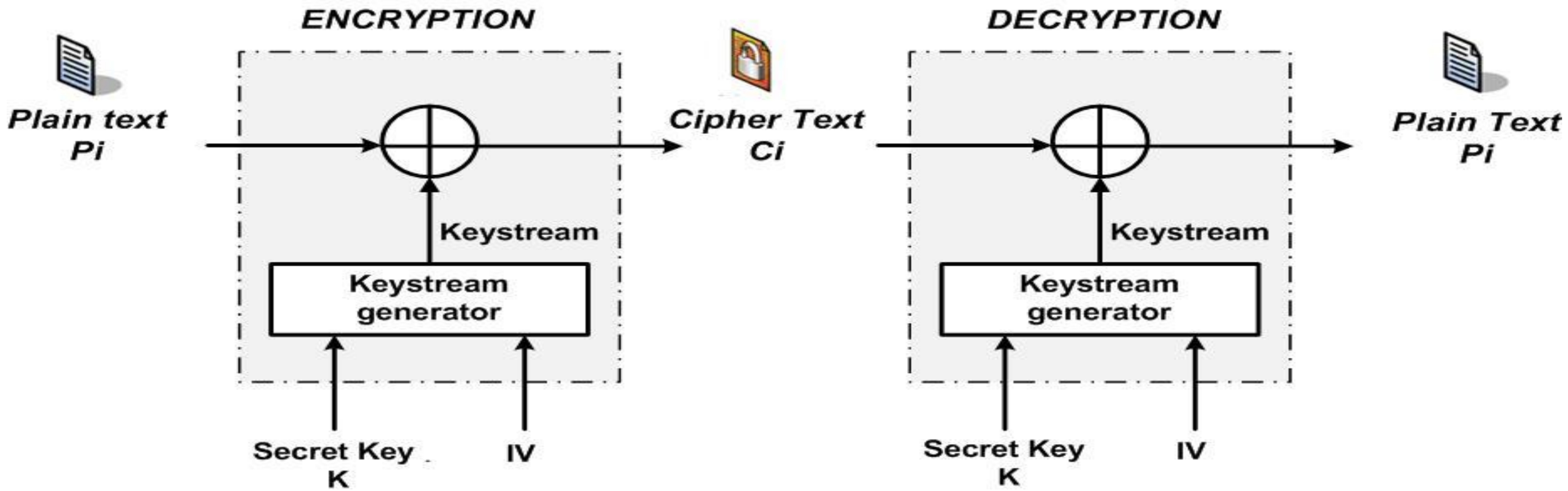
Design of efficient pseudo random number generators of chaotic sequences (PRNGs-CS) & performance evaluation

- **General scheme of a stream cipher**
- **General structure of the proposed secure PRNGs-CS**
- **Architecture description of the proposed PRNGs-CS**
  - Ultra-weak coupling technique & chaotic mixing (Lozi, 2007 & 2012)
  - Perturbation technique (Tao, 2005, El Assad 2008)
  - Recursive structure & orbits multiplexing (El Assad et. al., 2008 & 2011)
  - Cascading technique (Li et. al., 2001)
- **Hardware implementation and security analysis of the proposed PRNGs-CS**
- **Performance analysis of stream ciphers based on the proposed**
- **PRNGs-CS**

General scheme of a stream cipher



Pi	*Xi*	*Ci*
0	0	0
0	1	1
1	0	1
1	1	0

Encryption: $Ci = Pi \oplus Xi$ **Decryption:** $Pi = Ci \oplus Xi$

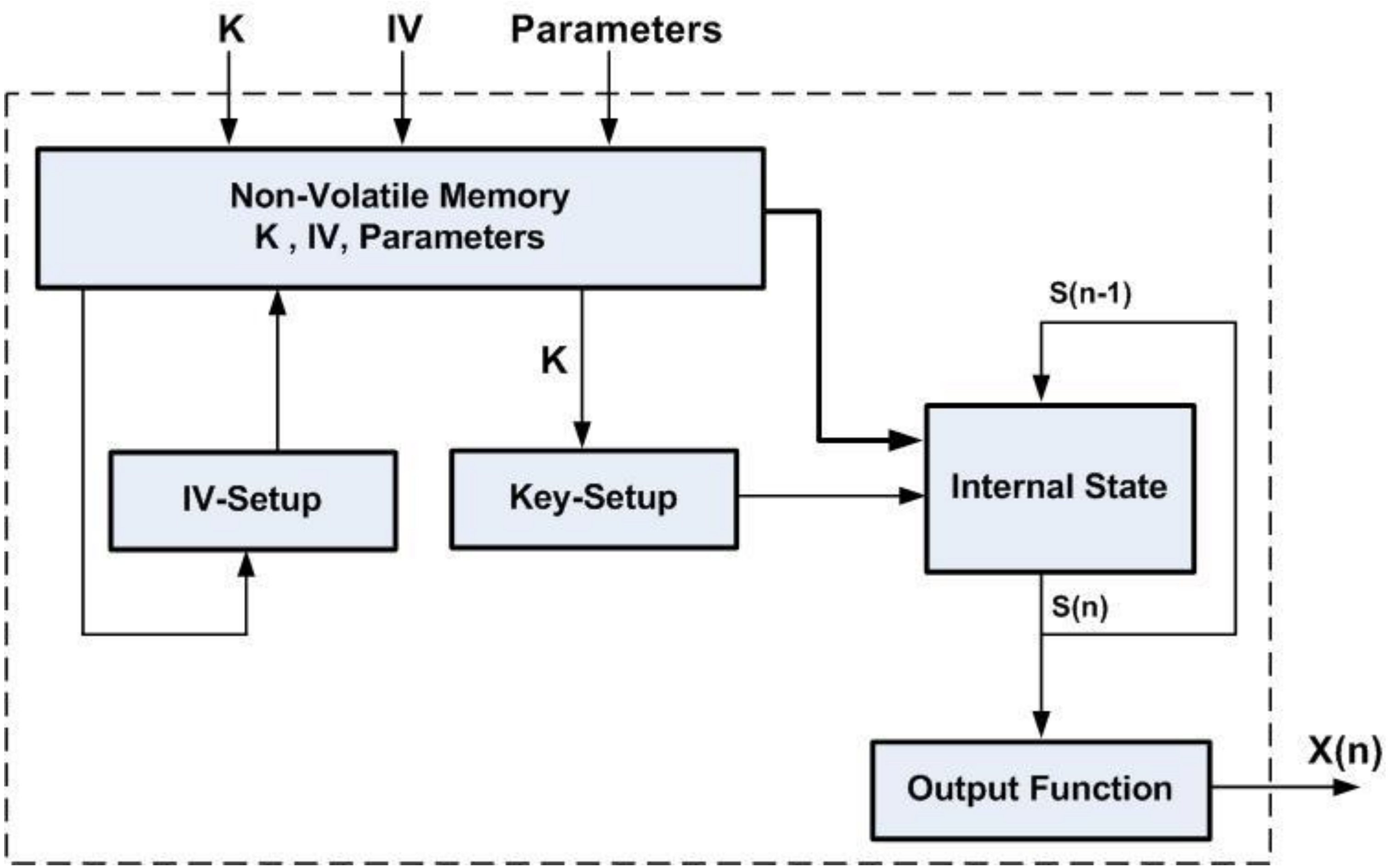
Encrypt $Pi = 0$*, depending on the keystream bit* $Xi = \begin{cases} 0 \\ 1 \end{cases}$ *gives* $Ci = \begin{cases} 0 \\ 1 \end{cases}$

If the keystream bit Xi is perfectly random, i.e., it is unpredictable and has exactly 50% chance to have the value 0 or 1, then both Ci also occur with a 50% likelihood. Likewise when we encrypt $Pi = 1$:

Encrypt $Pi = 1$*, depending on the keystream bit* $Xi = \begin{cases} 0 \\ 1 \end{cases}$ *gives* $Ci = \begin{cases} 1 \\ 0 \end{cases}$

The security of a stream cipher completely depends on the Keystream generator

General structure of the proposed secure PRNGs-CS

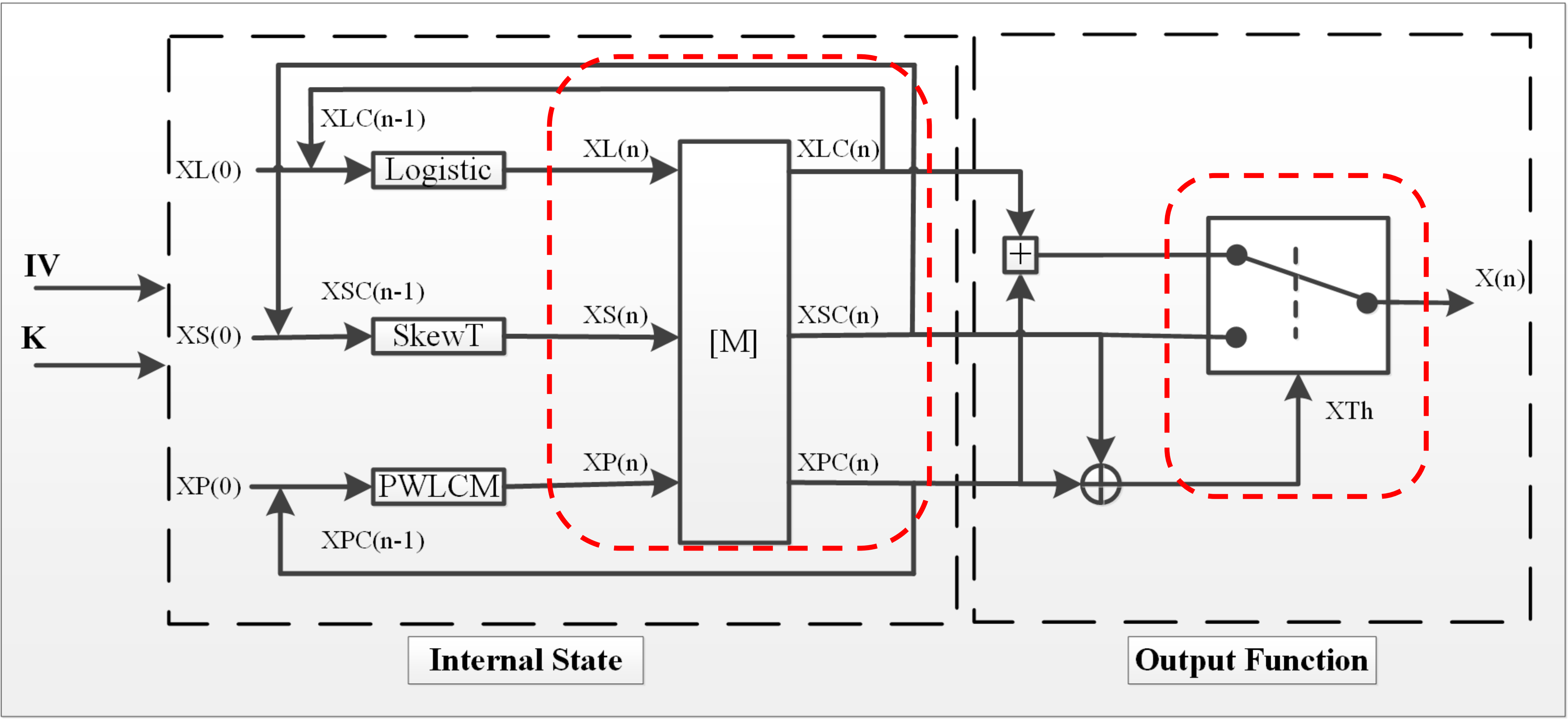


Keystream generator with internal feedback mode

The cryptographic complexity is in the internal state

LSP-PRNG: Ultra-weak coupling technique and chaotic mixing

PhD thesis: Ons Jallouli 2017, Fethi Dridi 2022



$$\begin{bmatrix} XLC(n) \\ XSC(n) \\ XPC(n) \end{bmatrix} = M \times \begin{bmatrix} XL(n) \\ XC(n) \\ XP(n) \end{bmatrix} \qquad M = \begin{bmatrix} \left(2^N - \varepsilon_{12} - \varepsilon_{13}\right) & \varepsilon_{12} & \varepsilon_{13} \\ \varepsilon_{21} & \left(2^N - \varepsilon_{21} - \varepsilon_{23}\right) & \varepsilon_{23} \\ \varepsilon_{31} & \varepsilon_{32} & \left(2^N - \varepsilon_{31} - \varepsilon_{32}\right) \end{bmatrix}$$

$$\varepsilon_{ij} \in \left[1, 2^k\right], \qquad k \leq 5$$

- **LSP-PRNG**

The initial conditions and parameters of the chaotic maps and the matrix M are supplied from the secret key $|K| = 189\ bits$ as follows:

$$XL0 = K(0\ to\ 31)$$
$$XS0 = K(32\ to\ 63)$$
$$XP0 = K(64\ to\ 95)$$
$$Ps = K(96\ to\ 127)$$
$$Pp = K(128\ to\ 158)$$
$$\varepsilon_{12} = K(159\ to\ 163)$$
$$\varepsilon_{13} = K(164\ to\ 168)$$
$$\varepsilon_{21} = K(169\ to\ 173)$$
$$\varepsilon_{23} = K(174\ to\ 178)$$
$$\varepsilon_{31} = K(179\ to\ 183)$$
$$\varepsilon_{32} = K(184\ to\ 188)$$

The initial vector IV supplies IVL, IVS, IVP as follows:

$$|IV| = 86\ bits, \text{ where: } \begin{cases} IVL = IV(0\ to\ 31) \\ IVS = IV(32\ to\ 63) \\ IVP = IV(64\ to\ 95) \end{cases}$$

The initial values $XL(0)$, $XS(0)$, and $XP(0)$ of the three chaotic maps are calculated as follows:

$$XL(0) = mod\ [(XL0 + IVin), 2^N]$$
$$XS(0) = mod\ [(XS0 + IVin), 2^N]$$
$$XP(0) = mod\ [(XP0 + IVin), 2^N]$$

With:

$$IVin = IVL \oplus IVS \oplus IVP$$

- **LSP-PRNG: first output** $X(1)$

$$XL(1) = Logistic\{mod[XL(0), 2^N]\}$$
$$XS(1) = SkewT\{mod[XS(0), 2^N], P_s\}$$
$$XP(1) = PWLCM\{mod[XP(0), 2^N], P_p\}$$

Chaotic maps coupling

$$\begin{bmatrix} XLC(1) \\ XSC(1) \\ XPC(1) \end{bmatrix} = M \times \begin{bmatrix} XL(1) \\ XS(1) \\ XP(1) \end{bmatrix}$$

$$X_{th}(1) = XPC(1) \oplus XSC(1)$$

If $X_{th} < T$ then
$X(1) = mod\{[XPC(1) + XLC(1)], 2^N\}$
else
$X(1) = XSC(1)$
end

$Threshod\text{: } T = 0.8 \times 2^N$

- **LSP-PRNG output** $X(n)\text{: } 2 \leq n \leq Ns$

$$XL(n) = Logistic\{mod[XLC(n-1), 2^N]\}$$
$$XS(n) = SkewT\{mod[XSC(n-1), 2^N], P_s\}$$
$$XP(n) = PWLCM\{mod[XPC(n-1), 2^N], P_p\}$$

Chaotic maps coupling

$$\begin{bmatrix} XLC(n) \\ XSC(n) \\ XPC(n) \end{bmatrix} = M \times \begin{bmatrix} XL(n) \\ XS(n) \\ XP(n) \end{bmatrix}$$

$$X_{th}(n) = XPC(n) \oplus XSC(n)$$

If $X_{th} < T$ then
$X(n) = mod\{[XPC(n) + XLC(n)], 2^N\}$
else
$X(n) = XSC(n)$
end

Security analysis of PRNGs-CS

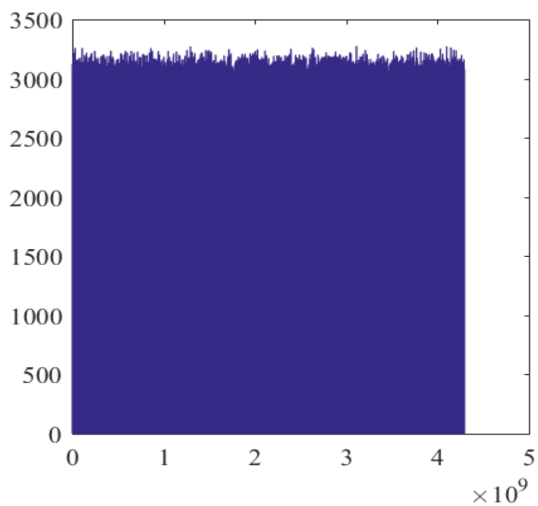
- **Statistical analysis:**
  - **NIST test**
  - **Uniformity test (histogram and chi-square)**
- **Key sensitivity analysis (Hamming distance)**
- **Key space**

Statistical analysis of the LSP-PRNG: Uniformity, NIST test & Key sensitivity

Uniformity test:

- Visually uniform histogram
- Chi-squared distribution

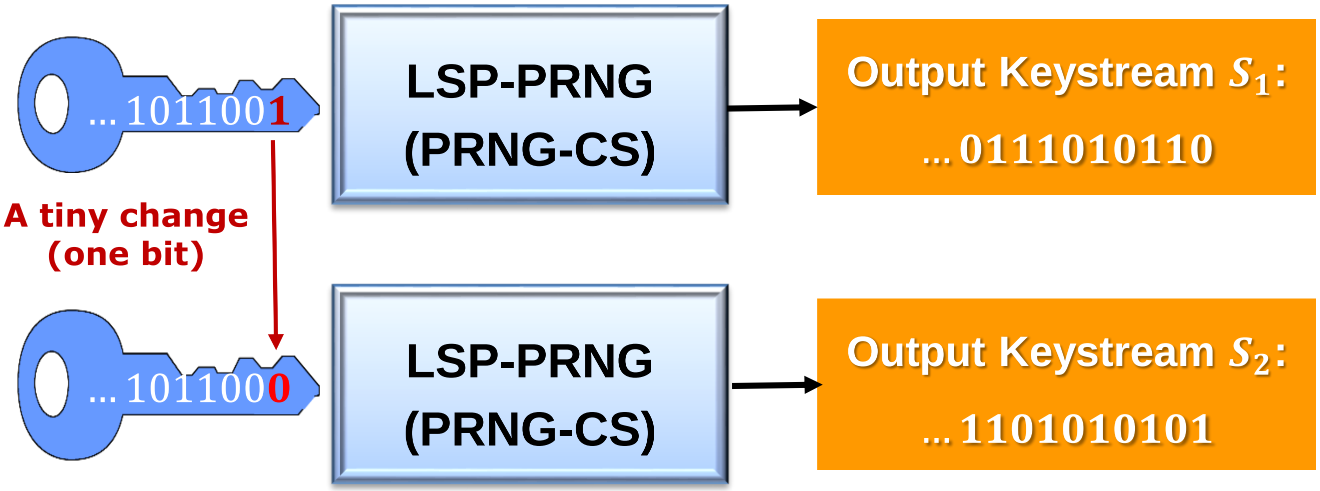
$\Leftrightarrow \chi^2_{ex} < \chi^2_{th}(N_c - 1, \alpha)$

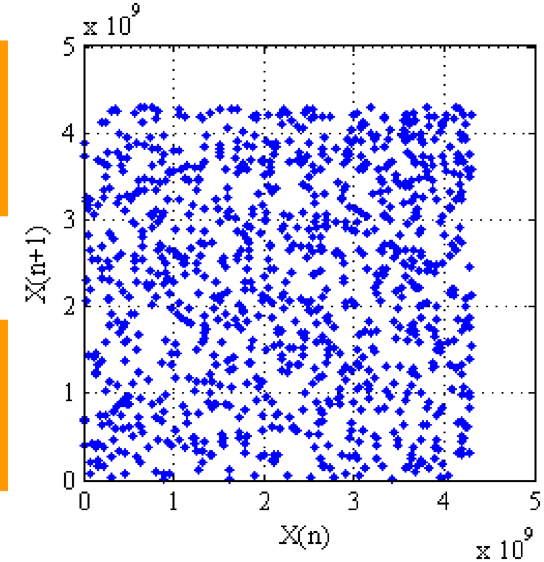


	LSP-PRNG
χ^2_{ex}	941.58 OK
χ^2_{th}	1073.64 For $\alpha = \mathbf{0.05}$ and $N_c = 1000$, $N_s = 10^8$ bits

