

Key Addition Layer

Key Expansion Nr

Ciphertext (16 bytes)

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	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
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:

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

- 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 $\boldsymbol{P(x) = x^8 + x^4 + x^3 + x + 1}$

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

×

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

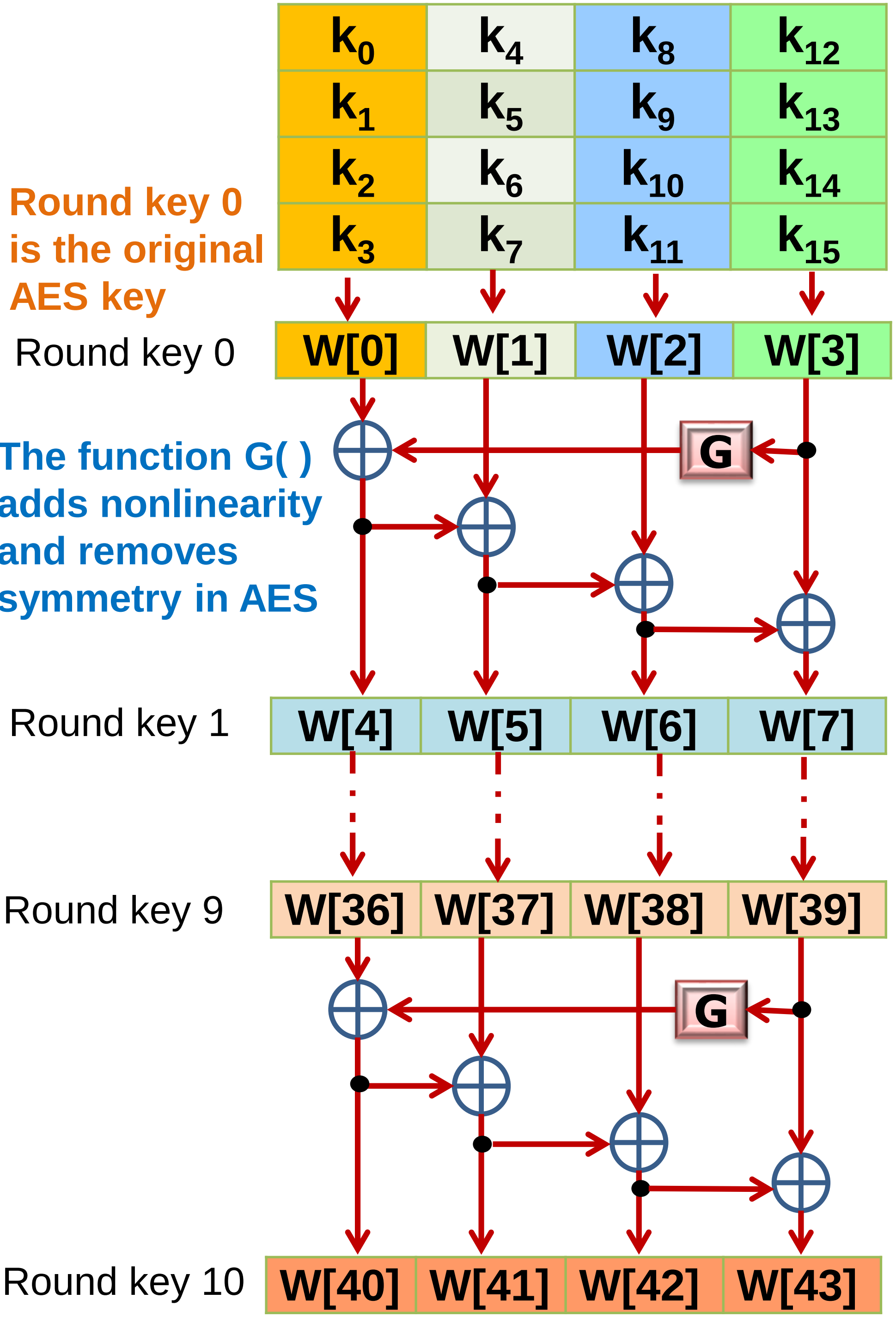
$$\begin{bmatrix} 04 & e0 & 48 & 28 \\ 66 & cb & f8 & 06 \\ 81 & 19 & d3 & 26 \\ e5 & 9a & 7a & 4c \end{bmatrix} \oplus \begin{bmatrix} a0 & 88 & 23 & 2a \\ fa & 54 & a3 & 6c \\ fe & 2c & 39 & 76 \\ 17 & b1 & 39 & 05 \end{bmatrix} = \begin{bmatrix} a4 & 68 & 6b & 02 \\ 9c & 9f & 5b & 6a \\ 7f & 35 &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

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

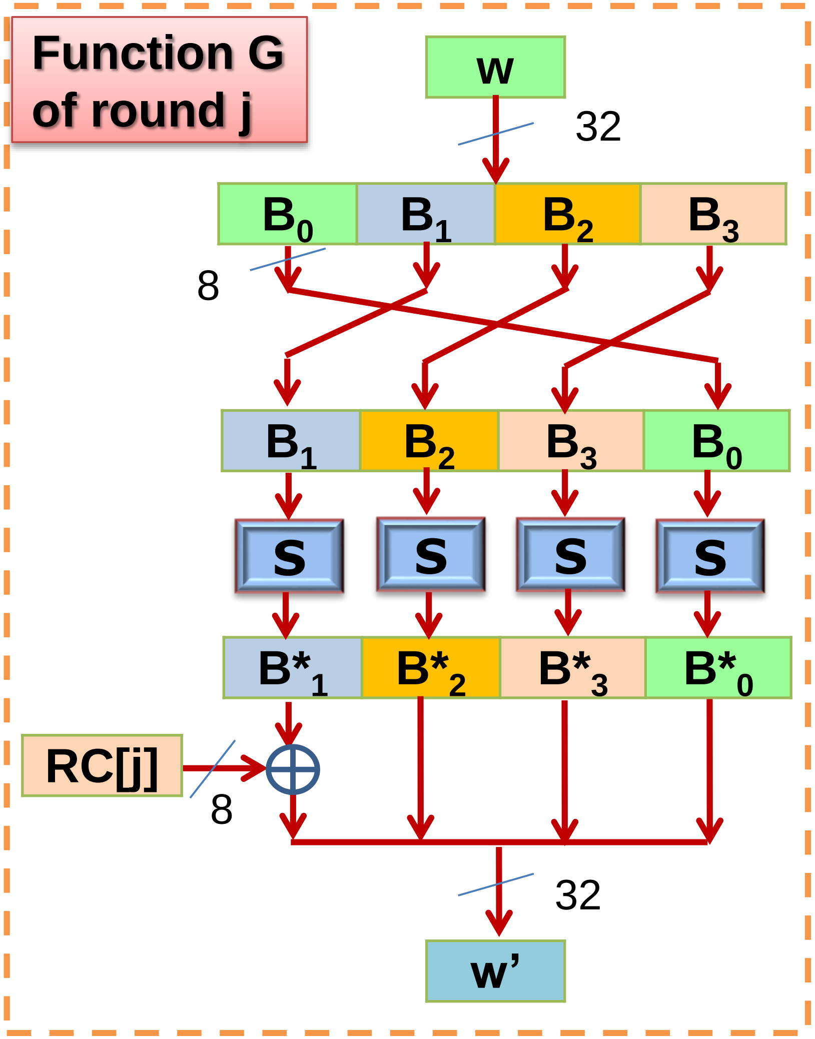
Key Expansion algorithm for 128-bit Key AES



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

The function G( ) adds nonlinearity and removes symmetry in AES



Function G of round j

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.

The set of these 256 polynomials is the finite field $GF(2^8)$.

- 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:

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

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)

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

$$
\begin{array}{rrrrrrrrr}
 & & & \multicolumn{6}{l}{A(x) \times B(x) = x^7 + x^5 + x^4 + 1} \\
 & & & & \multicolumn{5}{r}{\times \left(x^5 + x^2 + 1\right)} \\
\hline
 & & & x^7 + & & x^5 + & x^4 + & & 1 \\
 & & x^9 + & x^7 + & x^6 + & & & x^2 & \\
x^{12} + & x^{10} + & x^9 + & & & x^5 & & & \\
\hline
x^{12} + & x^{10} + & & & x^6 & & + x^4 + & x^2 + & 1
\end{array}
$$

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)

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

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: $R(x) = C(x) \; mod \; P(x)$

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

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:

$x^k \, mod \, P(x) = [P(x) - x^k]$ ***in AES:*** $x^8 \, mod \, P(x) = x^4 + x^3 + x + 1$ **(1)**

Consider:

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

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

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:

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

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

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

Indeed:

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

Multiplication by a higher power of x can be achieved by repeated Eq (2). By adding intermediate results, multiplication by any constant in $GF\left(2^8\right)$ 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
x	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
	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: $A(x) = x^7 + x^5 + x^4 + 1 = (10110001) = \{\mathrm{b1}\} = \{xy\}$

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

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

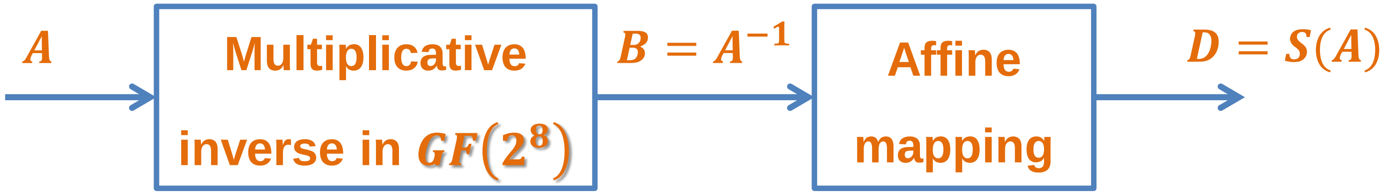
Mathematical description of the AES S-Box

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

1. **Map each byte $A \in \boldsymbol{GF}(2^8)$ 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 $C = (01100011) = \{63\}$



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:

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

From multiplicative inverse:

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

From affine mapping:

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

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

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

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

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

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

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

Multiplication Inverse

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

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

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: $r_0 = 84,\ r_1 = 30$

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

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

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

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

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

As long as $(r_0 - qr_1) > 0$. Then:

$$gcd(r_0, r_1) = gcd\big((r_0 - qr_1), r_1\big) = gcd(r_0\, mod\, r_1, r_1) = gcd(r_1, r_0\, mod\, r_1) \qquad (4)$$

Because $r_0\, mod\, r_1 < r_1$

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

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

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

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

Example:

i	$r_{i-2} = q_{i-1} \times r_{i-1} + r_i \quad 0 \le r_i < r_{i-1}$	$gcd(r_0, r_1) = gcd(973, 301)$
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	$r_2 = q_3 \times r_3 + r_4 \quad 0 < r_4 < r_3$	$70 = 3 \times 21 + 7 \quad 0 < 7 < 21$
⋮	⋮ ⋮	$21 = 3 \times 7 + 0$
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	$r_2 = q_3 \times r_3 + r_4 \quad 0 < r_4 < r_3$	$r_4 = [s_4] \times r_0 + [t_4] \times r_1$
$\vdots$	$\vdots \quad \vdots$	$\vdots$
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:

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

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

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

$\boldsymbol{s_0 = 1}$, $\boldsymbol{s_1 = 0}$, $\boldsymbol{t_0 = 0}$, $\boldsymbol{t_1 = 1}$.

