ISCAS

Vancouver • 2004

Tutorial Notes

Tutorial 7:
Cryptography:
Circuits and Systems Approach

Presented by:
O. Koufopavlou, University of Patras
N. Sklavos, University of Patras
P. Kitsos, University of Patras

Sunday Afternoon, May 23, 13:15 - 16:15

ISCAS

Vancouver • 2004

Tutorial Notes

Tutorial 7:
Cryptography:
Circuits and Systems Approach

Presented by:

O. Koufopavlou, University of Patras

N. Sklavos, University of Patras

P. Kitsos, University of Patras

Sunday Afternoon, May 23, 13:15 - 16:15

Cryptography:
Circuits and Systems Approach

Paris Kitsos, Odysseas Koufopavlou, and Nicolas Sklavos

Electrical and Computer Engineering Department
University of Patras, Patras, Greece

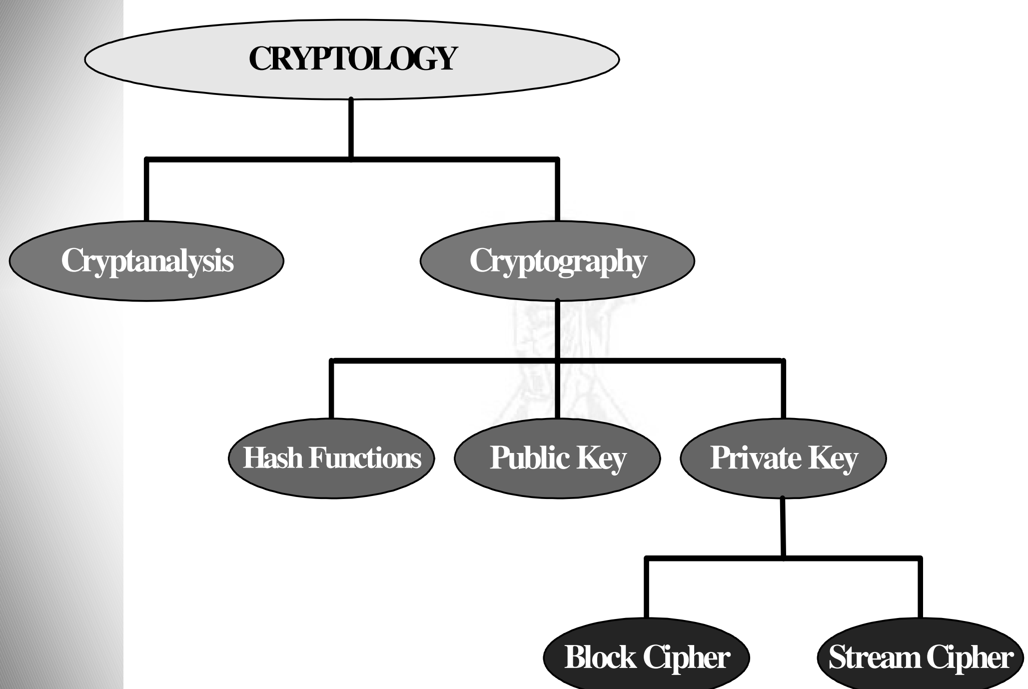
{pkitsos, odysseas, nsklavos}@ee.upatras.gr

Presentation Outline (Section I)

- Introduction to Security
- Encryption Algorithms: Basic Categories
- Wireless Networking Protocols
- Security and Wireless Communications

2

Introduction to Cryptography



3

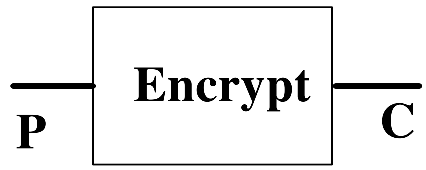
Cryptanalysis

- Used to break or attack cryptography systems
- Attack can be brute force (exhaustive search on the key-space)
- Exploit vulnerabilities in the cryptography system or the way it is used

Cryptography

- A special process of computation used to protect a message
- The “security” of the system is based on the difficulty of the “inverse” computation
- Are there full secure algorithms?

Cryptography (continued)



C = S(P)

S( ) = Encryption function

P = H(C)

H( ) = Decryption function

- The function H must be secret, otherwise it is easy to compute the message P.
- But, two people if want to have bidirectional traffic, they must share the two functions (S, and H). Is this good?

6

The Need for the Key

The answer is NO!!!! Because…..