	LSP-PRNG	
NIST test	P-value	Prop %
Frequency	0.494	100
Block-frequency	0.760	99
Cumulative-sums (2)	0.757	100
Runs	0.367	100
Longest-run	0.983	99
Rank	0.720	98
FFT	0.575	98
Non-periodic-templates (148)	0.527	99.115
Overlapping-templates	0.596	99
Universal	0.335	98
Approximate Entropy	0.475	99
Random-excursions (8)	0.352	99.792
Random-excursions-variant (18)	0.468	98.611
Serial (2)	0.460	99
Linear-complexity	0.401	99

Key sensitivity, mapping and auto & cross correlation





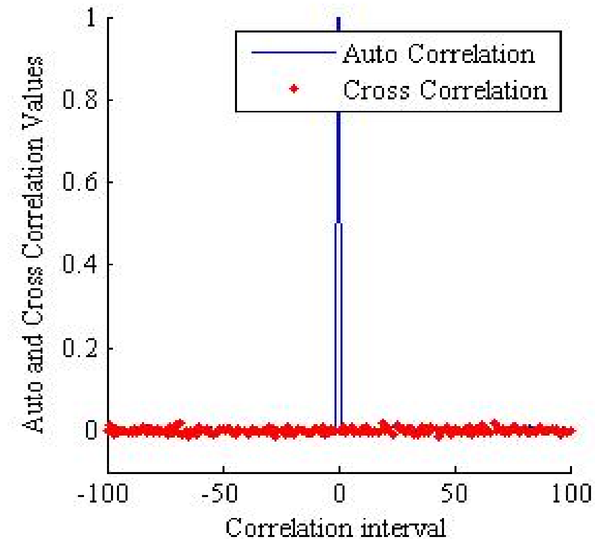
Hamming distance:

$$HD(S_1, S_2) = \frac{1}{Nb} \sum_{i=1}^{Nb} \left(S_1(i) \oplus S_2(i) \right)$$

Nb is the number of bits in the sequence

PRNG-CS	LSP-PRNG
Key space $(> 2^{128})$	2^{189}
Average $H_D(S_1, S_2)$ (optimal value: 50%)	50.0022

It is computationally infeasible to retrieve the secret key from the generated sequences



Average over 100 secret keys

- **Computing performance of the LSP stream cipher**

  - **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

  - **Computing by Hardware:**

VHDL using Vivado design (V.2017.2)
PYNQ-Z2 FPGA

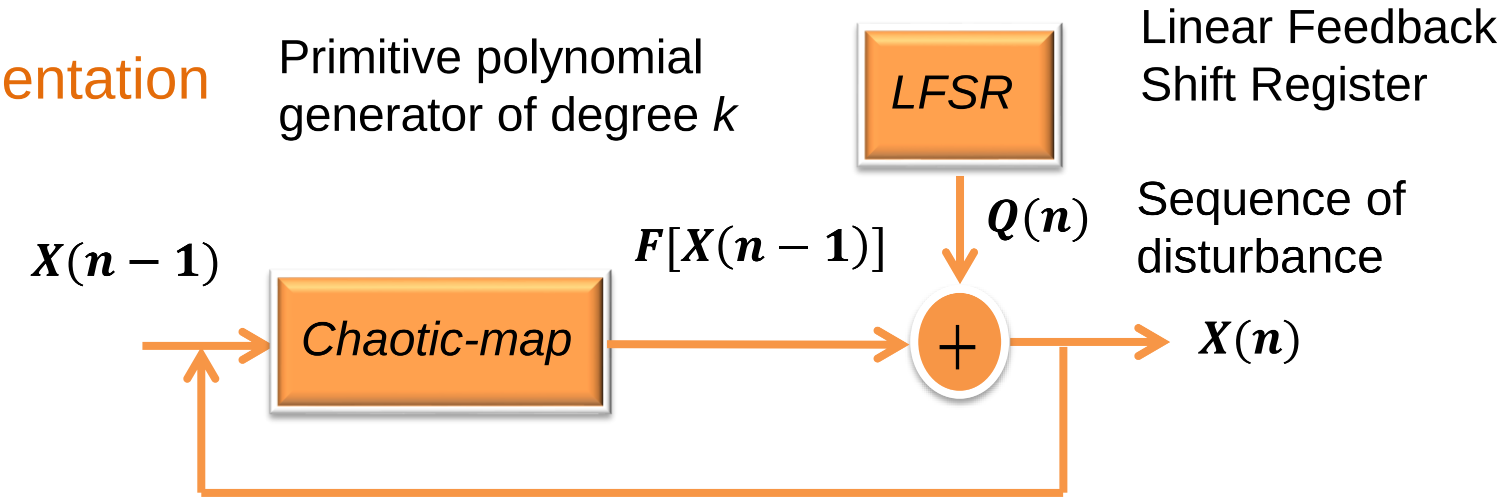
Image size (Bytes): 512 x 512 x 3 = 786,432 Bytes

LSP stream cipher	
Generation time (μs)	8511
Throughput (Mbps)	739.21
NCpB	28.14

LSP-PRNG			
Resources used	Area	LUTs	10,420 /19.59 %
		FFs	448 /0.42 %
		Slices	3,160 /23.71 %
	DSPs		13 /5.91 %
Speed	Max. Freq. (MHz)		32.41
	Throughput (Mbps)		1,037.27
Efficiency (Mbps/Slices)			0.32
Power (W)			0.146

Perturbation Technique

Implementation



$$X(n) = x_{N-1}(n)x_{N-2}(n)\cdots x_i(n)\cdots x_1(n)x_0(n) \qquad x_i(n) \in A_b = [0,1]$$

Perturbation every Δ iterations Δ : Average orbit of the chaotic-map without perturbation

$$If \quad n = l \times \Delta \qquad l = 1, 2, \cdots$$

$$x_i(n) = \begin{cases} F[x_i(n-1)] & k \le i \le N-1 \\ F[x_i(n-1)] \oplus q_i(n) & 0 \le i \le k-1 \end{cases}$$

$$Else$$

$$No\ perturbation: X(n) = F[X(n-1)]$$

Lower length of the orbit: $o_{min} = \Delta \times \left(2^k - 1\right)$

- Disturbance characteristics:
  - Controllable long cycle length and uniform distribution
  - Not degrade the good statistical properties of chaos dynamics

$$SNR = 20 \times \log_{10}\left[\frac{2^N\text{: maximum chaotic signal amplitude}}{2^k\text{: maximum disturbance signal amplitude}}\right] \geq 40\ dB$$

$$\log_{10}[2^{N-k}] \geq 2\ dB\ \rightarrow 2^{N-k} \geq 10^2 \rightarrow N-k \geq \log_2(10^2)\ \rightarrow N-k \geq 7\ \rightarrow k \leq N-7$$

With $N = 32\ \rightarrow k \leq 23$

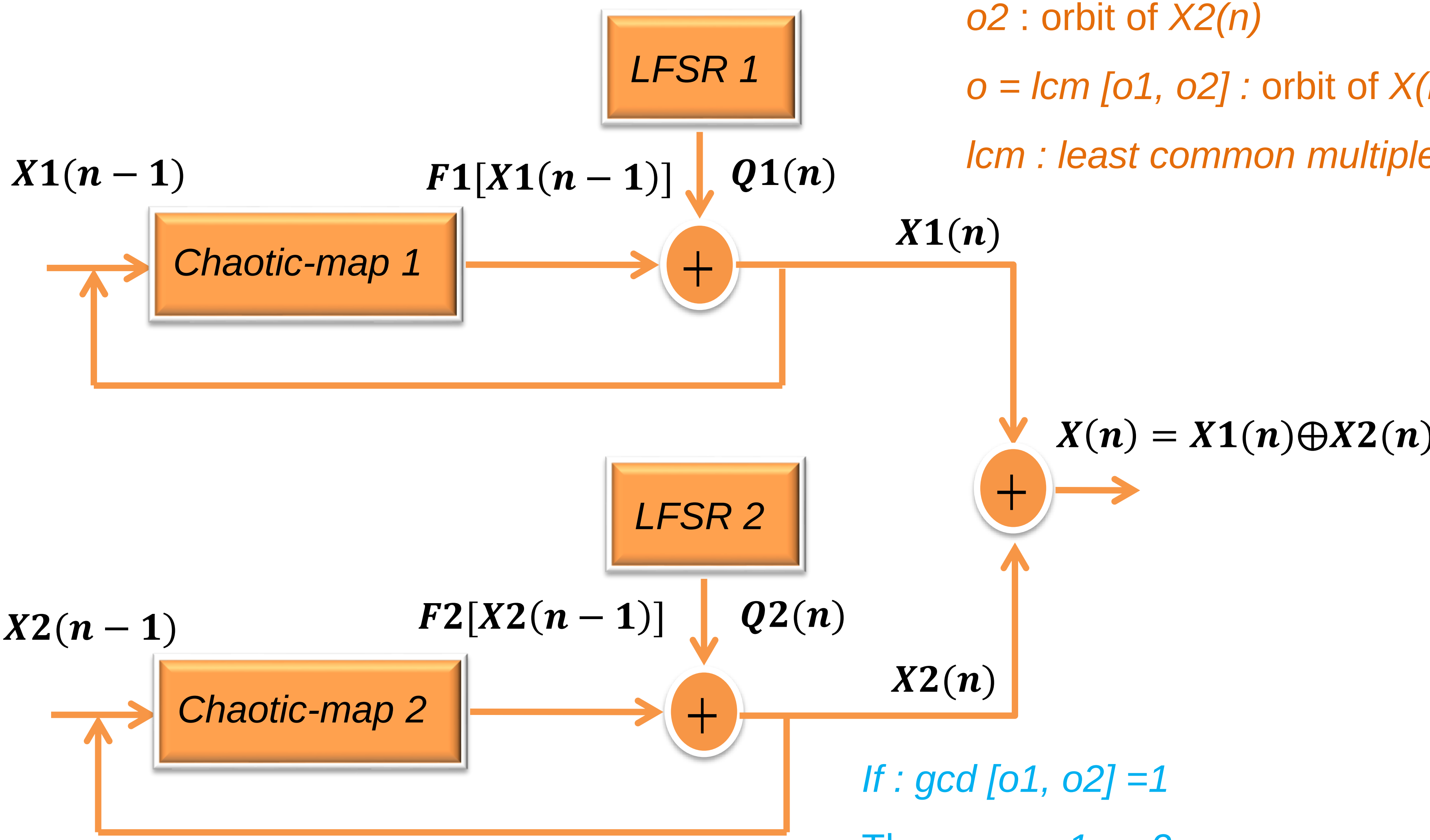
Cascading Technique

o1 : orbit of *X1(n)*

o2 : orbit of *X2(n)*

o = lcm [o1, o2] : orbit of *X(n)*

lcm : least common multiple

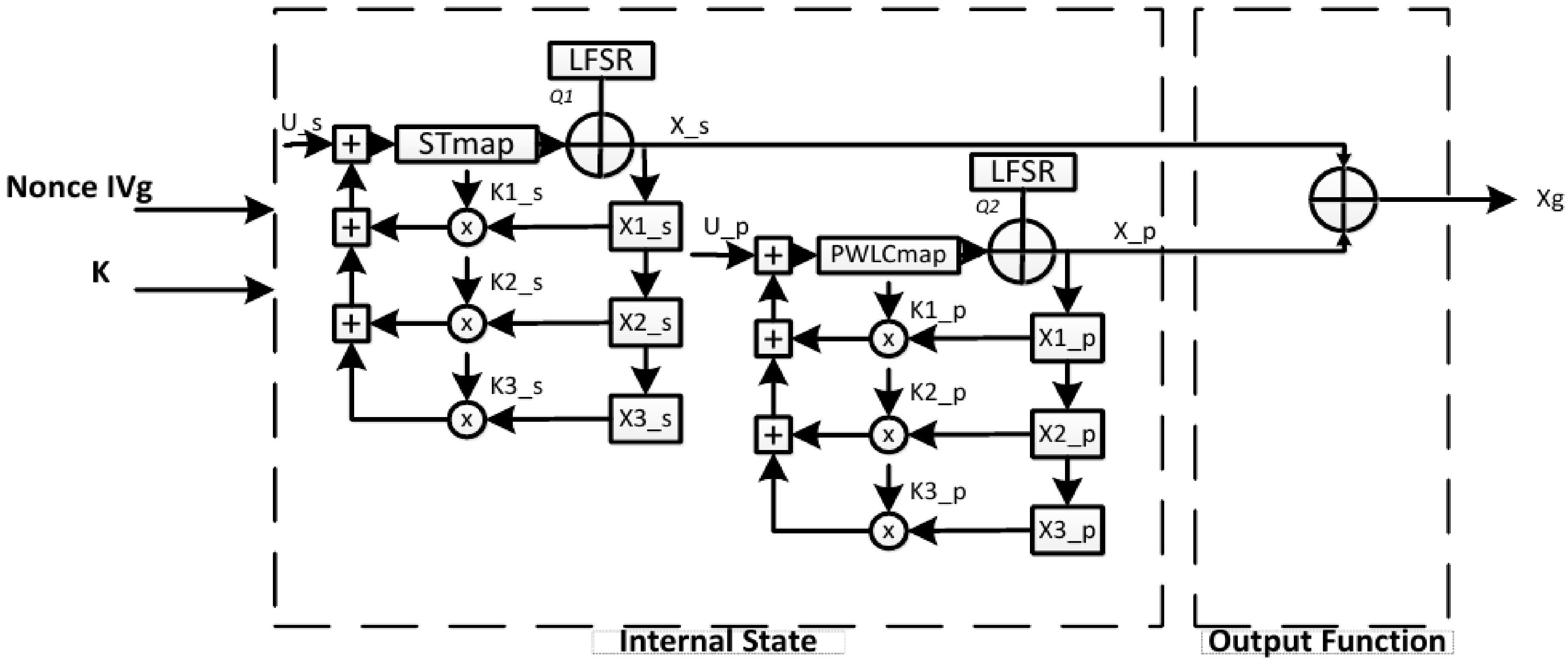


If : gcd [o1, o2] =1

Then : *o = o1 x o2*

gcd : greatest common divisor

Basic chaotic generator-PRNG (BCG-PRNG): Patent 2011



$$Xs(n) = Skew\_Tent\left\{mod\left[\left(Us + Xs(0) + \sum_{i=1}^{j} Kis \times Xs(n-i)\right), 2^N\right], Ps\right\} \oplus Q1(n)$$

$$Xp(n) = PWLC\left\{mod\left[\left(Up + Xp(0) + \sum_{i=1}^{j} Kip \times Xp(n-i)\right), 2^N\right], Pp\right\} \oplus Q2(n)$$

$1 \leq j \leq 3$

$$X_g(n) = Xs(n) \oplus Xp(n)$$

BCG-PRNG : Advantages

- **Generic scheme**
- **Long orbit of *Xg(n)*:** $o_{min} = lcm[\Delta_s \times (2^{k1} - 1), \Delta_p \times (2^{k2} - 1)]$

With: $N = 32,\ k1 = 21, k2 = 23\ and\ \Delta_{nom} \cong 2^{\frac{N}{2} \times 3} = 2^{48} \Longrightarrow 2^{71} \leq o_{min} \leq 2^{140}$

- **Large secret key space:** Brute-Force Attack infeasible

Delay d	Key size (bits) of the Skew-tent recursive cell [Nic + Np] x N + k1	Key size (bits) of the PWLCM recursive cell [Nic + (Np-1)] x N + N - 1 + k2	Key size (bits)
3	4N + 4N + k1 = 256 + 21 = 277	4N + 3N + (N-1) + k2 = 255 + 23 = 278	555
2	3N + 3N + k1 = 192 + 21 = 213	3N + 2N + (N-1) + k2 = 191 + 23 = 214	427
1	2N +2N + k1 = 128 + 21 = 149	2N + N + (N-1) + k2 = 127 + 23 = 150	299

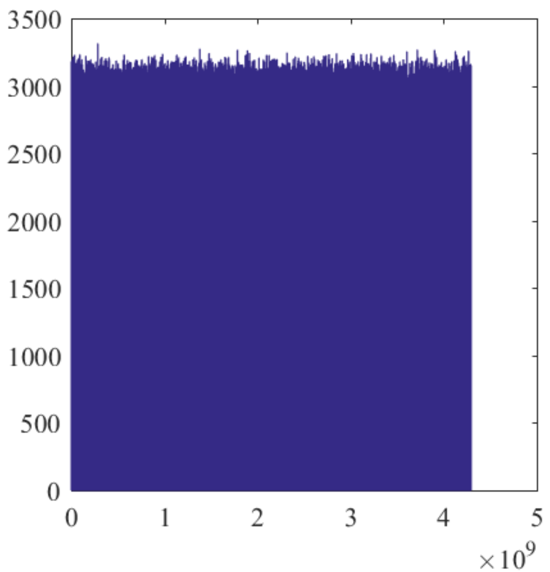
Speed of a Brute-Force Attack: (Nb of keys to be tested and the speed of each test)

With **key size = 128 bits**, there are **2^{128} possible keys.** Assuming a computer can try a **million keys a second**, it will take **[2^{128} / (10^6 x 3600 x 24 x 365)] > 10^{25} years old**, a very long time, because **the universe is only 10^{10} years old**.

- **Statistical analysis of the BCG-PRNG: Uniformity, NIST test & Key sensitivity**

- **Uniformity test:**

$$\chi^2_{ex} < \chi^2_{th}(N_c - 1, \alpha)$$



	BCG-PRNG		
	Delay = 1	Delay = 2	Delay = 3
χ^2_{ex}	**1022.96**	**990.13**	**860.35**
χ^2_{th}	1073.64 $\alpha = 0.05$ and $N_c = 1000$, $N_s = 10^8$ bits		

	BCG-PRNG	
NIST test (Delay = 1)	P-value	Prop %
Frequency	0.081	100
Block-frequency	0.616	100
Cumulative-sums (2)	0.790	100
Runs	0.494	99
Longest-run	0.350	97
Rank	0.658	100
FFT	0.213	100
Non-periodic-templates (148)	0.514	99.01
Overlapping-templates	0.575	99
Universal	0.898	99
Approximate Entropy	0.437	98
Random-excursions (8)	0.418	99.24
Random-excursions-variant (18)	0.364	99.75
Serial (2)	0.395	99.5
Linear-complexity	0.081	96

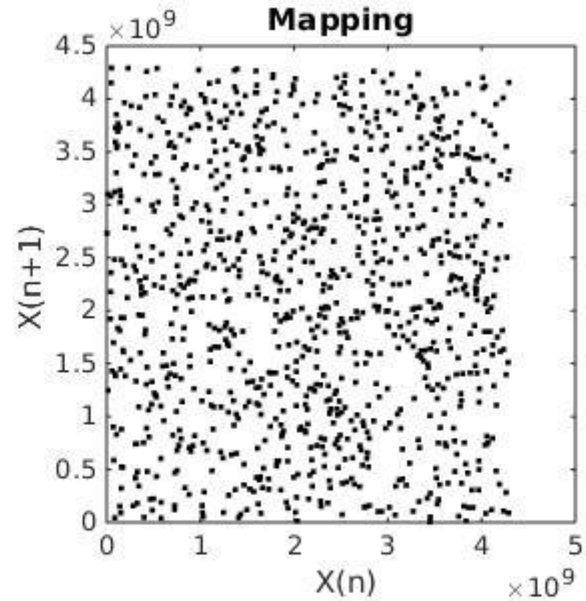
- **Key sensitivity, mapping and auto & cross correlation**

Hamming distance:

$$HD(S_1, S_2) = \frac{1}{Nb}\sum_{i=1}^{Nb}(S_1(i) \oplus S_2(i))$$

Nb is the number of bits in the sequence

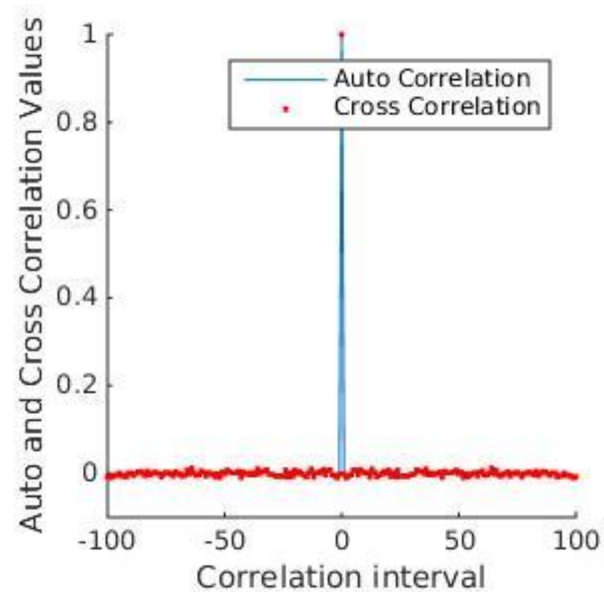
BCG-PRNG
Key space: 2^{299}
Average $H_D(S_1, S_2)$ = 0.499993



- **Computing by software: C language**

Computer: Intel ® Core™ i5-4300M, CPU @ 2.6 GHz, memory 15.6 GB, operating system: Ubuntu 14.04 Linux using GNU GCC compiler

BCG stream cipher (Delay = 1)	
Generation time (μs)	8099
Throughput (Mbps)	776,82
NCpB	26.77



Comparison of computational performance with stream ciphers, software-oriented from the eSTREAM project

Image size (Bytes): 512 x 512 x 3 = 786,432 Bytes

Stream cipher	Enc Time (µs)	ET (Mbit/s)	NCpB (Cycles/B)
LSP stream	8511	739.21	28.14
BCG stream	8099	776.82	26.77
Rabbit	3256	1842.6	9.5
HC-128	4895	1225.6	14.4
Salsa20/12	3389	1770	9.9
SOSEMANUK	3570	1680	10.5
AES-CTR	-	-	21.2

Note: eSTREAM ciphers are not secure enough [Manifavas et al., 2016].