AES Decryption

Chaos-based Cryptography

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**

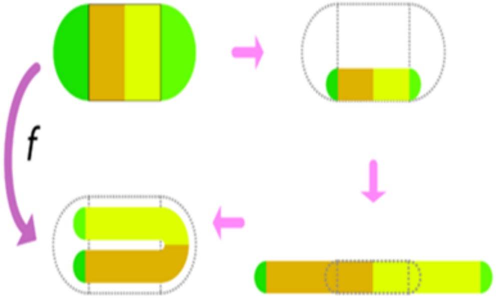
Dynamical non-linear systems can generate chaos

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

Recursion relations, iterated maps or simply maps

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

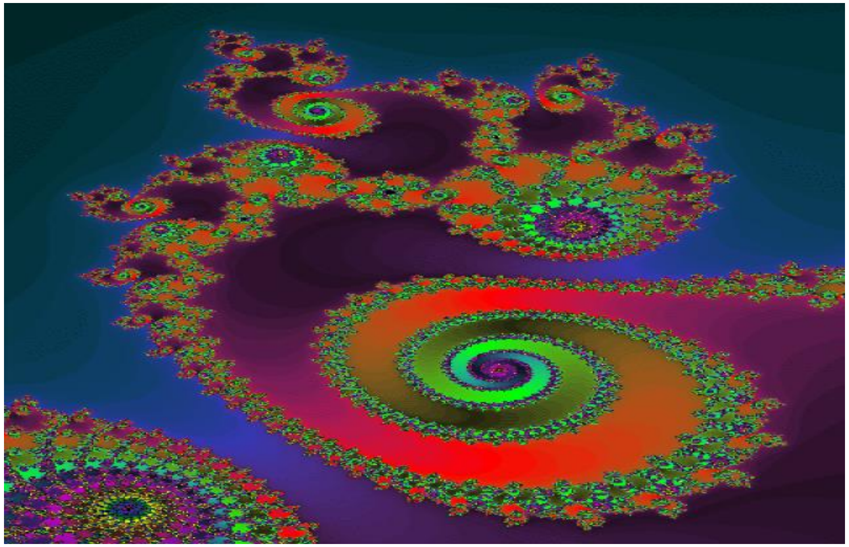
Flow: continuous evolution of field lines in the phase space



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 →∞.
  - Periodic behaviour for discrete-time dynamical system
  - Sensitive to initial conditions and initial parameters (Secret Key)

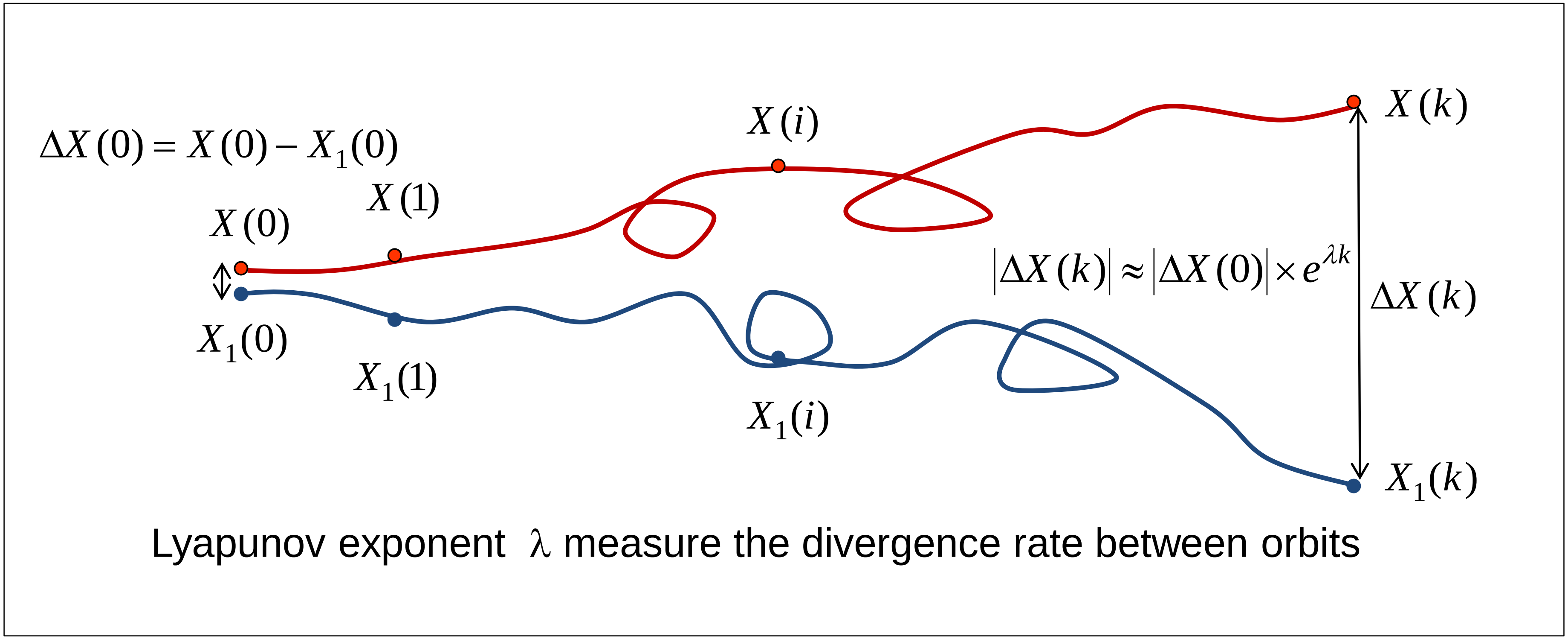
    Means that nearby trajectories separate exponentially fast, which means the system has positive Lyapunov exponent.

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)**



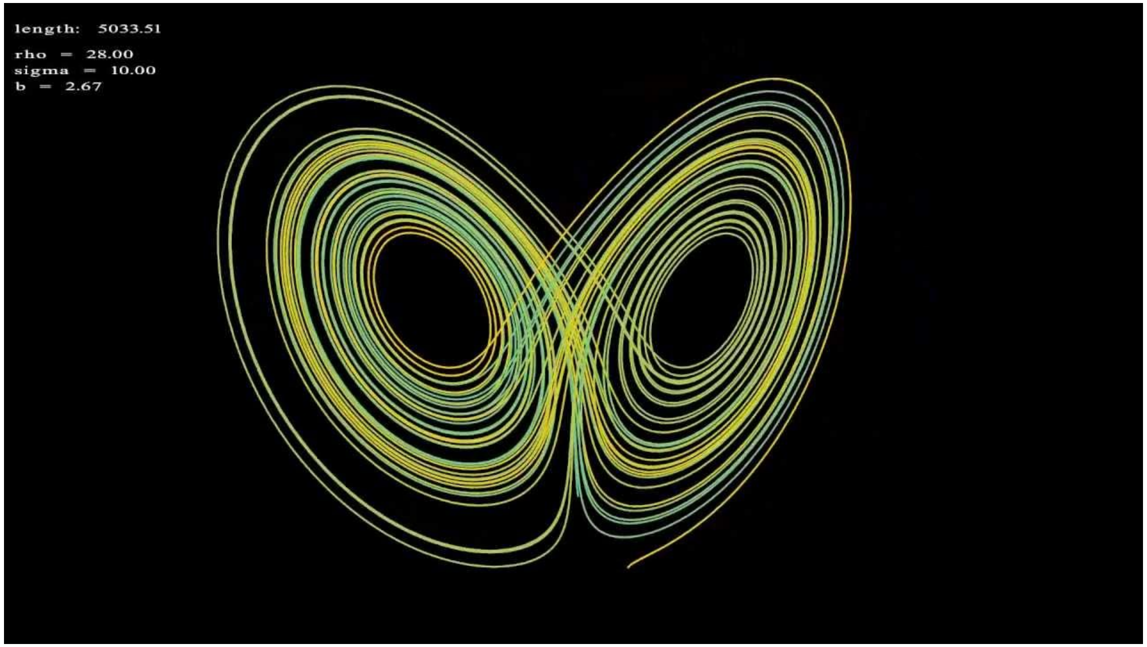
$\Delta X(0) = X(0) - X_1(0)$
$X(0)$
$X(1)$
$X(i)$
$X(k)$
$X_1(0)$
$X_1(1)$
$X_1(i)$
$X_1(k)$
$|\Delta X(k)| \approx |\Delta X(0)| \times e^{\lambda k}$
$\Delta X(k)$

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

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 cryptography

- Chaotic maps used as PRNG:

1-D: Logistic, PWLCM, Skew Tent

3-D: Lorenz, Chebyshev

4-D: Chebyshev polynomial, Lorenz Hyperchaos, Chen Hyperchaos, Qi Hyperchaos.

