WARSAW UNIVERSITY OF TECHNOLOGY
DEVELOPMENT PROGRAMME

CRYPSTIC

Wednesday 10th March, 2010: 11:00 -13:00

Cryptography using Chaos

DIT

GCHQ

J M Blackledge

Stokes Professor
Dublin Institute of Technology
http://eleceng.dit.ie/blackledge

Distinguished Professor
Warsaw University of Technology

sfi science foundation ireland fondúireacht eolaíochta éireann

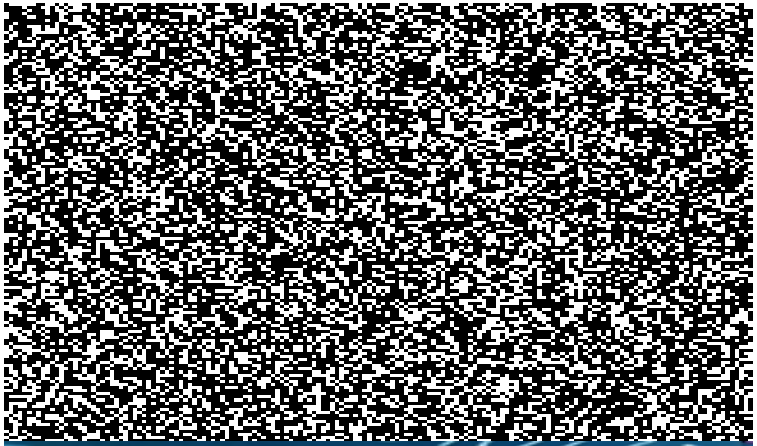
HUMAN CAPITAL
NATIONAL COHESION STRATEGY

LexiconData

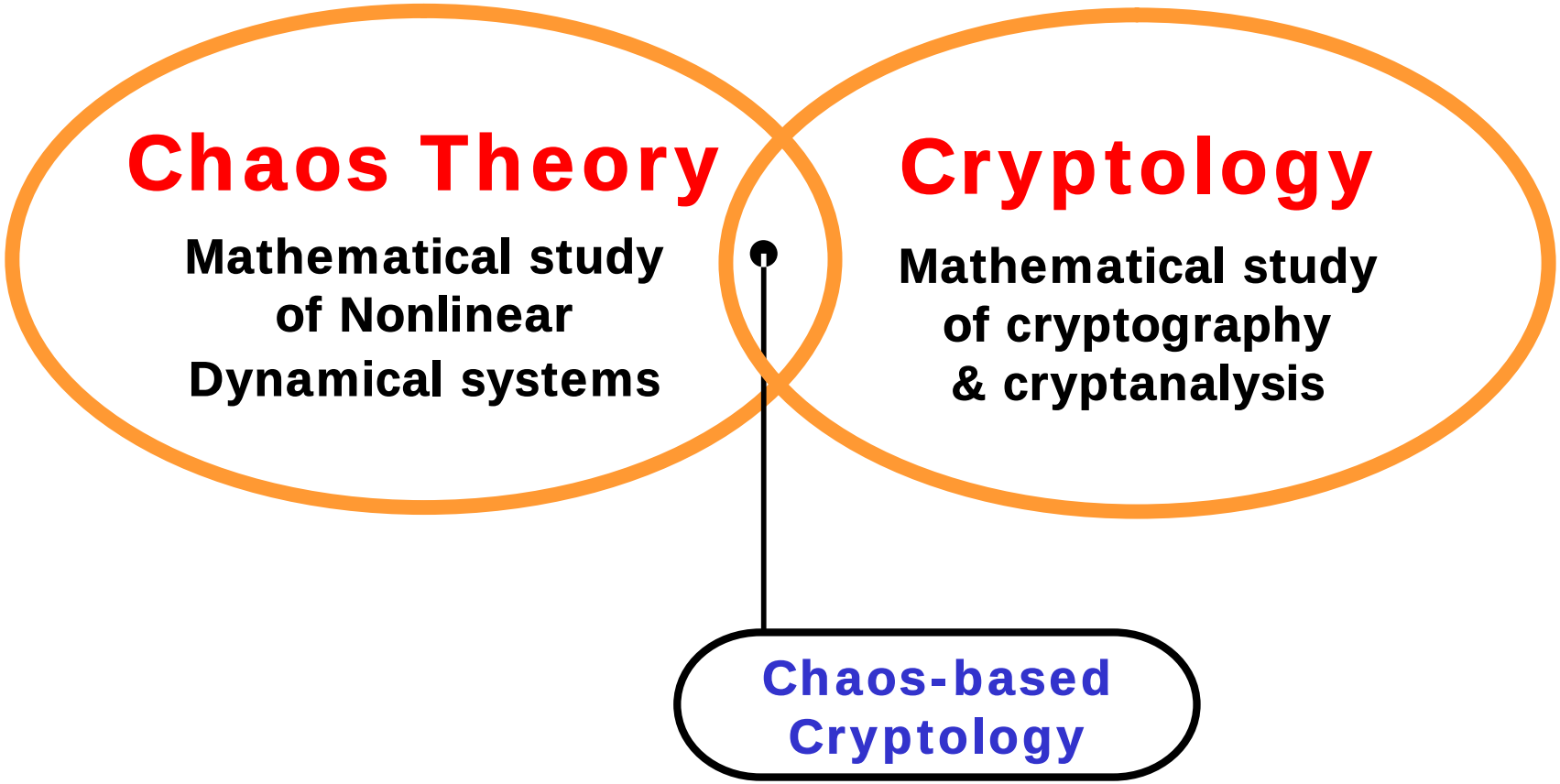
EUROPEAN UNION
EUROPEAN
SOCIAL FUND

Lectures co-financed by the European Union in scope of the European Social Fund

What is the Problem ?



The Concept

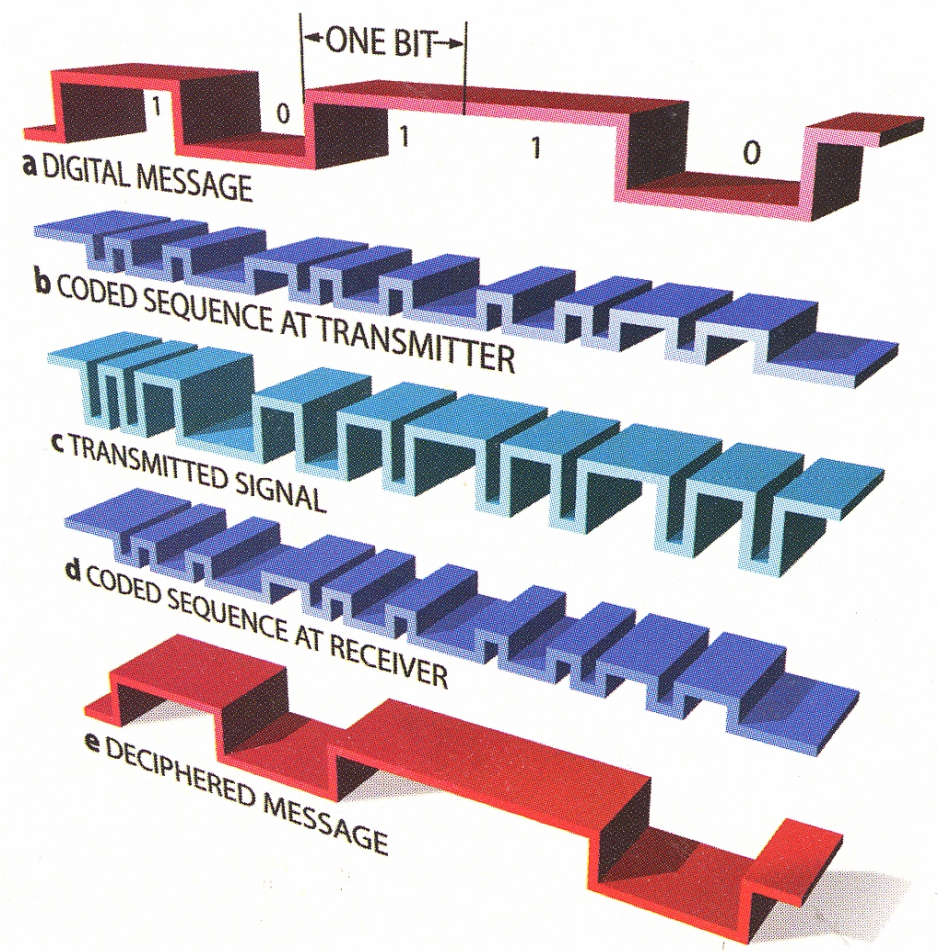

Chaos Theory
Mathematical study of Nonlinear Dynamical systems
Cryptology
Mathematical study of cryptography & cryptanalysis
Chaos-based Cryptology

Why use Chaos ?

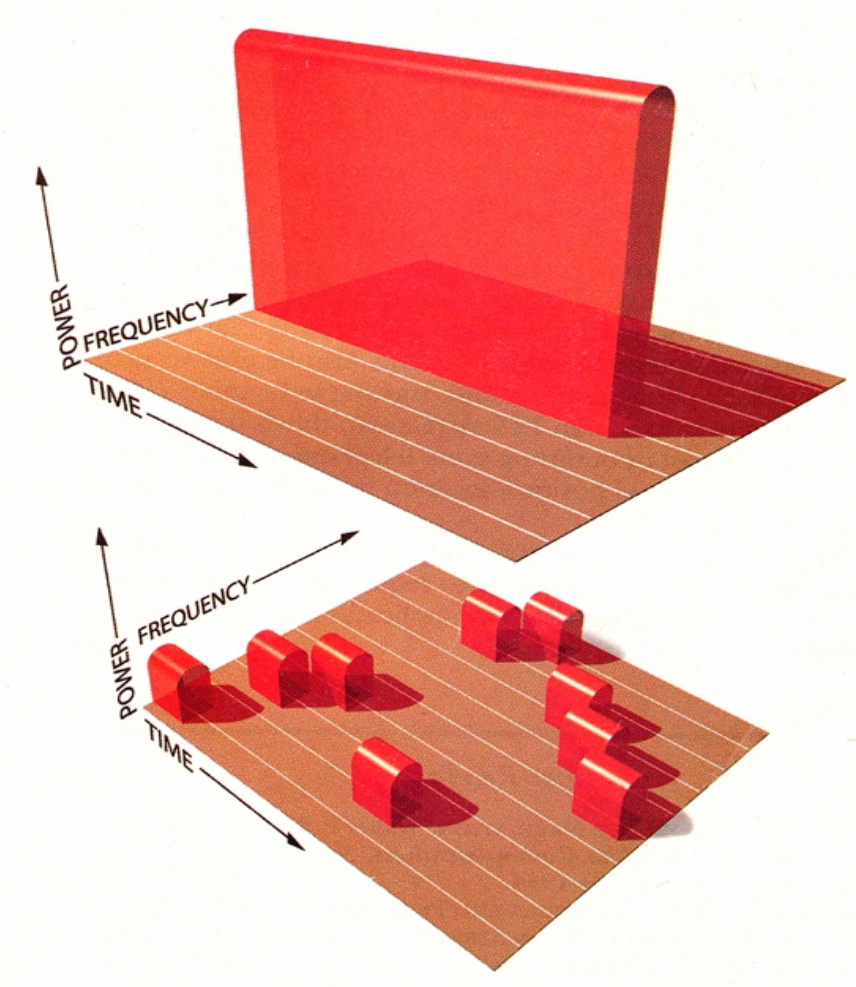
A *Complexity Theoretic Approach* based on *Multi-Algorithmicity*

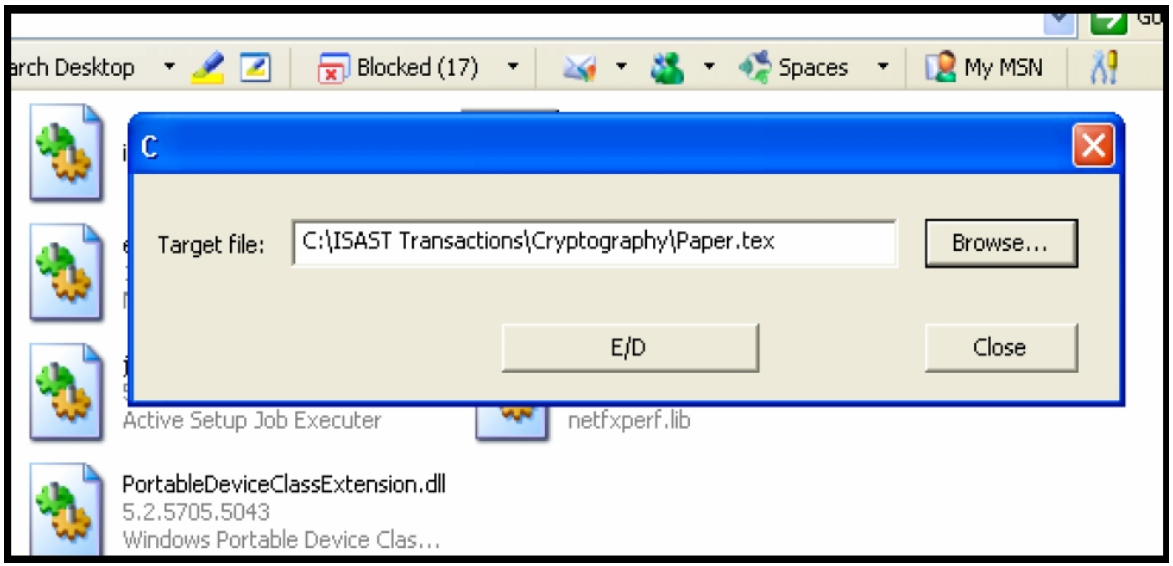
Applications

Stream coding



Spread Spectrum





LexiconData

http://www.lexicon-data.com

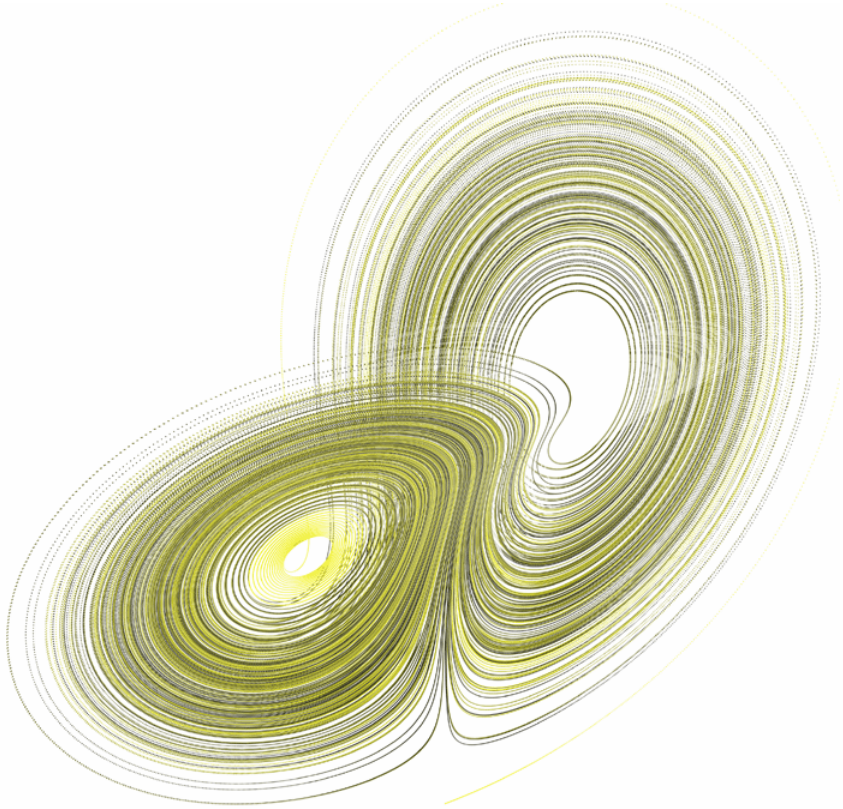
Principal Publication

Multi-algorithmic Cryptography using Deterministic Chaos with Applications to Mobile Communications, J M Blackledge, International Society for Advanced Science & Technology, Transactions on Electronics and Signal Processing, No. 1, Vol. 2,23 - 64, 2008; http://eleceng.dit.ie/papers/107.pdf

Contents of Presentation I

Part I:

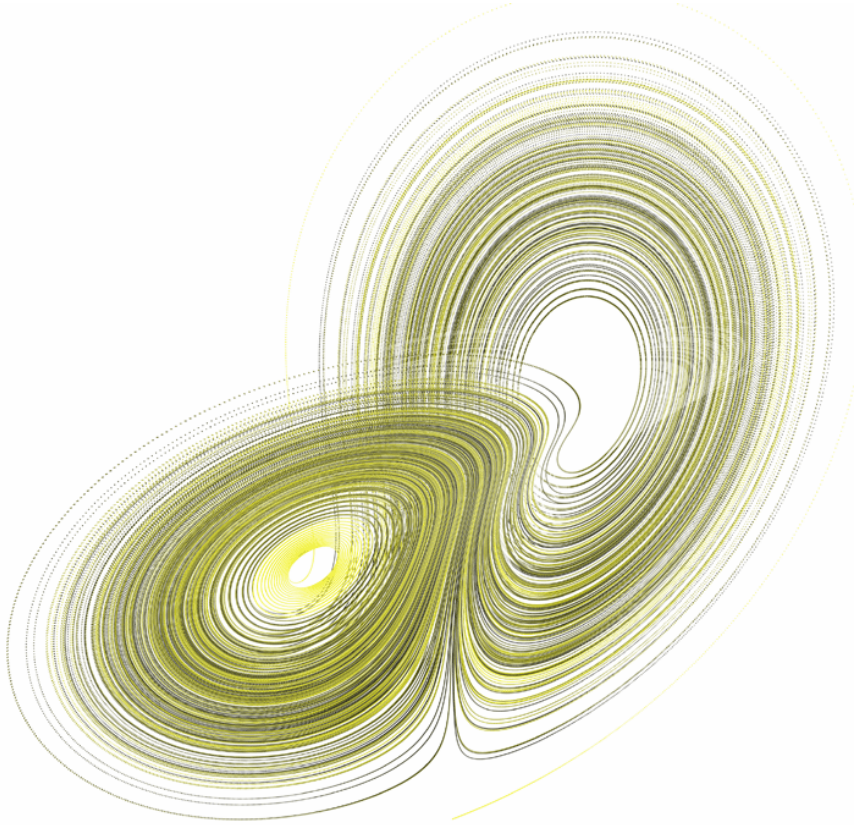
- Basic Concepts in Cryptography
- Substitution Ciphers
- Principal Conditions
- Example Algorithms
- Diffusion and Confusion
- Kerchhoff-Shannon Principle
- Summary
- Interval (10 Minutes)