Chaos-based stream ciphers are used to enhance the security issue.

[Manifavas et al., 2016]: Manifavas, C., Hatzivasilis, G., Fysarakis, K., & Papaefstathiou, Y. (2016). A survey of lightweight stream ciphers for embedded systems. Security and Communication Networks, 9(10), 1226-1246

Structure of the chaotic generator

Generator of chaotic Sequences

and corresponding generating

system WO Patent

WO/2011/121,218 A1, Oct 6, 2011

PCT Extension:

United States

US-8781116 B2, July 15, 2014.

Europe

EP-2553567 B1, Sept 3, 2014.

Japan:

JP 5876032 B2, Mars 02, 2016

China :

CN-103124955 B, April 20, 2016.

2958057

Truly unbreakable cipher: One-Time Pad

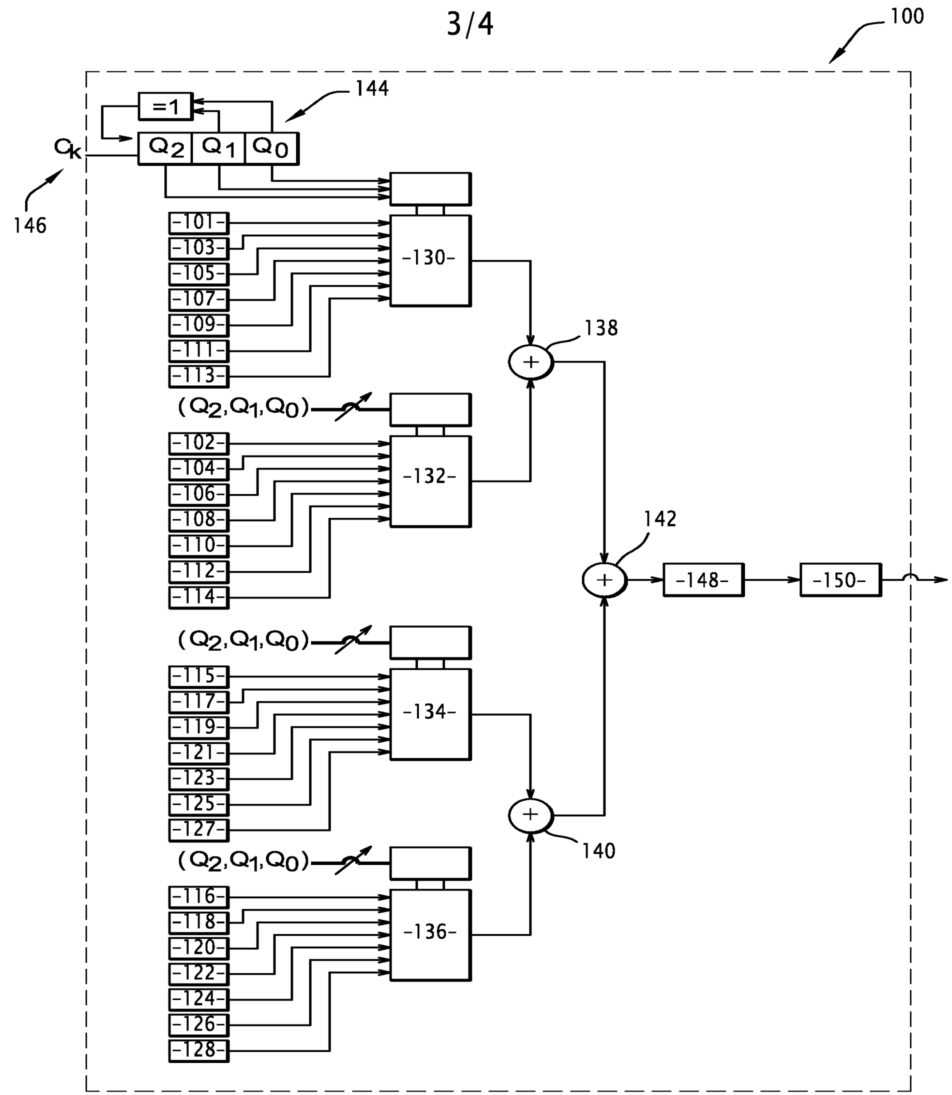


FIG.3

For each state *j = 1,2,…,7 of the LFSR*

Point 142 : $$o_{j\min_{j=1,2,\cdots,7}} = lcm\left[o_{j\min 1}, o_{j\min 2}\right]$$

Point 138 : $$o_{j\min 1_{j=1,2,\cdots,7}} = lcm\left\{\left[2^{k_{(2j-1)}} - 1\right] \times \Delta_{k_{(2j-1)}}, \left[2^{k_{(2j)}} - 1\right] \times \Delta_{k_{(2j)}}\right\}$$

Point 140 :

$$o_{j\min 2_{j=1,2,\cdots,7}} = lcm\left\{\left[2^{k_{(14+2j-1)}} - 1\right] \times \Delta_{k_{(14+2j-1)}}, \left[2^{k_{(14+2j)}} - 1\right] \times \Delta_{k_{(14+2j)}}\right\}$$

$$T_{Ck} = Min\left(o_{j\min_{j=1,2,\cdots,7}}\right)$$

$$o_{\min} = 7 \times T_{Ck}\left[1 - p\%\right]$$

Comparison of hardware performance (FPGA) with some stream ciphers

Cipher	Device	Freq. (MHz)		Slices	Throughput (Mbps)	Efficiency (Mbps/slices)
		Clock Freq.	Max. Freq.			
LSP-SC [Dridi et al 2021-b]	$Pynq\ Z2$	125	32.41	3,160	1,037.27	0.32
Improved BCG SC [Gautier et al 2019]	$Zynq\ 7000$	–	18.5	2,363	565	–
LST-SC [Dridi et al 2022]	$Pynq\ Z2$	125	36.84	1,049	1,179.07	1.12
Chaos-ring [Koyuncu et al 2020]	$Virtex-6$	125	464.688	1,050	464.688	0.44
Lorenz's chaotic system [Tanougast, 2011]	$Virtex-\mathrm{II}$	50	15.598	1,926	124	0.06
ZUC [Kitsos et al 2013 Grain-V1 Mickey-V2 Trivium	Spartan XC3S700A-4FG48	-	38 177 250 326	1147 318 98 149	1216 177 250 326	1.074 0.558 2.554 2.186

[Manifavas et al., 2016]: Manifavas, C., Hatzivasilis, G., Fysarakis, K., & Papaefstathiou, Y. (2016). A survey of lightweight stream ciphers for embedded systems. Security and Communication Networks, 9(10), 1226-1246

[Dridi et al 2021-b], "The Design and FPGA-Based Implementation of a Stream Cipher Based on a Secure Chaotic Generator", Appl. Sci. 2021, 11, 625. https://doi.org/10.3390/app11020625

[Gautier et al 2019],"Hardware implementation of lightweight chaos-based stream cipher," in *International Conference on Cyber-Technologies and Cyber-Systems*, 2019, pp. 5–pages.

[Koyuncu et al 2020], "Design, fpga implementation and statistical analysis of chaos-ring based dual entropy core true random number generator," *Analog Integrated Circuits and Signal Processing*, vol. 102, no. 2, pp. 445–456, 2020.

[Tanougast, 2011], "Hardware implementation of chaos based cipher: Design of embedded systems for security applications," in Chaos-Based Cryptography. Springer, 2011, pp. 297–330.

[Kitsos et al 2013], "FPGA-based performance analysis of stream ciphers ZUC, Snow3g, Grain V1, Mickey V2, Trivium and E0", Microprocessors and Microsystems, Elsevier.

Security analysis of stream ciphers

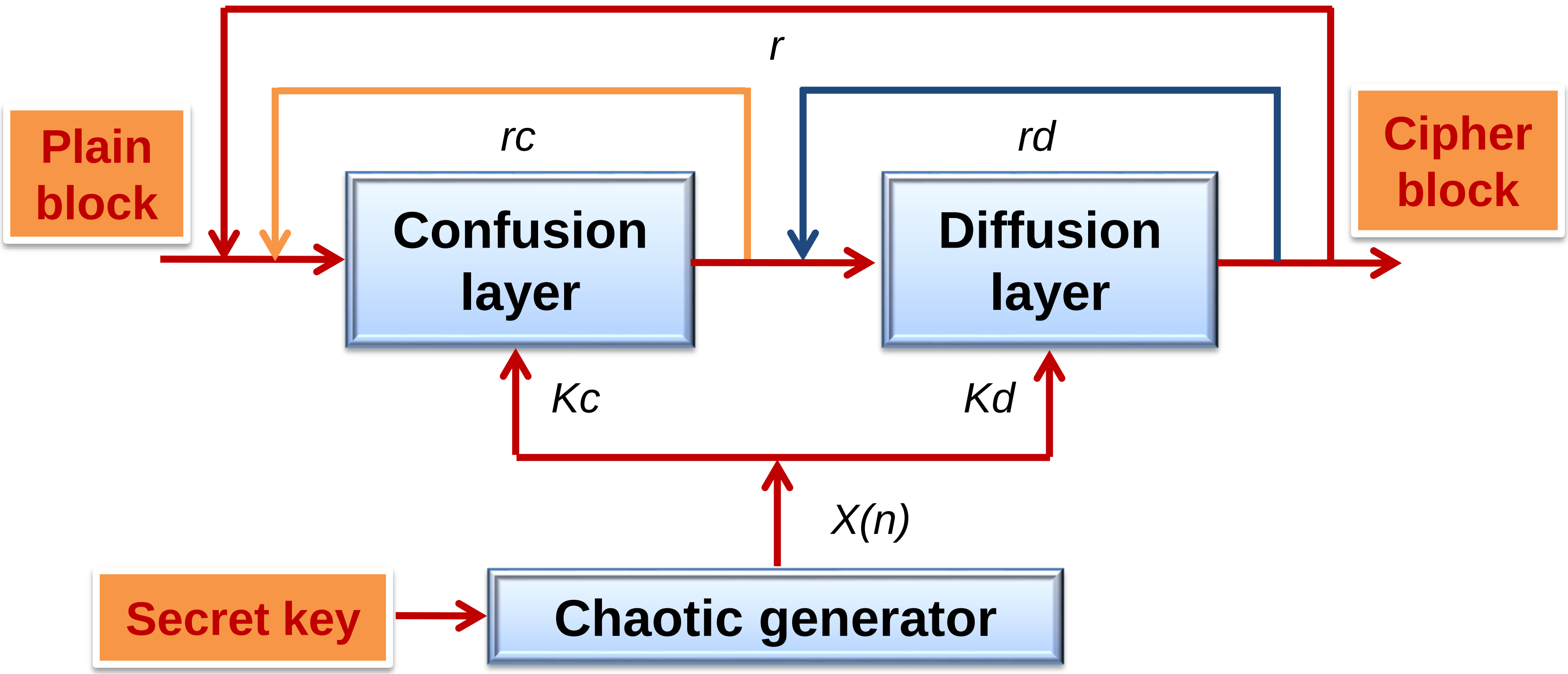
- **Key size and sensitivity analysis**
  - NPCR (Number of Pixels Change Rate)
  - UACI (Unified Average Changing Intensity)
  - HD (Hamming Distance)
- **Statistical analysis:**
  - Histogram and Chi-square analysis
  - Entropy analysis
  - Correlation analysis

Security analysis of block ciphers uses the same tests of stream ciphers + additional tests

❑ Design of efficient chaos-based cryptosystems (block ciphers) and performance evaluation

- General structure of chaos-based cryptosystems: Encryption side
- Chaos-based cryptosystems for block ciphers, 1st type:
  - Dynamic Adjustment of the Chaos-based Security in Real-time Energy Harvesting Sensors
  - A new chaos-based image encryption system
- Chaos-based cryptosystems for block ciphers, 2nd type:
  - An image encryption scheme using reverse 2-dimensional chaotic map and dependent diffusion
  - Fast and Secure Chaos-Based Cryptosystem for Images

General structure of chaos-based cryptosystems: Encryption side



Shannon [1949]

Confusion : measures how a change in the secret key affects the ciphered massage

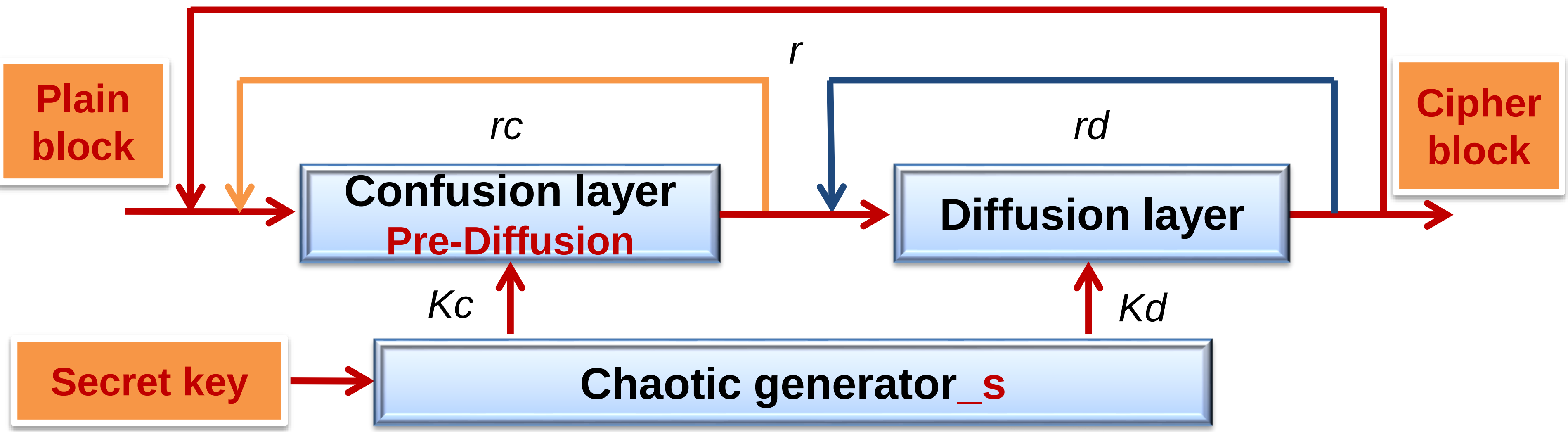
Diffusion : assesses how a change in the plain message affects the ciphered one

Fridrich [1998]

Most popular structure adopted in many chaos-based cryptosystems

Chaos-based cryptosystems for block ciphers

- **1st type : Separate layers of confusion and diffusion**



Both layers required image-scanning to obtain ciphered image

Confusion layer:

- **Pixel 2D-Permutation (Cat map; Standard map; Baker map)**

Image pixels are relocated without changing their values, an operation of **Substitution** (linear operation).

- **Pixel 1-D Substitution (Finite state Skew tent map: a non linear function)**

Image pixel values are substituted without or with Key-dependent on each round

Diffusion Layer (nonlinear operation):

1-D diffusion (Discrete Logistic map, Discrete Skew tent map)

Chaos-based cryptosystems: 1st type

Logistic map as diffusion layer (with real values $\in]0,1[$

$$\begin{cases} c(i) = v(i) \oplus q\{f[c(i-1)], L\} \\ c(-1) = q\{[4Kd \times (1-Kd)], \qquad L = 8\} \end{cases}$$

$$\begin{cases} f[c(i-1)] = 4 \times c(i-1) \times [1 - c(i-1)] \\ q[b, L] = \lfloor b \times 2^L \rfloor, \qquad b = 0.b_1 b_2 \cdots b_L, \qquad b_k \; is \; 0 \; or \; 1 \end{cases}$$

v_i is the value of the ith pixel of the permuted image

$c(i-1)$ and $c(i)$ are the values of the $(i-1)th$ and ith pixels of the diffused image, Kd is the diffusion key.

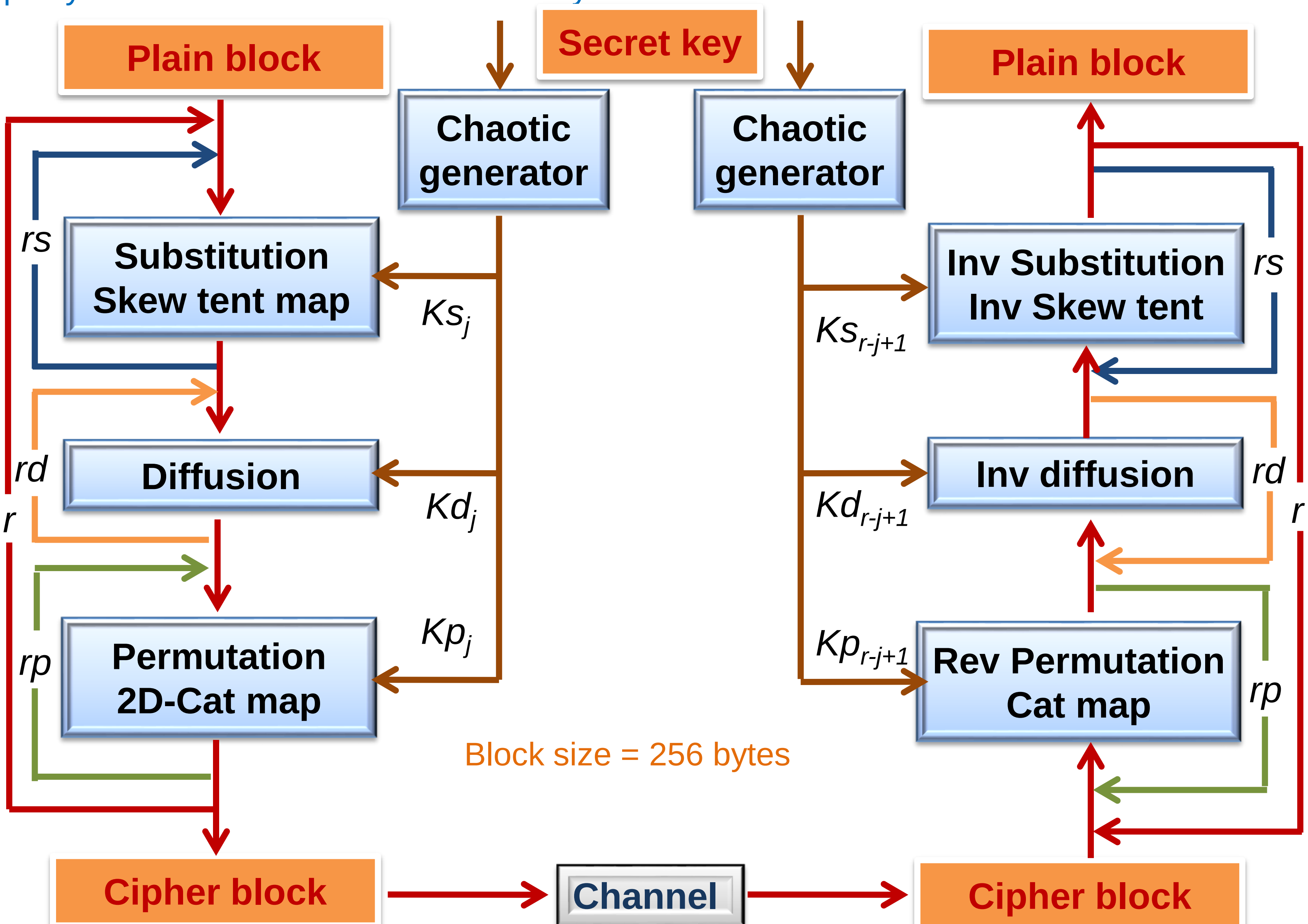
Chaotic generator_s of dynamic keys (encryption keys):