- Chaotic maps used as permutation layer :

2-D : Cat, Standard, and Baker map

- Chaotic map used as nonlinear substitution layer :

1-D : Skew Tent

- Effects of the finite precision N

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

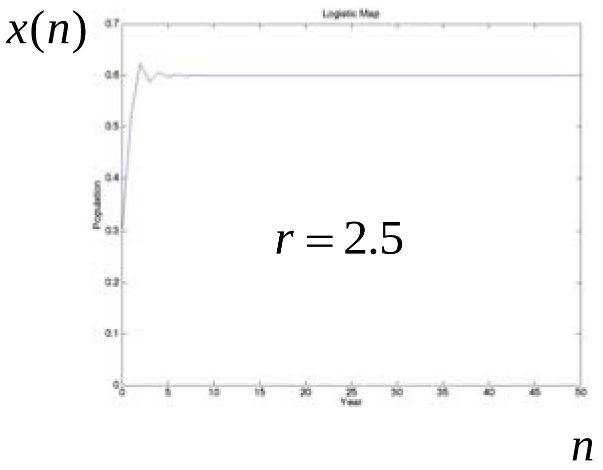
- **Discrete Logistic map: quantized on N-bit (N = 32 bits)**

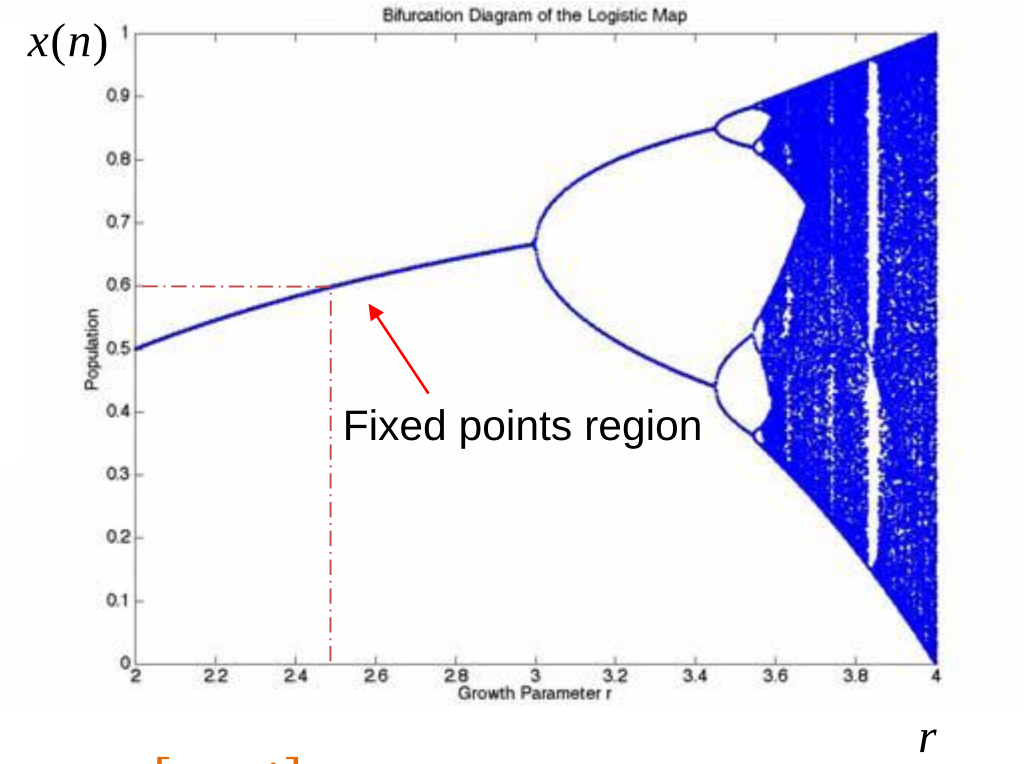
$$X(n) = \begin{cases} \left\lfloor \dfrac{X(n-1)[2^N - X(n-1)]}{2^{N-2}} \right\rfloor & if \; X(n-1) \neq \left[3 \times 2^{N-2} - 1, \quad 2^{N-1}\right] \\ 3 \times 2^{N-2} - 1 & if \quad X(n-1) = 3 \times 2^{N-2} \\ 2^N - 1 & if \quad X(n-1) = 2^{N-1} \end{cases}$$

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;*** $\quad x(n), X(n)$: $dynamical \; variables$

Logistic map





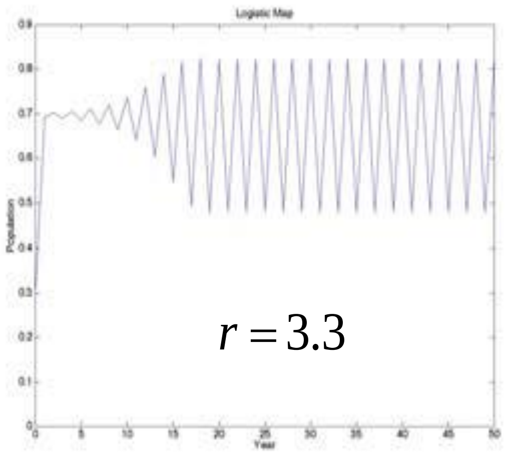
Fixed points region

Three fixed points:

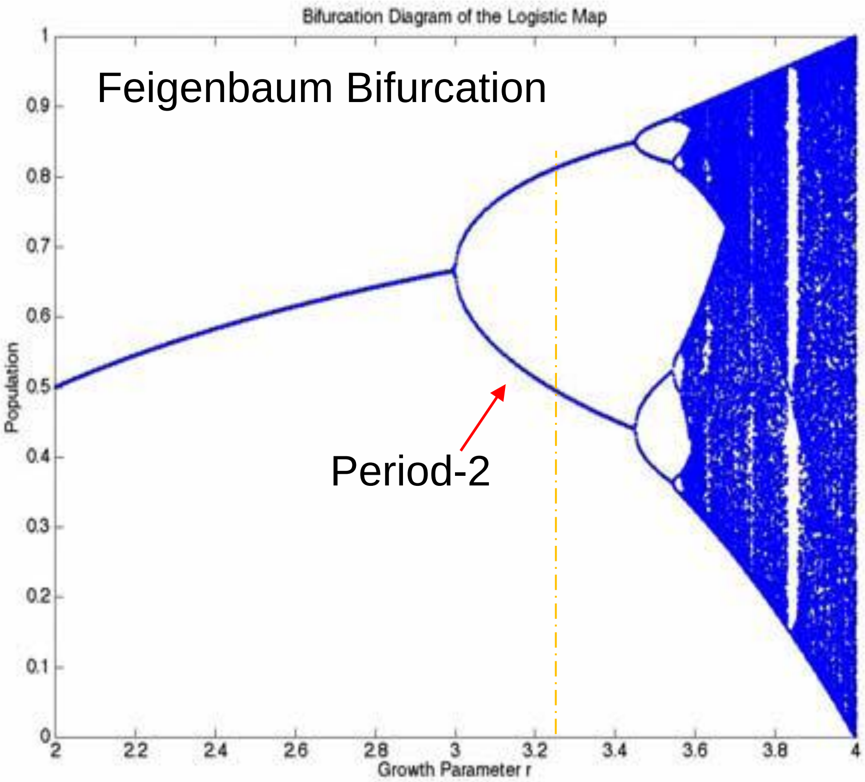
$$\boldsymbol{x(n) = f[x(n-1)] = x(n-1) = \left[1 - \frac{1}{r}\right]}$$

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

Logistic map



$r = 3.3$



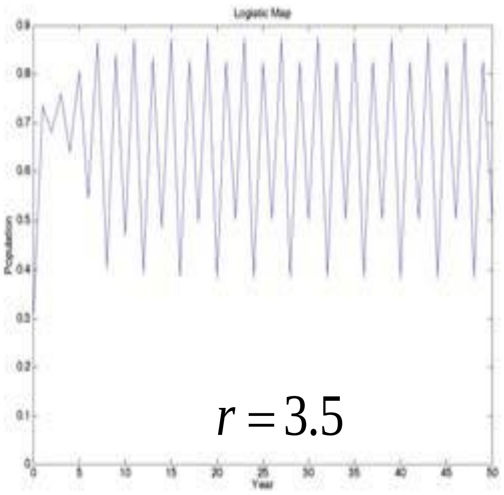
Feigenbaum Bifurcation

Period-2

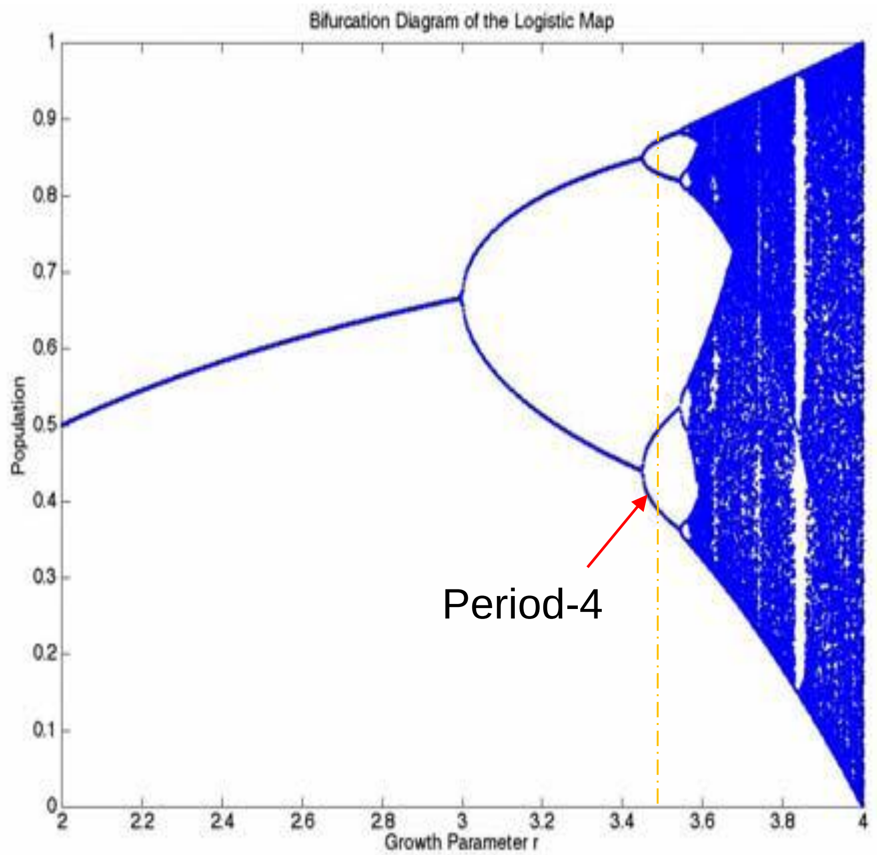
If initial condition is changed, the sequence always converge to the same cycle of period-2, but with a different rate