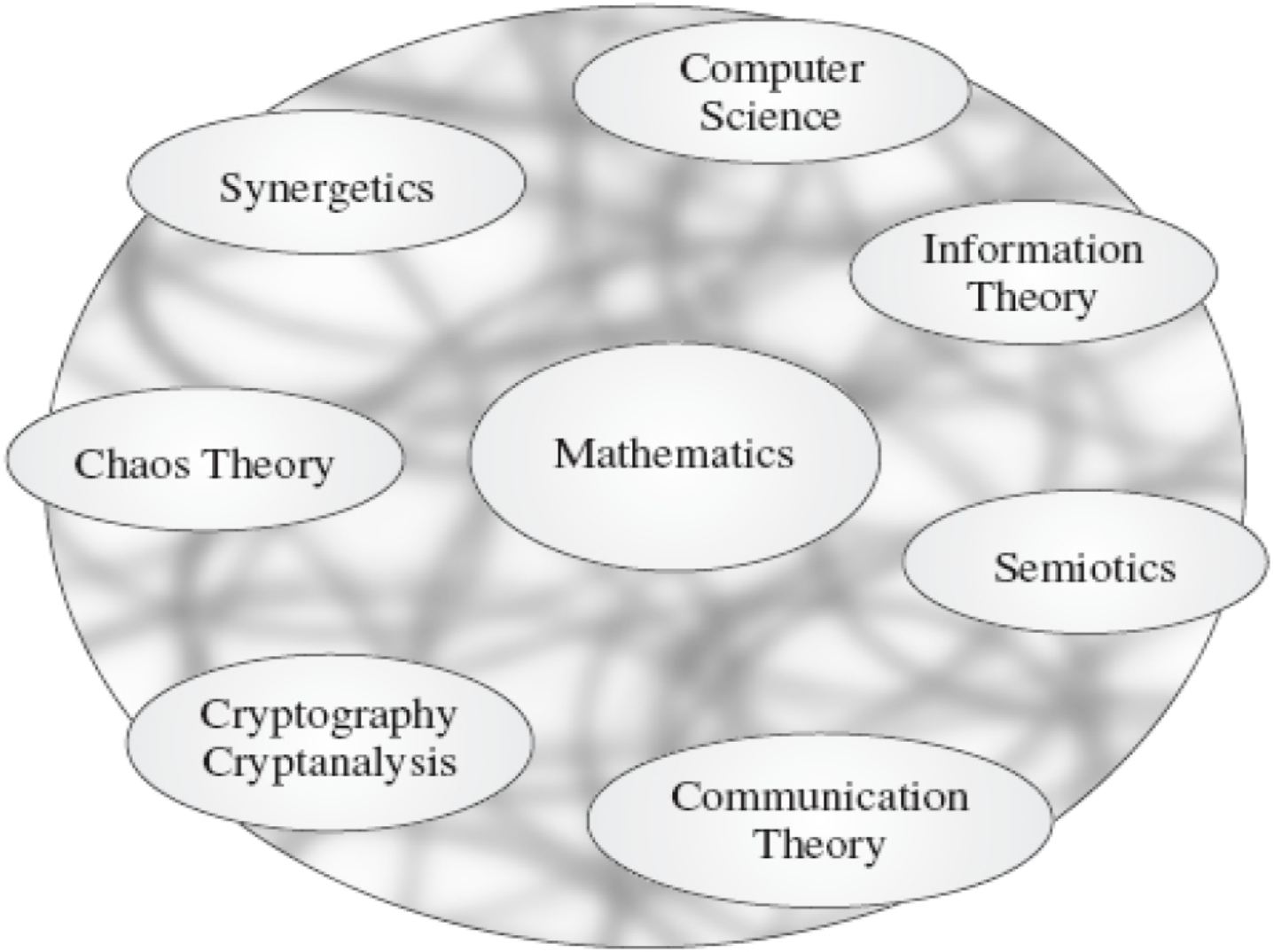
Contents of Presentation II

Part II:

- Multi-algorithmicity
- Designing Chaotic Algorithms
- Software Development
- Applications
- ***Crypstic***
- Demonstration
- Research Project Proposal
- Questions

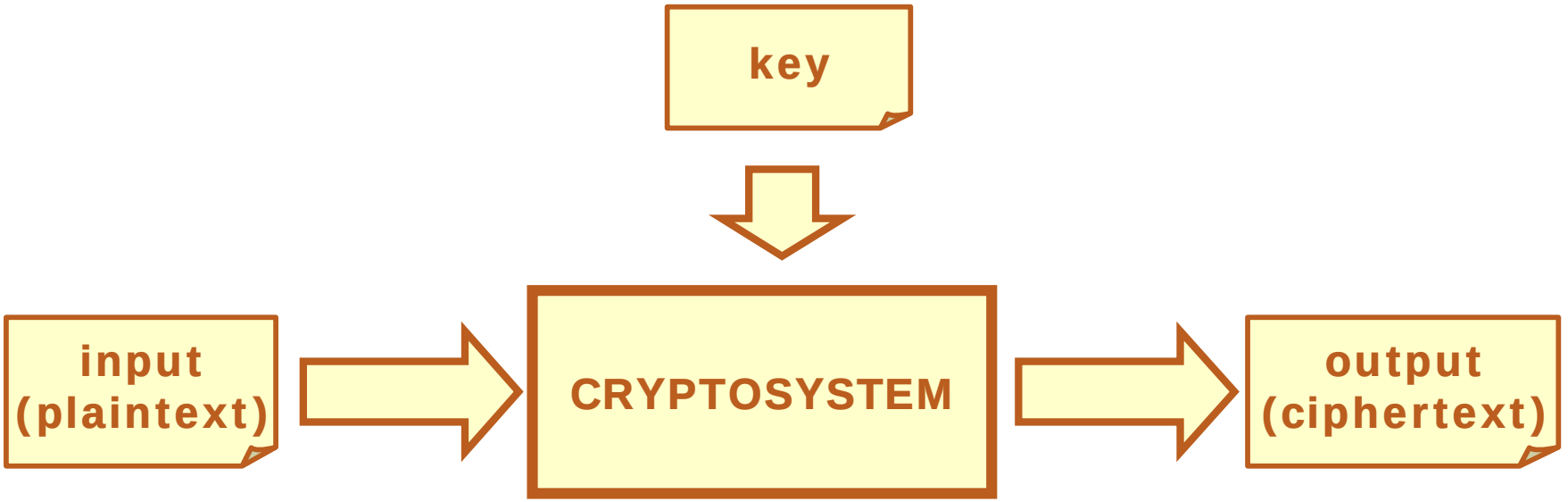


Cryptology:
Contributing Subject Areas



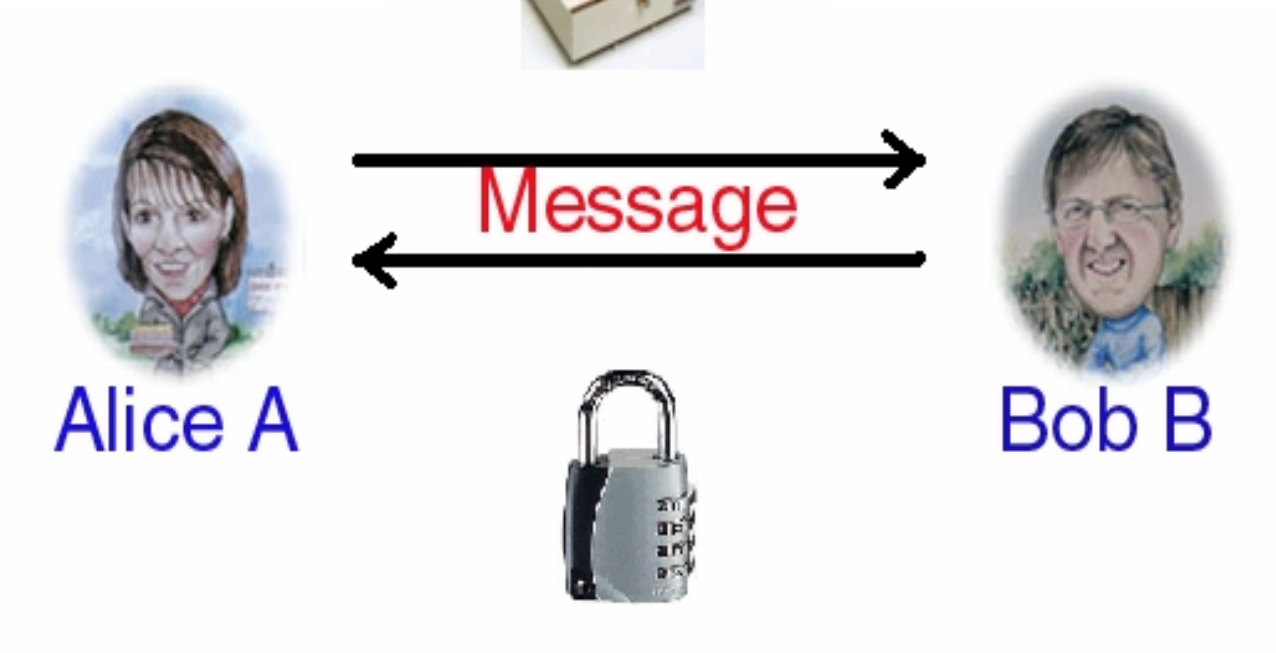
What is a Cryptosystem?

A cryptosystem is a computer program transforming information in a key-dependent and apparently unpredictable manner



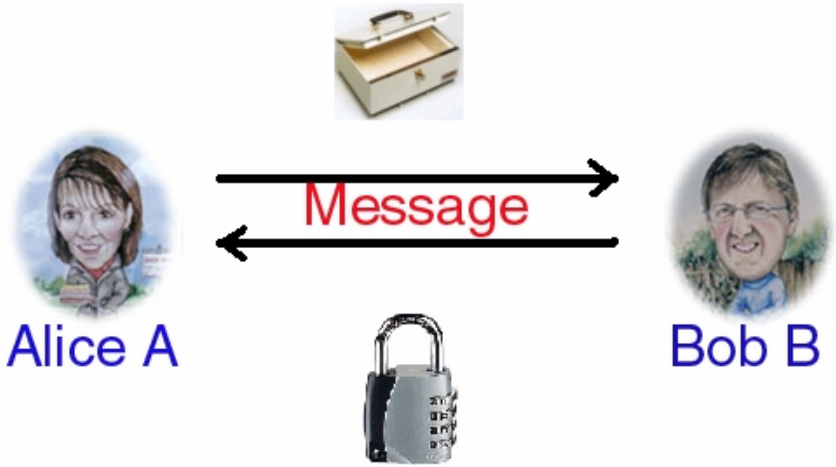
Basic Concepts in Cryptography

$$C = E_K(P) \qquad P = D_K(C)$$



- Box strength : strength of **Encryptor E/D**
- Combination # : strength of **Key K (length of #)**

Symmetric Encryption



- A & B agree on combination # *a priori*
- A & B undertake the same lock/unlock process – a *symmetric* process
- **Vulnerable to attack if interceptor obtains combination # when A & B agree upon it**
- **Problem: How should A & B exchange the key?**

Multiple Encryption

- Uses many locks or **Keys K_n**

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

$$P = D_{K_1}(D_{K_2}(D_{K_2}(C)))$$

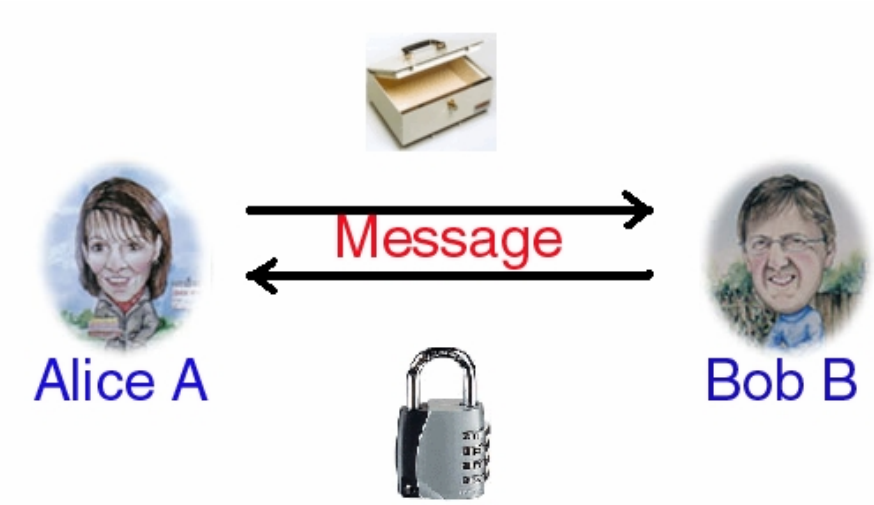
- Based on application of the same encryption/decryption algorithm ***E/D***

- Used to increase effective key length, e.g. ***Digital Encryption Standard 3 (DES3)***

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

Asymmetric Encryption

- A sends B an open lock with combination known only to A.



- B secures box with lock & sends box (with message) back to A – an *asymmetric* process

- **A is vulnerable to receiving disinformation if open lock is intercepted**

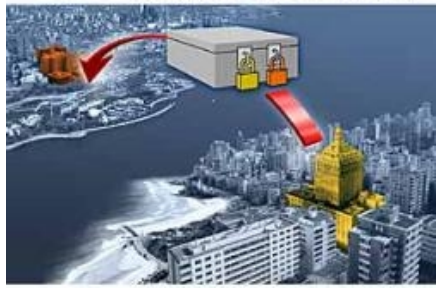
- **Problem: How can A authenticate the message from B?**

Three-Way-Pass Protocol

- A locks box with combination # known only to A and sends it to B

- B locks box with another lock and a combination # known only to B and sends it back to A

- A (partially) unlocks box and sends it back to B

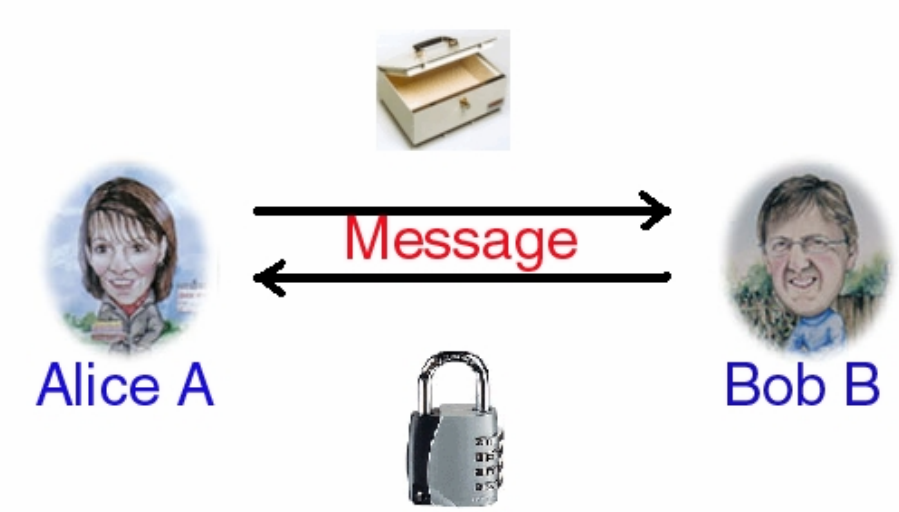


- B (completely) unlocks box to recover message

Protocol is vulnerable to 3-pass interception

Public/Private Key Encryption

- A locks box with a public combination # unique to B - a public key
- Some ‘property’ of this public key is known only to B



- This ‘property’ (the *private key)* allows B to unlock the box
- Vulnerability of method depends on the ‘property’ which depends on the design details of the lock

Principal issues in Cryptography

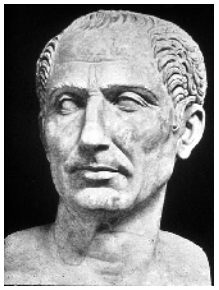
- Cryptographic systems should be designed with respect to three components:

  - cyphertext generation

  - key exchange

  - authenticity

- Each component tends to rely on separate and distinct methods of approach



A Simple Cipher: The Caesar Cipher

In his book *The Life of Julius Caesar*, Seutonius states that:

If he had anything confidential to say, he wrote it in cipher, that is, by so changing the order of the letters of the alphabet that not a word could be made out. If anyone wishes to decipher these, and get at their meaning, he must substitute the fourth letter of the alphabet, namely D, for A, and so with the others.

```
   ABCDEFGHIJKLMNOPQRSTUVWXYZ
K: DEFGHIJKLMNOPQRSTUVWXYZABC

P: HAIL CAESAR DICTATOR FOR LIFE
C: KDLO FDHVDU GLFWDWRU IRU OLIH
```

Computing the Caesar Cipher using Modular Arithmetic

- Let $A=0, B=1, C=2, \ldots, Z=25$ and K be the ***shifting key***
- Let P_i denote the ***Plaintext string***
- Caesar ciphertext is given by $C_i = (P_i + K)\mathrm{MOD}(26)$
- Decryption is based on $P_i = (C_i - K)\mathrm{MOD}(26)$
- MOD? **If** $P_i + K$ or $P_i - K$ are not in the range 0-25, **Then** subtract or add 26, respectively

The Vigenere Cipher

- The Caesar cipher is a ***monoalphabetic*** substitution cipher

- The key is easily recovered through statistical analysis, i.e. searching for the most frequently occurring letter in the cipher

- *Vigenere* encryption is based on an obvious extension to produce a ***polyalphabetic*** substitution cipher, i.e.