Logistic, Skew tent, PWLCM, Lorenz, PRNG-CS (combined maps)

Dynamic Adjustment of the Chaos-based Security in Real-time Energy Harvesting Sensors

[Farajallah et al., 2013]

Cryptosystem based on variable control keys



Equations of the Skew tent map and inverse Skew tent maps

Finite state Skew tent map as byte substitution layer :

Block size = 256 bytes

Robust nonlinear layer, resists to the chosen cipher text attack

$$Y = S_a(X) = \begin{cases} \left\lceil \frac{Q}{a} X \right\rceil & 0 \leq X \leq a \\ \left\lfloor \frac{Q}{Q-a}(Q-X) \right\rfloor + 1 & a < X < Q \end{cases}$$

$$Q = 2^8 = 256, \qquad X = [0, 1, 2, \cdots, 255]$$

Structure of the dynamic key Ks

$$Ks = \left[Ks_1 \left\| Ks_2 \right\| \cdots \| Ks_r \right]$$

$$Ks_j = \left[a_{j,1} \left\| a_{j,2} \right\| \cdots \| a_{j,rs} \right], j = 1, 2, \cdots, r$$

$$1 \leq a_{j,i} < Q \qquad i = 1, 2, \cdots, rs$$

Inverse Skew tent map

$$X = S_a^{-1}(Y) = \begin{cases} X1 & if \;\; m(Y) = Y \; and \; \frac{X1}{a} > \frac{Q - X2}{Q-a} \\ X2 & if \;\; m(Y) = Y \; and \; \frac{X1}{a} \leq \frac{Q - X2}{Q-a} \\ X1 & if \;\; m(Y) = Y + 1 \end{cases}$$

With: $X1 = \left\lfloor \frac{a}{Q} Y \right\rfloor$

$$X2 = \left\lceil \left(\frac{a}{Q} - 1 \right) Y + Q \right\rceil$$

$$m(Y) = Y + X1 - \left\lceil \frac{a}{Q} Y \right\rceil + 1$$

$\lfloor \;\; \rfloor$: ***Floor function***

$\lceil \;\; \rceil$: ***Ceiling function***

One to one mapping: can be implemented by lookup tables

Chaotic generator: A simplified version of the basic chaotic generator of our Patent

Modified 2-D Cat map as permutation layer

Permutation process changes each pixel position without changing its value

$$\begin{bmatrix} i_n \\ j_n \end{bmatrix} = Mod\left(\begin{bmatrix} 1 & u \\ v & 1+uv \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} ri+rj \\ rj \end{bmatrix}, \begin{bmatrix} M \\ M \end{bmatrix} \right) \qquad 0 \le u, v, ri, rj \le M-1$$

Where i, j and i_n, j_n are the original and permuted pixel positions of the $M \times M$ *square* matrix, with here $M = \sqrt{256} = 16 = 2^4$, $\text{so } |u| = |v| = |ri| = |rj| = 4\ bits$

The Cat map is bijective, so each point in the square matrix is transformed to another point uniquely.

Structure of the dynamic key Kp

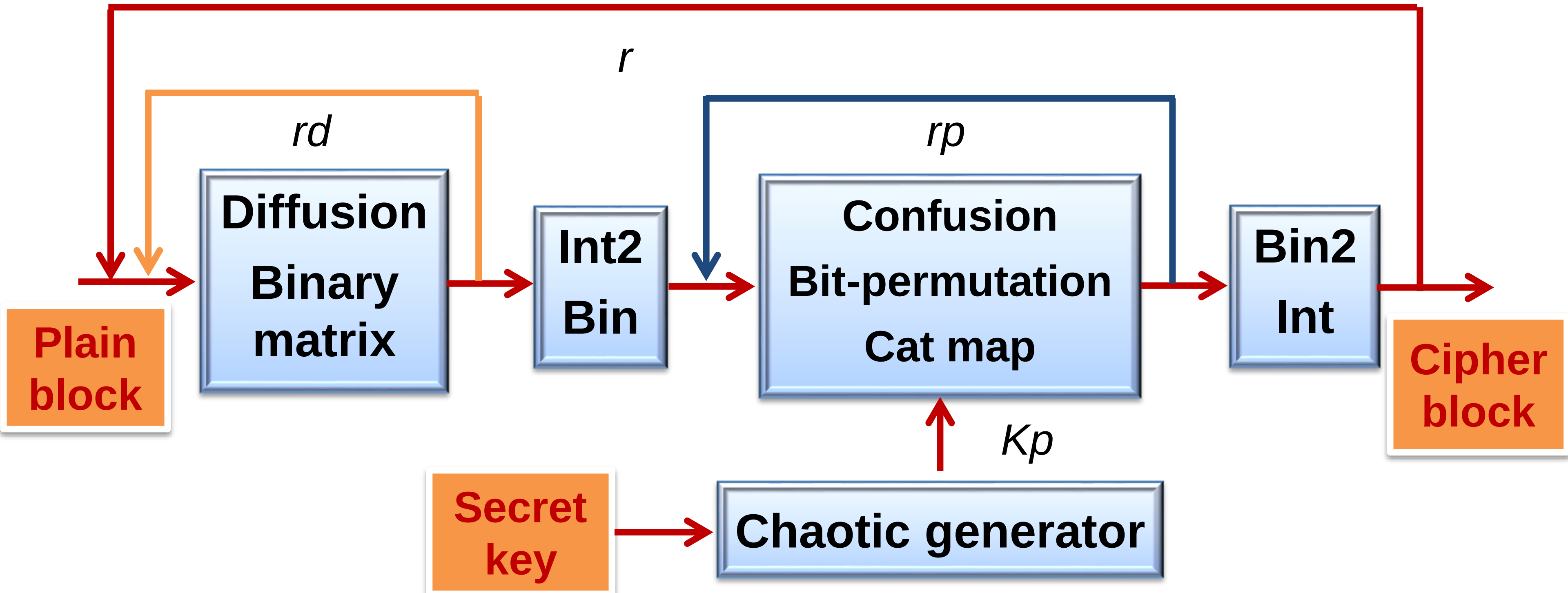
$$Kp = \left[Kp_1 \middle\| Kp_2 \middle\| \cdots \middle\| Kp_r \right]$$

$$Kp_j = \left[Kp_{j,1} \middle\| Kp_{j,2} \middle\| \cdots \middle\| Kp_{j,rp} \right] \quad j = 1, 2, \cdots, r$$

$$Kp_{j,k} = \left[u_{j,k}, v_{j,k}, ri_{j,k}, rj_{j,k} \right] \qquad k = 1, 2, \cdots, rp \qquad and\ \ here\ \left| Kp_{j,k} \right| = 4 \times 4 = 16\ bits$$

Example, how it works

[El Assad & Farajallah, 2016]: A new chaos-based image encryption system



Int2Bin: Implemented as a nonlinear converter

2D cat : Efficient formulation for C implementation

When a bit-permutation layer is applied on a block, it performs, on one scan, bit relocating and byte substitution with a change of its value.

Diffusion binary matrix $[DM]$

The diffusion and inverse diffusion layer operate on blocks of 32 bytes each (i.e. 256 bits)

Diffusion layer

$$\begin{bmatrix} Od_{\mathbf{0}} \\ Od_2 \\ \vdots \\ Od_{\mathbf{31}} \end{bmatrix} = [\boldsymbol{D}M] \odot \begin{bmatrix} O_{\mathbf{0}} \\ O_2 \\ \vdots \\ O_{\mathbf{31}} \end{bmatrix}$$

Inverse Diffusion layer

$$\begin{bmatrix} O_{\mathbf{0}} \\ O_2 \\ \vdots \\ O_{\mathbf{31}} \end{bmatrix} = [\boldsymbol{D}M]^{-\mathbf{1}} \odot \begin{bmatrix} Od_{\mathbf{0}} \\ Od_2 \\ \vdots \\ Od_{\mathbf{31}} \end{bmatrix}$$

$[DM]$ is the binary diffusion square matrix of 32×32, which is invertible

$O_i, Od_i \in [0, 255]$

$\odot$ is an matrix operator defined as shown by the first diffused byte $Od_{\mathbf{0}}$ and the first inverse diffused byte $O_{\mathbf{0}}$

$$Od_{\mathbf{0}} = O_0 \oplus O_2 \oplus O_3 \oplus O_4 \oplus O_5 \oplus O_8 \oplus O_9 \oplus O_{10} \oplus O_{12} \oplus O_{13} \oplus O_{17} \oplus O_{18} \oplus O_{19} \oplus O_{24} \oplus O_{25} \oplus O_{29} \oplus O_{31}$$

$$O_{\mathbf{0}} = Od_{\mathbf{0}} \oplus Od_{\mathbf{8}} \oplus Od_{\mathbf{9}} \oplus Od_{\mathbf{10}} \oplus Od_{\mathbf{16}} \oplus Od_{17} \oplus Od_{\mathbf{19}} \oplus Od_{\mathbf{20}} \oplus Od_{22} \oplus Od_{23} \oplus Od_{24} \oplus Od_{\mathbf{25}} \oplus Od_{27} \oplus Od_{\mathbf{29}} \oplus Od_{31}$$

[Koo et al 2006] On Constructing of a 32× 32 Binary Matrix as a Diffusion Layer r for a 256-bit block cipher
[Li et al 2011] Impossible Dif ferential Cryptanalysis of SPN Ciphers

$[DM] =$

```
1 0 1 1 1 1 0 0 1 1 1 0 1 1 0 0 0 1 1 1 0 0 0 0 1 1 0 0 0 1 0 1
1 1 0 1 0 1 1 0 0 1 1 1 0 1 1 0 1 0 1 1 0 0 0 0 0 1 1 0 1 0 1 0
1 1 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 1 0 1 0 0 0 0 0 0 1 1 0 1 0 1
0 1 1 1 1 0 0 1 1 1 0 1 1 0 0 1 1 1 1 0 0 0 0 0 1 0 0 1 1 0 1 0
1 1 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 0 0 1 1 0 0 0
0 1 1 0 1 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0 1 0 0
0 0 1 1 0 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 1 0
1 0 0 1 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 1
1 1 1 0 0 1 0 0 1 1 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0
0 1 1 1 0 0 1 0 0 1 1 1 1 1 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0
1 0 1 1 0 0 0 1 1 0 1 1 0 1 1 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0
1 1 0 1 1 0 0 0 1 1 0 1 0 0 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
1 1 0 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 0 0 1 1 0 0 0 0 0 0 1 1 1 0
0 1 1 0 1 1 0 1 1 1 0 0 1 0 1 1 0 1 1 0 0 1 1 0 0 0 0 0 0 1 1 1
0 0 1 1 1 1 1 0 0 1 1 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 1 1
1 0 0 1 0 1 1 1 0 0 1 1 1 1 1 0 1 0 0 1 1 0 0 1 0 0 0 0 1 1 0 1
0 1 1 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0
1 0 1 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0
1 1 0 1 0 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0
1 1 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0 0 0 0 0 1
0 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 1 1 1 1 0 0 1 0 1 1
0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 0 1 0 0 0 1 1 0 1 0 1 1 0 1 1 0 1
0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0 1 1 1 0 0 0 1 1 1 1 1 0
0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0 1 0 0 1 1 1 1 0 0 1 0 1 1 1
1 1 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1 1 0 1 1 0 0 0 1 1 1 0 0 0 0
0 1 1 0 1 0 0 0 1 0 1 1 0 0 0 0 0 0 1 1 0 1 1 0 1 0 1 1 0 0 0 0
0 0 1 1 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 1 0 0 1 1 1 1 0 1 0 0 0 0
1 0 0 1 0 0 1 0 1 1 1 0 0 0 0 0 1 1 0 0 1 0 0 1 1 1 1 0 0 0 0 0
0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 1 0 1 1 0 0 0 0 1 1 1 0
1 0 1 0 0 1 0 0 0 0 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 0 0 0 1 1 1
0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 1 0 0 1 0 1 1 1 0 0 0 0 0 1 0 1 1
1 0 1 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 0 1 0 1 1 1 0 0 0 0 1 1 0 1
```

$[DM]^{-1} =$

```
1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 1 1 0 1 1 1 1 0 1 0 1 0 1
0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 0 1 1 0 1 1 1 1 0 1 0 1 0
0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 1 0 1 0 1
0 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 0 1 1 0 1 1 1 1 0 1 1 1 0 1 0
0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1 1 0 0 1 0 0 1 1 0 1 1 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 1 1 0 0 1 0 0 1 1 0 1 1 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 1 1 0 1 1 0 0 1 1 0 1 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 1 0 0 1 1 0 1 1 0 1 1 1 0 0 0 0 0
1 1 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 1
0 1 1 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 0 0
1 0 1 1 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 0
1 1 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1
0 0 0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 0 1 1 0 1 1 0 1 1 0 1 1 1
0 0 0 0 1 1 1 0 0 1 1 0 1 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 1 0 1 1
0 0 0 0 0 1 1 1 0 0 1 1 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 1 0 1
0 0 0 0 1 0 1 1 1 0 0 1 0 0 1 0 0 0 0 0 1 0 1 1 0 1 1 1 1 1 1 0
1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 0 1 0 0 0 0 0
1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 0 1 0 0 0 0
0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 0 1 0 0 0 0 0
1 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 0 1 0 1 0 0 0 0
1 0 1 1 0 0 1 1 0 0 0 0 1 1 0 1 1 1 1 0 1 0 0 0 0 0 0 0 1 0 1 1
1 1 0 1 1 0 0 1 0 0 0 0 1 1 1 0 0 1 1 1 0 1 0 0 0 0 0 0 1 1 0 1
1 1 1 0 1 1 0 0 0 0 0 0 0 1 1 1 1 0 1 1 0 0 1 0 0 0 0 0 1 1 1 0
0 1 1 1 0 1 1 0 0 0 0 0 1 0 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 1 1 1
1 1 0 1 0 1 1 1 1 0 1 1 1 0 1 1 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0
1 1 1 0 1 0 1 1 1 1 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0
0 1 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0
1 0 1 1 1 1 1 0 0 1 1 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0
0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0 1 0
1 0 1 0 0 0 0 0 1 1 0 0 1 0 1 1 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 1
0 1 0 1 0 0 0 0 0 1 1 0 1 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0
1 0 1 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0
```

Modified 2-D Cat map as permutation layer

The permutation process operates on the bits, it changes each bit position and the values of input bytes

$$\begin{bmatrix} i_n \\ j_n \end{bmatrix} = Mod\left(\begin{bmatrix} 1 & u \\ v & 1+uv \end{bmatrix}\begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} ri+rj \\ rj \end{bmatrix}, \begin{bmatrix} M \\ M \end{bmatrix}\right) \qquad 0 \le u, v, ri, rj \le M-1$$

Smallest example with 8 bytes = 64 bits, so $M = 8$

Structure of the dynamic key Kp

$$Kp = \left[Kp_1 \middle\| Kp_2 \middle\| \cdots \middle\| Kp_r \right]$$

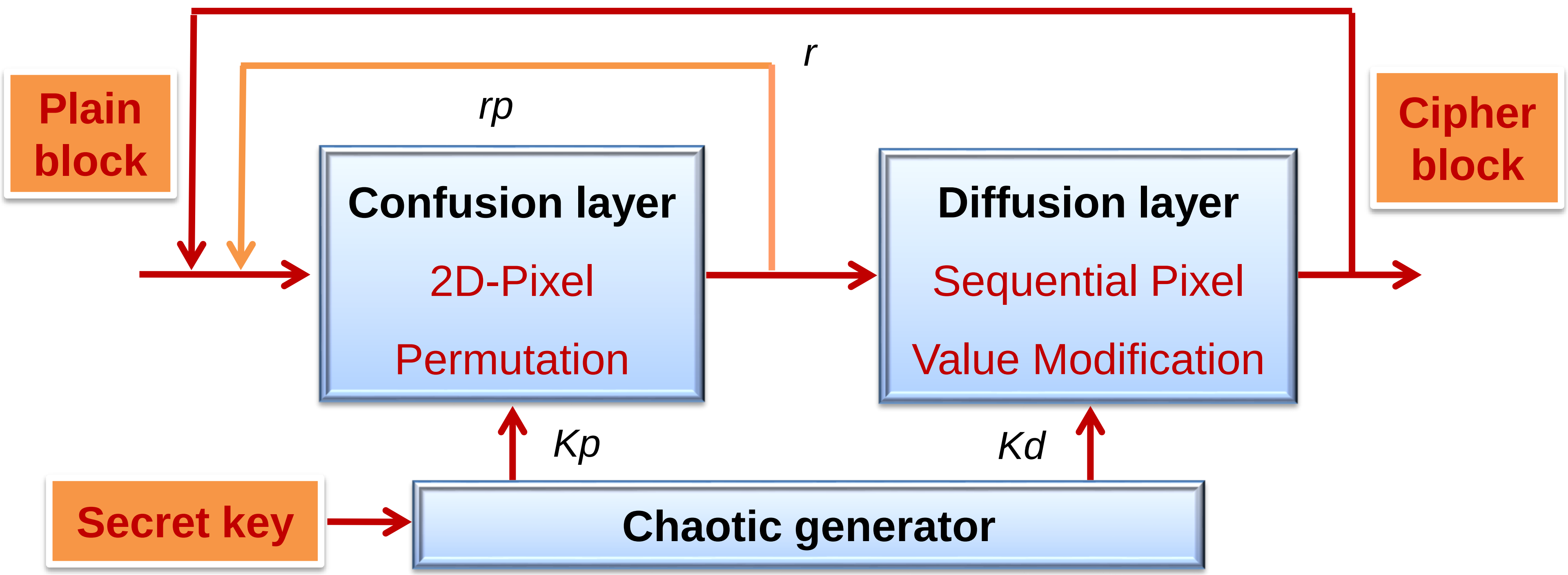
$$Kp_j = \left[Kp_{j,1} \middle\| Kp_{j,2} \middle\| \cdots \middle\| Kp_{j,rp} \right] \quad j = 1, 2, \cdots, r$$

$$Kp_{j,k} = \left[u_{j,k}, v_{j,k}, ri_{j,k}, rj_{j,k} \right] \qquad k = 1, 2, \cdots, rp$$

$$Here{:}\ M = \sqrt{32 \times 8 = 256} = 16, so\ the\ |Kp|\ for\ a\ needed\ values\ of\ r\ and\ rp$$

Chaos-based cryptosystems, 2nd type: Principle

- **2nd type : Combined layers of confusion and diffusion**

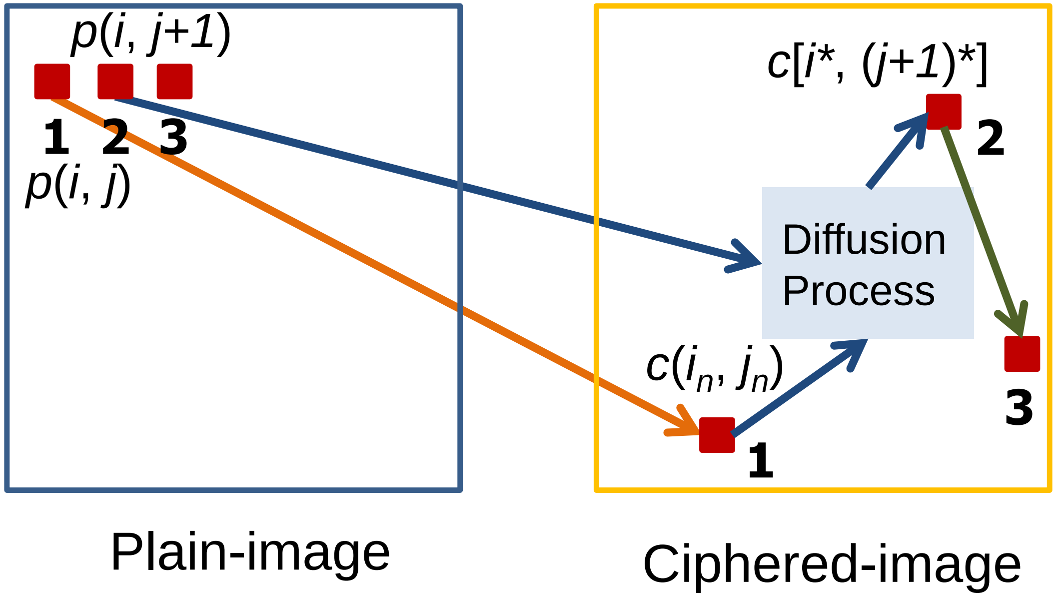


The confusion and diffusion processes are performed simultaneously in a single scan of plain-image pixels.