Bifurcations mark the transition from order into chaos

Logistic map



$r = 3.5$



Period-4

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

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

Logistic map

~ 3.569946 – period doubling region ends and chaos begins

3.828427 – small period tripling window opens up

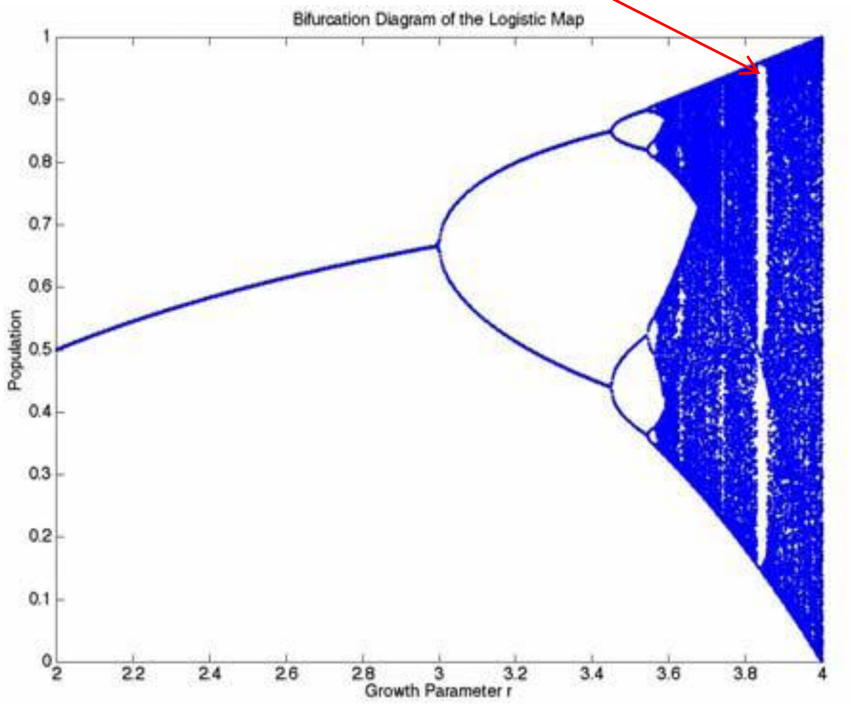
~ 3.855 – period tripling cascade ends and chaos resumes

~ 4.0 chaos reigns

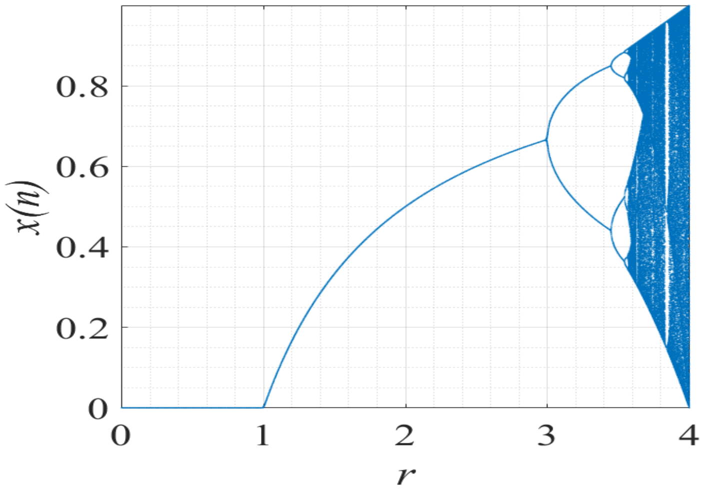
The sequence follows a geometric progression, but soon looks like a mess.

Messy regions are cyclically interspersed with clear “windows”.

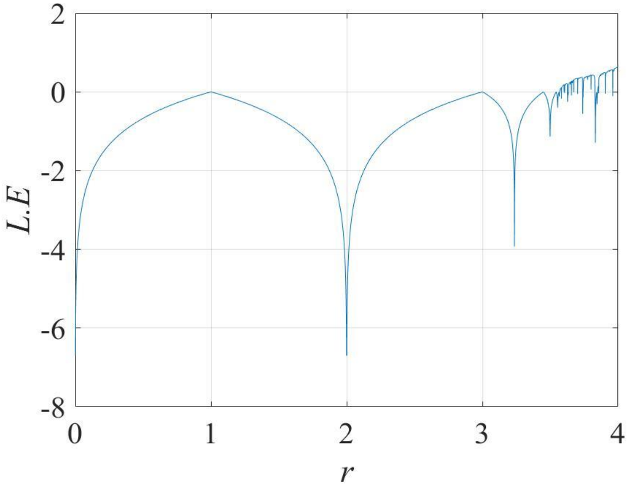
Existence of period-3 windows implies chaos



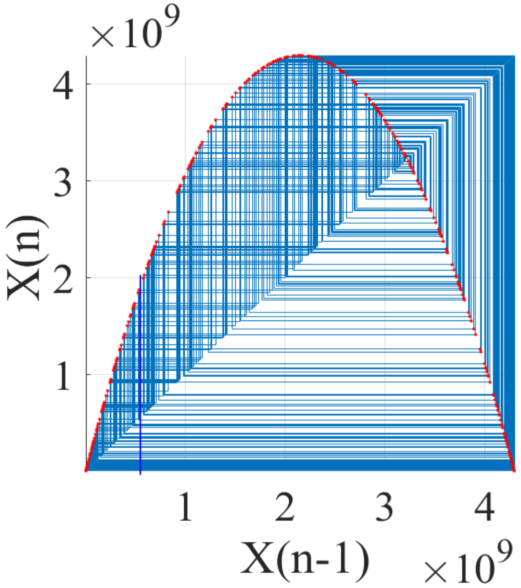
Chaos does not necessarily imply disorder
Chaos is the “randomness” in predicting the next iteration



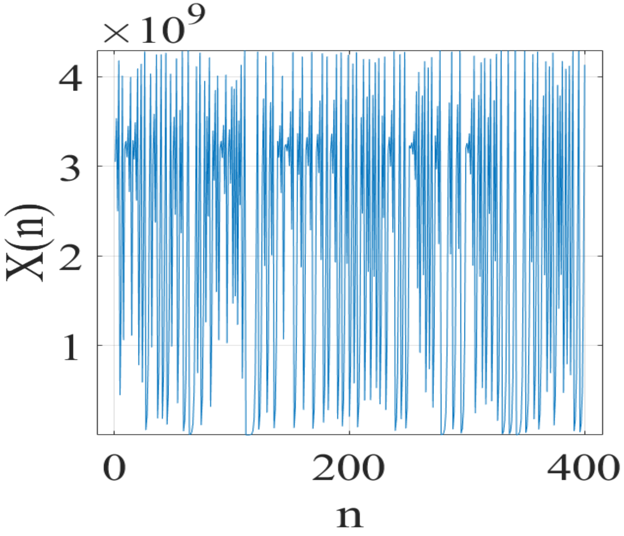
Bifurcation Diagram



Lyapunov Exponent



Strange Attractor: cobweb trajectory



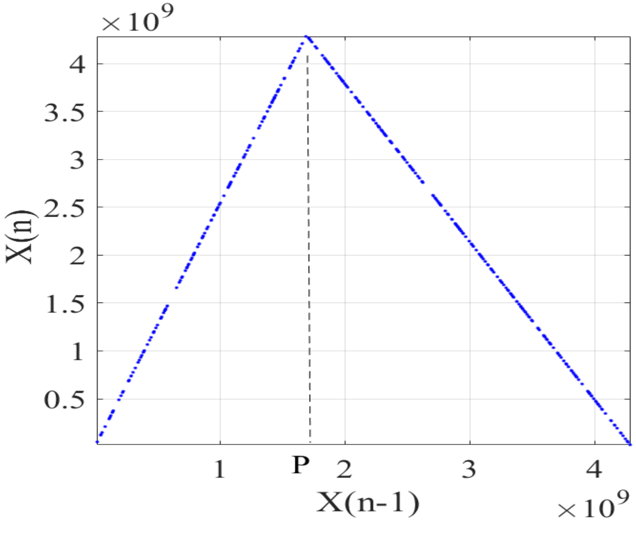
Discrete Variation

Logistic Map

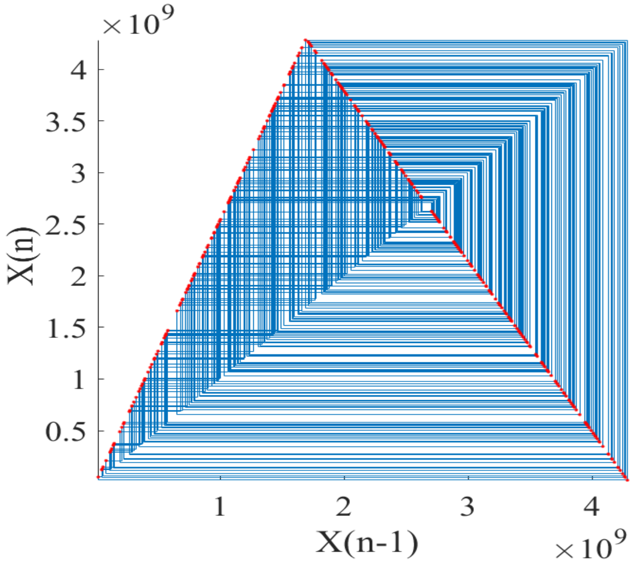
Discrete Skew Tent Map

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

$1 \leq X(n-1) \leq 2^N - 1,$ $1 \leq P \leq 2^N - 1$: Control parameter, $N = 32$ bits



Mapping



Attractor

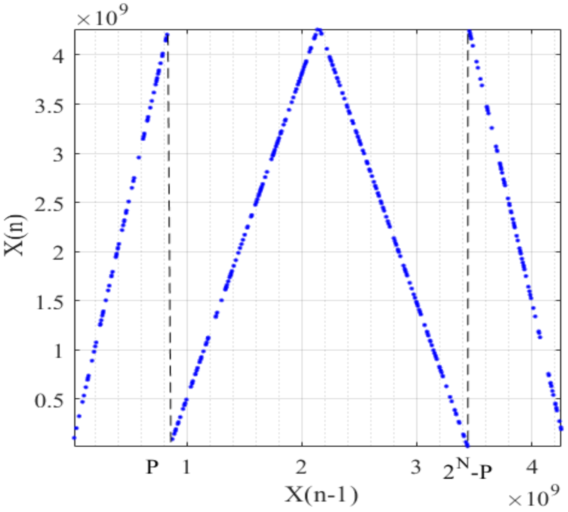
Better cryptographic performances than the Logistic map