- **First:** the description of S might be long, and hard to share
- **Second:** the description of S might be long, and hard to keep in secret

For example, can be recovered by reengineering the hardware module

- **Solution:** let S and H be public, but let H also depend on a short key K
- Easier to share, easier to keep secret (memorize, or store in hardware)

7

Type of Cryptographic Systems

- Totally Secret
- Public or Symmetric (Secret Key)
- Public Key

Totally Secret Systems

All aspects of the system are secret

- Encryption / Decryption
- The key

So one workgroup must used the same algorithm.

When a member abandon, the algorithm
must be change.

Example: The Microsoft Xbox™

Public Algorithms (Secret Key)

- The Algorithms are known, but the parameters (Keys) are secret.

$$C = S_K(P)$$
$$P = H_K(C)$$

- Use the same key (K) for both encryption and decryption.
- Consist of

  i) Block Ciphers - DES, RIJNDAEL, KASUMI

  ii) Stream Ciphers - RC4

10

Block Ciphers
Data Encryption Standard (DES)

- It is a Feistel cipher
- Based on the IBM Lucifer cipher
- Many operational modes- ECB (Electronic Codebook), OFB (Output Feedback), Cipher Block Chaining (CBC) e.t.g
- 64-bit plaintext, 56-bit key, 16 rounds.

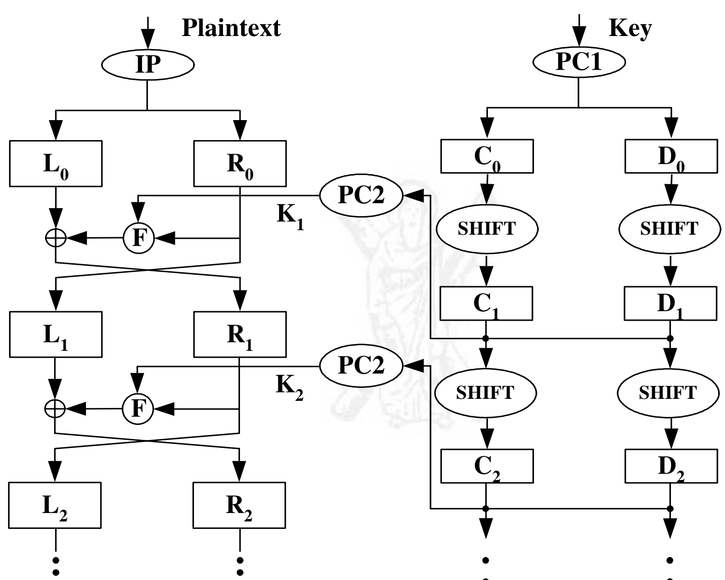
11

Feistel Cipher

- Plaintext must be even number of bits, 2n.
- Plaintext, m, is spitted into 2 halves $m=(m_0, m_1)$.
- Key has sub-keys $(K_1, K_2,\ldots,K_n)$.
- Each sub-key describes a transformation f_{ki} of n bits into n bits.
- $m_{i+1}= m_{i-1}+f_{ki}(m_i)$ where m is the output of any round.
- The same hardware is used for both encryption and decryption.

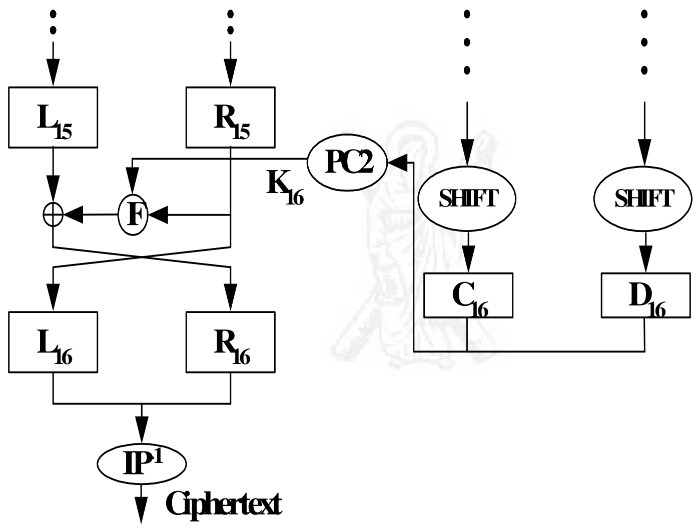
12

Data Encryption Standard (DES) (continued)