It is more robust against cryptanalysis and faster than 1st type cryptosystems.

[Zhang et al., 2013]: An image encryption scheme using reverse 2-dimensional chaotic map and dependent diffusion

The diffusion process at the pixel level is governed by the confusion one



$$\begin{cases} [i_n, j_n] = Cat[(i, j, u, v, ri, rj), M] \\ c(i_n, j_n) = p(i, j) \oplus f(z) \\ z = c(i_n, j_n) \end{cases}$$

$$\begin{cases} f(z) = \left[\left\lfloor r\dfrac{z}{1000} \times \left(1 - \dfrac{z}{1000}\right)\right\rfloor\right] \times 1000 \; mod \; 256 \\ z(-1) = [\lfloor 4Kd \times (1 - Kd) \rfloor] \times 1000 \; mod \; 256 \end{cases}$$

with z is a temporary variable storing the value of the previous ciphered pixel

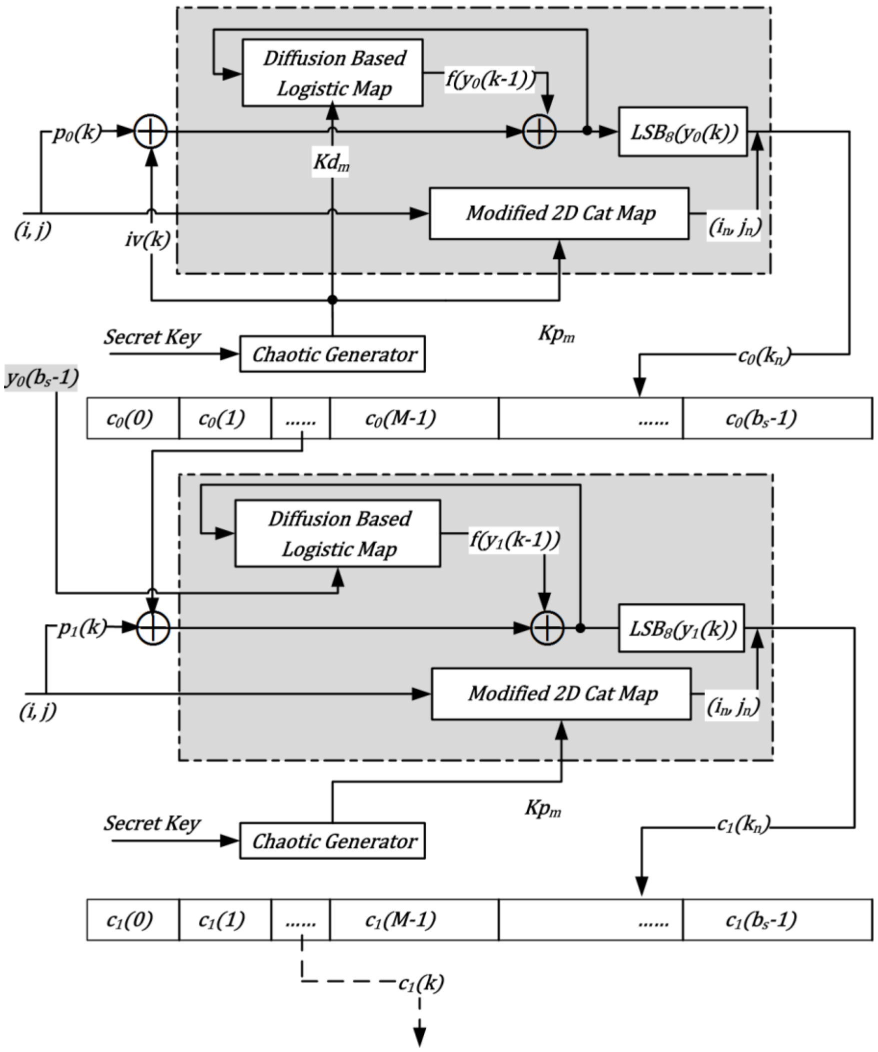
$r \in [3.56996, 4]$: the control parameter
$f(z)$: Logistic map

As the ranges of $f(z)$ and z are both [0,255], a look-up table can be used to reduce the execution time

$z(-1)$ is the initial value of z *and* Kd *the diffusion key,* $Kd = 0.33456434300001$ for example

[Farajallah et al., 2016]: Fast and secure chaos-based cryptosystem for images

PhD thesis: Mousa Farajallah 2015



$$y_l(k) = p_l(k) \oplus s_{l-1}(k) \oplus f(y_l(k-1))$$

$$c_l(k_n) = LSB_8[y_l(k)]$$

$$s_{l-1}(k) = \begin{cases} iv(k) & if \;\; l = 0 \\ c_{l-1}(k) & if \;\; l > 0 \end{cases}$$

$$k_n = i_n \times M + j_n$$

$$k = i \times M + j$$

Diffusion process :

V1: Discrete Logistic map with N = 32 bits

V2: Discrete Skew tent map with N = 32 bits

V3 : Look up table with N = 8 bits, of the Skew-tent map (as diffusion layer)

Advantages of chaos-based cryptosystems: 2nd type:

- The sensitivity to any modifications in the plain-image is increased.

  Indeed, ciphered pixels $c(i_n, j_n)$ are influenced by both the diffusion key Kd and the previously ciphered pixels z.

- The confusion effect can't be removed using a homogeneous plain-image HI:

$$HI \xrightarrow{K_{p1}} C1$$

$$HI \xrightarrow{K_{p2}} C2 \neq C1$$

In separate confusion – diffusion architecture : $c(i) = v(i) \oplus q\{f[c(i-1)], L\}$

$$HI \xrightarrow{K_{p1}} c1(i) = v \oplus q\{f[c1(i-1)], L\} \rightarrow C1$$

$$HI \xrightarrow{K_{p2}} c2(i) = v \oplus q\{f[c2(i-1)], L\} \rightarrow C2 = C1$$

Computing Performance

Average Encryption / Decryption time

Encryption Throughput

Number of needed Cycles per Bytes

$$\boldsymbol{ET(MByte/s) = \frac{Image\ Size\ (MBytes)}{Average\ Encyption\ Time\ (second)}}$$

$$\boldsymbol{NCpB = \frac{CPU\ Speed\ (Hertz)}{ET(Byte/s)}}$$

Average is done by encrypting the image under test at least 100 times with different secret keys each time

Results are carried out by using :

C language, PC: 3.1 GHz processor Intel Core TM i3-2100 CPU, 4GB RAM

Windows 7, 32-bit operating system.

Computing performance: comparison

Lena image of size 256 X 256 X 3 Bytes

Crypto3-V1 : Discrete Logistic map-32 bit (as diffusion)

Crypto3-V2: Discrete Skew tent map-32 bit (as diffusion)

Crypto3-V3: Look up table-8 bit of the Skew tent map (as diffusion)

Cryptosystem	Enc / Dec times (ms)	ET (Mbyte/s)	Cycles per Byte
[Farajallah et al., 2013]	9.9 / 32.4	18.9	157
[El Assad & Farajallah, 2016]	8.38 / 8.48	22.3	132
[Farajallah et al., 2016] -V1	2.1 / 2.6	93.9	32
[Farajallah et al., 2016] –V2	4.15 / 4.79	45.3	65
[Farajallah et al., 2016] –V3	1.3 / 1.4	140.7	21
[Zhang et al., 2013]	7.5 / 8.25	25	122
[Wang et al., 2011]	7.79 / 8.39	24.1	208
[Wong et al., 2008]	15.59 / 16.77	7.2	417
AES	1.75 / 1.8	122	24

- **Confusion**

  Makes the relationship between ciphertext statistics and secret key value as complex as possible to thwart an attacker's attempts to discover the secret key.

  It obscures the relationship between the plaintext and ciphertext.

  If we change a single bit in the secret key, then (statistically) half of the bits in the ciphertext should change.

  It is difficult for an attacker to find the secret key from the ciphertext.

  It is used by both block and stream ciphers

  Confusion level is measured by: Histogram uniformity, correlation analysis, information entropy, Hamming distance: HD(P, C) and HD[C(K1), C(K2)]

- **Diffusion**

  Spreads the statistics of the plaintext into long-range statistics of the ciphertext.

  If we change a single bit of the plaintext, then (statistically) half of the bits in the ciphertext should change, and similarly, if we change one bit of the ciphertext, then approximately one half of the plaintext bits should change.

  Makes the statistical relationship between plaintext and ciphertext as complex as possible, so that it is difficult for an attacker to deduce the secret key.

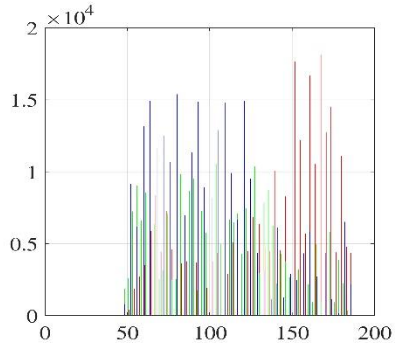
  It is used by block ciphers only.

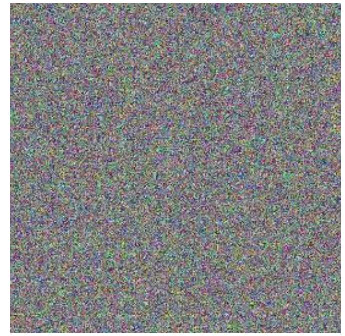
  Diffusion level is measured by HD[C(P1), C(P2)], NPCR and UACI parameters.

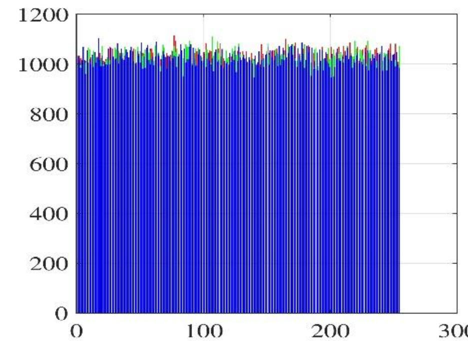
Confusion property

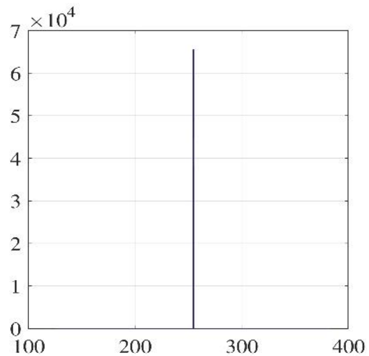
- **Histogram, chi-square test: χ^2**

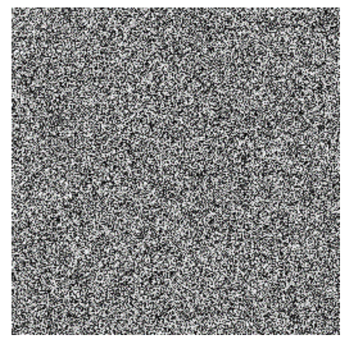
Plain image	Histogram	Ciphered image	Histogram

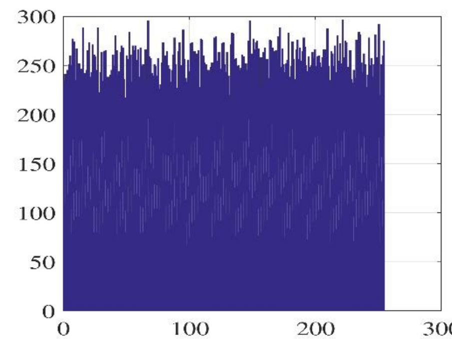


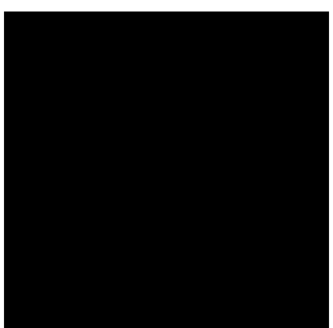


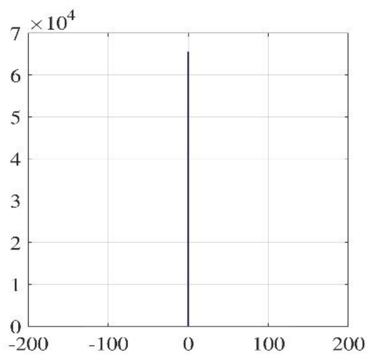


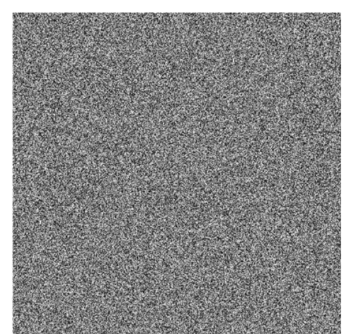


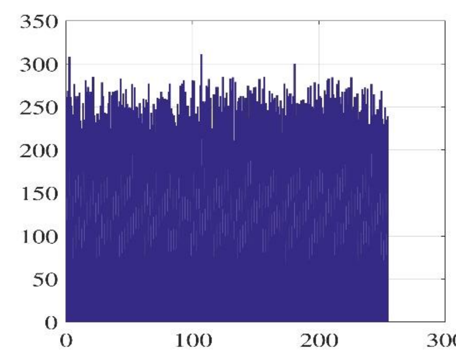












- **Uniformity:** $\chi^2_{ex} < \chi^2_{th}(N_c - 1, \alpha) = \chi^2_{th}(255, 0.05) = 293.2478$

- **Redundancy:**

$$H(C) = -\sum_{0}^{N_c-1} Pr(c_i) \times \log_2 Pr(c_i)$$

$H(C)$: Information entropy of the ciphered image
$H(P)$: Information entropy of the plain image

$Pr(c_i)$ is the probability of occurrence of each level c_i ∈ [i = 0, 1, 2, ..., 255].
If $Pr(c_i) = 2^{-8}$, $H(C) = 8$, the maximal value.

Image	Size	χ^2_{ex}	Entropy $H(P)$	Entropy $H(C)$
Lena	512 ×512 ×3	258.0559	5.6822	7.9998
White	256 ×256 ×1	255.4313	0	7.9968
Black	256 ×256 ×1	253.2374	0	7.9974
Goldhill	512 ×512 ×3	252.7116	7.6220	7.9997

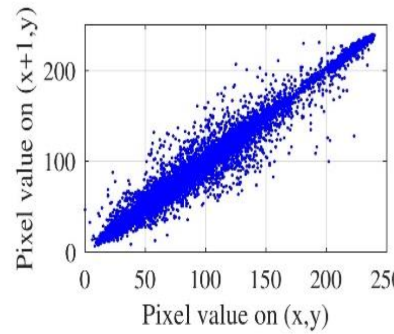
▪ Correlation analysis

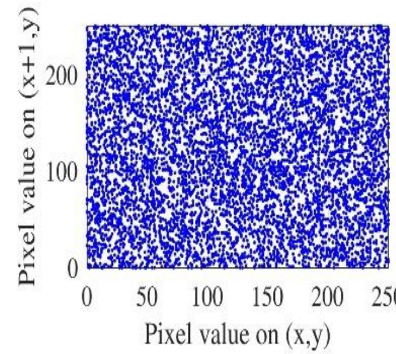
Correlation coefficient ρ_{xy}

$$\rho_{xy} = \frac{\sum_{i=1}^{N}(x_i - \bar{x}) \times (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}}$$

N=8000 pairs (x, y) of two adjacent pixels randomly selected in vertical, horizontal, and diagonal directions from the original and encrypted images.

Goldhill image





Correlation of adjacent pixels of Goldhill plain image and its cipher image in horizontal direction

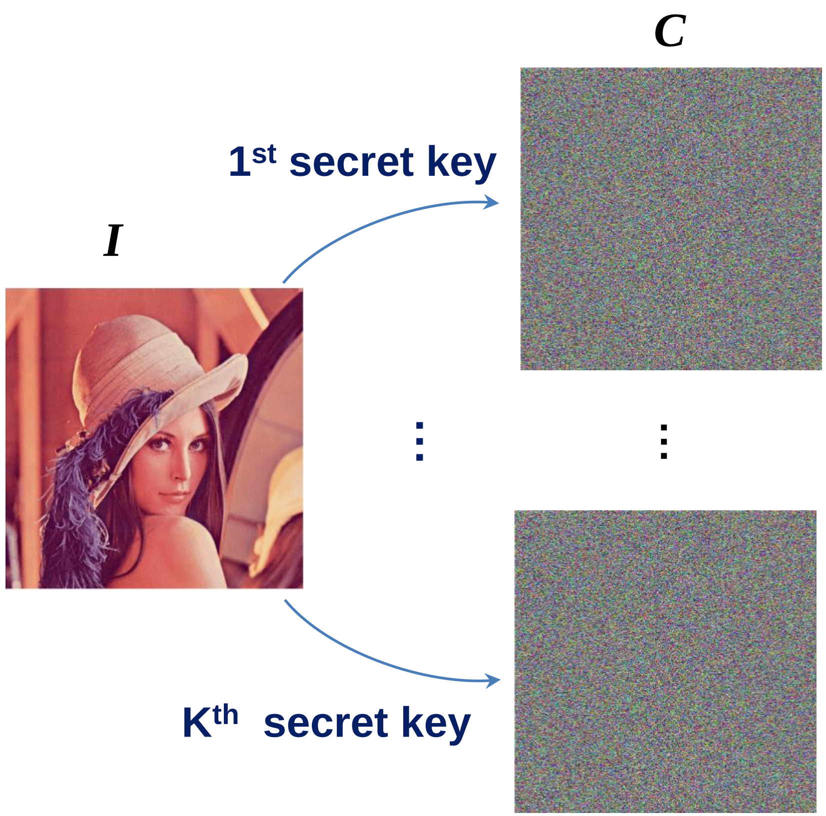
Image	Size		Plain image			Ciphered image		
			H	V	D	H	V	D
Lena	512 ×512 ×3	R	0.97524	0.98533	0.96489	-0.00028	0.00229	-0.00107
		G	0.96666	0.98009	0.95345	-0.00184	-0.00069	0.00190
		B	0.93391	0.95554	0.91848	0.00160	-0.00181	-0.00105
White	256×256 ×1		-	-	-	-0.00105	-0.00146	0.00152
Black	256×256 ×1		-	-	-	0.00017	-0.00176	-0.00203
Goldhill	512 ×512 ×3	R	0.97764	0.97647	0.95983	-0.00059	-0.00031	-0.00133
		G	0.98196	0.98501	0.97002	-0.00089	-0.00171	0.00052
		B	0.98444	0.98646	0.97345	-0.00089	0.00003	-0.00157

- **Complexity between ciphered image, plain image and secret key**

Hamming distance :

$$HD(I, C) = \frac{1}{Nb}\sum_{i=1}^{Nb}[I(i) \oplus C(i)]$$

Nb is the number of bits of the test image



$K = 100$ random secret keys

Image	Size	Average $HD(\%)$ optimal value: 50%
Lena	512 ×512 ×3	49.9978
White	256 ×256 ×1	50.0012
Black	256 ×256 ×1	49.9949
Goldhill	512 ×512 ×3	50.0032

▪ Key sensitivity test

A good encryption scheme should be sensitive to the secret key in process of both encryption and decryption.

$$I1 \xrightarrow{Key} C1 \qquad C1 \xrightarrow{Key} I1$$

$$I1 \xrightarrow{Key\ with\ 1\ bit\ change} C2 \neq C1 \qquad C1 \xrightarrow{Key\ with\ 1\ bit\ change} I2 \neq I1$$

$K = 100$ random secret keys

Hamming distance :

$$\boldsymbol{HD(C1, C2) = \frac{1}{Nb}\sum_{i=1}^{Nb}[C1(i) \oplus C2(i)]}$$

Nb is the number of bits of the test image

Image	**Size**	$\boldsymbol{HD}(\%)$ **optimal value: 50%**
Lena	512 ×512 ×3	49.9907
White	256 ×256 ×1	50.0000
Black	256 ×256 ×1	50.0094
Goldhill	512 ×512 ×3	49.9941

Diffusion property

- **Plaintext sensitivity attack:**

To resist the chosen plaintext attack and the differential attack, the cryptosystem should be highly sensitive to one bit LSB change in the plaintext.