$$C_i = (P_i + K_i)\mathrm{MOD}(26)$$

$$P_i = (C_i - K_i)\mathrm{MOD}(26)$$

Vernam Cipher (1919)

- Substitution cipher based on generating an array of random integers to form a vector $\mathbf{n}$

- Cipher is given by (vector addition)

$$\mathbf{c} = \mathbf{p} + \mathbf{n}$$

- Number code used for $\mathbf{p}$ (and $\mathbf{c}$) must be standardised, e.g. 7-bit ASCII code so that

$$c_i = (p_i + n_i)\mathrm{MOD}(127)$$

Example of a Vernam Cipher

DEC	BIN	CHAR	DEC	BIN	CHAR	DEC	BIN	CHAR
0	0000000	null	43	0101011	+	86	1010110	V
1	0000001	suh	44	0101100	,	87	1010111	W
2	0000010	stx	45	0101101	-	88	1011000	X
3	0000011	etx	46	0101110	.	89	1011001	Y
4	0000100	eot	47	0101111	/	90	1011010	Z
5	0000101	enq	48	0110000	0	91	1011011	[
6	0000110	ack	49	0110001	1	92	1011100	&
7	0000111	bel	50	0110010	2	93	1011101	]
8	0001000	bs	51	0110011	3	94	1011110	^
9	0001001	ht	52	0110100	4	95	1011111	_
12	0001010	lf	53	0110101	5	96	1100000	‘
11	0001011	vt	54	0110110	6	97	1100001	a
12	0001100	ff	55	0110111	7	98	1100010	b
13	0001101	cr	56	0111000	8	99	1100011	c
14	0001110	so	57	0111001	9	100	1100100	d
15	0001111	si	58	0111010	:	101	1100101	e
16	0010000	dle	59	0111011	;	102	1100110	f
17	0010001	dc1	60	0111100	<	103	1100111	g
18	0010010	dc2	61	0111101	=	104	1101000	h
19	0010011	dc3	62	0111110	>	105	1101001	i
20	0010100	dc4	63	0111111	?	106	1101010	j
21	0010101	nak	64	1000000	@	107	1101011	k
22	0010110	syn	65	1000001	A	108	1101100	l
23	0010111	etb	66	1000010	B	109	1101101	m
24	0011000	can	67	1000011	C	110	1101110	n
25	0011001	em	68	1000100	D	111	1101111	o
26	0011010	sub	69	1000101	E	112	1110000	p
27	0011011	esc	70	1000110	F	113	1110001	q
28	0011100	fs	71	1000111	G	114	1110010	r
29	0011101	gs	72	1001000	H	115	1110011	s
30	0011110	rs	73	1001001	I	116	1110100	t
31	0011111	us	74	1001010	J	117	1110101	u
32	0100000	sp	75	1001011	K	118	1110110	v
33	0100001	!	76	1001100	L	119	1110111	w
34	0100010	”	77	1001101	M	120	1111000	x
35	0100011	#	78	1001110	N	121	1111001	y
36	0100100	$	79	1001111	O	122	1111010	z
37	0100101	%	80	1010000	P	123	1111011	{
38	0100110	&	81	1010001	Q	124	1111100	|
39	0100111	’	82	1010010	R	125	1111101	}
40	0101000	(	83	1010011	S	126	1111110	~
41	0101001	)	84	1010100	T	127	1111111	del
42	0101010	*	85	1010101	U			

Plaintext:
Attack at 03:00am.

DEC ASCII code	Cipher	Addition	MOD(127)
65	5	70	70
116	0	116	116
116	66	182	55
97	1	98	98
99	8	107	107
107	6	113	113
32	13	45	45
97	27	124	124
116	67	183	56
32	31	63	63
48	9	57	57
51	16	67	67
58	17	75	75
48	19	67	67
48	10	58	58
97	18	115	115
109	2	111	111
46	9	55	55

Ciphertext:
Ft7bkq-|8?9CKC:so7

Substitution (Stream) Ciphers

- Plaintext character substituted for randomly selected character generated by a cipher $\mathbf{n}$

$$\mathbf{c} = \mathbf{n} + \mathbf{p}$$ Decimal space

- Usually implemented using bit-wise operators, operating on binary strings, e.g.

$$\mathbf{c} = \mathbf{n} \oplus \mathbf{p}$$ Binary space

$$\mathbf{p} = \mathbf{n} \oplus \mathbf{c} = \mathbf{n} \oplus \mathbf{n} \oplus \mathbf{p}$$

8-bit XOR based Encryption

Plaintext:
Attack at 03:00am.

8-bit ASCII code	8-bit cipher	XOR
01000001	00000101	01000100
01110100	00000000	01110100
01110100	01000010	00110110
01100001	00000001	01100000
01100011	00001000	01101011
01101011	00000110	01101101
00100000	00001101	00101101
01100001	00011011	01111010
01110100	01000011	00110111
00100000	00011111	00011111
00110000	00001001	00111001
00110011	00010000	00100011
00111010	00010001	00101011
00110000	00001001	00111001
00110000	00001010	00111010
01100001	00010010	01110011
01101101	10000000	01101111
00101110	00001001	00100111

Ciphertext:
Dt6‘km-z7&9#+9:so’

Principal Conditions for

$$\mathbf{c} = \mathbf{n} + \mathbf{p} \quad | \quad \mathbf{c} = \mathbf{n} \oplus \mathbf{p}$$

- $\mathbf{n}$ - ***the cipher*** – is generated by some physical effect or computed using a numerical algorithm that can be ***seeded*** by a key $\boldsymbol{K}$

- The algorithm should produce random numbers with no statistical bias – **maximum *confusion***

- $\mathbf{n}$ should be ultra-sensitive to $\boldsymbol{K}$ **:**

  a change of 1 bit in $\boldsymbol{K}$ should potentially effect all the bits of $\mathbf{n}$ – **maximum *diffusion***

- $\mathbf{n}$ must have a ***long cycle length***

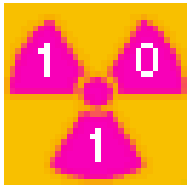
Examples of Cipher Generation I

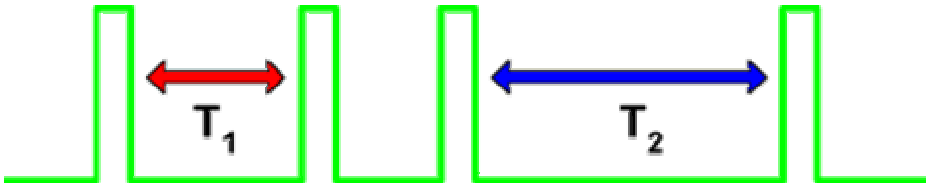
- SIGSALLY (Green Hornet): AT & T (1942-46)
- Noise generated using a vacuum tube and stored on a phonograph record
- Record used to mask 1-to-1 voice signals
- Distribution of noise sources strictly controlled
- Records were in effect ***one-time-pads***




Examples of Cipher Generation II

- HotBits





http://www.fourmilab.ch/hotbits/

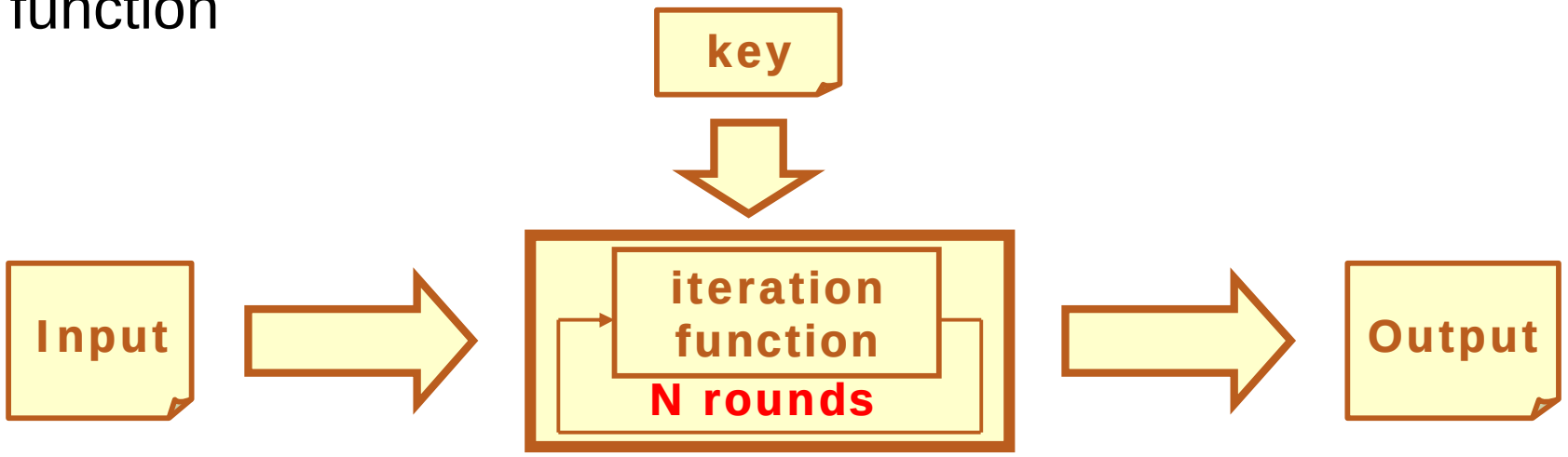
- Atmospheric radio noise
  http://www.random.org/

- **Quantum Mechanical noise using a reverse biased semiconductor junction** http://www.araneus.fi/

Iterative Cryptosystems

- Most cryptographic systems are based on a series of so-called round transformations, which are relatively simple and produce Pseudo Random Number Streams ***Pseudo Random Number Generators (PRNG)***

- A PRNG is a function or an algorithm that produces a sequence of numbers from a relatively short seed (initial conditions: password, plaintext) based on some iteration function



The mod Function

- Modular based functions tend to behave more erratically than conventional functions

- *a*mod(*b*) gives the remainder of a/b, e.g.

23mod(7) = 2, 6mod(8) = 6

***a*mod(*b*)=*a*-*b*floor(*a*/*b*)**

x	1	2	3	4	5	6	7	8
2^x	2	4	8	16	32	64	128	256
$2^x \bmod 13$	2	4	8	3	6	12	11	9

Example Algorithms for Computing

$$\mathbf{n} \equiv (n_0, n_1, n_2 \ldots)$$

- Blum Blum Shub generator $n_{i+1} = n_i^2 \bmod(pq)$
  where p and q are two prime numbers

- Blum Mercali generator $n_{i+1} = p^{n_i} \bmod(q)$
  where q is a prime and p is an odd prime

- RSA (Rivest, Shamir and Adleman) generator

$$n_{i+1} = n_i^e \bmod(pq)$$

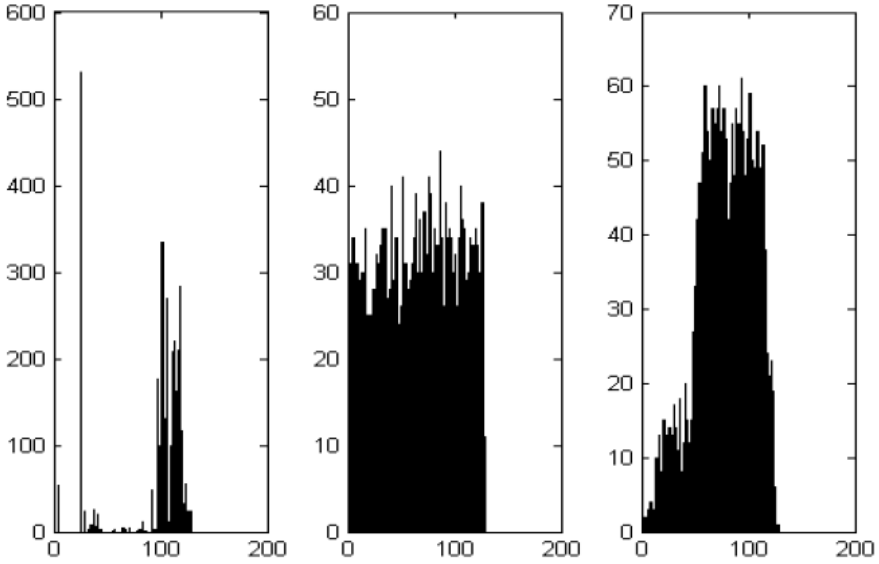
where e is a relative prime of p-1 and q-1

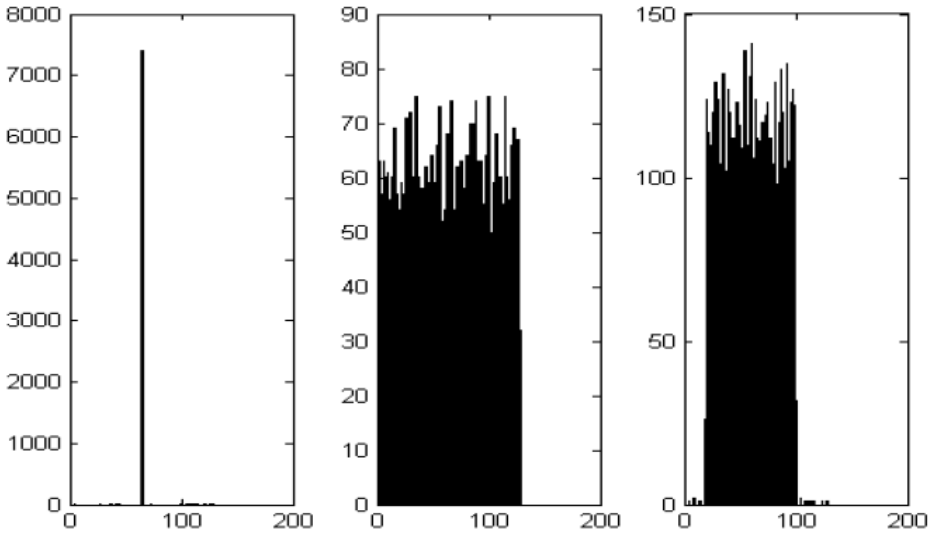
Maximum Entropy Encryption

- Encryption process changes the ***statistics of cipher***
- Statistics of the ciphertext become non-uniform
- Solution is to ***pad the plaintext*** (with '?' = 63 for 7-bit ASCII)

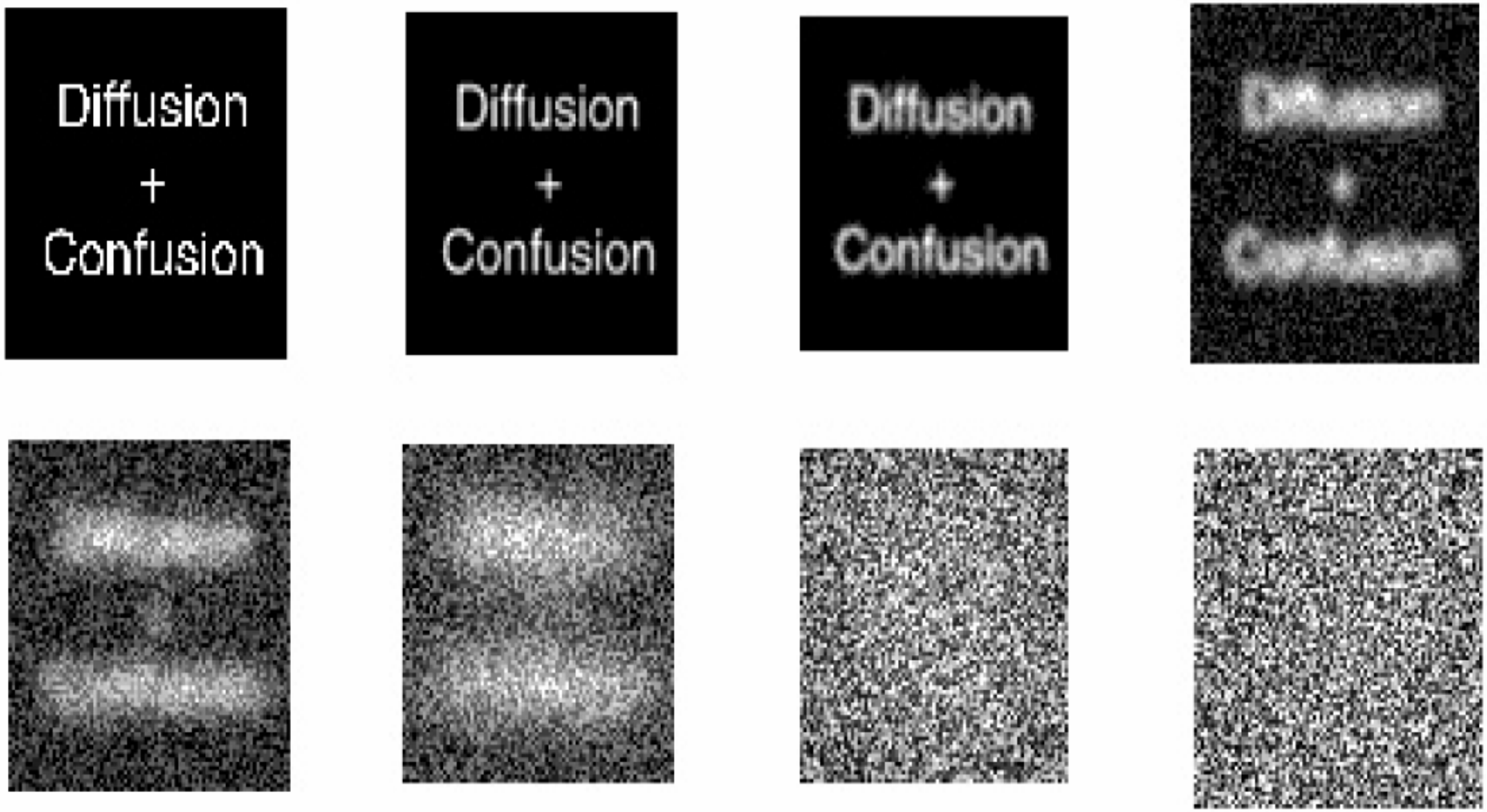
$$\mathbf{c} = \mathbf{n} + \mathbf{p}$$