13

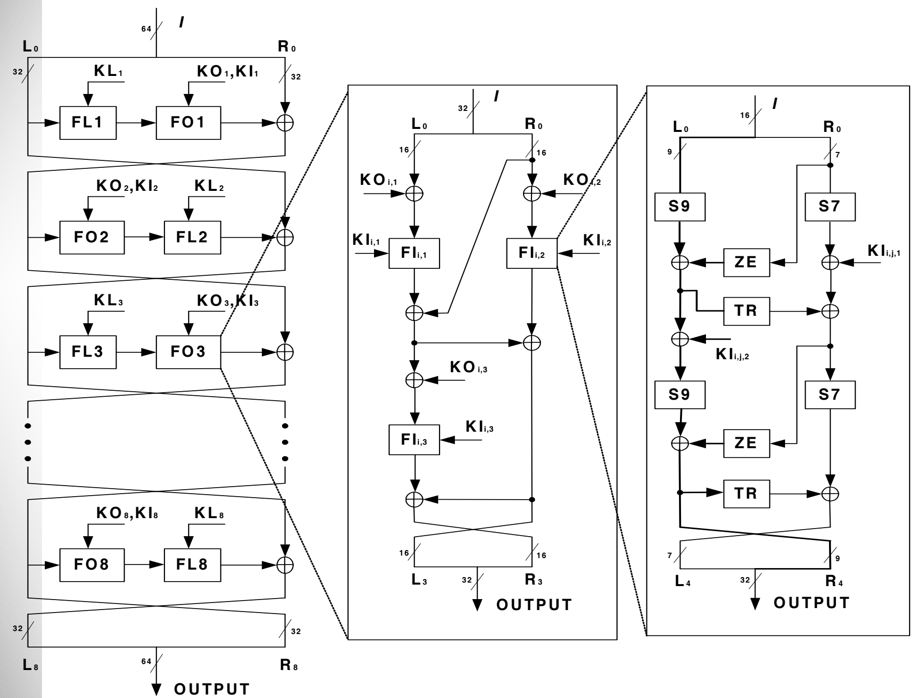
Data Encryption Standard (DES) (continued)



KASUMI

- Adopted by 3GPP as a basic component of the confidentiality algorithm *f8*, and the integrity algorithm *f9*.
- It is developed by Mitsubishi Electric
- It is a Feistel cipher.
- 64-bit plaintext, 128-bit key, 8 rounds.
- The odd rounds has different structure than even rounds.
- It is designed in order to provide security against differential and linear cryptanalysis

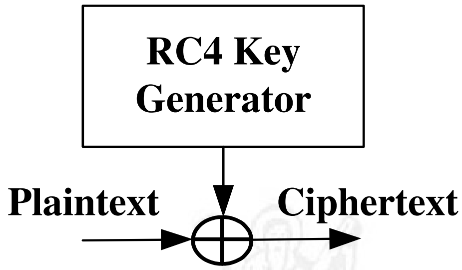
KASUMI (continued)



Stream Cipher RC4

- Developed in 1987 by Ron Rivest for RSA Data Security.
- Variable-key-size stream cipher.
- Byte-oriented operations.
- Based on the use of a random permutations.
- Works in OFB (Output Feedback) mode of operation.

Stream Cipher RC4 (continued)



$i = (i + 1) \bmod 256$

$j = (j + S_i) \bmod 256$

swap S_i and S_j

$t = (S_i + S_j) \bmod 256$

$K = S_t$

But the S_i is constructed as:

For $i = 0$ to 255;

$j = (j+S_i+K_i) \bmod 256$

swap S_i and S_j

18

Public Key Systems

- Use two keys, one for encryption (E), and one for decryption (D).

$$C = S_E(P)$$

$$P = H_D(C)$$

- Very difficult to compute the D from E.
- Each user has a public E and private D.
- Have low time performance.
- Examples:RSA, Diffie-Hellman Key Exchange

19

RSA

- Public key: Choose two integers h and n
- Plaintext: message m
- Encryption: $\mathbf{c = m^h \bmod (n)}$
- Decryption: $\mathbf{m = c^d \bmod (n)}$

where

h: Known public encryption key

d: Private decryption key

n: Known

RSA (continued)

- To generate the d and n, two prime numbers p and q such that n=pq, are chosen.
- p and q are secret
- Choose d such that