Histogram is more uniform. Antagonist characteristics with the PWLCM

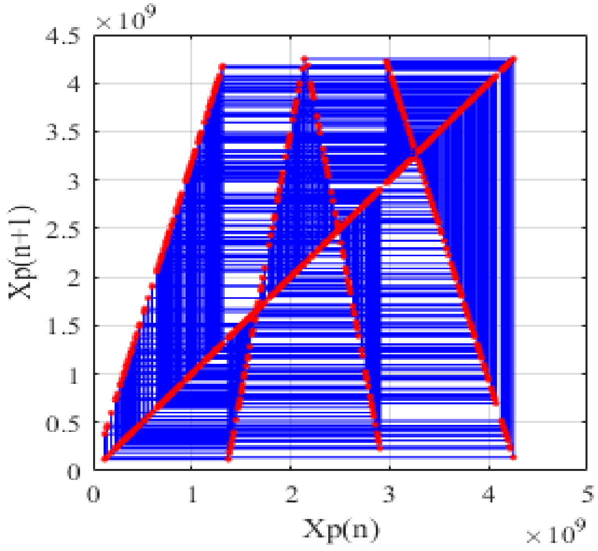
- **Discrete PWLCM Map**

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

$1 \leq X(n-1) \leq 2^N - 1,$ $1 \leq P \leq 2^{N-1} - 1$: Control parameter, $N = 32$ bits



Mapping

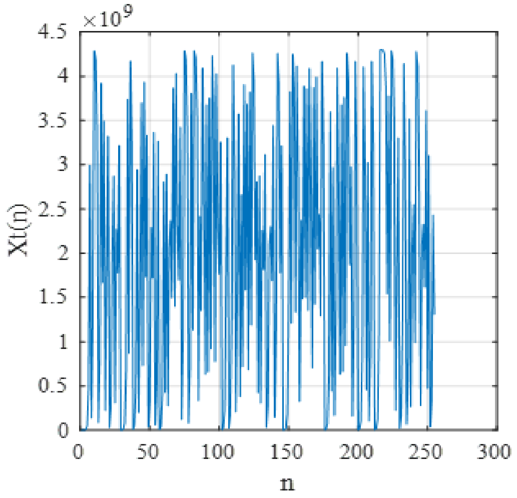


Attractor

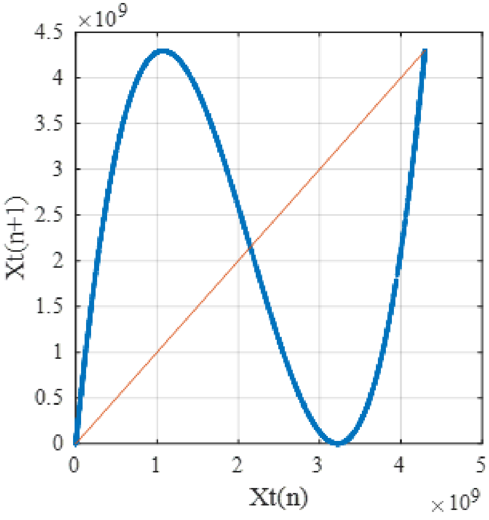
Discrete 3-D Chebyshev map

$$X(n) = \begin{cases} 2^{N-1} \qquad if \quad X(n) = 0 \quad or \quad 2^{N} \quad or \;\; 2^{N-1} \\ \left\lfloor 2^{-2N+2} \times \left\{ \begin{matrix} 4 \times \left[X(n-1) - 2^{N-1}\right]^{3} \\ -3 \times \; 2^{2N-2} \times \left[X(n-1) - 2^{N-1}\right] \end{matrix} \right\} + 2^{N-1}, otherwise \right\rfloor \end{cases}$$

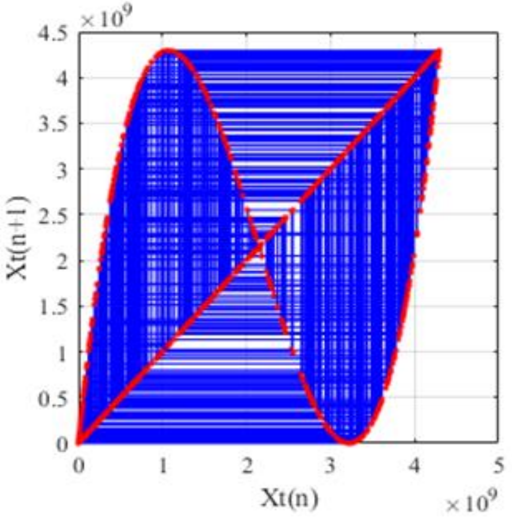
$1 \leq X(n-1) \leq 2^{N} - 1, \qquad N = 32$ bits



Discrete Variation



Mapping



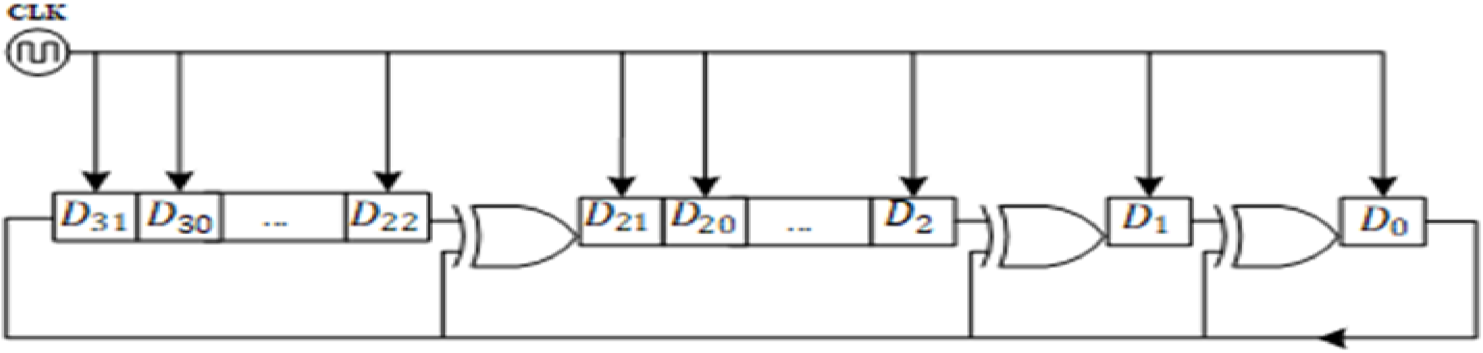
Attractor

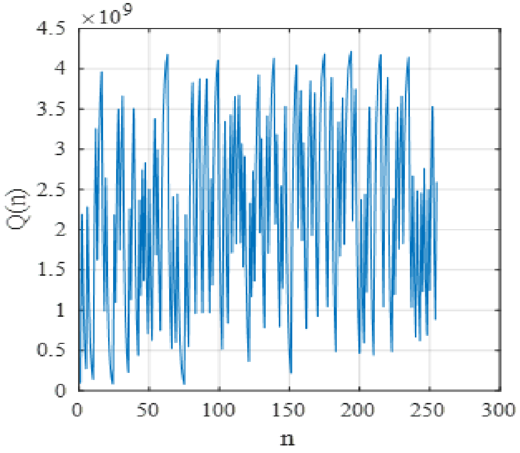
- ## Linear Feedback Shift Register (LSFR)

Primitive polynomial

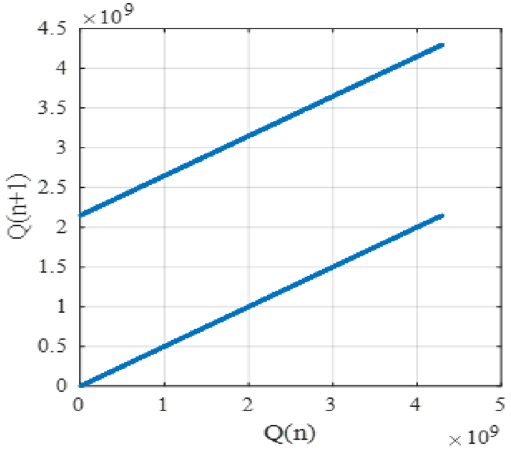
$$Q(n) = x^{32} + x^{22} + x^{2} + x + 1, \quad 1 \le Q(n) \le 2^{N} - 1, \qquad l = 2^{32} - 1$$