$$I = \sum_{i=1}^{N} P_i \log_2 P_i$$

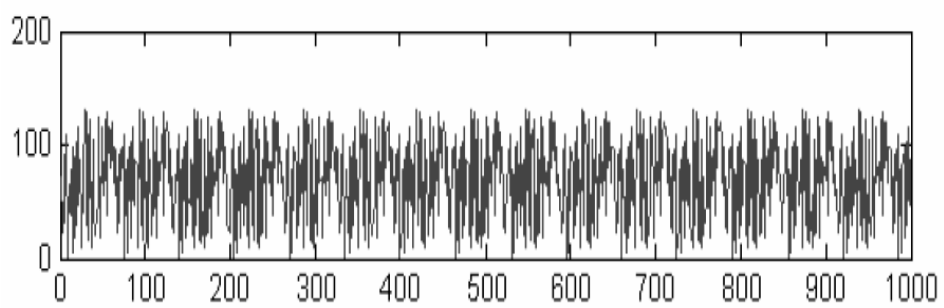


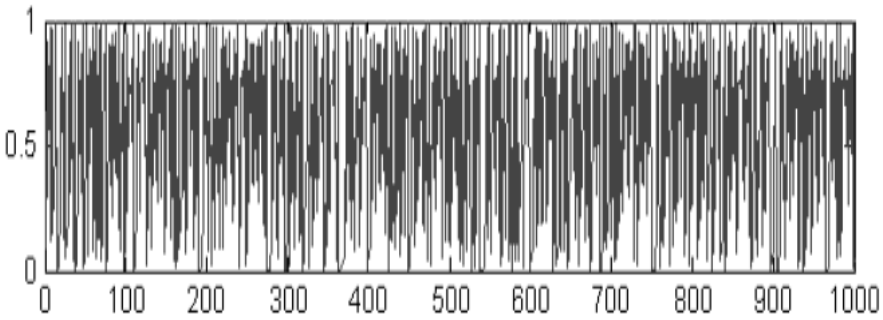


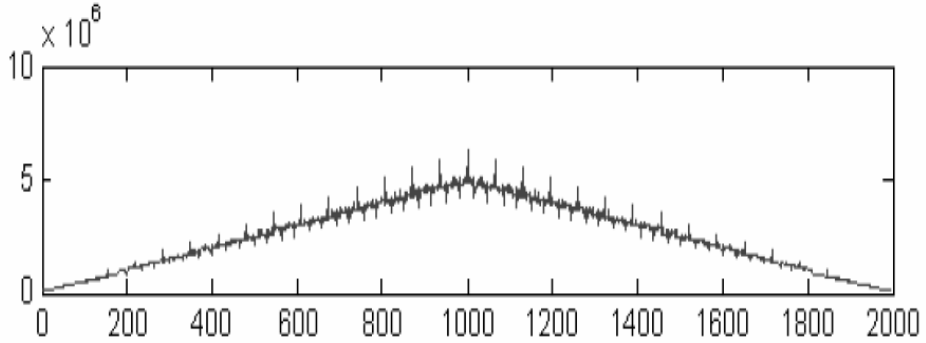
Diffusion + Confusion

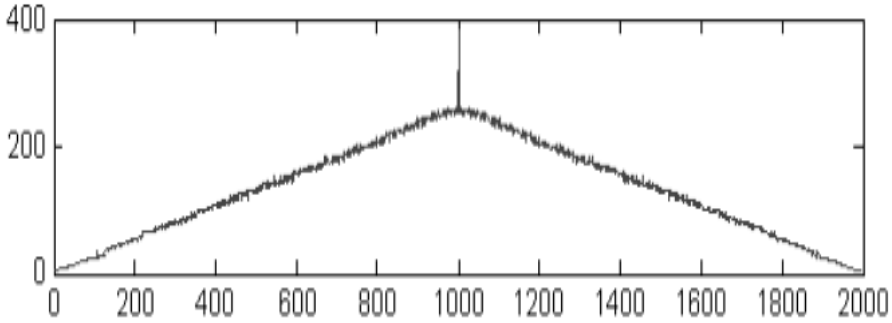


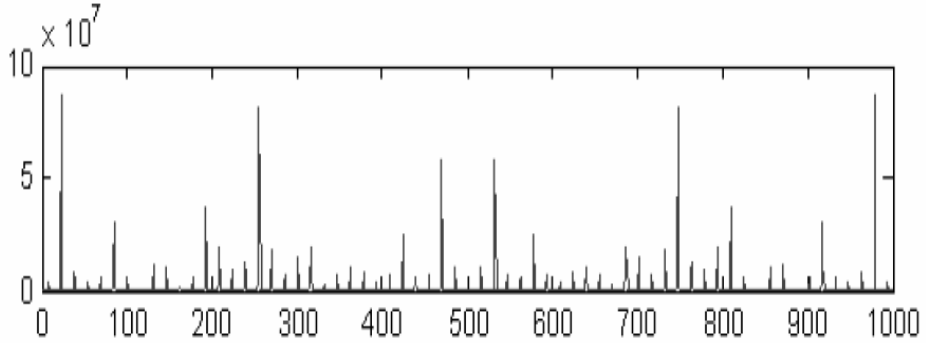
Cycle Length Analysis using Autocorrelation & Power Spectrum

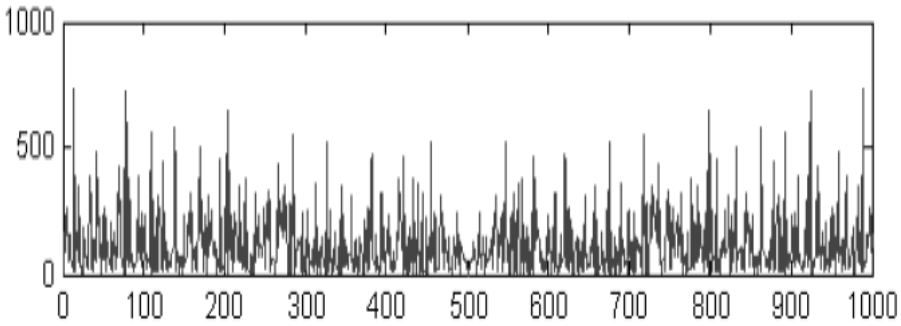












Kerchhoff-Shannon Principle

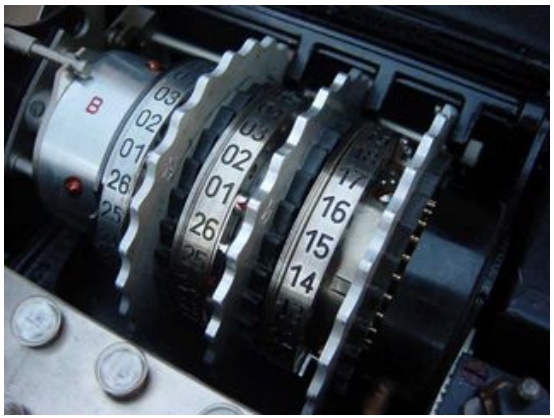
- Kerchhoff's Principle:

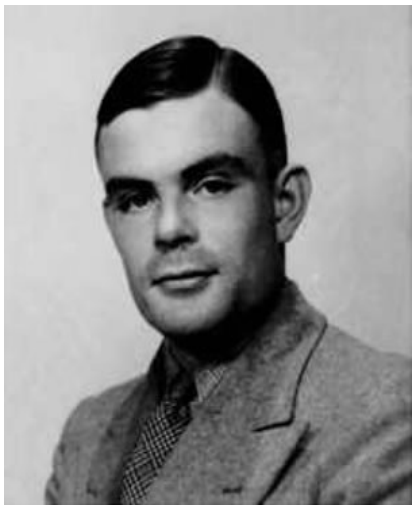
  'A cryptosystem should be secure even if everything about the system, except the key, is public knowledge'

- Shannon's Principle:

  'The enemy knows the system', i.e.

THE ALGORITHM





Some Golden Rules

- ***Security is a process not a product***
- **Never underestimate the enemy**
- **The longer that any cryptosystem, or part thereof, remains of the same type with the same function, the more vulnerable the system becomes to a successful attack inclusive of *THE ALGORITHM***
- **If you want to know what you are eating then grow it and cook it yourself**

The RSA Algorithm

The **R**ivest, **S**hamir & **A**dleman algorithm is as follows:

- Prime numbers p & q are chosen together with $e < pq$
- A obtains public key for B - given by (e, pq) - and sends

$$C_i = P_i^e \mathrm{mod}(pq), \quad i = 1, 2, ..., N$$

- B has a private key d such that ed-1 is divisible by (p-1)(q-1), i.e. d is the solution of

$$de = \mathrm{mod}[(p-1)(q-1)]$$

- B recovers message using

$$P_i = C_i^d \mathrm{mod}(pq)$$

Important Points

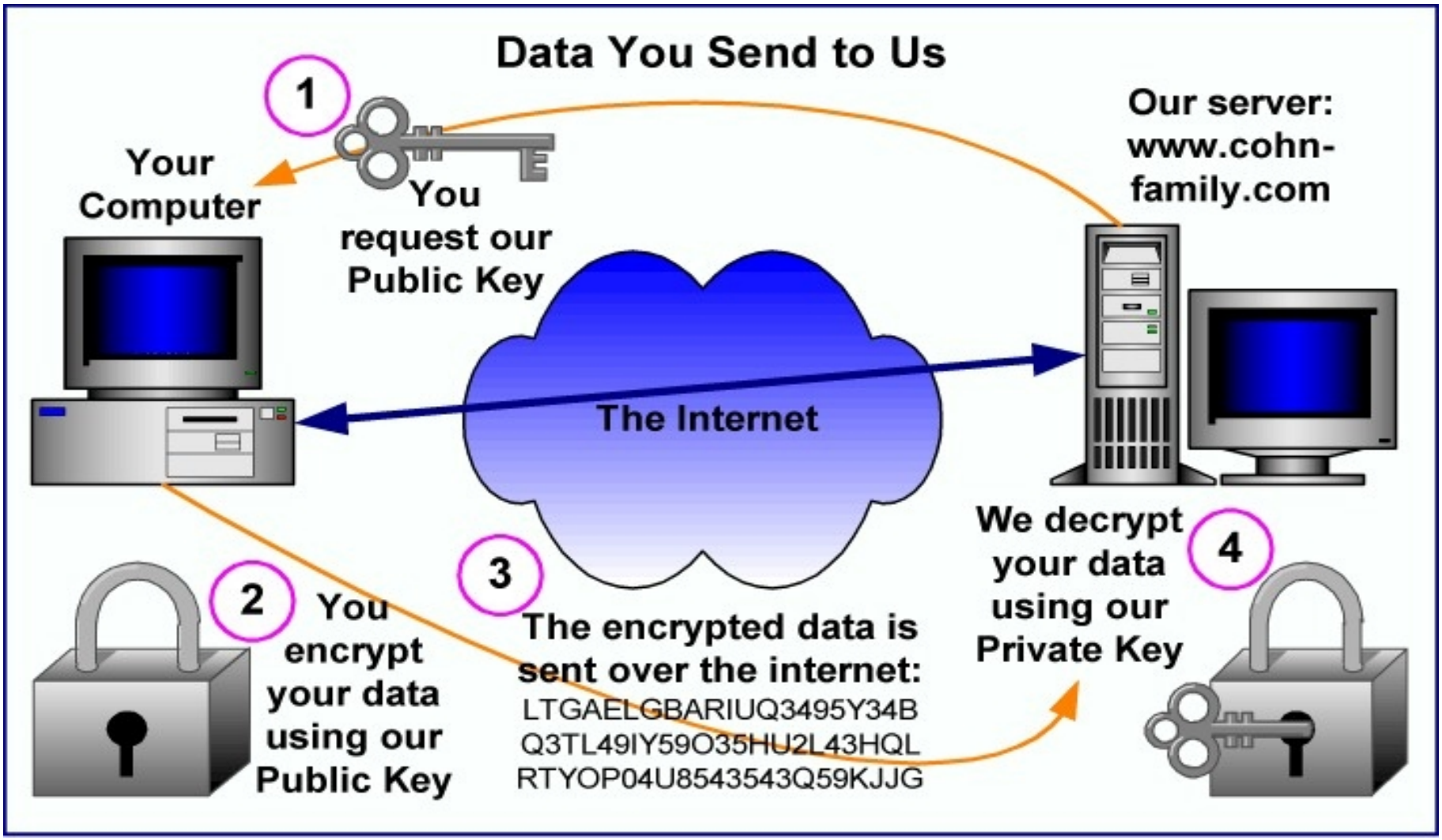
$$de = \mathrm{mod}[(p-1)(q-1)]$$

- To compute d, e must be a relative prime of (p-1)(q-1). This means that e & (p-1)(q-1) have no common factors except 1

- The prime numbers p & q and the number $e < pq$ must be distributed to Alice and Bob in such a way that they are unique to Alice and Bob on the condition that d exists!

- This requires an appropriate infrastructure to be established by a trusted third party who's 'business' is to distribute values of e, pq & d to its clients – a **P**ublic **K**ey **I**nfrastructure (PKI)

Internet Communications



Vulnerability to an Attack

- e and pq are known and p and q **must** be prime numbers - elements of a large but (assumed) known set.

- To attack the cipher, d must be found and it is known that d is the solution of

$$de = \mathrm{mod}[(p-1)(q-1)]$$

which is only solvable if $e < pq$ is a relative prime of (p-1)(q-1).

- An attack is based on searching through prime numbers whose magnitudes are consistent with the product pq until the *relative prime condition* is established for factors p and q.

Public Key Infrastructure (PKI)

- A PKI is required in order to distribute public keys,
  i.e., different but appropriate values of e and pq,
  for use in public key cryptography (RSA algorithm)

- Requires the establishment of appropriate authorities and directory services for the generation, management and certification of public keys

- Vulnerable to authorities (operating in UK) having to conform to the **Regulation of Investigatory Powers** Act (UK) 2000, Section 49

Summary

- Encryption systems belong to two basic classes:

  - symmetric

  - asymmetric

- Encryption algorithm should provide a cipher with the following basic properties:

  - Maximum entropy of cipher

  - Maximum diffusion of key

  - Long cycle length of cipher

- Encryption algorithm is taken to be public knowledge

  The Kerchhoff-Shannon Principle, e.g. RSA Algorithm

In the Following Lecture…

- We shall investigate the properties of chaotic signals
- Consider a multi-algorithmic approach for designing *encryption engines*
- Provide an overview of Crypstic
- Provide a demonstration of the product

Questions

+

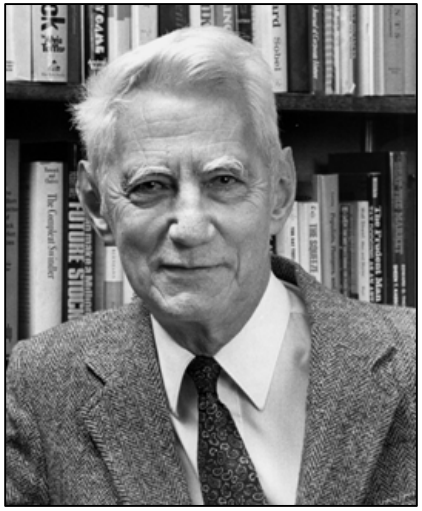
Interval (10 Minutes)

Part II: Contents

- Chaos and Cryptography
- Iteration Functions Systems
- Chaos and Pseudo-Chaos
- The Lyapunov Exponent
- Designing Chaos-based Encryption Algorithms
- Multi-algoritjmicity
- ***Crypstic***
- Demonstration of ***Crypstic***
- Q & A

Cryptography using Chaos

Founders of Modern Cryptography

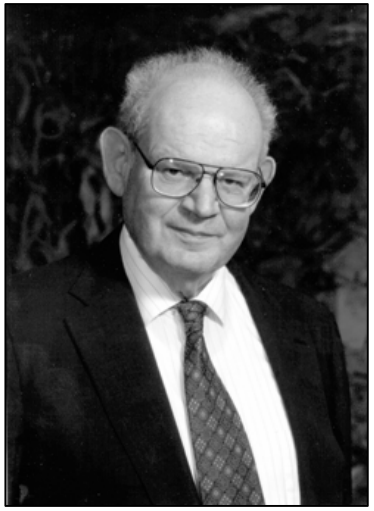


Claude Shannon

Vladimir Kotelnikov

Founders of Chaos Theory

Mitchell Feigenbaum



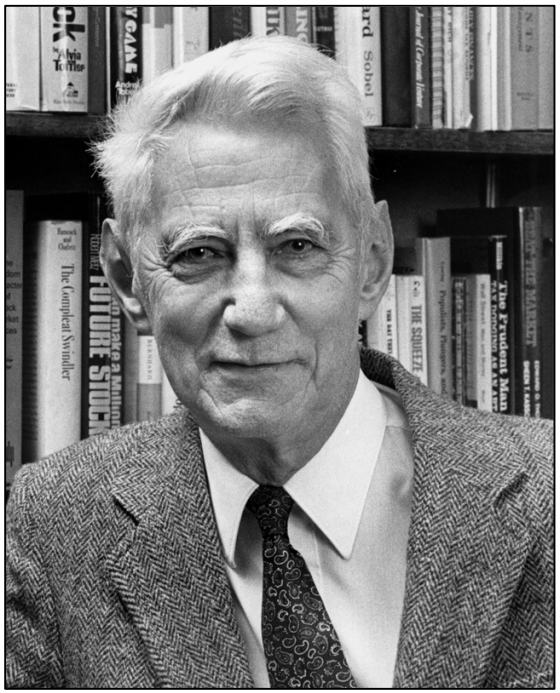
Benoit Mandelbrot

Algorithm(s) for $\mathbf{n}$

$$\mathbf{c = n + p}$$

Iteration Function Systems (IFS)

Brief History of Chaos-based Cryptography



Claude Shannon
1916 - 2001

- Early 1950s: Shannon explicitly mentions that the basic stretch-and-fold mechanism of chaos can be used in cryptology.
- Silent period until the late1980s.
  - Chaos theory becomes popular
  - Cryptography becomes more important
- ~ 30 publications in 1990s
  - Various ciphers suggested
  - Focus on analog circuits
- 2000++: Chaos begins to be recognized
  - spread spectrum for military communications
  - launch of Crypstic by Lexicon Data Limited

Chaos and Cryptology: *Similarities 1*

- Deterministic
  - chaotic map
  - encryption algorithm

- Complex and Unpredictable

  random-like behavior for any external observer with no *a priori* knowledge of the algorithm and initial condition - key

Chaos and Cryptology
Similarities 2

Chaos		Cryptography
sensitivity to initial conditions	⇔	**key-dependent confusion & diffusion**

- Small variations of any variable changes the outputs considerably

- Modification of 1 bit of the plaintext or key should change all bits of the ciphertext with probability 50%.