$GCD(d,\varphi(n))=1$

$\varphi(n) = (p-1)(q-1)$

φ ~ Euler's Function

Diffie_Hellman Key Exchange

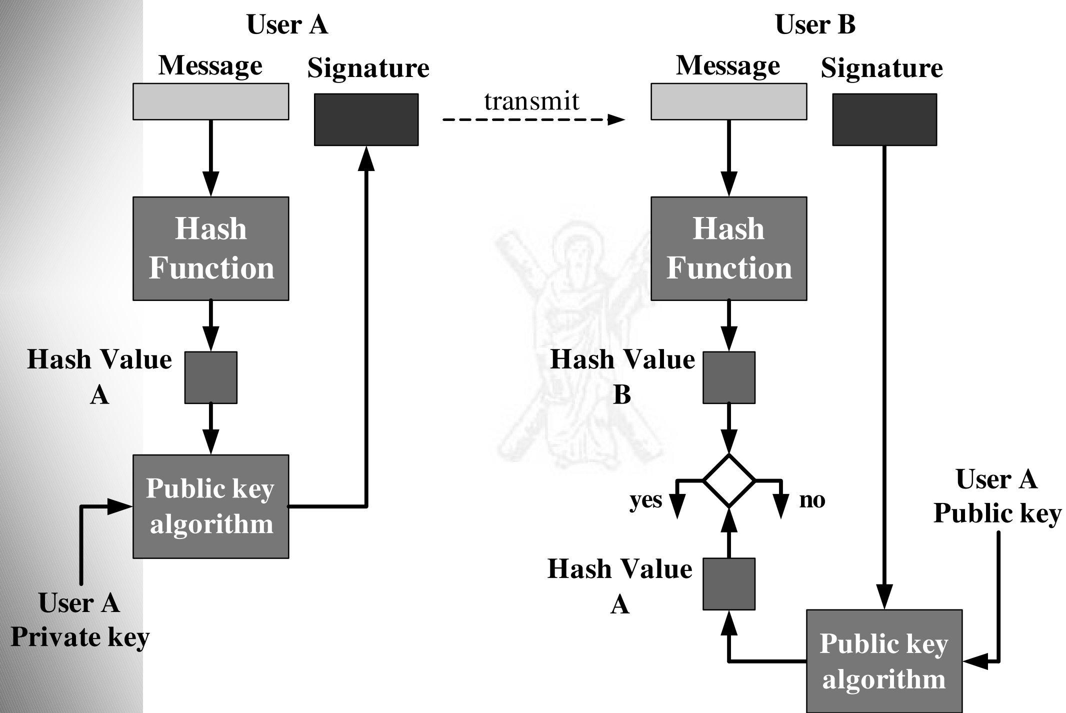
- Choose
  a prime number n, and
  a public integer number g
- User1 computes the
  $A=g^x \bmod n$ (x random integer) and send it to User2
- User2 computes the
  $B=g^y \bmod n$ (y random integer) and send it to User1
- User1 computes $k=B^x \bmod n$
- User2 computes $l=A^y \bmod n$
  $k=l=g^{xy}$ use as the key

Digital Signatures

- A copy of signed digital document is identical to the original
- Generated and verified by a public key algorithm and a hash function
- Message origin authentication and Message integrity

Schemes: Hash + Digital signature algorithm
Algorithms: RSA, DSA, EC-DSA

Digital Signatures (continued)