We evaluate the plaintext sensitivity as follows:

$$I \xrightarrow{Key} C$$

$$I1 = I \text{ with } 1 \text{ bit } LSB \text{ change} \xrightarrow{Key} C1 \neq C$$

Hamming distance :

$$HD(C, C1) = \frac{1}{Nb} \sum_{i=1}^{Nb} [C1(i) \oplus C2(i)]$$

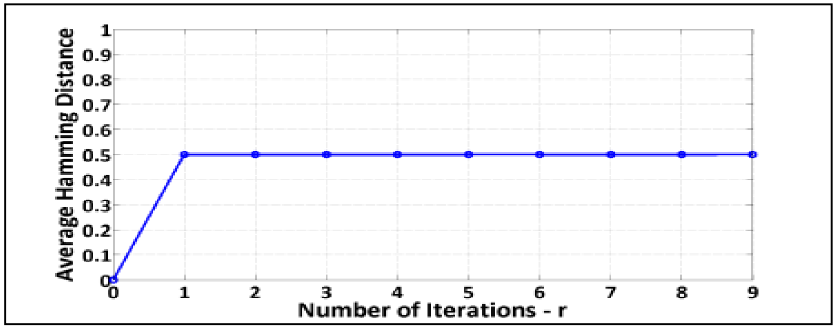
Black image: $I = [0, 0, \dots, 0]$ and $I1 = [0, 0, \dots 1_i, \dots, 0]$

If the Hamming distance is close to 50% (probability of bit changes close to 1/2), then the previous attacks would become ineffective.

This test gives also the minimum number of rounds r, needed to overcome the plaintext sensitivity attack.

- **Plaintext sensitivity attack:** number of rounds r needed

Average Hamming distance (over 1000 randomly chosen pixel positions in turn to change their 1-bit LSB) versus the number of rounds r.
With $r = 1$, the effect avalanche is reached.



- $NPCR$ and $UACI$ criteria

Number of pixel change rate: $NPCR$

$$NPCR = \frac{1}{L \times C \times P} \sum_{k=1}^{P} \sum_{i=1}^{L} \sum_{j=1}^{C} D(i,j,k) \times 100\%$$

$$D(i,j,k) = \begin{cases} 0 & if \;\; C(i,j,k) = C1(i,j,k) \\ 1 & if \;\; C(i,j,k) \neq C1(i,j,k) \end{cases}$$

Unified average changing intensity: $UACI$

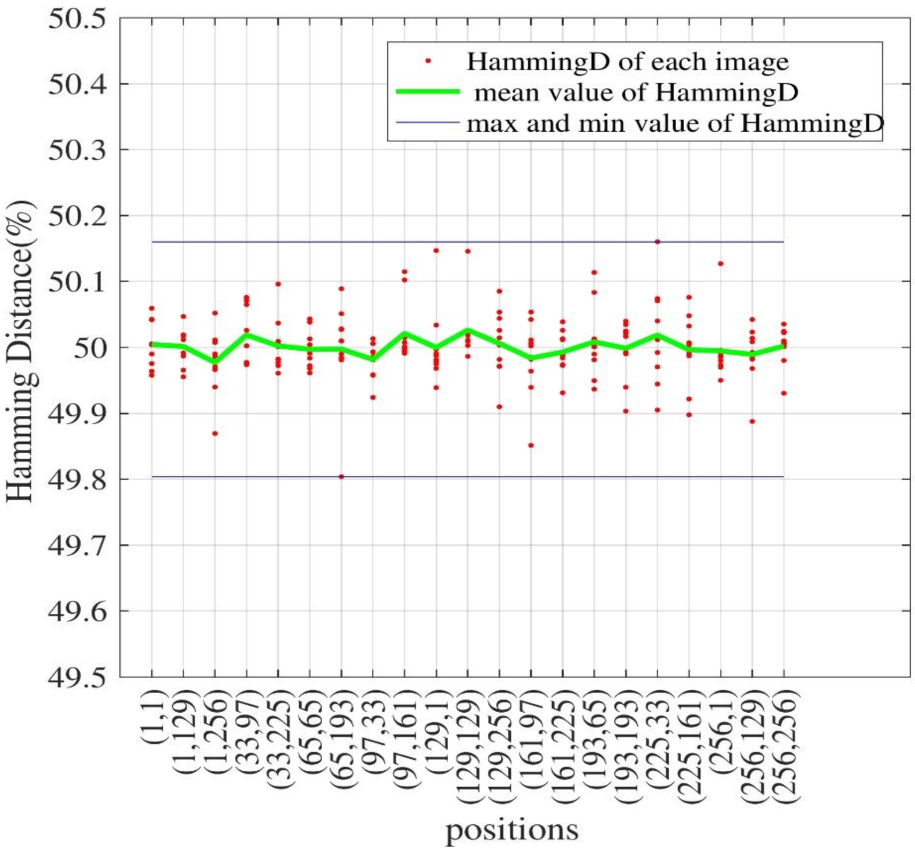
$$UACI = \frac{1}{L \times C \times P} \times \frac{1}{255} \sum_{k=1}^{P} \sum_{i=1}^{L} \sum_{j=1}^{C} |C(i,j,k) - C1(i,j,k)| \times 100\%$$

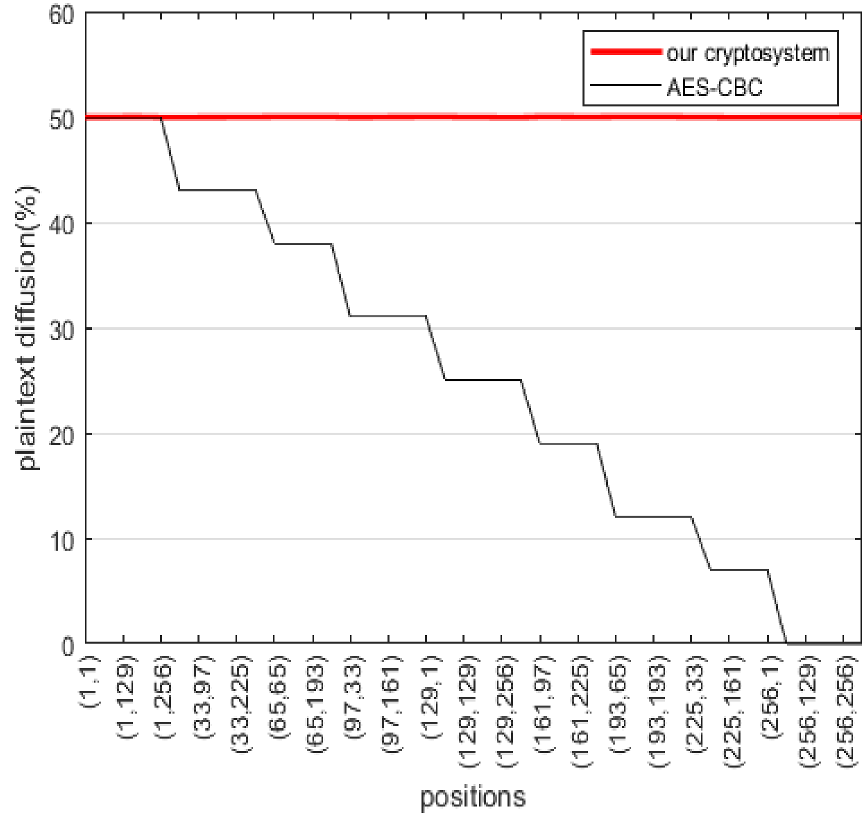
For two random images the expected values of $NPCR$ and $UACI$ are:

$E(NPCR) = 99.609\ \%$

$E(UACI) = 33.463\ \%$

▪ Plaintext sensitivity attack





Hamming distance for 10 tested images at each position (21 positions).

Average values over images are shown in green line

- **Plaintext sensitivity attack**

Image	Size		Plaintext sensitivity		
		HD (%)	$NPCR$ (%)	$UACI$ (%)	HD (%)
Airplane	$512 \times 512 \times 3$	49.9972	99.6102	33.4659	49.9972
Black	$256 \times 256 \times 1$	50.0143	99.6067	33.4840	50.0143
Bridge	$512 \times 512 \times 1$	50.0060	99.6114	33.4960	50.0060
Cameraman	$256 \times 256 \times 1$	49.9907	99.6068	33.4616	49.9907
Flowers	$256 \times 256 \times 3$	49.9954	99.6016	33.4563	49.9954
Goldhill	$512 \times 512 \times 3$	50.0003	99.6081	33.4460	50.0003
Kiel	$512 \times 512 \times 1$	49.9974	99.6075	33.4658	49.9974
Lena	$512 \times 512 \times 3$	50.0028	99.6092	33.4669	50.0028
Sailboat	$512 \times 512 \times 3$	50.0062	99.6058	33.4837	50.0062
White	$256 \times 256 \times 1$	49.9981	99.6008	33.5139	49.9981

Performance in terms of security analysis

Cryptanalytic Attacks: ordered, for an attacker, from the hardest type to the easiest:

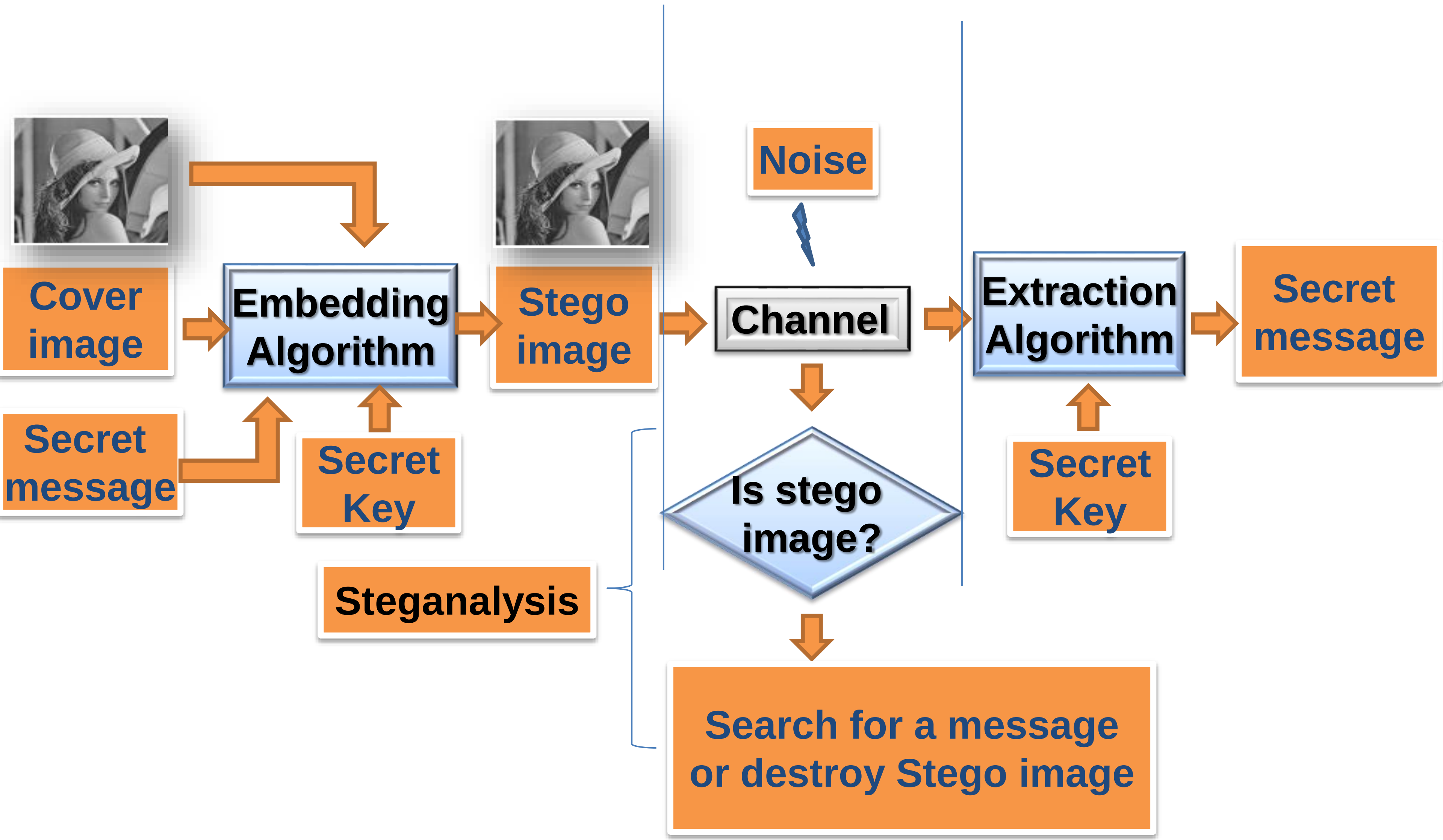
1) **Ciphertext only:** the attacker has the ciphertext of several messages.
2) **Known plaintext attack:** the attacker has access to the ciphertext of several messages and their corresponding plaintext.
3) **Chosen plaintext attack:** the attacker has obtained temporary access to the encryption machinery, and then he can choose a specific plaintext to encrypt and obtain the corresponding ciphertext.
4) **Chosen ciphertext attack**: the attacker has obtained temporary access to the decryption machinery, and then he can choose a specific ciphertext to decrypt and obtain the corresponding plaintext.

If a cryptosystem is able to resist chosen plaintext attack, then it is also resistant to all the other attacks. It is computationally secure

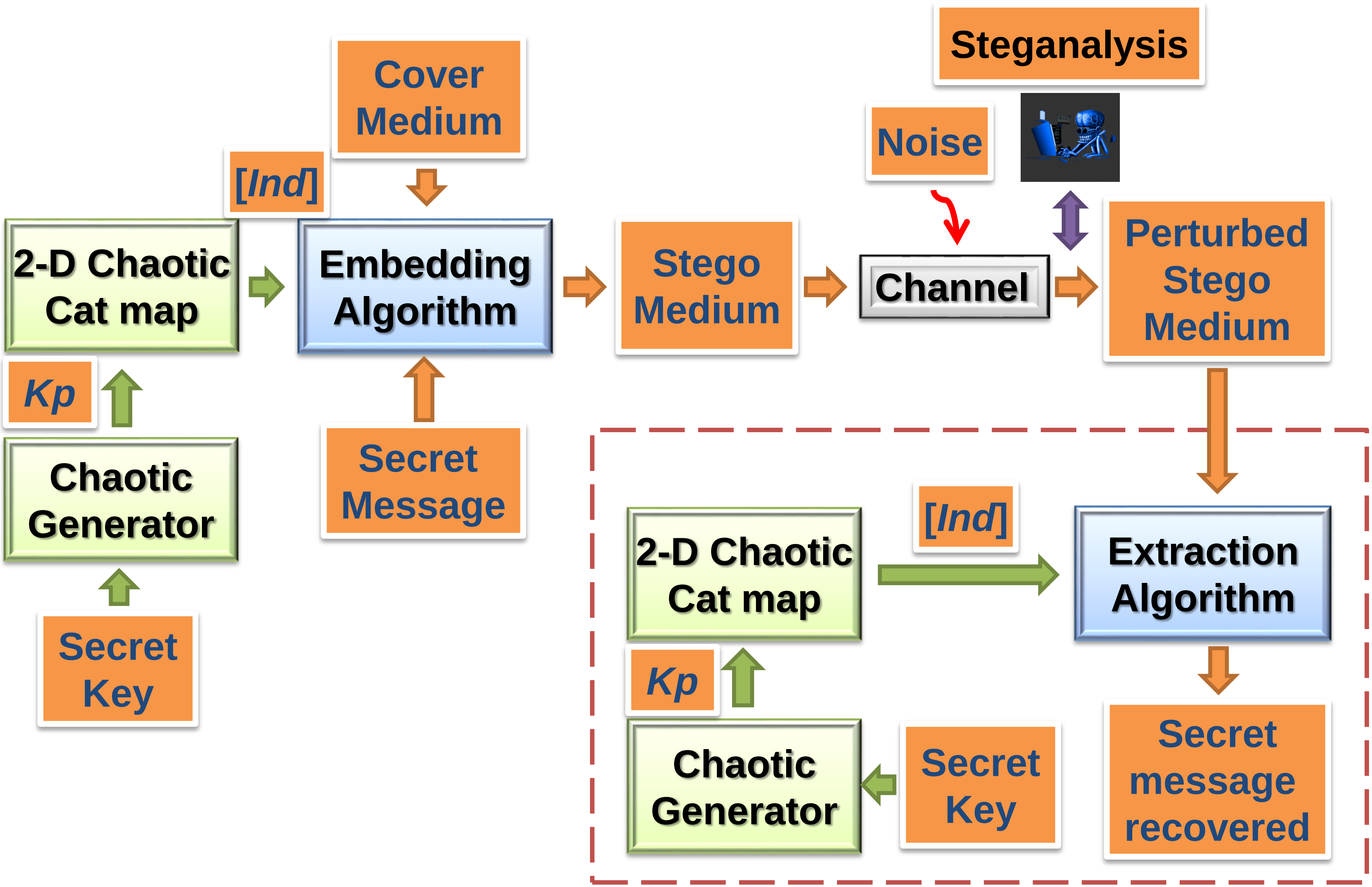
❑ Chaos-based steganography systems

- ❑ Principle of data hiding in spatial LSB domain
- ❑ Structure of the proposed chaos-based steganography system
- ❑ Enhanced Adaptive data hiding in Edge areas of images with spatial Low Significant Bit domain systems : EAE-LSB
- ❑ Enhanced Edge Adaptive Image Steganography Based on LSB Matching Revisited : EEA-LSBMR
- ❑ Comparative performances

Principle of data hiding in spatial LSB domain



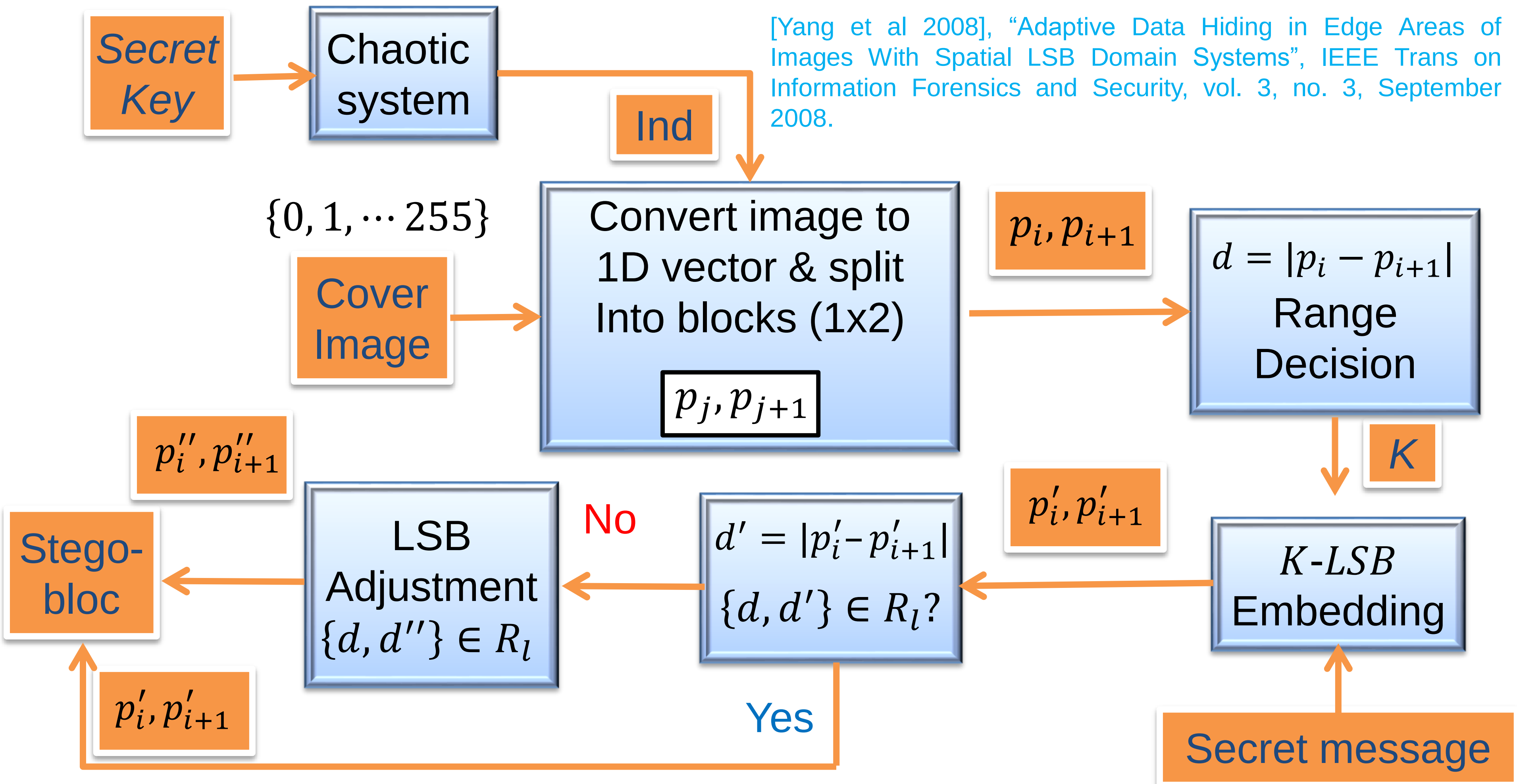
Structure of the proposed chaos-based steganography system



EAE-LSB

[Battikh et al 2014], "Chaos-based spatial steganography system for images", International Journal of Chaotic Computing (IJCC), Volume 3, Issue 1, June 2014/201

[Yang et al 2008], "Adaptive Data Hiding in Edge Areas of Images With Spatial LSB Domain Systems", IEEE Trans on Information Forensics and Security, vol. 3, no. 3, September 2008.