Chaos and Cryptology
Similarities 3

Chaos | **Cryptography**

topological transitivity with iterative process	⇔	**multi-round transformations**

- Bounded state space, self-mapping, extension of a state point over the whole state space

- Iterative transformations with a single chaotic map

Chaos and Cryptology
Principal Differences

- Chaotic systems are defined on real/complex numbers spaces (bounded continuous space) whereas cryptography uses binary sequences (*finite discrete space*).

- Chaos theory aims to understand the *asymptotic behavior* of iterative process whereas cryptography focuses on the properties of a number of first few iterations

Chaos Theory .v. Cryptography

Chaos Theory	Cryptography
Chaotic system	Pseudo-chaotic system
Nonlinear transform	Nonlinear transform
Infinite number of states	Finite states
Infinite number of iterations	Finite iterations
Initial state	Plaintext
Final state	Ciphertext
Initial condition(s) and/or parameter(s)	key
Asymptotic independence of initial and final states	Confusion
Sensitivity to initial condition(s) and parameter(s) mixing	Diffusion

Simple Example of an IFS: The Vurhulst Process

- Linear exponential model $x(t) = x_0 \exp(kt)$

$$\frac{dx}{dt} = kx$$

- Nonlinear model

$$\frac{x_{i+1} - x_i}{\Delta t} = kx_i(1 - x_i)$$

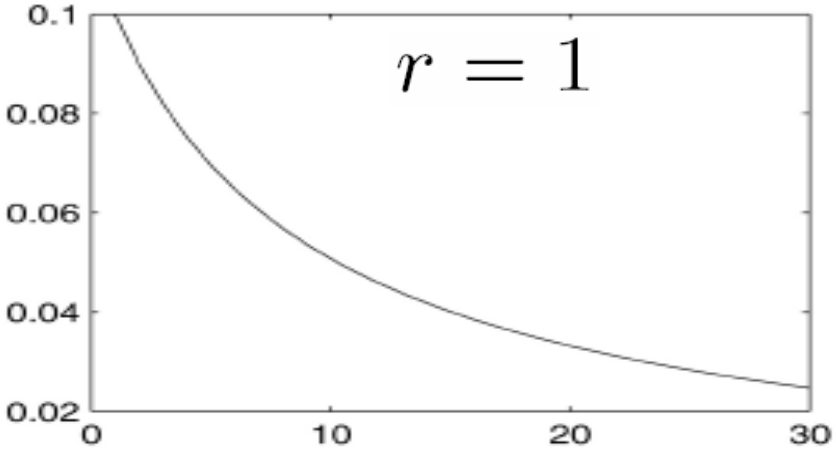
$$\frac{dx}{dt} = kx(1 - x)$$

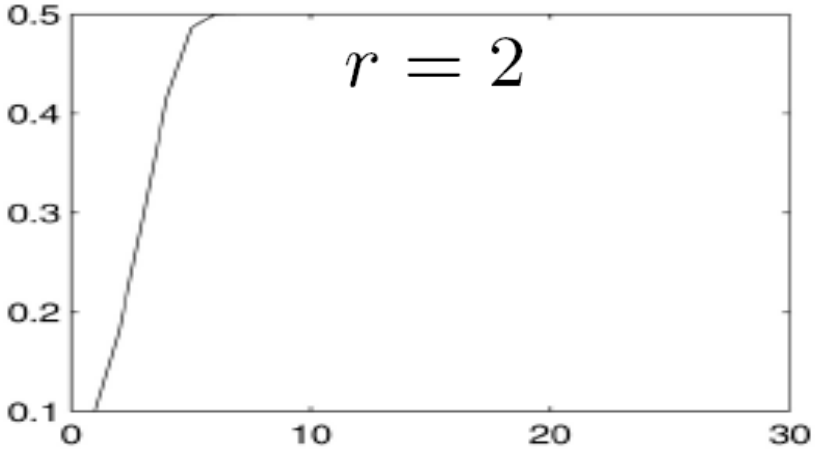
$$x_{i+1} = x_i + k\Delta t x_i(1 - x_i)$$

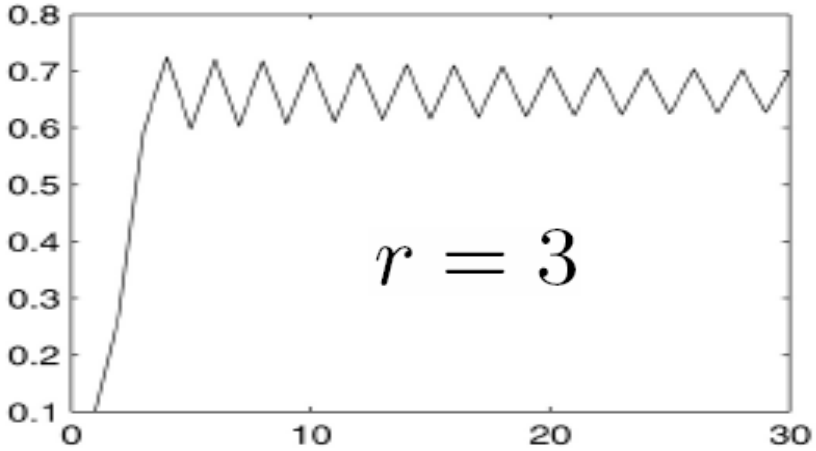
$$x_{i+1} = rx_i(1 - x_i)$$

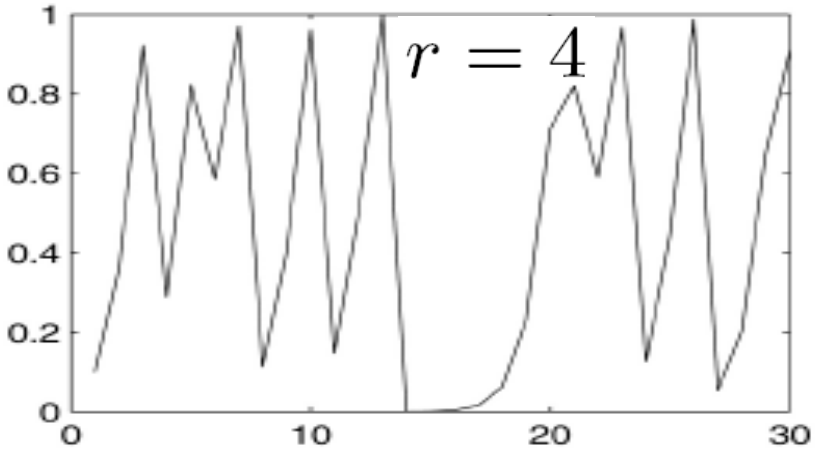
Example Iteration Function System (IFS)

$$x_{i+1} = r x_i (1 - x_i)$$



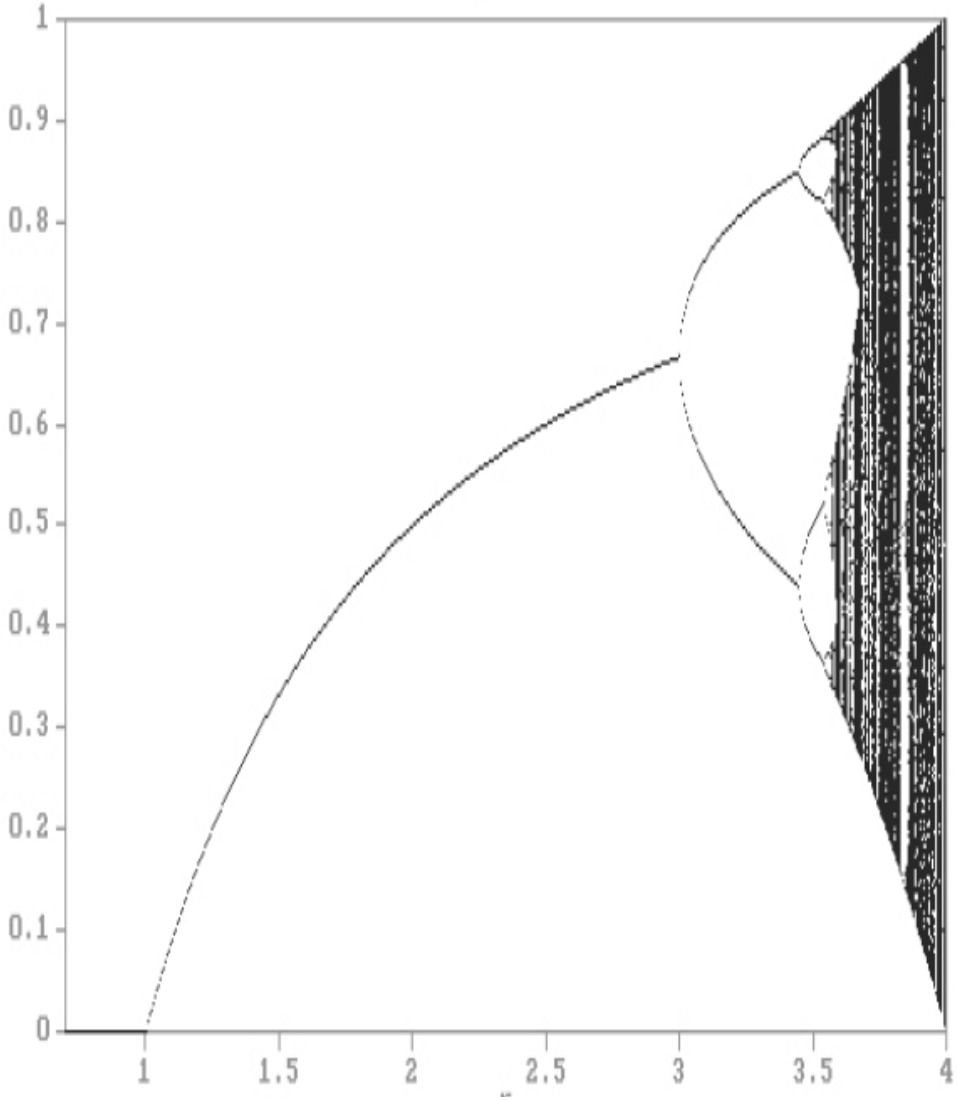


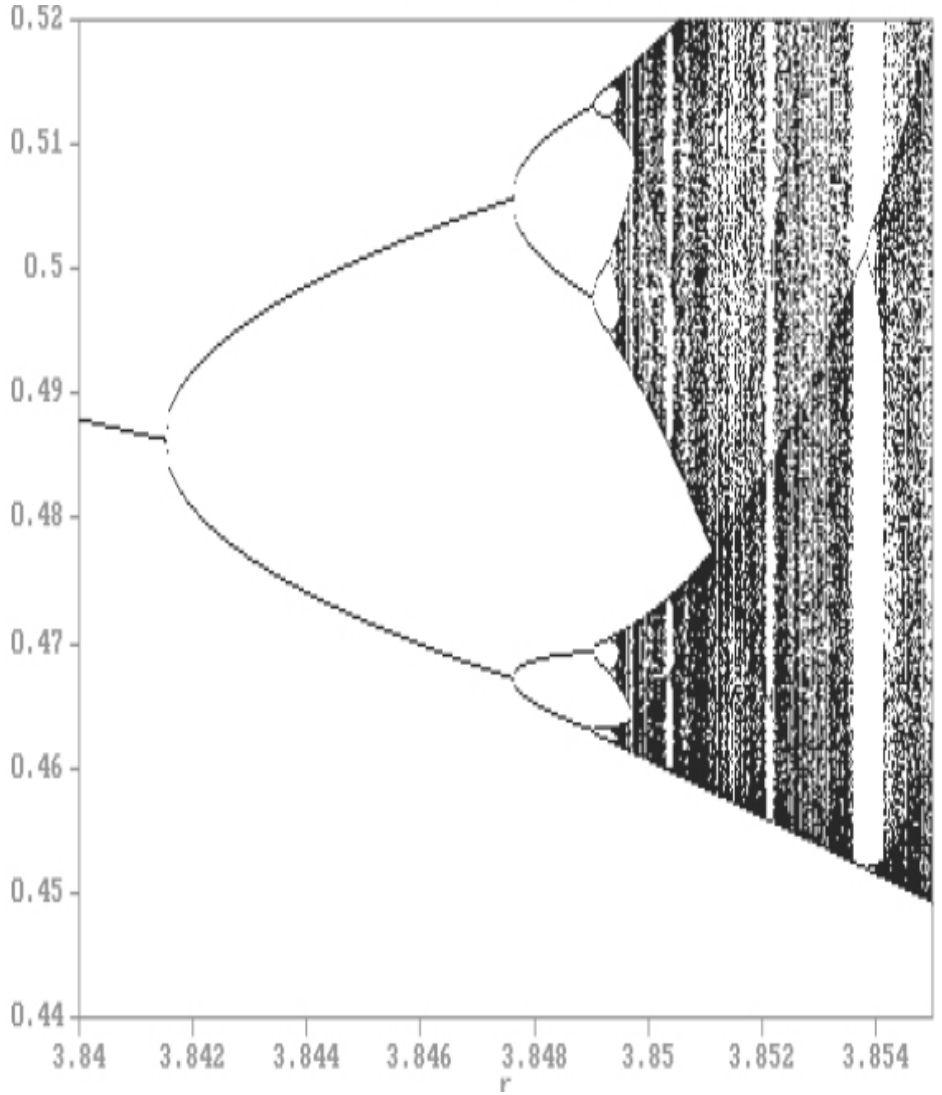




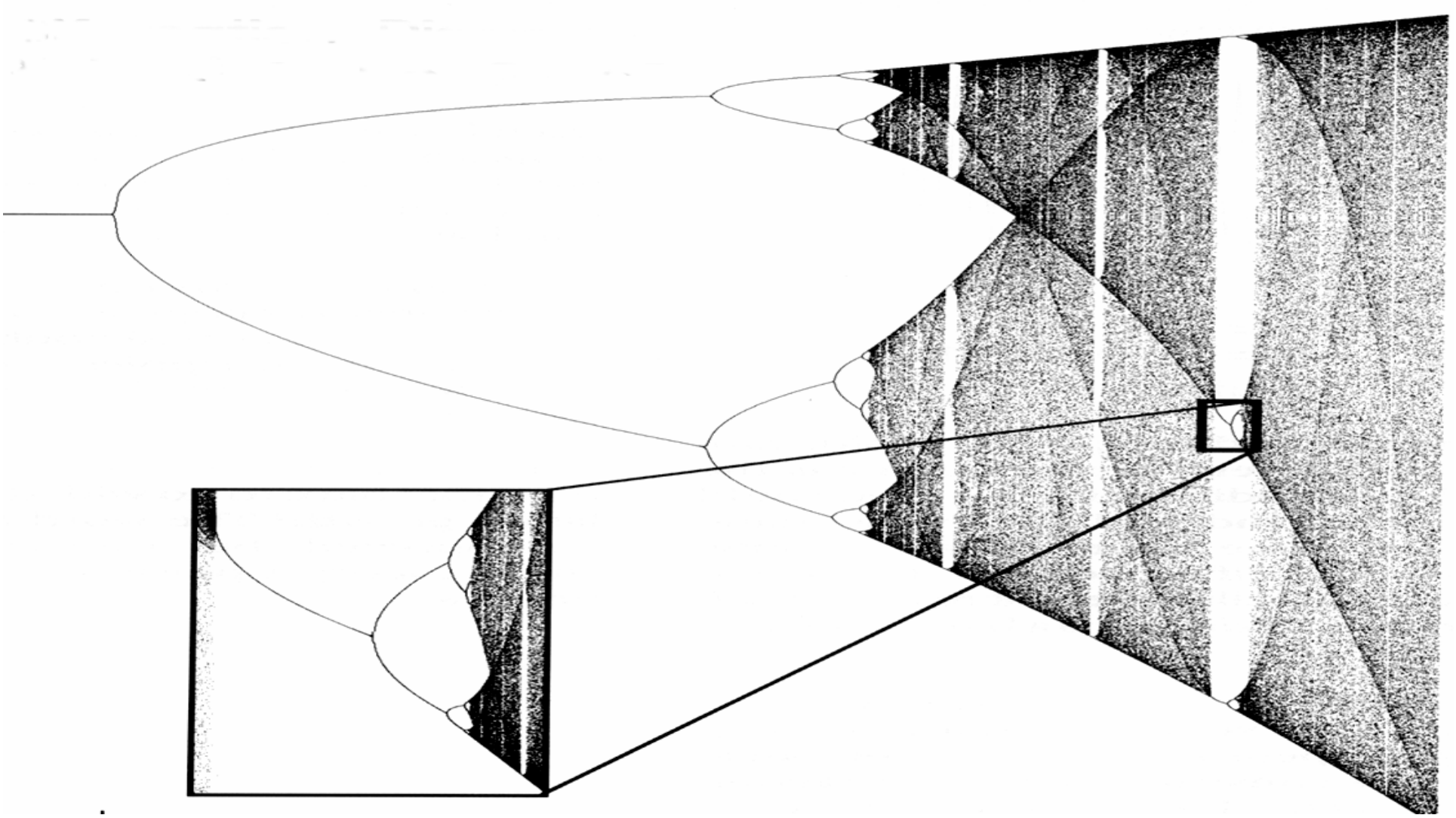
Feigenbaum Diagram

$$x_{i+1} = r x_i (1 - x_i)$$





Self-Affine Characteristics



Properties of Chaotic Systems Required for Cryptography

- Sensitivity to the initial conditions

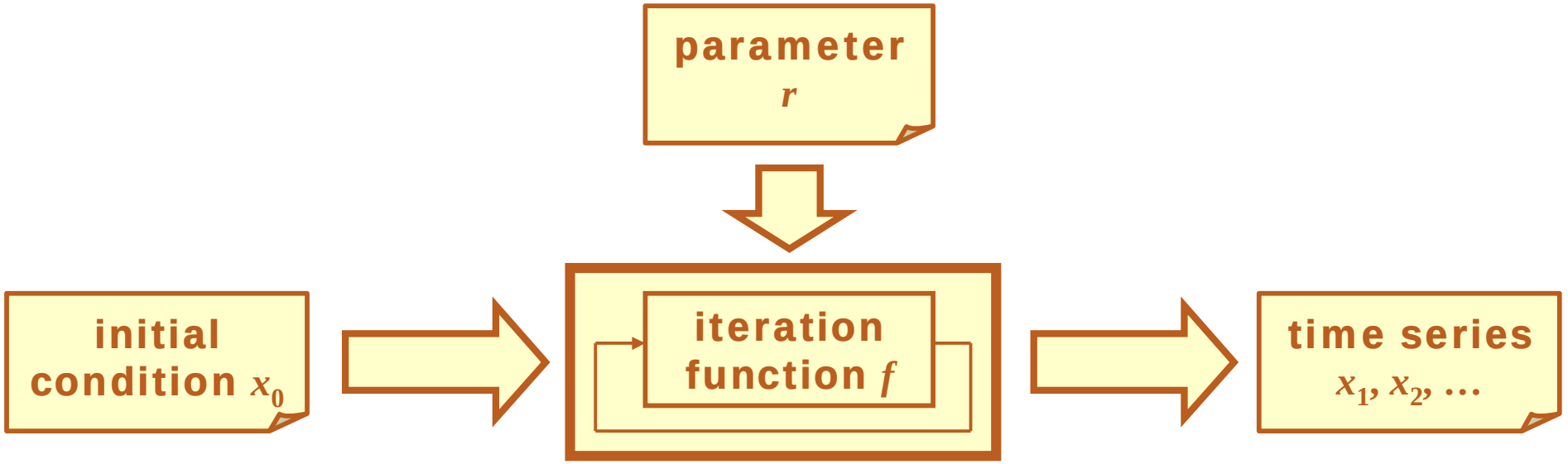
  It is impossible to predict the behaviour of the system even if we have partial knowledge of its organization.