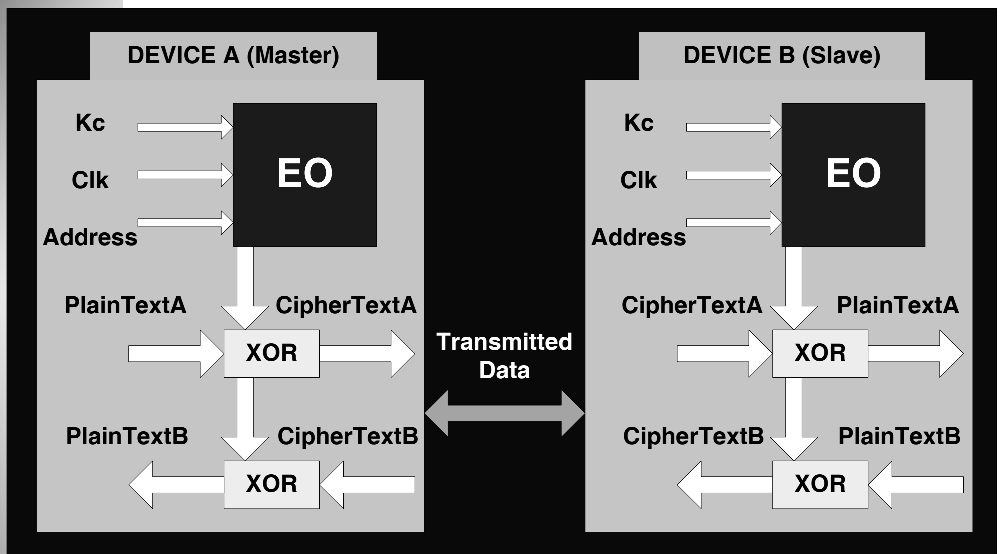
24

Wireless Communications Protocols

- **Bluetooth:**
  Eo Function, Safer+
- **Wireless Application Protocol (WAP):**
  WTLS Security Layer
- **IEEE 802.11:**
  WEP and 802.11a-d, AES and 802.11i
- **Universal Mobile Telecommunications System (UMTS):**
  Kasumi, AES
- **HIPERLAN:**
  DES, Triple-DES

25

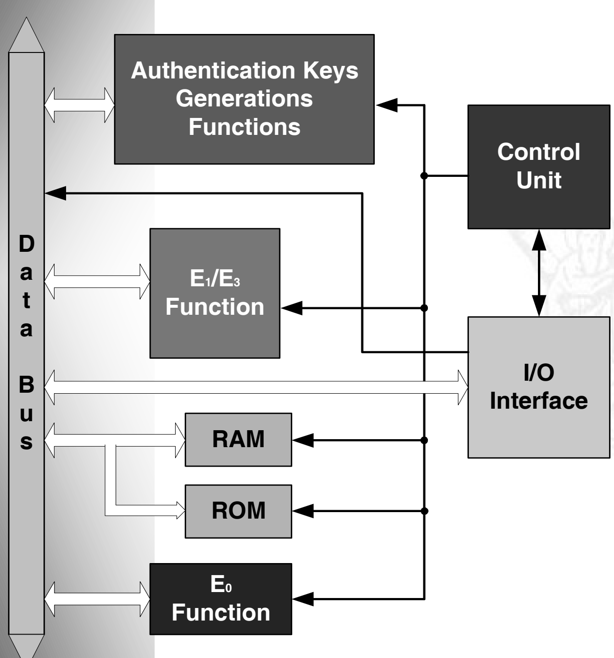
Bluetooth Security



- Eo: Encryption Decryption Scheme

26

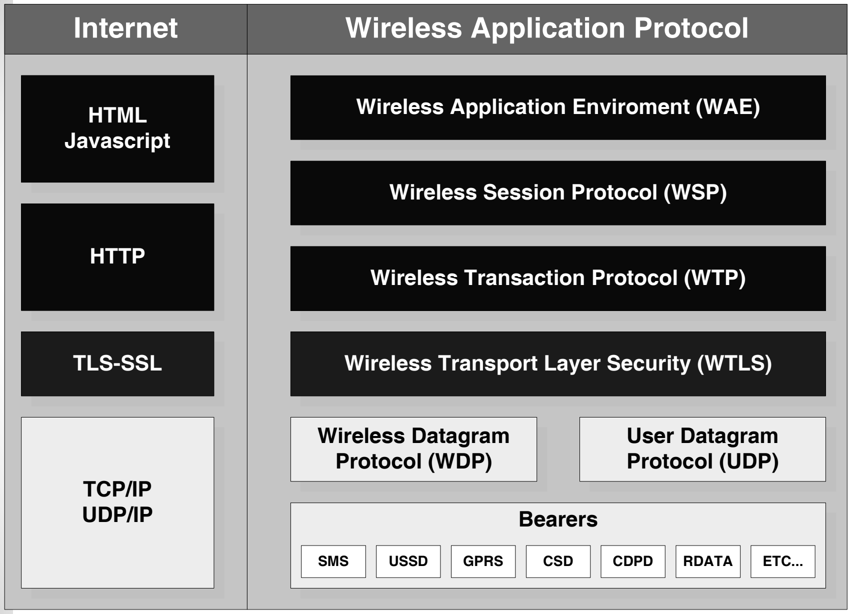
Bluetooth Security (continued)