Galois structure

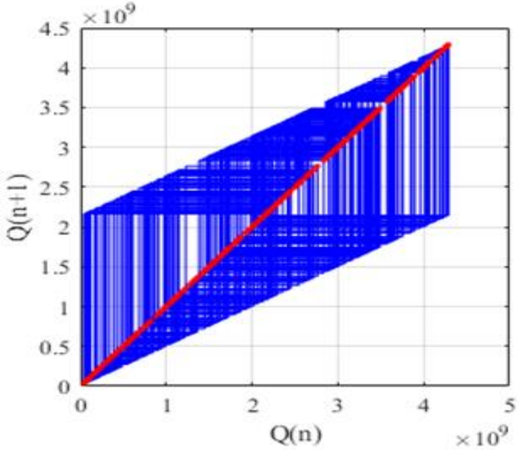




Discrete Variation

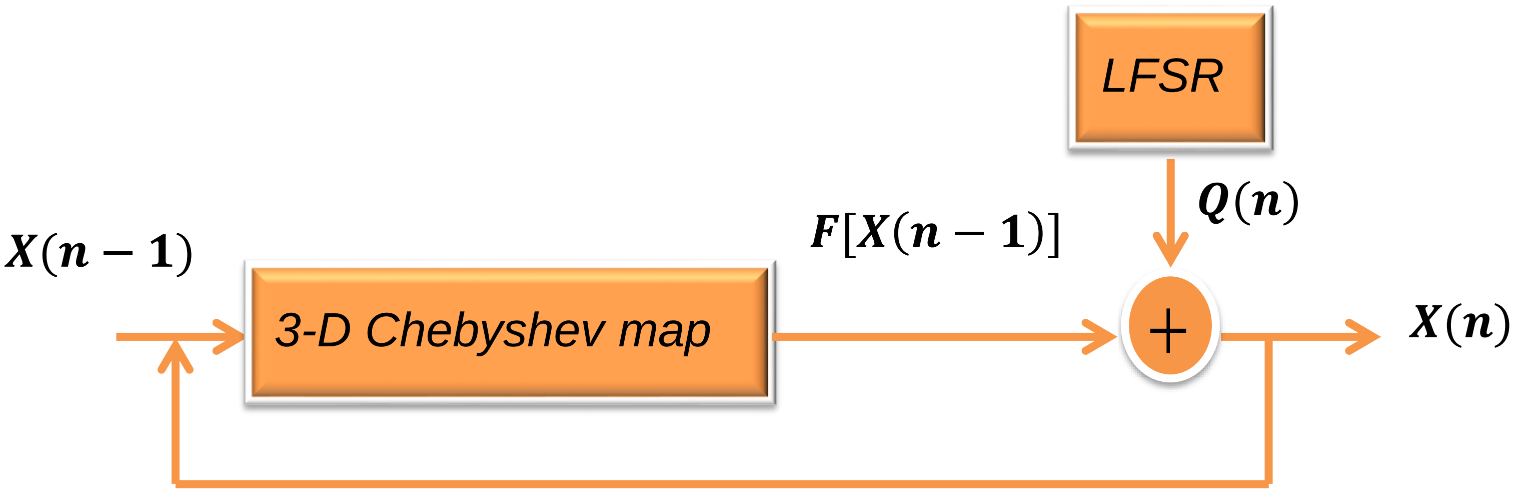


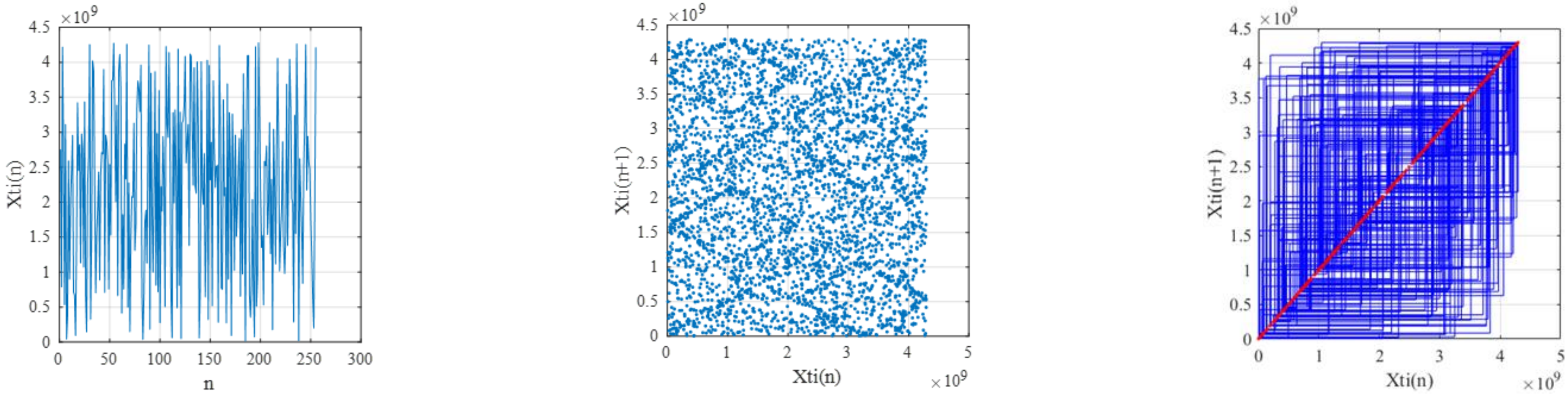
Mapping



Attractor

- **Discrete 3-D Chebyshev map coupled with an LFSR**





- ➢ **Random mapping vs known mapping of a Skew Tent, PWLCM, Logistic, and a 3-D Chebyshev map**

- ➢ **The technique of coupling a chaotic card with an LFSR improves the cryptographic properties of this chaotic map.**

- **Statistical analysis of chaotic maps: Uniformity and NIST test**

1) Uniformity test: Histogram and chi-square χ^2 test

- **Visually uniform histogram**
- **Chi-squared distribution** $\Leftrightarrow \chi^2_{ex} < \chi^2_{th}(N_c - 1, \alpha)$

$$\chi^2_{ex} = \sum_{i=0}^{Nc-1} \frac{(O_i - E_i)^2}{E_i}$$

N_c is the number of classes (sub-intervals) or degrees of freedom, chosen here $N_c = 1000$

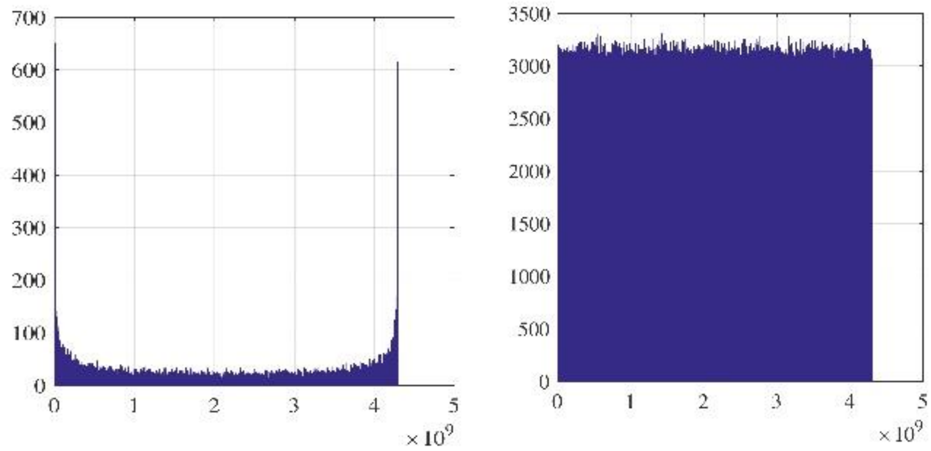
O_i is the number of observed (calculated) samples in the *i-th* class

E_i is the expected number of samples in a uniform distribution, $E_i = N_s / N_c$

N_s is the number of generated samples of 32 bits each, chosen here $N_s = 3{,}125{,}000 = 10^8\ bits$

α is the significance level or probability level, chosen here $\alpha = 0.05$

Uniformity test for the Logistic map and the 3-D Chebyshev map with LFSR



	Logistic map	3-D Chebyshev map with LFSR
χ^2_{ex}	38,698 KO	999.48 OK
χ^2_{th}	1073.64	1073.64

2) NIST test

For all experiments, we generate 100 different sequences of 3,125,100 32-bit samples using 100 random secret keys. But we only used 3,125,000 samples per sequence (i.e. 10^8 bits). The first 100 samples generated are produced by the system internally but are not used (to deviate from transitional regime).

Nist test consists of a battery of **188** tests and sub-tests, globally **15** different tests, to conclude regarding the randomness or non-randomness of binary sequences.

Nist test uses as input a sequence S of $n = 10^6$ bits, then divides them to m binary sequences S_k, $k = 1, m$ (chosen here m=100).

For each test, a set of $m\ P\_values$ is produced (based on the standard normal or chi-square as references distributions).

	Frequency (test 1)	Block Frequency (test 2)	……..	Linear Complexity (test 15)
S_1	$P_{1,1}$	$P_{1,2}$	……..	$P_{1,15}$
S_2	$P_{2,1}$	$P_{2,2}$	……..	$P_{2,15}$
……..	……..	……..	……..	……..
S_m	$P_{m,1}$	$P_{m,2}$	……..	$P_{m,15}$

A sequence passes a test (the sequence appears to be random) whenever the $P\_values \geq \alpha$, where a is the level of significance of the test. The value of α is set for all the tests.

For a fixed α, a certain percentage of m P_values are expected to indicate failure. Indeed, an $\alpha = 0.01$, indicates that 1 % of the m sequences are expected to fail.

- A $P\_value \geq \alpha = 0.01$, would mean that the sequence would be random with a confidence of $(1 - \alpha) = 99\ \%$.
- A $P\_value < \alpha = 0.01$, would mean that the conclusion was that the sequence is non-random with a confidence of $(1 - \alpha) = 99\ \%$.

Remark:

- The minimum pass rate for each statistical test, with the exception of the 8 Random Excursion tests and the 18 Random Excursion Variant tests, is approximately = 0.960150.

- The minimum pass rate for the 8 Random Excursion tests and the 18 Random Excursion Variant tests is approximately 0.952091. These tests are applicable only to 62 sequences instead of 100 sequences.

Interpretation of empirical results:

- The distribution of *P_values* to check for uniformity
- The examination of the Proportion of sequences that pass a statistical test

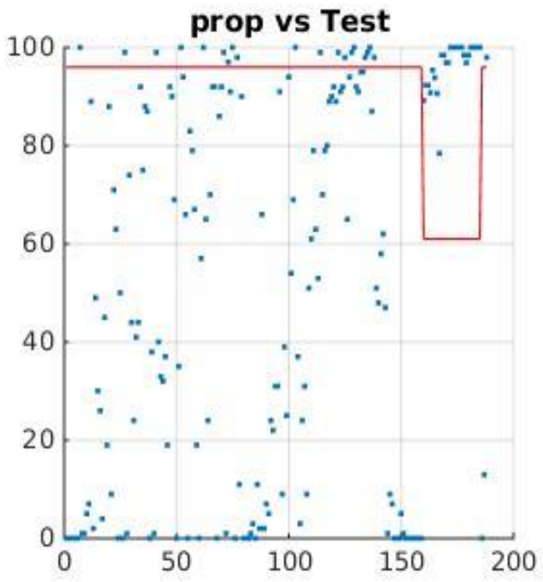
Final Analysis Report

```
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

 0  .1   .2   .3   .4   .5   .6   .7   .8  .9   1 <-----  P-values
-------------------------------------------------------------------------------
 C1   C2   C3   C4   C5   C6   C7   C8   C9  C10   P-VALUE_T  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------------
 F1   F2   F3   F4    F5   F6   F7   F8    F9   F10 <-----  Frequency of P-values

 13    8   12    6    10   10   10   10    10   11  0.946308   0.9600     frequency
  4    6   13   12    12   15   11   12     4   11  0.137282   0.9800     block-frequency
 15   10    8    8     9   13    8    9     8   12  0.779188   0.9800     cumulative-sums
  9   10    6   11     9   15    8    6     9   17  0.249284   0.9900     runs
 12    9   10   14     5    4   15    8     9   14  0.171867   0.9900     longest-run
 14   11    6   12     8    6   13    3    21    6  0.002758   1.0000     rank
 12   13   10   13    13    6   10    7     7    9  0.678686   0.9900     fft
  9    7    8   11    13    9    9   14    12    8  0.834308   0.9900     nonperiodic-templates (148)
 12   11   11   17    11    9    9    7     7    6  0.419021   0.9900     overlapping-templates
  4   17   10   12     9   10    4   12    13    9  0.122325   1.0000     universal
  8   10   10   10    10   12    9   14     8    9  0.964295   0.9900     approximate entropy
  3    2   11    6     7    8    8    4     8    5  0.253551   0.9677     random-excursions (8)
  6    2    7    8     6   11    6    8     5    3  0.350485   0.9677     random-excursions-variant (18)
  8   10   10    6    10   12   10   14     9   11  0.897763   0.9900     serial
 19    9    8    7     7   13    6    9     7   15  0.058984   0.9900     linear-complexity
```