- Topological transitivity

  The state point stays within a bounded state space and approaches infinitely closely to any point of the state space.

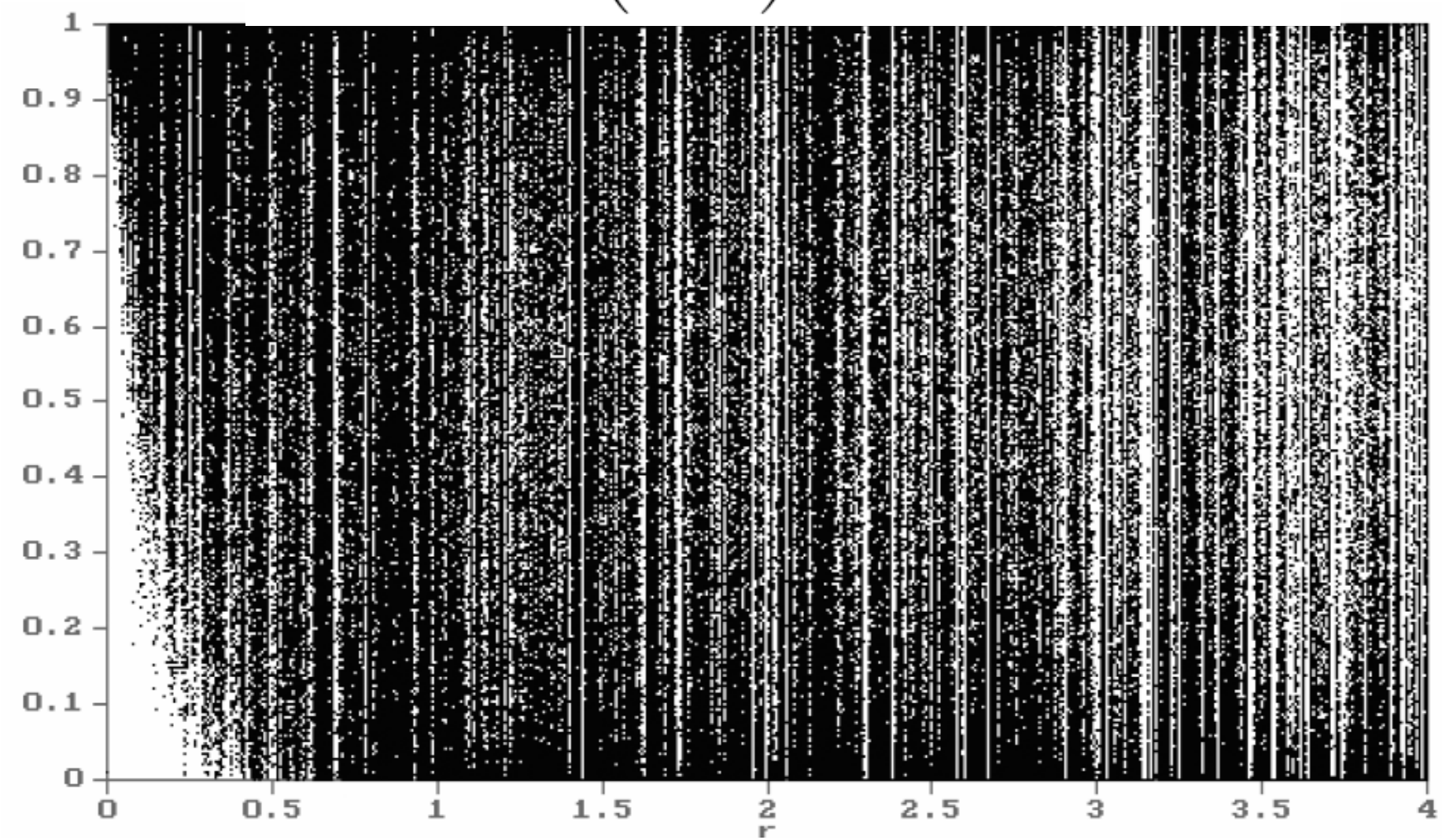
A Deterministic Chaotic System

- Deterministic system is defined by a IFS $f(x)$
- Input is initial condition x_0 and parameter r
- Output is a sequence of states: $x_1,\ x_2,\ x_3, \ldots$ where $x_{i+1} = f(x_i,\ r)$



Matthews Cipher

$$x_{i+1} = (1+r)\left(1+\frac{1}{r}\right)^{r} x_i(1-x_i)^r, \quad r \in (0,4]$$



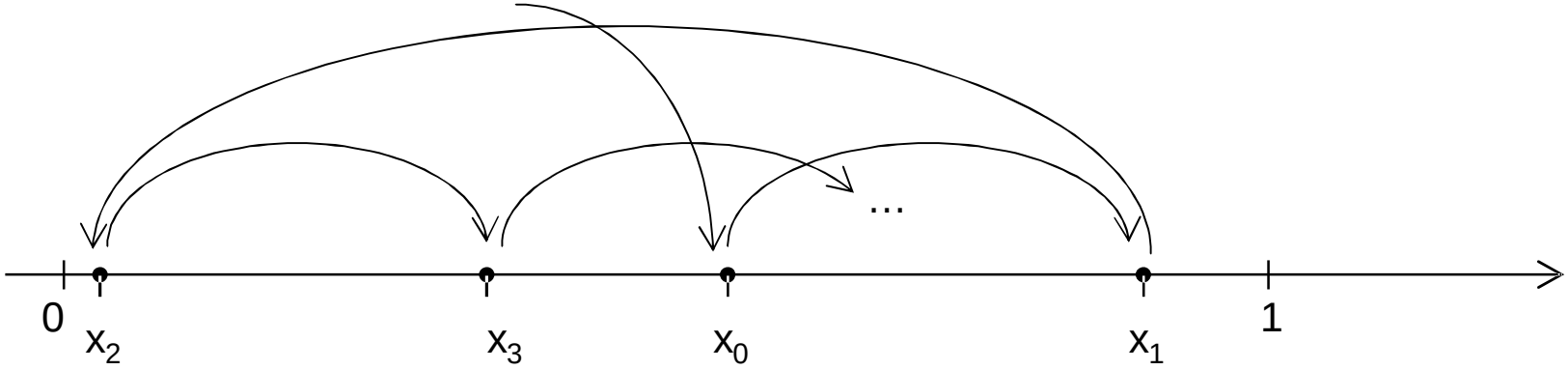
Chaos and Pseudo-Chaos

- True Chaos has an ***infinite*** number of states

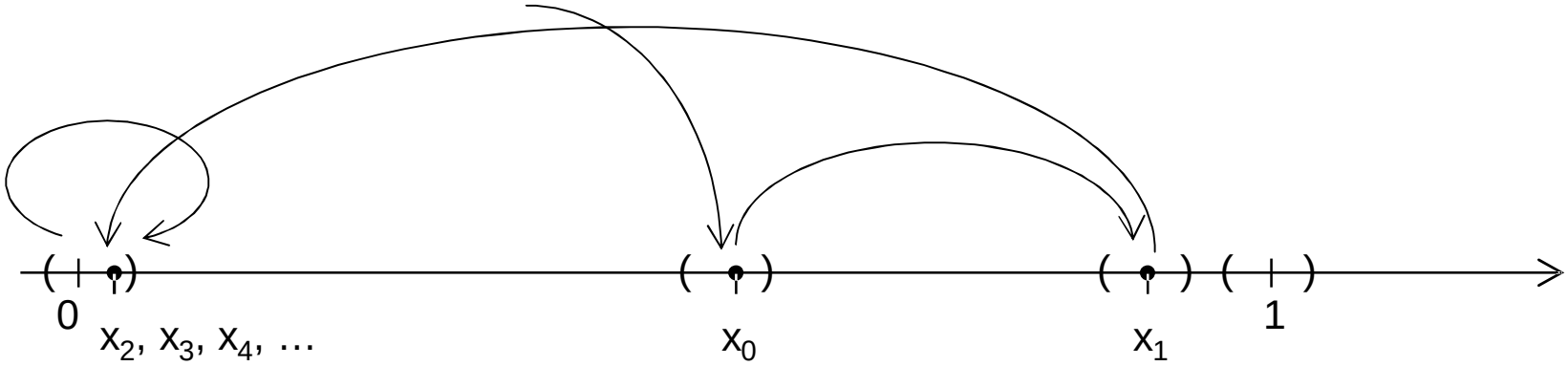
- Pseudo-Chaos has a ***finite*** number of states
  - Involves approximation of continuous chaos with floating- or fixed-point arithmetic
  - Leads to discrete chaos-like system with low cycle lengths

Floating-point Approximation

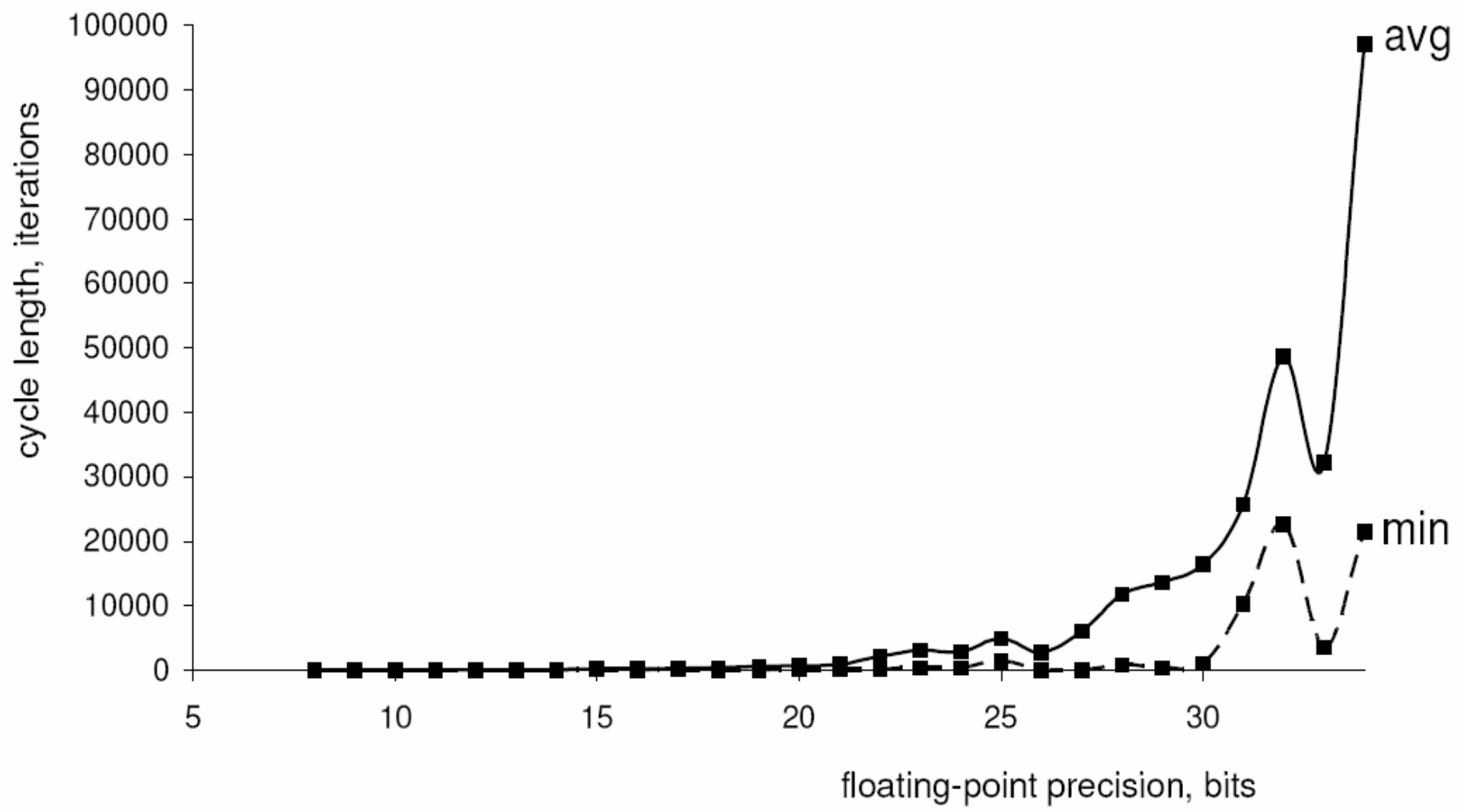
Continuous Chaos



Floating-point Approximation

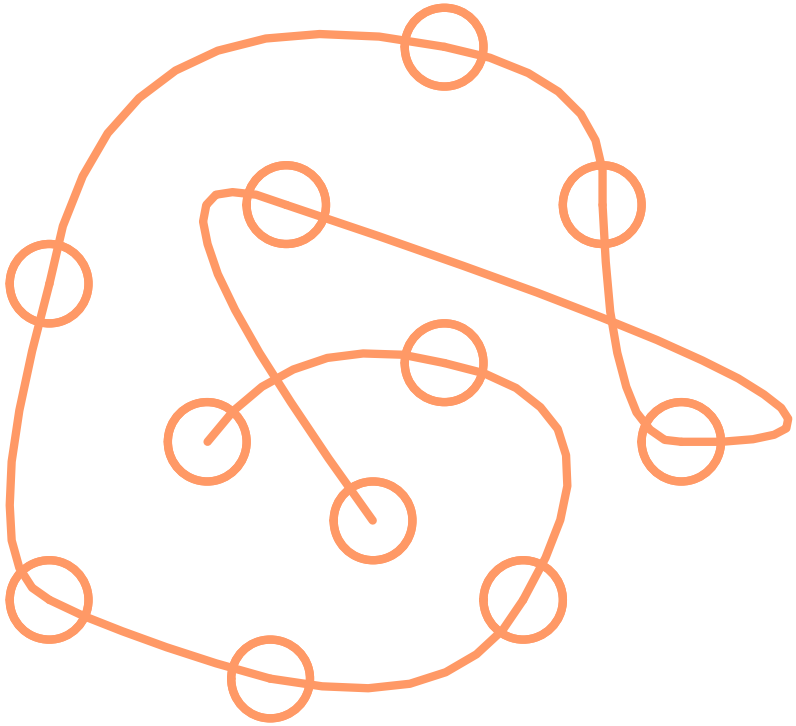


Example Cycle Length Distribution (Vurhulst Process)

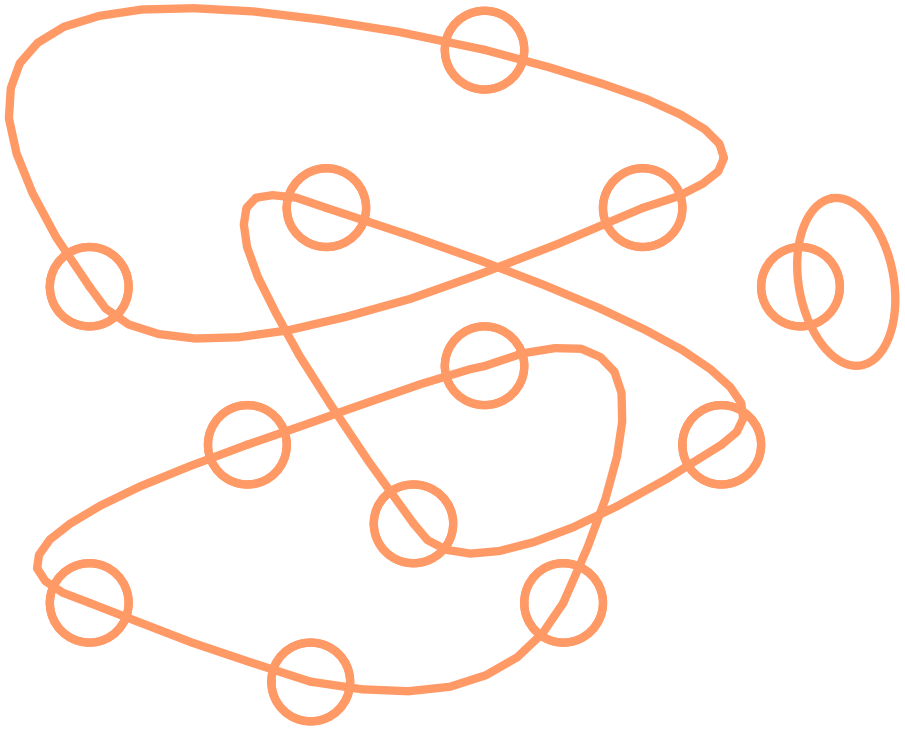


Chaos .v. Pseudo Chaos

Infinite orbit (Chaos)