Low Level: $K = 3$	Middle Level: $K = 4$	High Level: $K = 5$
$R_1 = [0, 15]$	$R_2 = [16, 31]$	$R_3 = [32, 255]$

0 ← Range → 255

EAE-LSB : Adaptive Embedding process

- Rearrange the image as a row vector V by raster scanning, and then divide V into non overlapping 2-pixel blocks: (p_j, p_{j+1})
- Select in a chaotic manner a block (p_i, p_{i+1}) from 1D V blocks vector
- Compute block difference $d = |pi - p_{i+1}|$, find its corresponding range R_l and identify K:

$R_1 = [0, 15] => K = 3$; $R_2 = [16, 31] => K = 4$; $R_3 = [32, 255] => K = 5$

- Hide $2K$ bits message in every block using K-LSB insertion => (p_i', p_{i+1}')
- Compute block difference $d' = |p_i' - p_{i+1}'|$, and test if $\{d, d'\}$ are in the same range R_l.
- If yes, than Stego-block => (p_i', p_{i+1}') is carrying the secret message.
- Else, apply the LSB adjustment process => Stego-block => (p_i'', p_{i+1}'')

EAE-LSB : LSB adjustment process

Input: (p'_i, p'_{i+1}), (p_i, p_{i+1}); Output: (p''_i, p''_{i+1}) $d \in R_l$, $d' \in R_t$, $l \neq t$

If $(d < d')$

if $(p'_i \geq p'_{i+1})$

$$(p''_i, p''_{i+1}) = Best\_Choice\_Of\ \{(p'_i, p'_{i+1} + 2^K), (p'_i - 2^K, p'_{i+1})\}$$

else

$$(p''_i, p''_{i+1}) = Best\_Choice\_Of\ \{(p'_i, p'_{i+1} - 2^K), (p'_i + 2^K, p'_{i+1})\}$$

Else $(d > d')$

if $(p'_i \geq p'_{i+1})$

$$(p''_i, p''_{i+1}) = Best\_Choice\_Of\ \{(p'_i, p'_{i+1} - 2), (p'_i + 2^K, p'_{i+1})\}$$

else

$$(p''_i, p''_{i+1}) = Best\_Choice\_Of\ \{(p'_i, p'_{i+1} + 2^K), (p'_i - 2^K, p'_{i+1})\}$$

End

$$Best\_Choice\_Of = MSE\ \{(p_i, p_{i+1}), (p''_i, p''_{i+1})\}$$

$$MSE = \{(pi - p''_i)^2 + (p_{i+1} - p''_{i+1})^2\}$$

MSE: Mean Squared Error

Example of Embedding and Adjustment process

Secret bits : 1011 0110

p_i d p_{i+1}

$d = |pi - p_{i+1}| = |149 - 173| = 24$

$d \in R_2 => K = 4$

1001 0101 149

173 1010 1101

p_i' d' p_{i+1}'

$d' = |p_i' - p_{i+1}'| = |155 - 166| = 11$

$d' \in R_1$

1001 **1011** 155

166 1010 **0110**

$d > d' \Rightarrow$ Adjustment process

Case: $p_i' \leq p_{i+1}' = 166$

$$(p_i'', p_{i+1}'') = Best\_Choice\_Of\ \{(p_i', p_{i+1}' + 2^K), (p_i' - 2^K, p_{i+1}')\}$$

$$(p_i'', p_{i+1}'') = Best\_Choice\_Of\ \{(155{,}182), (139{,}166)\}$$

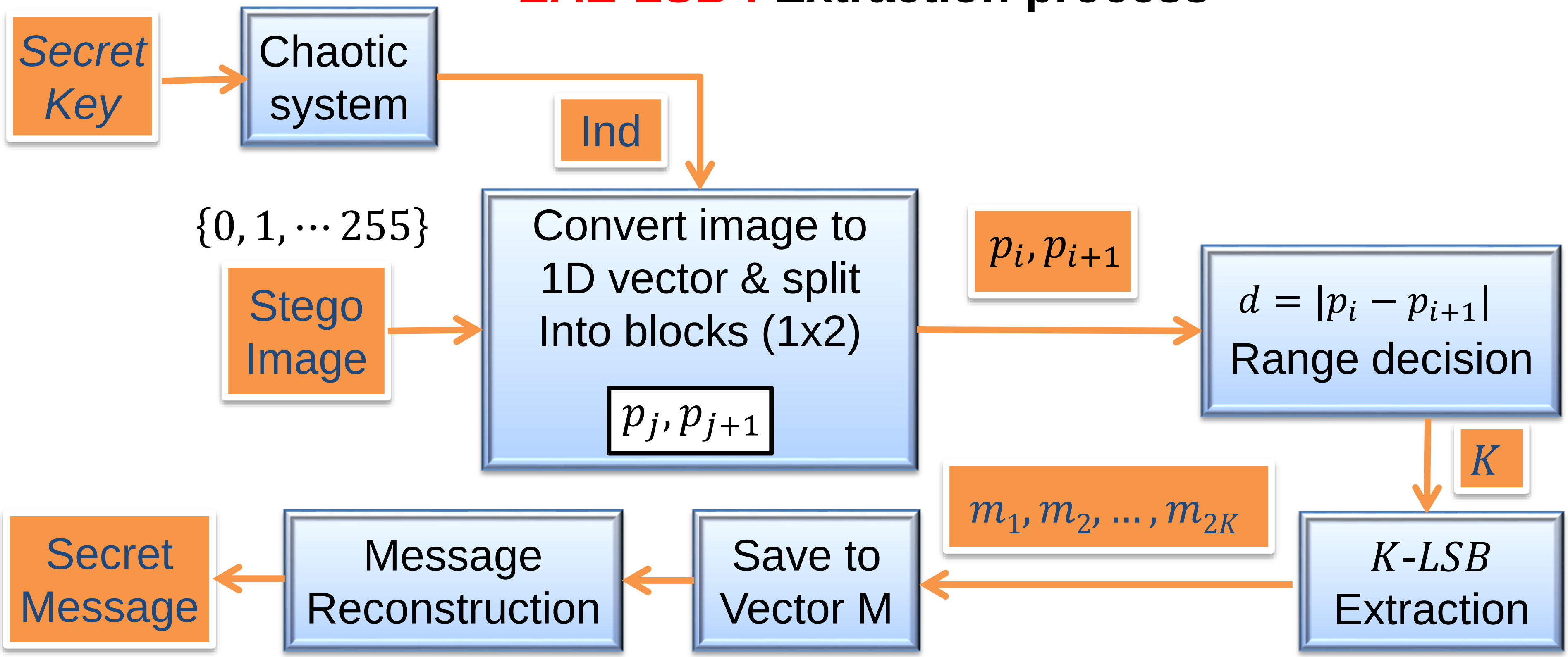
$$MSE = \{(p_i - p_i'')^2 + (p_{i+1} - p_{i+1}'')^2\}$$

$MSE1$: $(149 - 155)^2 + (173 - 182)^2 = 117$

$MSE2$: $(149 - 139)^2 + (173 - 166)^2 = 149$

$p_i'' = 155$ $p_{i+1}'' = 182$ $Best\_Choice$

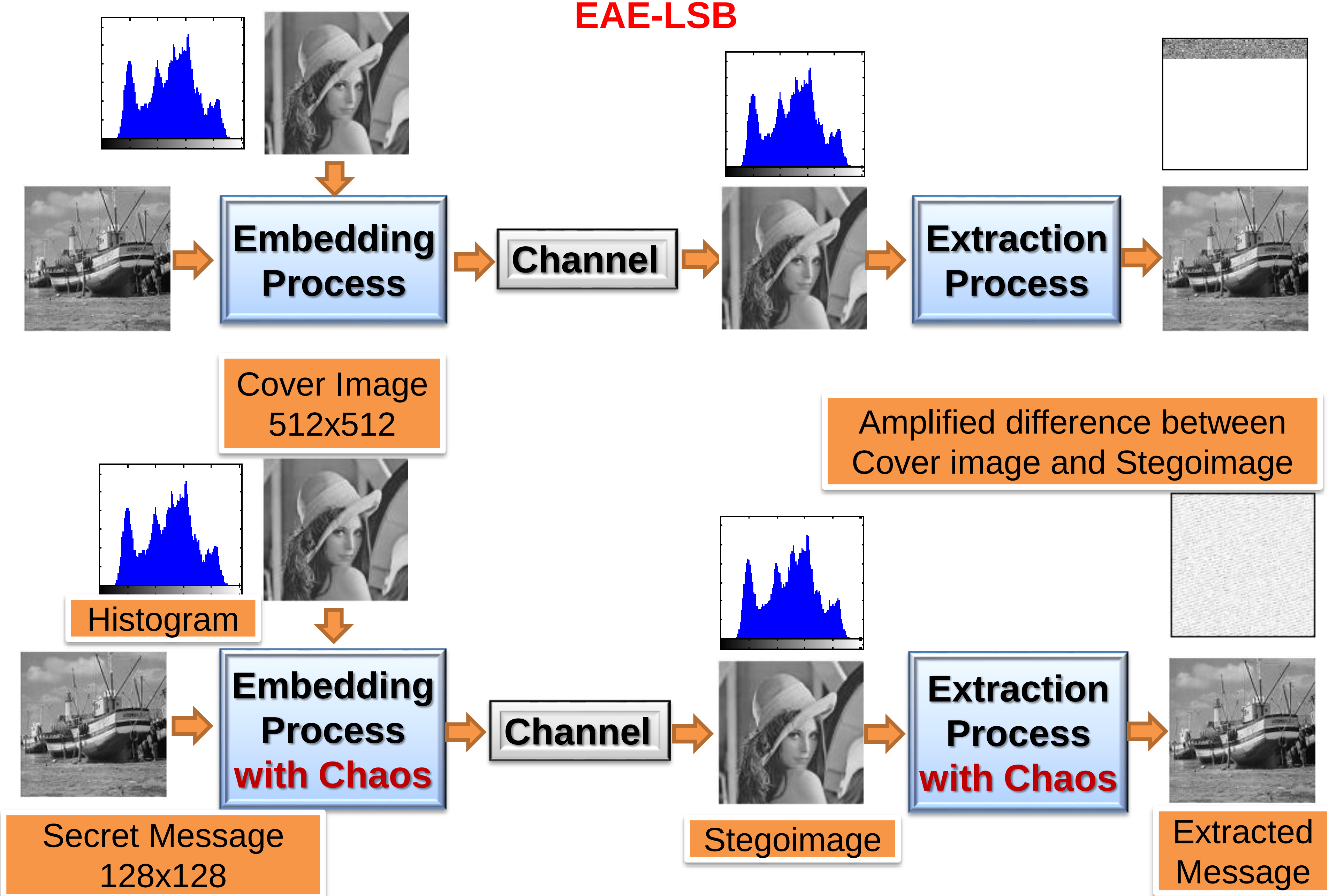
EAE-LSB : Extraction process



- Divide stego image in 2-pixel blocks as for insertion
- Select block chaotically (p_i, p_{i+1}) as for insertion
- Compute block difference d and identify K-$LSBs$ for the corresponding range
- Extract K-LSB secret bits from p_i, K-LSB secret bits from $p_{i+1}$and add to message vector M
- Reconstruct secret message from $2K$ bits sequence groups of M

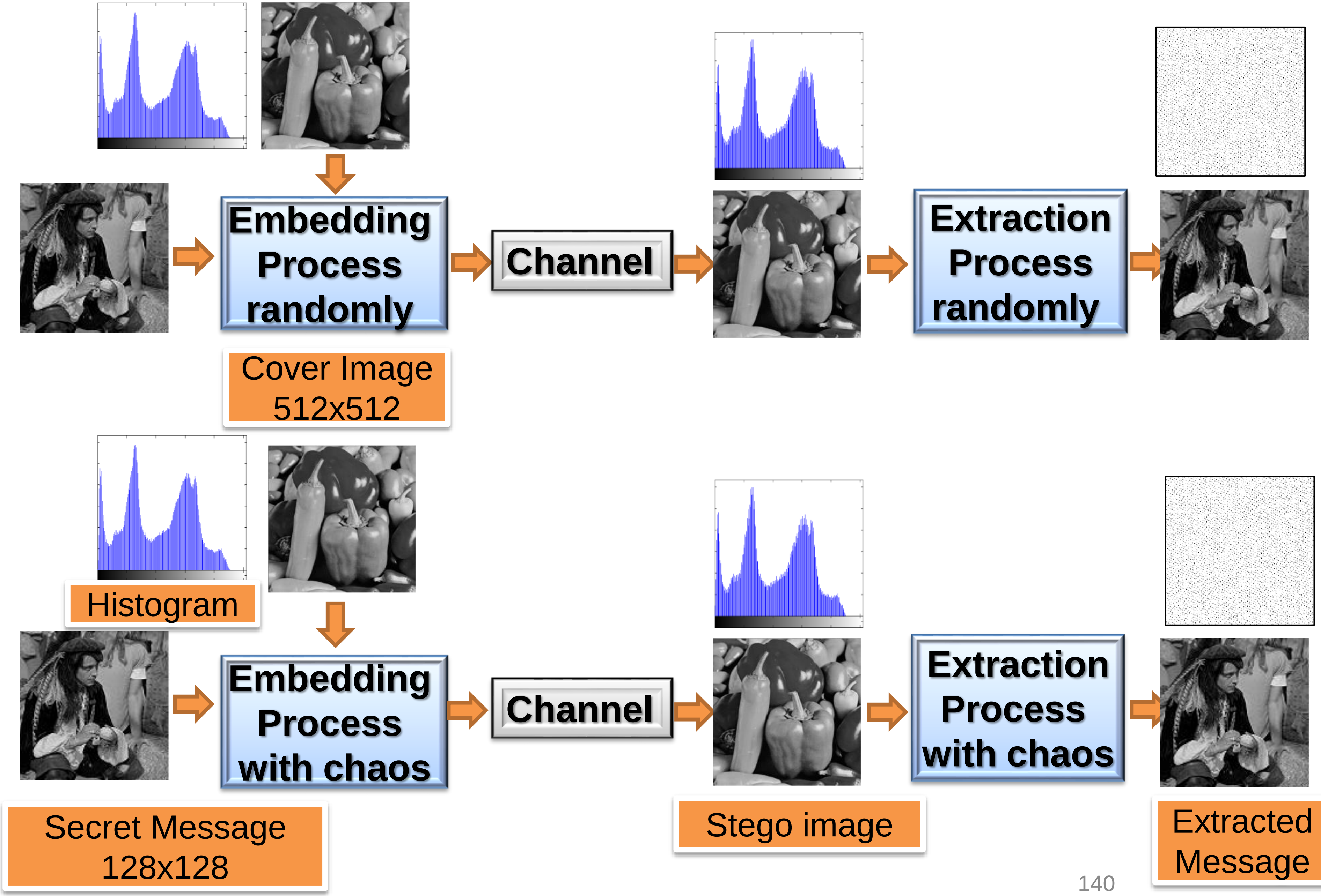
Experimental results : Embedding-Extraction without and with chaos

EAE-LSB



Experimental results : Embedding-Extraction without and with chaos

EEA-LSBMR



Experimental Results

- Performances in terms of secret message capacity and image quality

$$PSNR = 10 \times \log_{10}\left(\frac{M\text{ax}\, I^2(i,j)}{\frac{1}{M \times N}\left(\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left[I(i,j) - I_s(i,j)\right]^2\right)}\right)$$

Cover C	Message M	PSNR EAE-LSB	PSNR EEA-LSBMR
Lena (512x512)	32x32	60.03	70.35
	64x64	54.42	64.41
	100x100	50.33	60.51
	128x128	48.32	58.35
	256x256	42.49	--
Baboon (512x512)	32x32	57.55	70.52
	64x64	51.27	64.46
	100x100	47.19	60.59
	128x128	45.20	58.41
	256x256	39.40	--
Peppers (512x512)	32x32	59.43	69.71
	64x64	54.52	63.78
	100x100	50.25	59.86
	128x128	48.04	57,70
	256x256	42.42	--

Thanks for your Attention

I hope this lecture was clear and useful for you

There are any questions?

- **Appendix**
  - **Various block cipher modes: Symmetric key algorithms**
  - **Error Propagation**

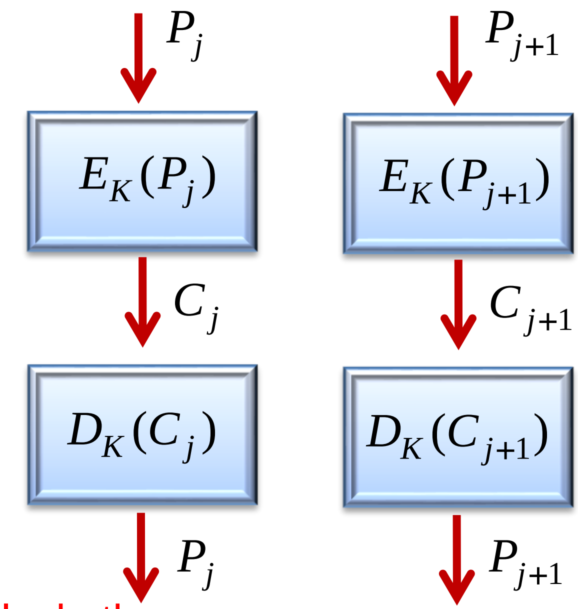
Various block cipher modes: Symmetric key algorithms

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

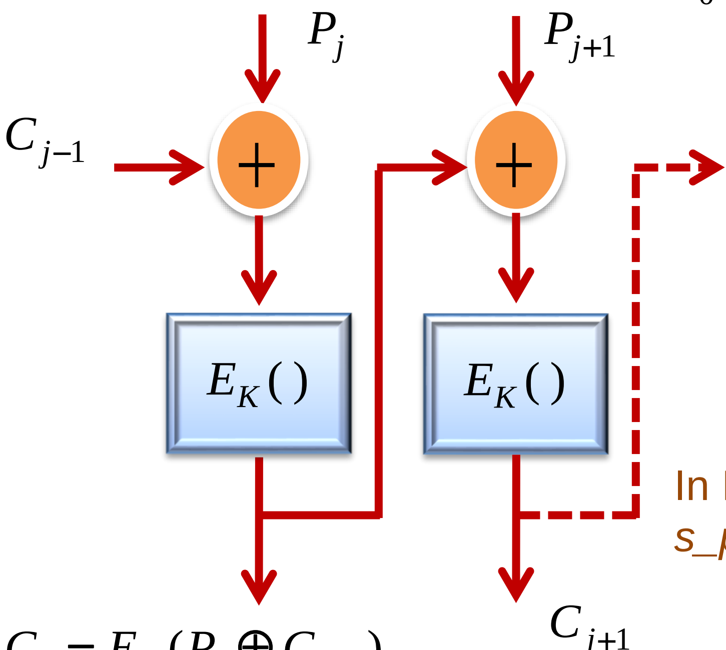
Disadvantages :