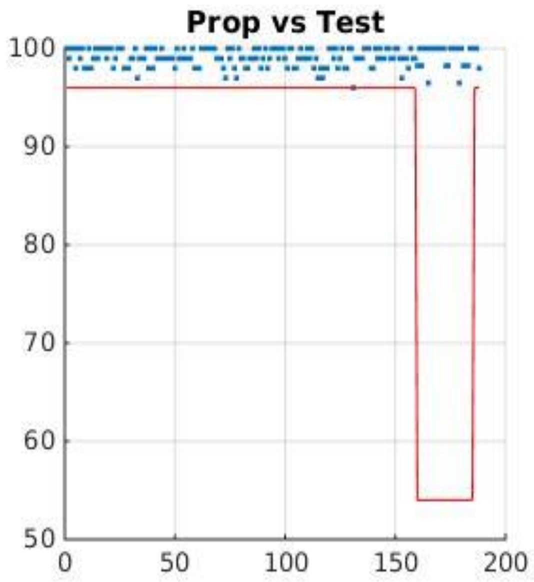
NIST test for the Logistic map and the 3-D Chebyshev map with LFSR

	Logistic		3-D Chebyshev	
NIST test	P-value	Prop %	P-value	Prop %
Frequency	0.000	92.000	0.249	100
Block-frequency	0.000	0.000	0.011	98
Cumulative-sums (2)	0.000	93.500	0.608	100
Runs	0.000	0.000	0.335	99
Longest-run	0.000	0.000	0.494	100
Rank	0.419	98.000	0.983	99
FFT	0.000	41.000	0.115	100
Non-periodic-templates (148)	0.036	61.304	0.524	99.020
Overlapping-templates	0.000	0.000	0.304	100
Universal	0.000	0.000	0.798	98
Approximate Entropy	0.000	0.000	0.213	100
Random-excursions (8)	0.002	90.367	0.284	99.256
Random-excursions-variant (18)	0.400	99.145	0.290	99.592
Serial (2)	0.000	0.500	0.259	99
Linear-complexity	0.081	97.000	0.514	100

Logistic

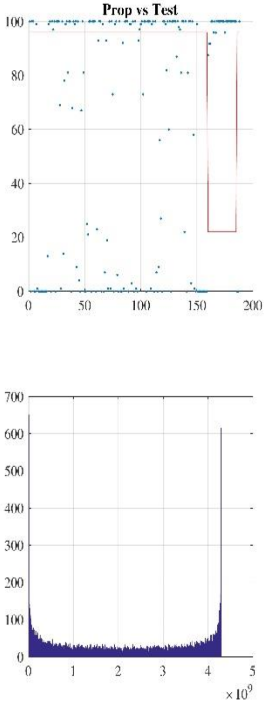


3-D Chebyshev



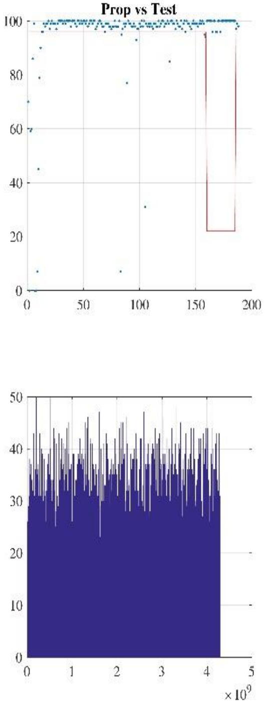
NIST test and uniformity test for studied chaotic maps

- Logistic



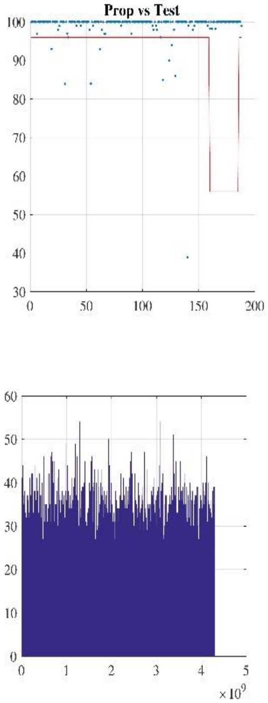
✗

- SkewTent



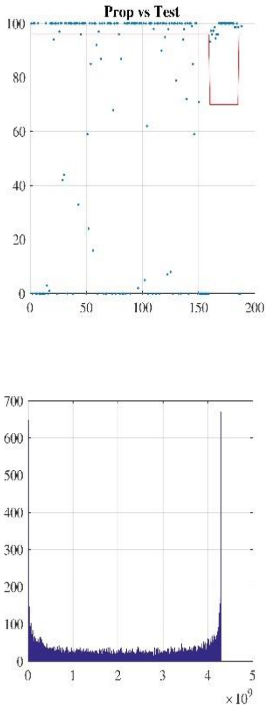
✗

- PWLCM



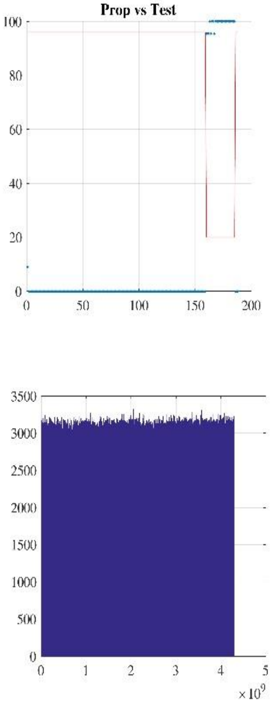
✗

- 3-D Chebyshev



✗

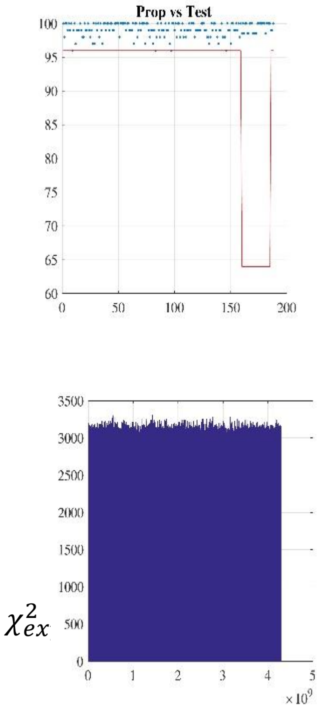
- LFSR



✓

- 3-D Chebyshev with LFSR

✓



✓

Computing performance

Computing by software: C language

Computer: Intel ® Core™ i5-4300M, CPU @ 2.6 GHz and memory 15.6 GB

Operating system: Ubuntu 14.04 Linux, using GNU GCC compiler

$$Bit\,rate(Mbit/s) = \frac{Generated\ data\ size\ (Mbits)}{Average\ generation\ time\ (second)}$$ **or Throughput (Mbps)**

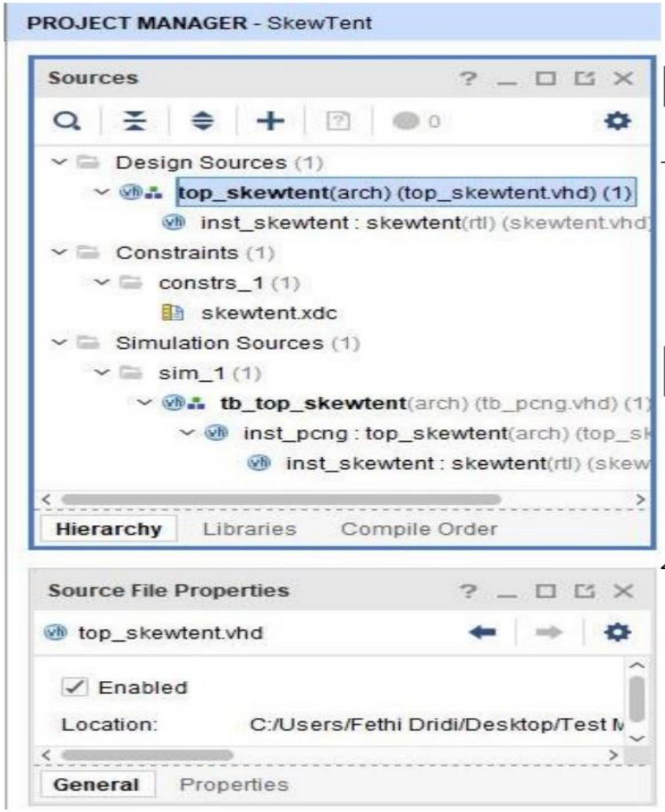
$$NCpB = \frac{CPU\ speed\ (Hz)}{Bit\,rate(Byte/s)}$$: **Number of needed Cycles to generate one Byte**

$NCpB$: permits to compare the computing performance of different systems working on different platforms

	Logistic	SkewTent	PWLCM
Generation time (μs)	317	422	514
Bit rate (Mbps)	3144	2368	1941
NCpB	3	8	10

○ Computing by hardware: VHDL description & FPGA

All studied chaotic systems were coded in VHDL using the Xilinx Vivado design suite (V.2017.2) and were implemented on a Xilinx XC7Z020 PYNQ-Z2 (7z020clg400-1) FPGA hardware platform.