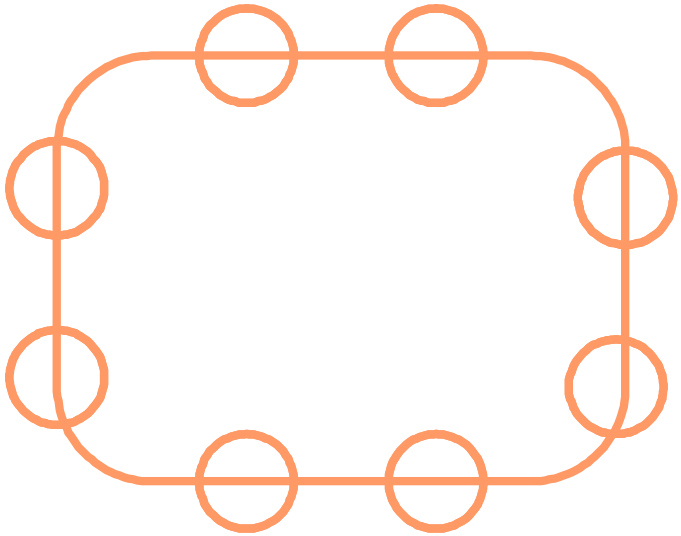


Finite orbits (Pseudo-Chaos)

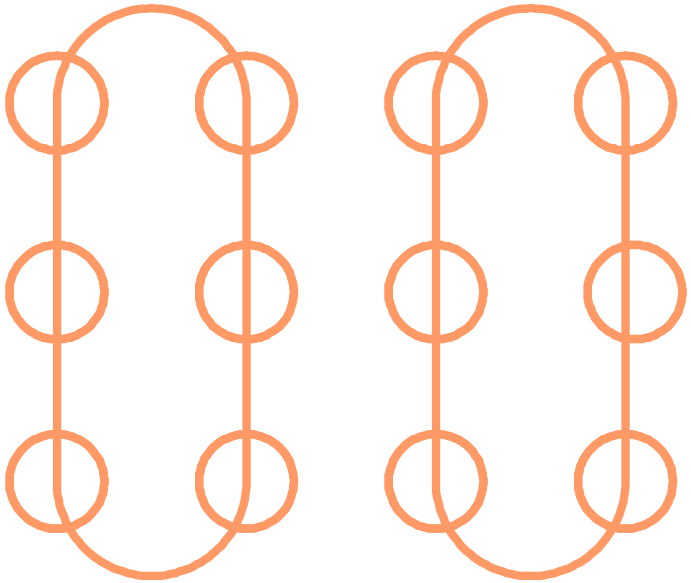


Cryptographically Good Orbits

Single orbit



Multiple orbits of the same length



Stability of an Iterative Process

- Consider the iterative process

$$x_{n+1} = f(x_n) = x + \epsilon_n$$

and a model for the error at each iteration given by

$$\epsilon_{n+1} = c \exp(n\lambda)$$

- Then

$$\epsilon_{n+1} = \epsilon_n \exp(\lambda)$$

Measure of Stability

$$\epsilon_{n+1} = \epsilon_n \exp(\lambda)$$

Rearranging and summing over N iterations:

$$\sum_{n=1}^{N} \ln\left(\frac{\epsilon_{n+1}}{\epsilon_n}\right) = N\lambda$$

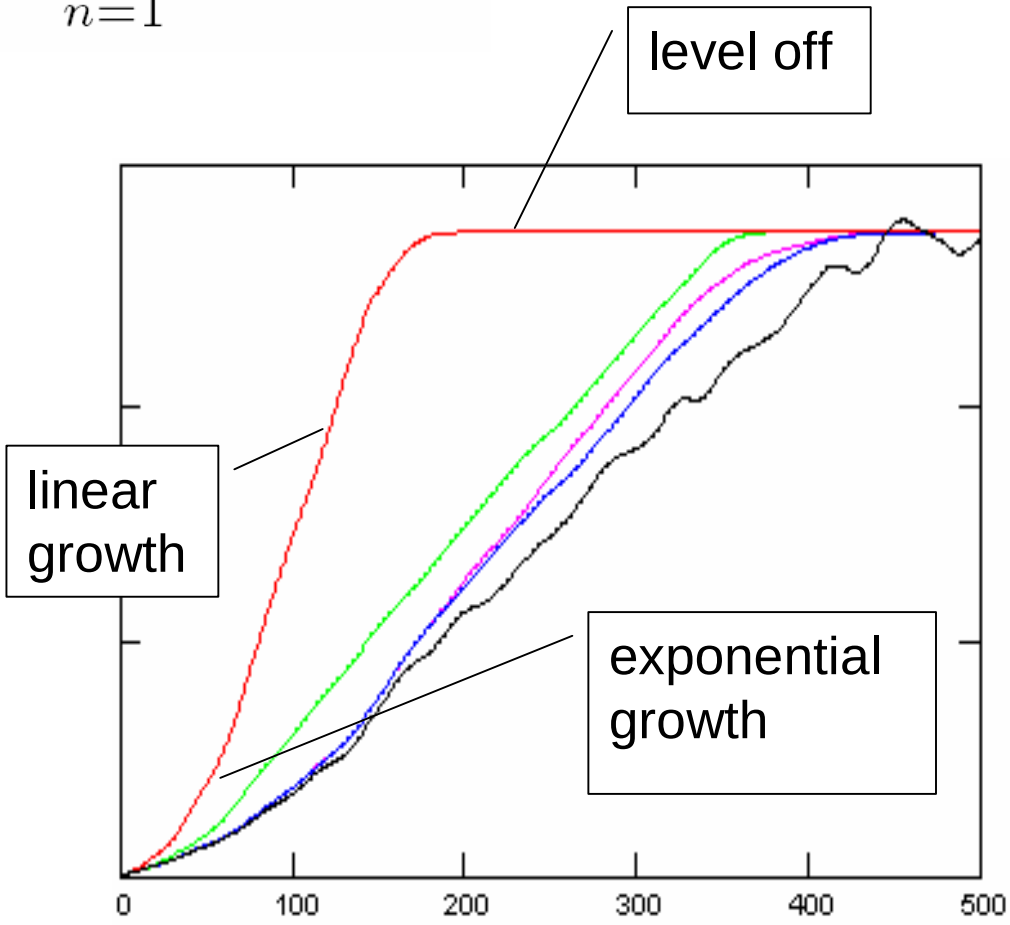
Thus

$$\lambda = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \ln\left(\frac{\epsilon_{n+1}}{\epsilon_n}\right)$$

The Lyapunov Exponent

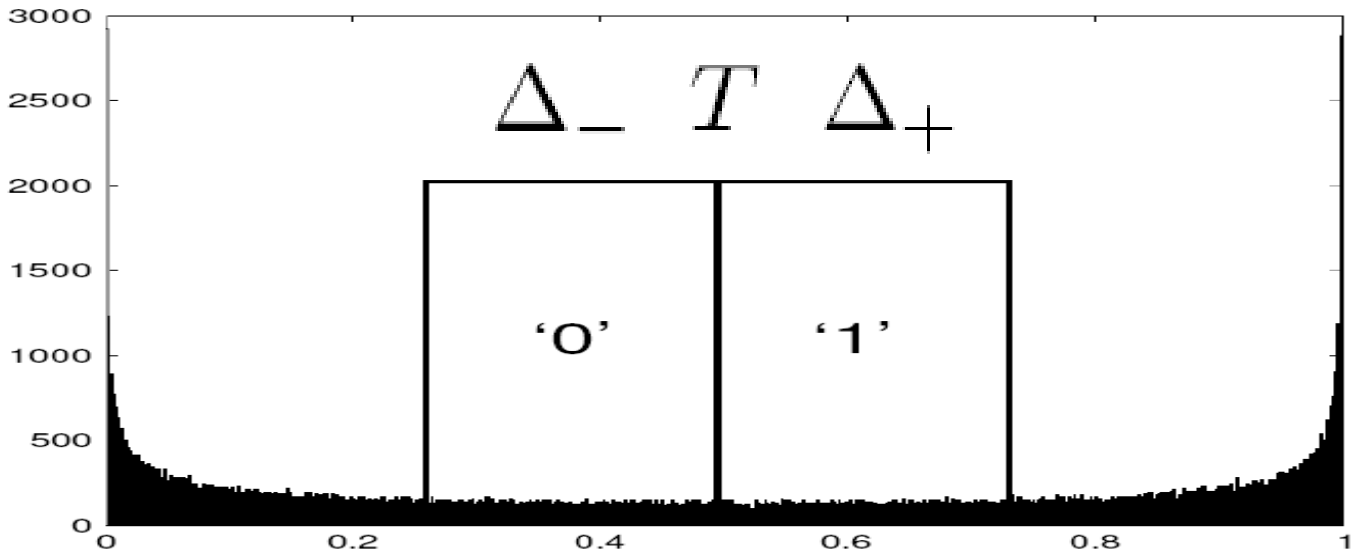
$$\lambda(x_0) = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \ln |f'(x_n)|$$

- Measures the sensitivity of an iterated function to the initial condition (key)
- Require the exponent to be:
  - >0 (chaotic behaviour)
  - approach 1 (extent of chaoticity)



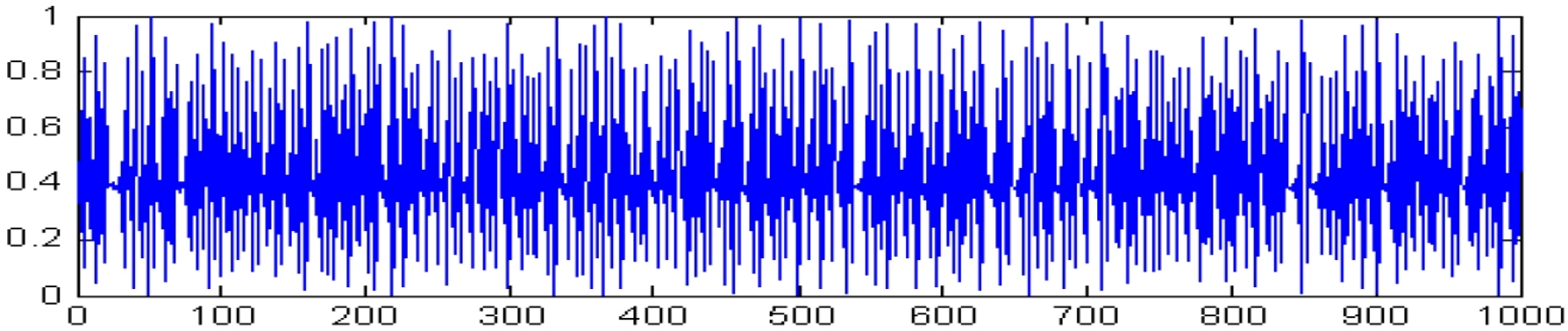
Maximum Entropy Ciphers

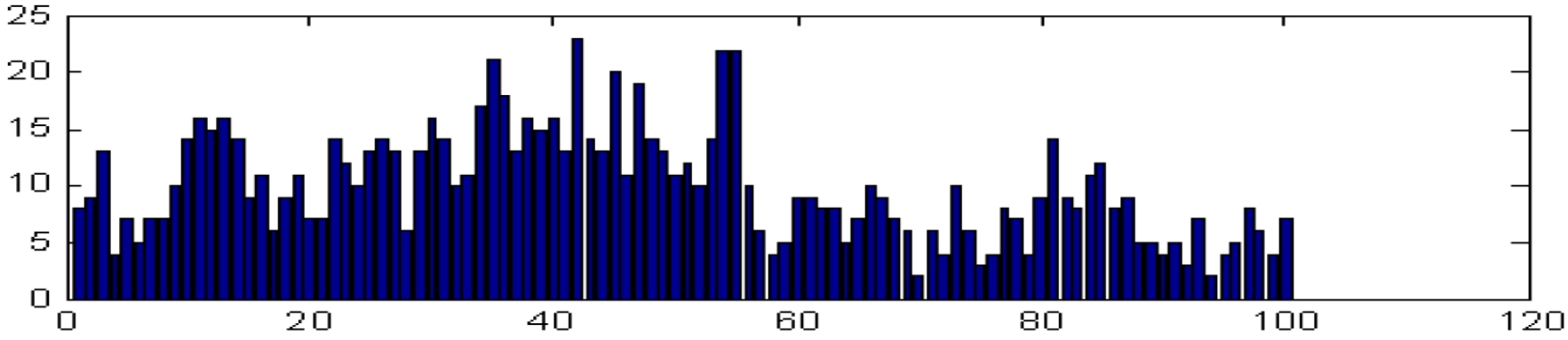
- PDFs of chaotic iterators are not uniform
- Bit stream cipher generated using a uniform PDF partitioning strategy to maximize entropy of cipher
- Encryption based on XOR operation



Example of a Chaotic Cipher with Poor Statistical Characteristics

$$x_{i+1} = r \mid 1 - \tan(\sin x_i) \mid, r = 1.5$$





Basic Design Steps

Step 1: 'Invent' a (non-linear) function f and apply the IFS $x_{i+1} = f(x_i, p_1, p_2, ...)$ such that $\mathbf{x}_\infty = 1$.

Step 2: Graph the output x_i and adjust parameters $p_1, p_2, ...$ until the output 'looks' chaotic.

Step 3: Compute the Lyaponuv Exponent and check that it is at least > 0.7.

Step 4: Check to see if cycle length of output is at least $> 10^6$.

Step 5: Graph the histogram of the output and observe if there is a significant region of the histogram over which it is 'relatively flat'.

Step 6: Set the values of the thresholds T and $\Delta_\pm$ based on 'observations' made in Step 5.

Chaos-based .v. Conventional Encryption Algorithms

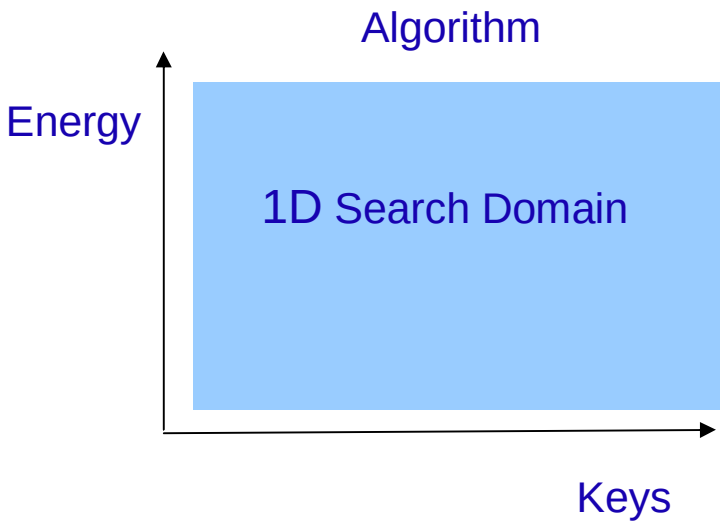
Chaos-Based Cryptography	Conventional Cryptography
Floating point arithmetic	Integer arithmetic
Computationally slow	Computationally fast
Based on any nonlinear function	Usually based on the mod function
Does not require prime numbers	Usually based on prime numbers
Low cycle lengths	High cycle lengths
Statistical bias	No statistical bias
Data redundant	Data compatible

Chaos-based cryptography has many disadvantages accept with regard to one important issue: can invent

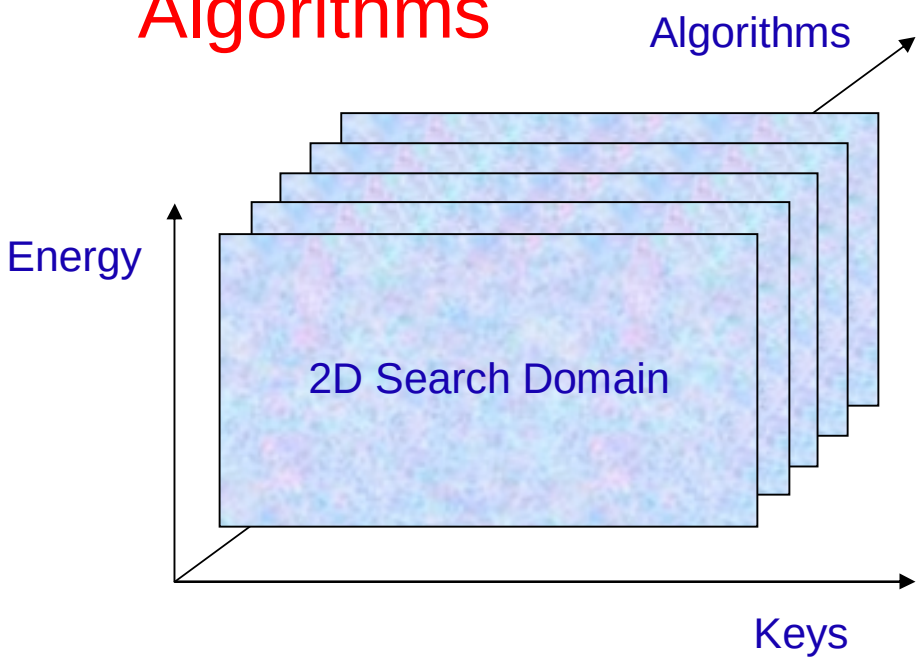
an unlimited number of algorithms

Multi-algorithmicity:
Meta-Encryption Engines

Single Encryption Algorithm



Multiple Encryption Algorithms



Chaotic Function Selection over Chaotic Block Lengths

- Analogous to the ***'M Algorithm'*** which is a method for combining multiple pseudo random streams to increase their security where one generators output is used to select a delayed output from another generator.