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



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

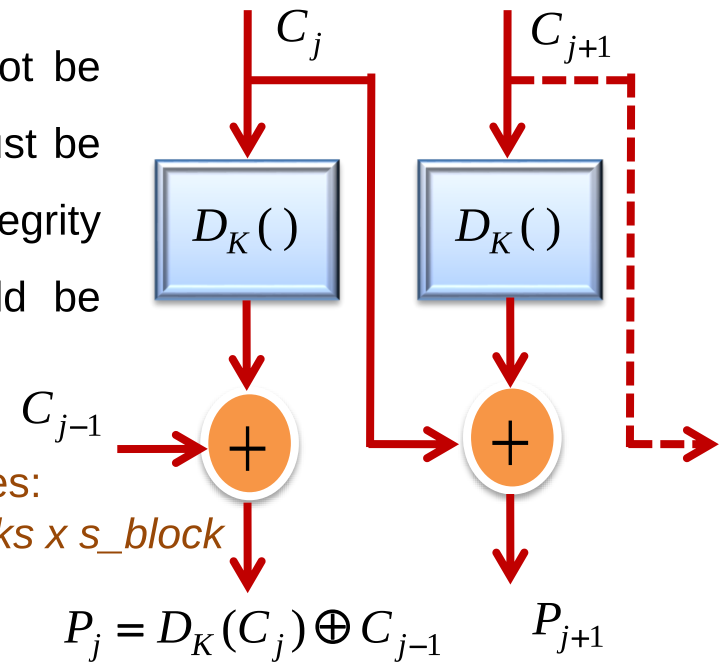
$$j = 1, \cdots, n\_blocks$$

$$C_0 = IV : Initialization\ Vector$$

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 :

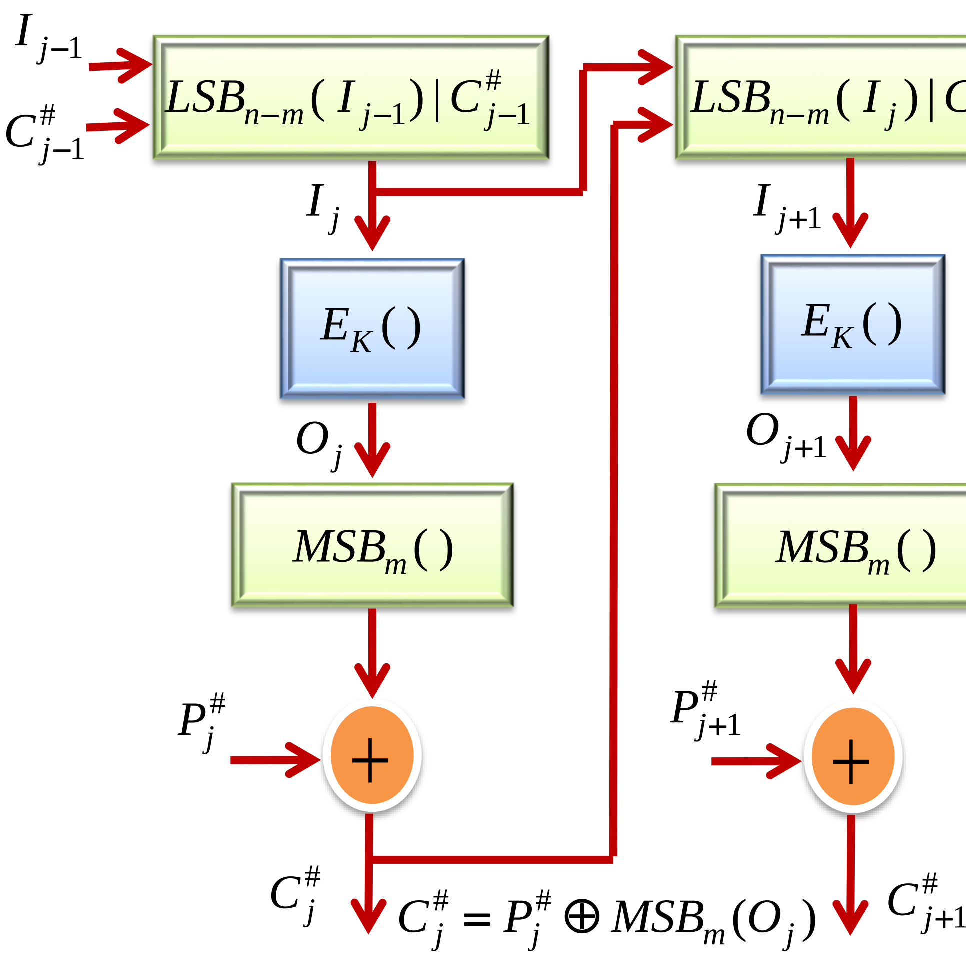
- Error Propagation: a single bit error in the ciphertext affects one block and one bit of the recovred plaintext.
- Encryption of blocks can't be performed in parallel, but decryption can be performed in parallel.
- Block structure must remains intact : if a bit is added or lost from the ciphertext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

CFB (Cipher Feedback) mode

Encryption process

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

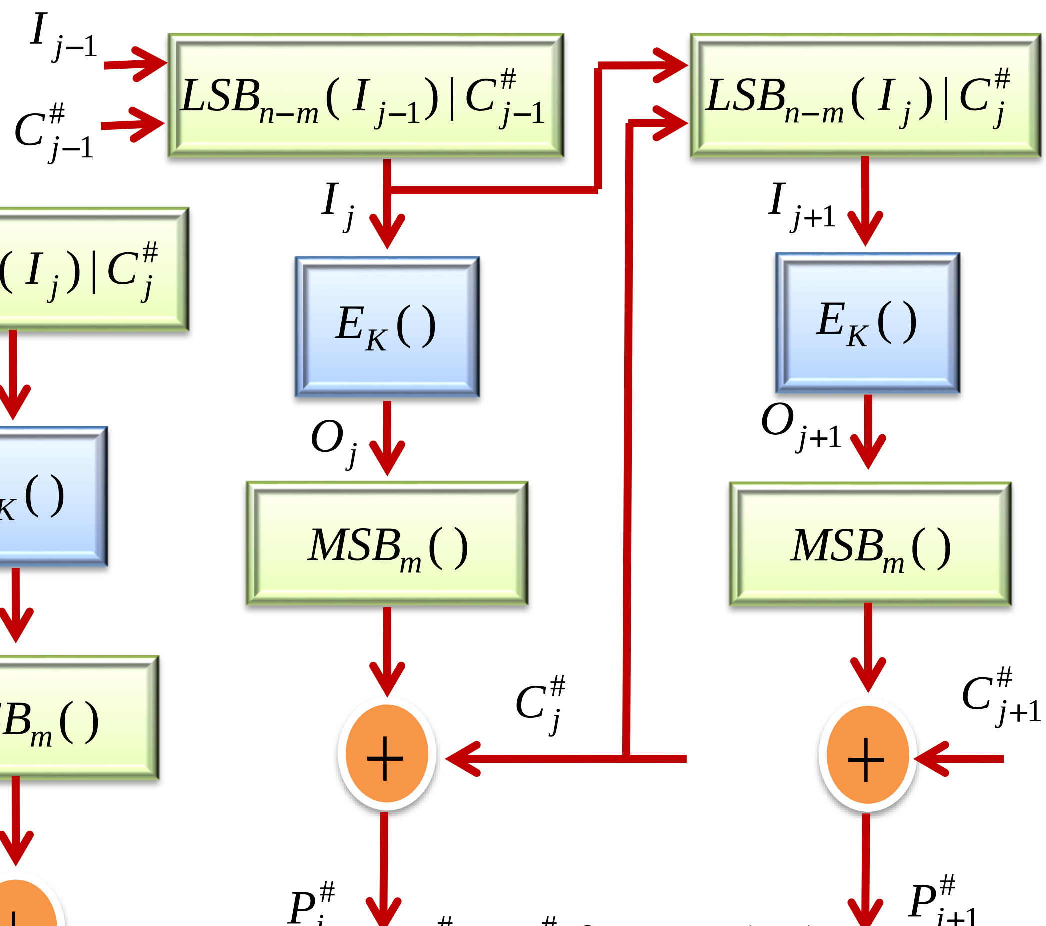
Size of I_j, O_j is *s_block* Size of $P^{\#}_j$, $C^{\#}_j$ *is m*



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

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.

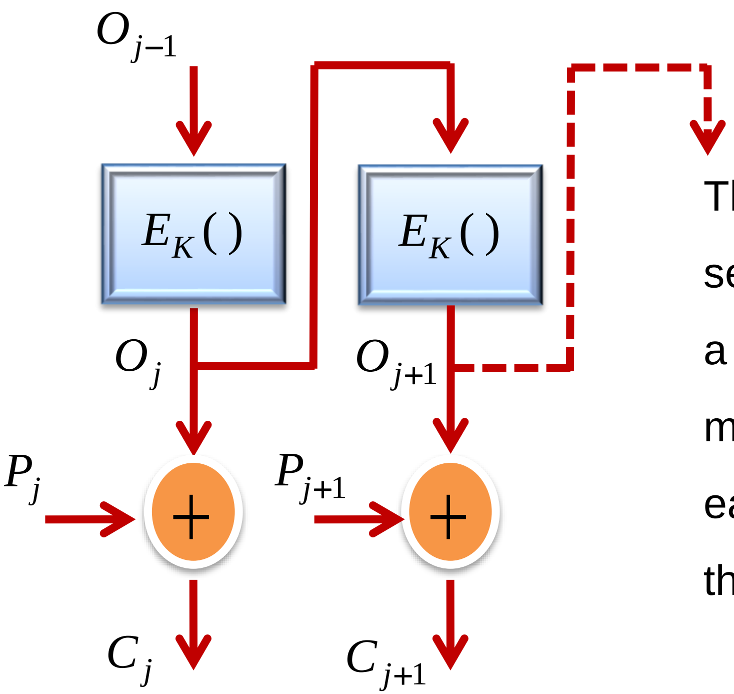
- Unlike CBC mode, in CFB mode, data can be encrypted in units *m* bits smaller than the block size *s-block*. This mean that it is not necessary to receive a complete block of data to begin the encryption process.
- It can be implemented as a self-synchronization stream cipher.

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



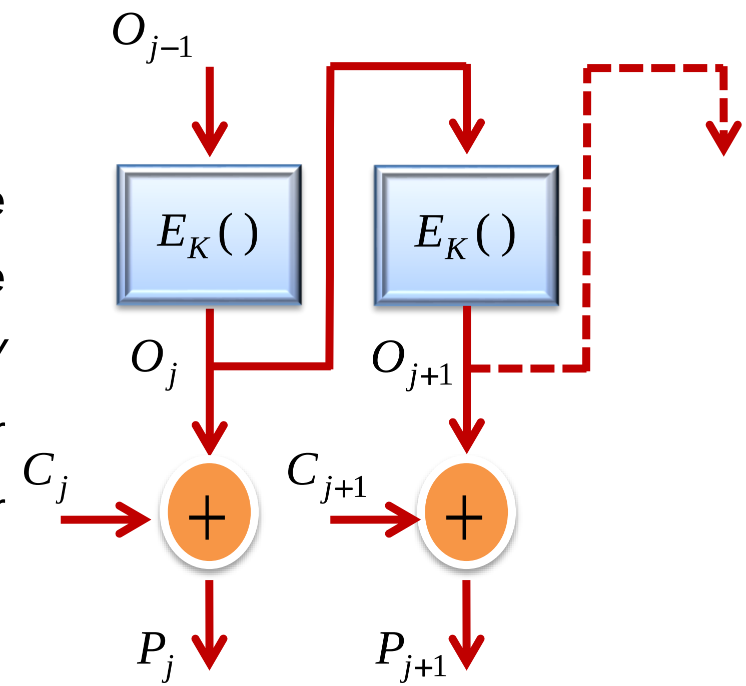
In OFB & CTR modes:

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.

Decryption process



$$C_j = P_j \oplus O_j$$

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

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

$$O_0 = IV$$

$$P_j = C_j \oplus O_j$$

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

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

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

OFB (Output Feedback) mode

Advantages :

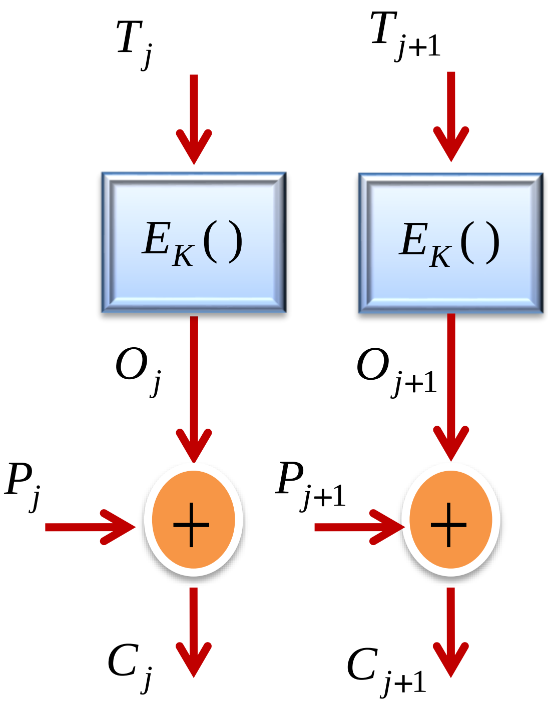
- Outputs O_j can be generated offline, before the plaintext or ciphertext data exists.
- It can be implemented as a synchronous stream cipher.
- Error Propagation : OFB mode has no error extension. A single bit error in the ciphertext causes a single bit in the recovered plaintext. This is useful for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

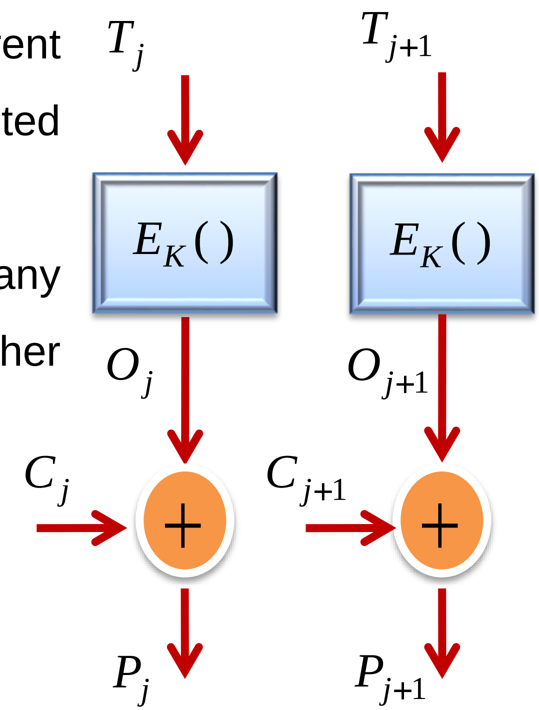
Encryption process



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



$$C_j = P_j \oplus O_j$$
$$j = 1,\cdots, n\_blocks - 1$$

$$O_j = E_K(T_j)$$
$$j = 1,\cdots, n\_blocks$$

$$P_j = C_j \oplus O_j$$
$$j = 1,\cdots, n\_blocks - 1$$

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

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

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 $P_1^{\#}, P_2^{\#}, \cdots, P_j^{\#}$ $\textit{for some } 1 \le j \le s\_block / m$
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. } $P_{e,c} \; Q_0 \; (1-P_{inv})$

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

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

Error Propagation

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

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 $$P_{e,d} = P_{e,c}$$

References

[Dworkin, 2001], “Recommendation for Block Cipher Modes of Operation; Methods and Techniques”. NIST Special Publication 800-38A, 2001 Edition.

[El Assad et al., 2008], “Design and analyses of efficient chaotic generators for cryptosystems" . WCECS, pp. 3-12, 2008, Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science, 2008.

[El Assad et Noura, 2011], “Generator of chaotic Sequences and corresponding generating system” WO Patent WO/2011/121,218,2011. PCT Extension: Europe : EP-2553567 B1, Sept 3, 2014 United States: US-8781116 B2, July 15, 2014. JP 5876032 B2, Mars 02, 2016
China : CN-103124955 B, April 20, 2016.

[Abu Taha et al., 2017], “Design and Efficient Implementation of a Chaos-based Stream Cipher”, International Journal of Internet Technology and Secured Transactions, Vol. 7, No. 2, Sept 2017, pp.89–114.

[Jallouli, et al., 2017], “Design and Analysis of two Stream Ciphers Based on Chaotic Coupling and Multiplexing techniques” MTAP, Multimedia Tools and Applications, June 2017, pp. 1-27

[Dridi et al., 2021-a], “Design, FPGA-based Implementation and Performance of a Pseudo-Chaotic Number Generator”, Advances in Electrical and Computer Engineering, Volume 21, Number 2, May 2021, pp. 41-48

[Dridi et al., 2021-b], “The Design and FPGA-Based Implementation of a Stream Cipher Based on a Secure Chaotic Generator”, Appl. Sci. 2021, 11, 625.

[Fridrich , 1998], “Symmetric ciphers based on two-dimensional chaotic maps”. Int. J. Bifurcat Chaos, vol. 8, no. 6, 1998, pp. 1259-1284.

[Lian et al., 2005a], “A bloc cipher based on a suitable use of the chaotic standard map“. Chaos Solitons and Fractals, vol. 26, 2005, pp. 117-129.

[Farajallah et al., 2013], ”Dynamic adjustment of the chaos-based security in real-time energy harvesting sensors”. IEEE, International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, Beijing, China, August 2013, pp. 282-289.

[El Assad, Farajallah, 2016], “A new Chaos-Based Image Encryption System”. Signal Processing: Image Communication 41, (2016) 144-157.

[Farajallah et al., 2016], “Fast and secure chaos-based cryptosystem for images”., International Journal of Bifurcation and Chaos, IJBC, Vol. 26, No. 2 (2016) 1650021 (21 pages).

[Lian et al., 2005b], “Security analysis of a chaos-based image encryption algorithm”. Physica A vol. 351, 2005, pp. 645-661.

[Lozi, 2012], ”Emergence of randomness from chaos”, International Journal of Bifurcation and Chaos, IJBC, Vol. 22, No. 2 (2012) 1250021 (15 pages).

[Masuda et al., 2006], “Chaotic block ciphers: from theory to practical algorithms”. IEEE Trans on Circuits and Systems-I, vol. 53, no. 6, 2006, pp. 1341–1352.

[Wang et al., 2011], “A new chaos-based fast image encryption algorithm”. Applied Soft Computing, vol. 11, 2011, pp. 514-522.

[Wong et al., 2008], “A fast image encryption scheme based on chaotic standard map“. Physics Letters A, vol. 372, no. 15, 2008, pp. 2645-2652.

[Zhang et al., 2013], “An image encryption scheme using reverse 2-dimentional chaotic map and dependent diffusion”. Commun Nonlinear Simulat, vo. 18, 2013, pp. 2066-2080.

[Wong et al., 2009], “An efficient diffusion approach for chaos-based image encryption”. Chaos Solitons and Fractals vol. 41, 2009, pp. 2652-2663.

[Wang et al., 2009], “A chaos-based image encryption algorithm with variable control parameters”. Chaos Solitons and Fractals vol. 41, 2009, pp. 1773-1783.

[Schneier, 1996], “Applied Cryptography — Protocols, Algorithms, and Source Code”, C. John Wiley & Sons, Inc., New York 2nd edition.

[Paar & Pelzl, 2010], “Understanding Cryptography, Springer edition.

[Stallings, 2014], “Cryptography and Network Security, Principles and Practice, Pearson sixth edition

[Mielikainen 2006], “LSB matching revisited”, IEEE Signal Processing Letters, vol. 13, no. 5, May 2006.

[Yang et al 2008], “Adaptive Data Hiding in Edge Areas of Images With Spatial LSB Domain Systems”, IEEE Trans on Information Forensics and Security, vol. 3, no. 3, September 2008.

Battikh et al 2014], ”Chaos-based spatial steganography system for images”, International Journal of Chaotic Computing (IJCC), Volume 3, Issue 1, June 2014/201

[Luo et al 2010], “Edge Adaptive Image Steganography Based on LSB Matching Revisited”, IEEE Trans on Information Forensics and Security, vol. 5, no. 2, June 2010

[Battikh et al, 2019], “Comparative Study of Three Steganographic Methods Using a Chaotic System and Their Universal Steganalysis Based on Three Feature Vectors”, Entropy 2019, 21, 748; doi:10.3390/e21080748.

[Abdoun et al 2019],” Design and Security Analysis of Two Robust Keyed Hash Functions based on Chaotic Neural Networks”, AIHC-2019, Journal of Ambient Intelligence and Humanized Computing, February 2019, 25 pages https://doi.org/10.1007/s12652-019-01244-y

[Abdoun et al 2020], “Designing Two Secure Keyed Hash Functions Based on Sponge Construction and Chaotic Neural Network”, Entropy 2020, 22, 1012; doi:10.3390/e22091012, (32 pages).

[Abdoun et al 2021], “Authenticated Encryption Based on Chaotic Neural Networks and Duplex Construction”, Symmetry 2021, 13(12), 2432; https://doi.org/10.3390/sym13122432 (23 pages).

[Preneel 2003], “Analysis and Design of Cryptographic Hash Functions”, Ph.D. Thesis, Katholieke Universiteit te Leuven, Leuven, Belgium, 1993.