PROJECT MANAGER - SkewTent

Sources

- Design Sources (1)
  - top_skewtent(arch) (top_skewtent.vhd) (1)
    - inst_skewtent : skewtent(rtl) (skewtent.vhd
- Constraints (1)
  - constrs_1 (1)
    - skewtent.xdc
- Simulation Sources (1)
  - sim_1 (1)
    - tb_top_skewtent(arch) (tb_pcng.vhd) (1
      - inst_pcng : top_skewtent(arch) (top_sk
        - inst_skewtent : skewtent(rtl) (skew

Hierarchy | Libraries | Compile Order

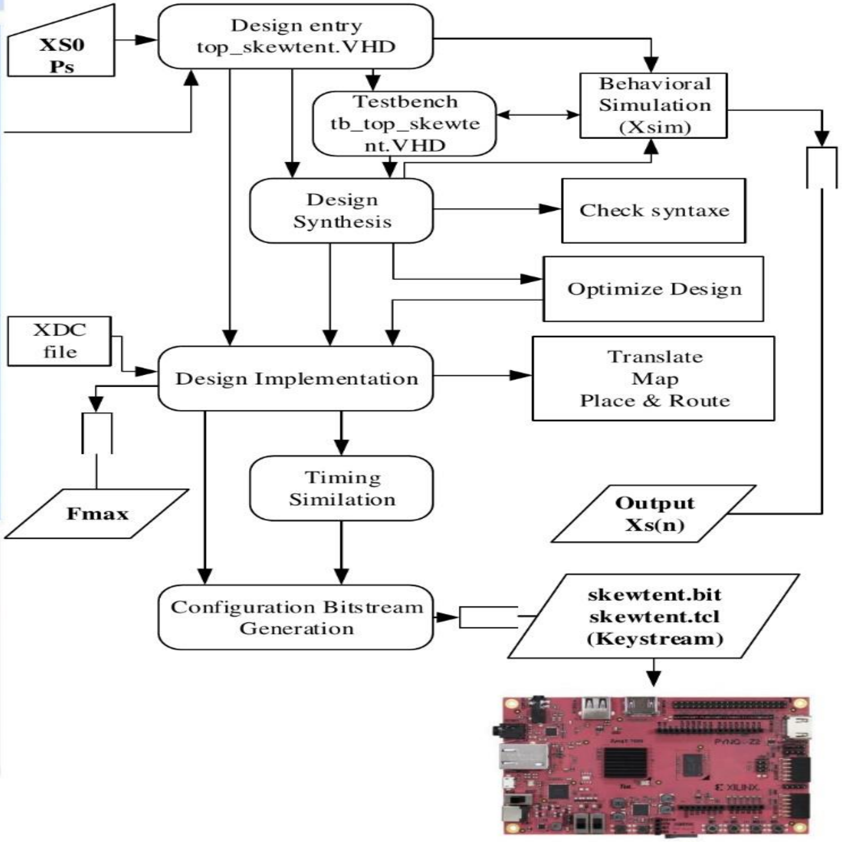
Source File Properties

top_skewtent.vhd

☑ Enabled

Location: C:/Users/Fethi Dridi/Desktop/Test M

General | Properties



XS0 Ps

Design entry top_skewtent.VHD

Testbench tb_top_skewtent.VHD

Behavioral Simulation (Xsim)

Design Synthesis

Check syntaxe

Optimize Design

XDC file

Design Implementation

Translate Map Place & Route

Fmax

Timing Similation

Output Xs(n)

Configuration Bitstream Generation

skewtent.bit skewtent.tcl (Keystream)

The programmable logic of the ZYNQ XC7Z020 provides 13,300 logic slices, each with four 6-input LUTs and 8 flip-flops, 630 KB block RAM, 220 DSP slices, and on-chip Xilinx analog-to-digital converter (XADC). It also has an external 125 MHz reference clock (PL CLK): $PL - F = 125\ MHz,\ PL - T = 8\ ns$.

- **Hardware metrics**

Maximum Frequency (MHz)

$$\boldsymbol{Max\_Freq = \frac{1}{Ti - WNSi}\ (MHz)}$$

Throughput (Mbps)

$$\boldsymbol{Throughput = N \times Max\_Freq\ (Mbps)}$$

Efficiency (Mbps/Slices)

$$\boldsymbol{Efficiency = \frac{Throughput}{Slices}\ (Mbps/Slices)}$$

Ti is the target clock period (ns) used during the implementation run ”i” and $WNSi$ is the Worst Negative Slack (ns) of the target clock used during the implementation run ”i” and must be positive, very close to zero.

$WNSi$ is the difference between the target clock period and the path delay between a pair of registers. The longest path delay $\tau i = Ti - WNSi$ determines the maximum frequency at which the design can operate.

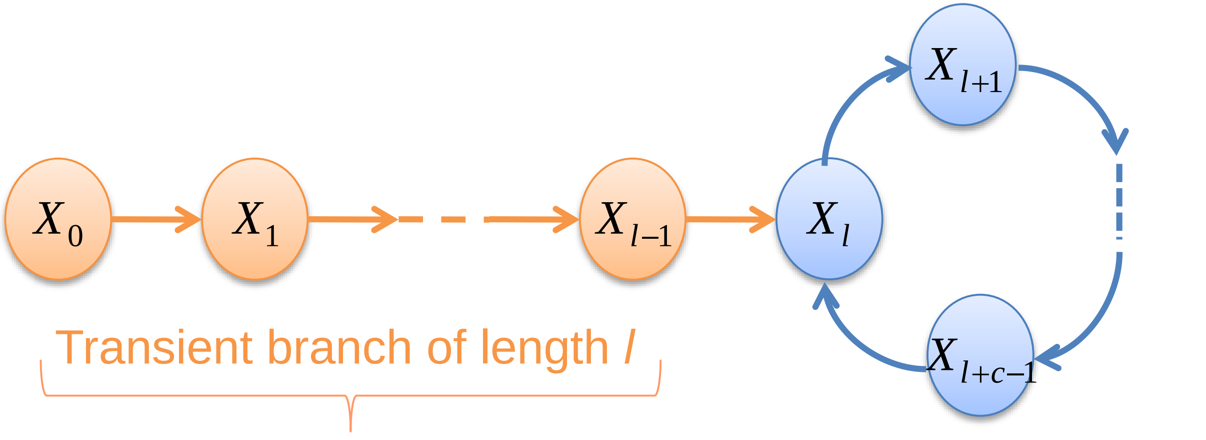
○ Hardware metrics

			Chaotic maps					
			Logistic	SkewTent	PWLCM	3-D Chebyshev	LFSR	3-D Ch with LFSR
Resources used	Area	LUTs	77 /0.14 %	2,830 /5.32 %	7,374 /13.86 %	268 /0.05 %	2 / 0.01 %	315 /0.59 %
		FFs	49 /0.05 %	57 /0.05 %	63 /0.06 %	47 /0.04 %	62 /0.06 %	78 /0.07 %
		Slices	33 /0.25 %	853 /6.41 %	2,171 /16.32 %	73 /0.55 %	19 /0.14 %	98 /0.74 %
	DSPs		4 /1.82 %	0 /0.00 %	0 /0.00 %	12 /5.45 %	0 /0.00 %	12 /5.45 %
Speed	WNSi (ns)		0.102	0.059	0.287	0.581	5.965	0.278
	Ti (ns)		12	28	32	24	8	22.80
	Max_Freq (MHz)		84.04	35.78	31.53	42.70	491.40	44.41
	Throughput (Mbps)		2,689.52	1,145.27	1,009.04	1,366.41	15,724.81	1,421.40
Efficiency (Mbps/Slices)			81.500	1.342	0.464	18.717	827.621	14.504
Power Consumption (W)			0.083	0.070	0.102	0.048	0.118	0.052

Effects of the finite precision N

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

$$\boldsymbol{X(n) = F[X(n-1)], \quad X(n) \in [1, 2^N - 1], \; n = 1, 2, \cdots \qquad Notation\colon X_n = X(n)}$$



Transient branch of length *l*

Cycle of period *c*

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

Pseudo-orbit of an integer chaotic values

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

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

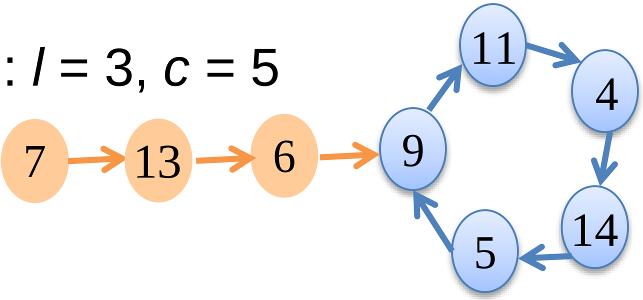
Were d is the number of delays of the recursive structure, if exist

Effects of the finite precision N

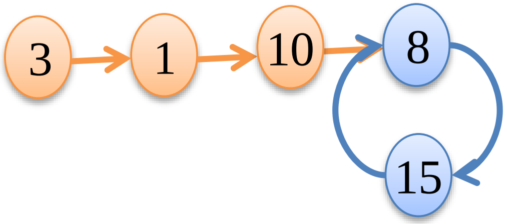
Example : *N* = 4 bits, and two I.Cs

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

1° I.C : *l* = 3, *c* = 5



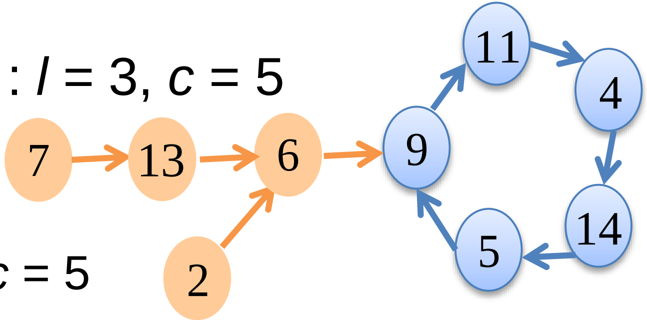
2° I.C : *l* = 3, *c* = 2



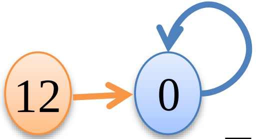
- Situation b : 2 different I.Cs give the same cycle *c*

1° I.C : *l* = 3, *c* = 5

2° I.C : *l* = 2, *c* = 5



- Situation c *l* = 1, *c* = 1



Fixed point

- **Observation: for classic 1-D chaotic maps used alone**
  - **Advantages:**
  - Simple equation
  - Easy implementation
  - Good computing performance
  - Disadvantages:
  - Weakness on security
  - Short size of the secret key
  - Short periodic orbits
  - Easily recognized functions