- The last floating point number of a current block cipher is used to seed the next block cipher

$$function[x(i)] = \begin{cases} function_{R_1}[x(i)], & i \in [1, L_1] \\ function_{R_2}[x(i)], & i \in [L_1 + 1, L_2] \\ \vdots & \vdots \\ function_{R_N}[x(i)], & i \in [L_{N-1} + 1, L_N] \end{cases}$$

Example Algorithms and Parameter Settings

Function $f(x)$	r	T	Δ_+	Δ_-
$rx(1-\tan(x/2))$	3.3725	0.5	0.3	0.3
$rx[1-x(1+x^2)]$	3.17	0.5	0.25	0.35
$rx[1-x\log(1+x)]$	2.816	0.6	0.3	0.2
$r(1-\mid 2x-1\mid^{1.456})$	0.9999	0.5	0.3	0.3
$\mid \sin(\pi r x^{1.09778}) \mid$	0.9990	0.6	0.25	0.25

Technology to License

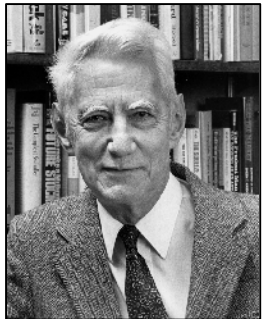
Encryption and Data Protection using Chaos

- Crypstic provides a **unique encryption engine** (unique set of algorithms) mounted on a single **pair of portable USB flash memory units**
- Includes **'Honey pot' disinformation**, e.g. other encryption applications
- **New meta-encryption engine** provided if Crypstic is compromised

Enigma

Scherbius

Shannon



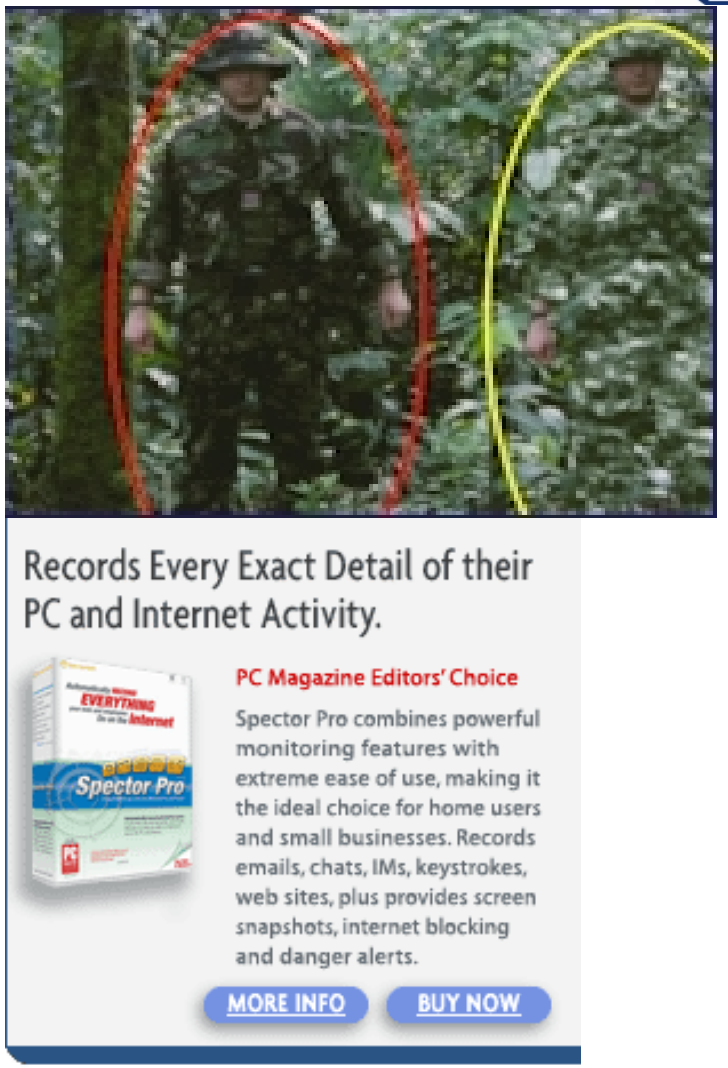
Crypstic™

Covert Access Through Obfuscation

- Camouflage encryption engine by embedding it in files of a similar type:
  a dll file

- Execution is based on renaming a known ***dll*** to a known ***exe*** file through deletion

- Requires that application is software engineered to be ***Forensically Inert***

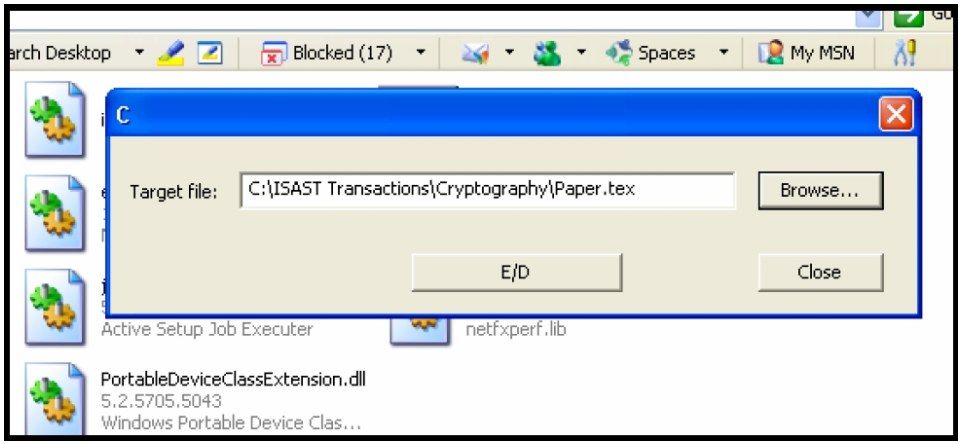


Demonstration of ***Crypstic***

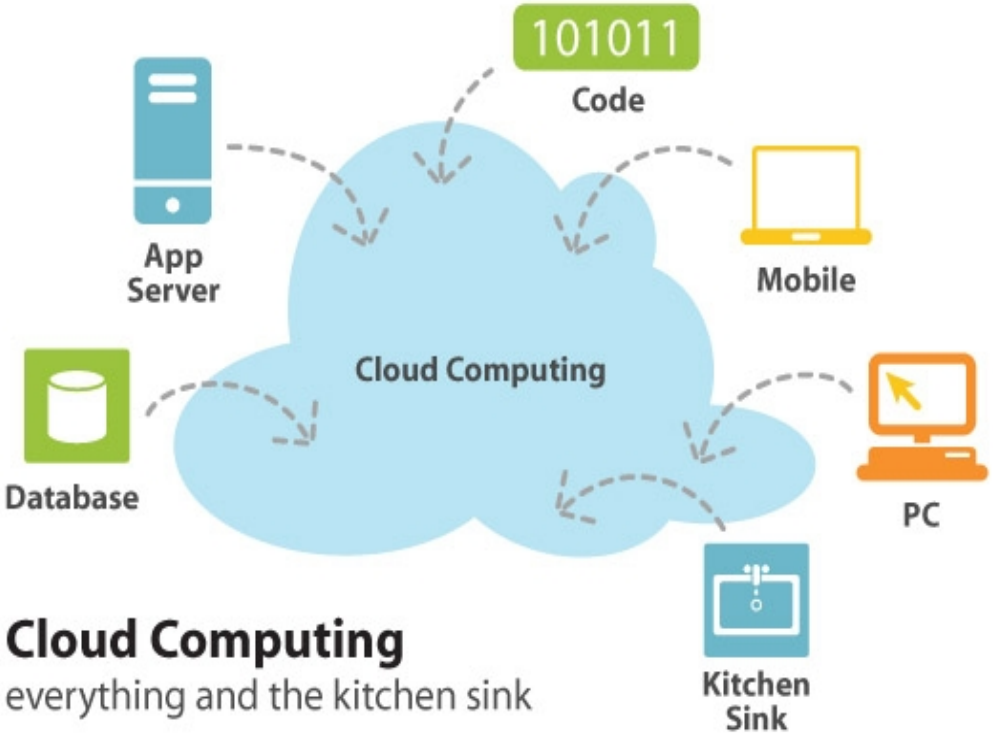
http://eleceng.dit.ie/arg/downloads/Crypstic.zip

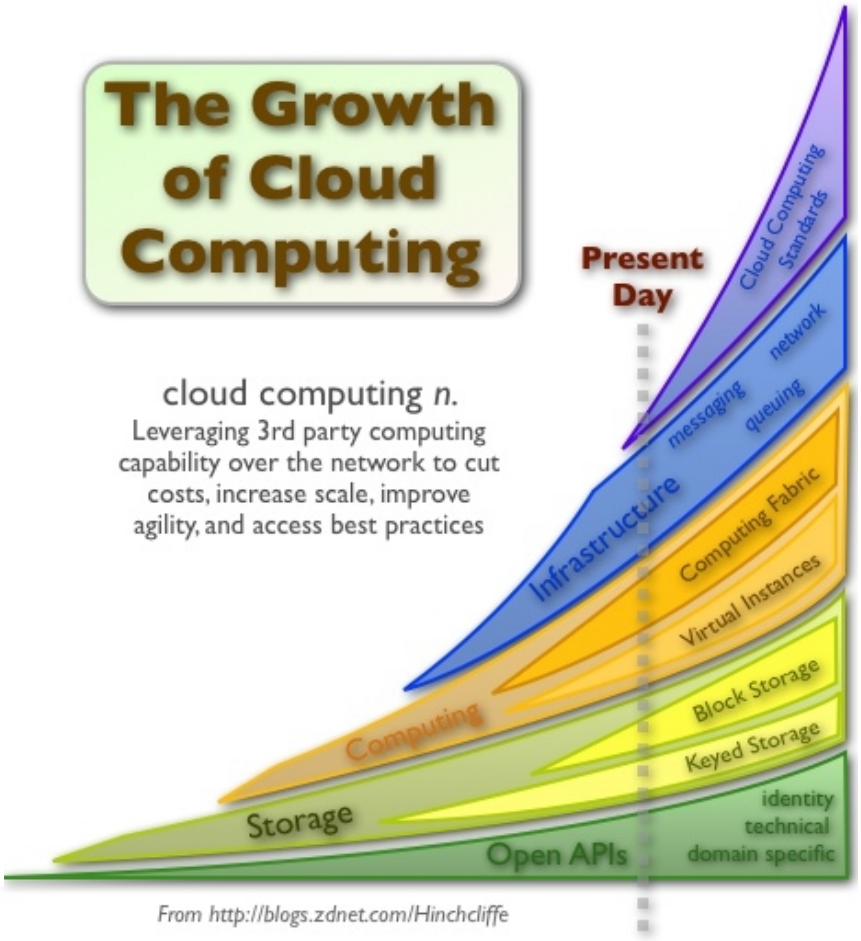
- **Multi-Algorithmic Block Encryption Engine**
  - Unique set of algorithms for each encryption engine
  - Algorithm selection & initiation seeded by file properties
  - Passes all statistical test recommended by NIST, USA

- **Implementation**
  - Flash memory
  - Forensically inert
  - Key-logging evasion



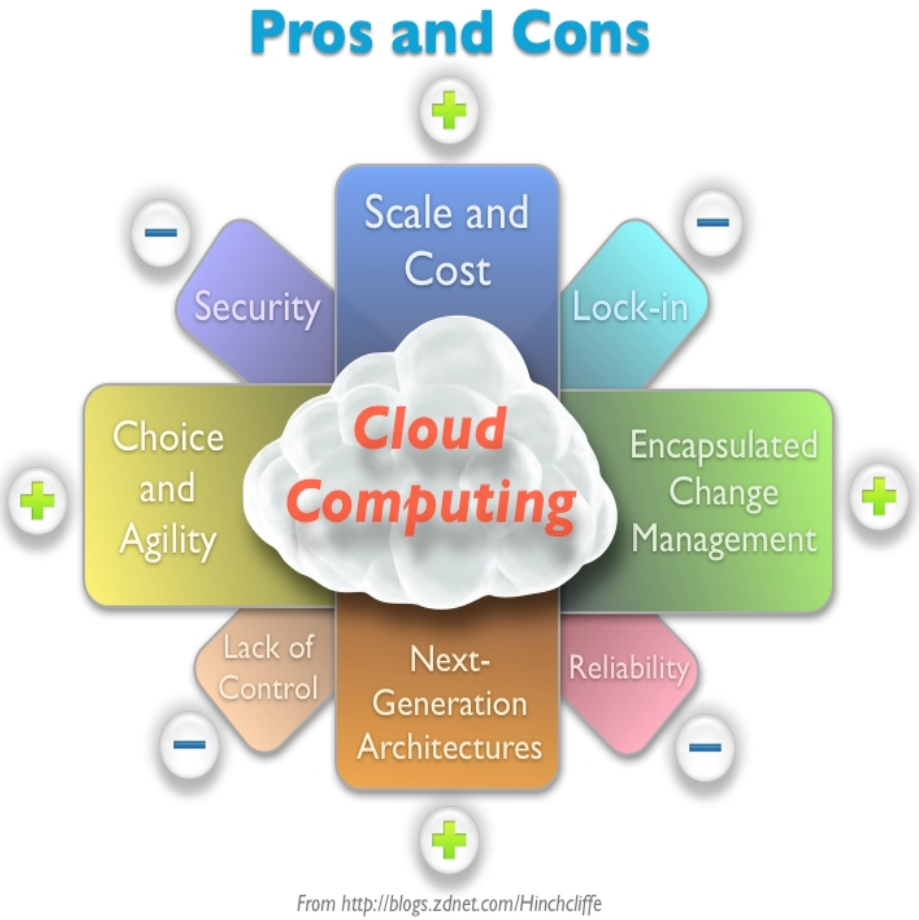
Applications to Cloud Computing





Advantages .v. Disadvantages

- Sovereignty is a potential major problem for the Cloud
- Need to treat the Cloud as a ***hostile territory***
- User-based security is the most likely solution



Cloud Security

- Cloud computing only represents 4% of current IT spend and is expected to more than double by 2012
- Software as a Service (SaaS) by itself is projected to nearly double from $9B to $17B (less than 10% of total market)
- User-security underpins acceptance of cloud architecture
- Each user has own encryption engine enabling both protection and control – ***PC + Crypstic***

Summary

- Chaos-based encryption has many disadvantages compared with conventional encryption algorithms:

  - computationally inefficient

  - low cycle lengths

- The principal advantage is that it provides the potential for developing an unlimited number of algorithms that can be used to produce a ***multi-algorithmic solution***

- Algorithms can be published so that approach conforms to the ***Kerchhoff-Shannon Principle*** in the knowledge that a **new set of chaos-based algorithms can be developed**.

Open Problems

- Structurally stable pseudo-chaotic systems

  Require a structurally stable cryptosystem, i.e. a system that has (almost) the same cycle length and Lyapunov exponents for all initial conditions. Most of the known pseudo-chaotic systems ***do not possess this property***

- Conditions of unpredictability for chaotic systems

  What properties of a chaotic system guarantee its computational unpredictability ?

Research Project Proposal 1:
Chaos based Asymmetric Encryption

- Asymmetric cryptographic systems are based on ***trapdoor functions***, i.e. functions that have a one-way property unless a secret parameter (trapdoor) is known

- No counterpart of a ***trapdoor transformation*** is, as yet, known in chaos theory

Research Project Proposal 2:
Forensically Inert Software Engineering

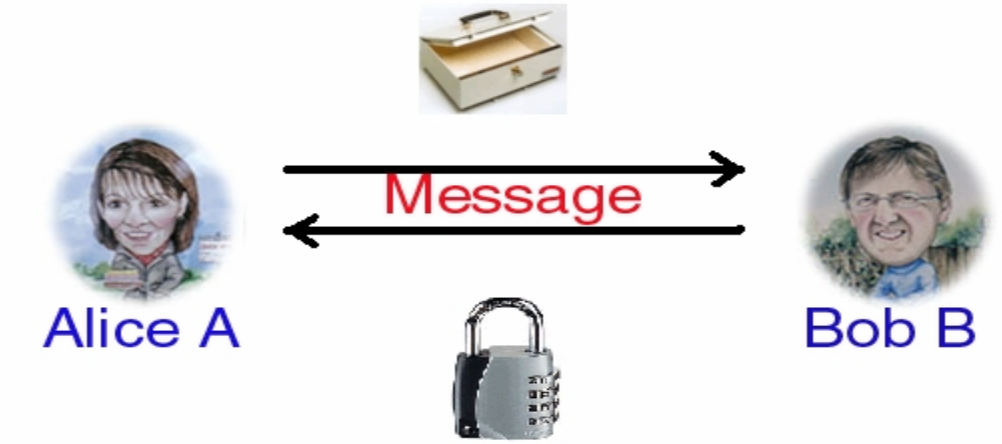
- Conventional software engineering

  Clarity > Efficiency :: Data > Process

- Forensically inert software engineering for

  Obfuscation > Efficiency :: Camouflage > Process

- ***What are the paradigms best suited to the development of forensically inert applications?***



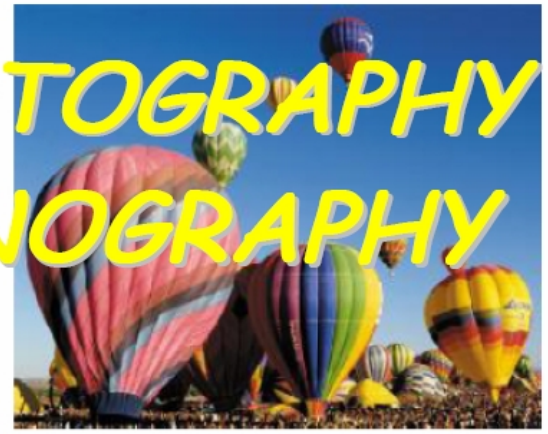
Attack ?

F7&^%p£#29hGS

Attack What ?

Have a nice day

COVERT CRYPTOGRAPHY AND STEGANOGRAPHY



Friday 12th March, 2008, 10:00 - 13:00

Institute of Heat Engineering,
Faculty of the Power and Aeronautical Engineering,
21/25 Nowowiejska St. Room TC 105 (first floor).


Dublin Institute of Technology
D·I·T
Institiúid Teicneolaíochta Átha Cliath


Politechnika Warszawska
Warsaw University of Technology


CAS

Q & A