- Safer+
- Authentication
- Encryption Key Generation

27

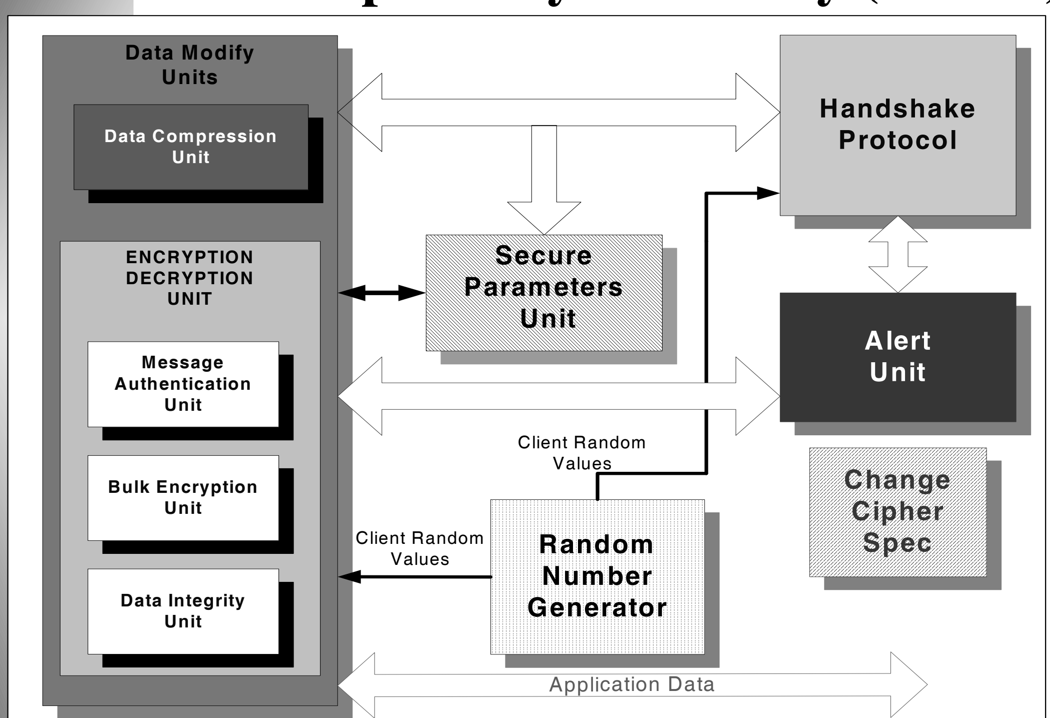
Wireless Application Protocol (WAP)



- Wireless Transport Layer Security (WTLS)

28

Wireless Transport Layer Security (WTLS)



29

Wireless Transport Layer Security (WTLS)

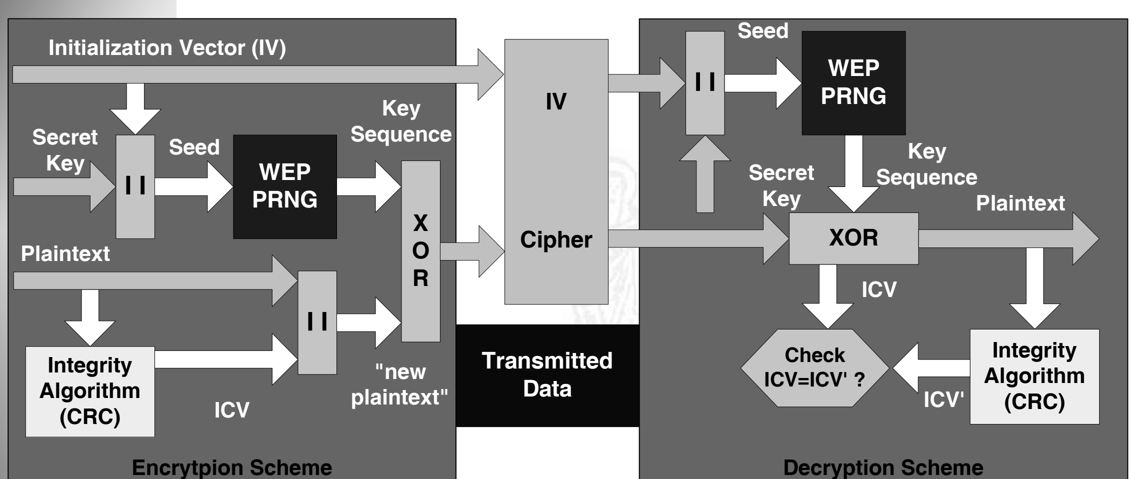
- *Bulk Encryption:*
  DES, IDEA, RC5

- *Message Authentication:*
  RSA, D-H, E.C.

- *Data Integrity:*
  SHA-1, MD5

30

IEEE 802.11 Security



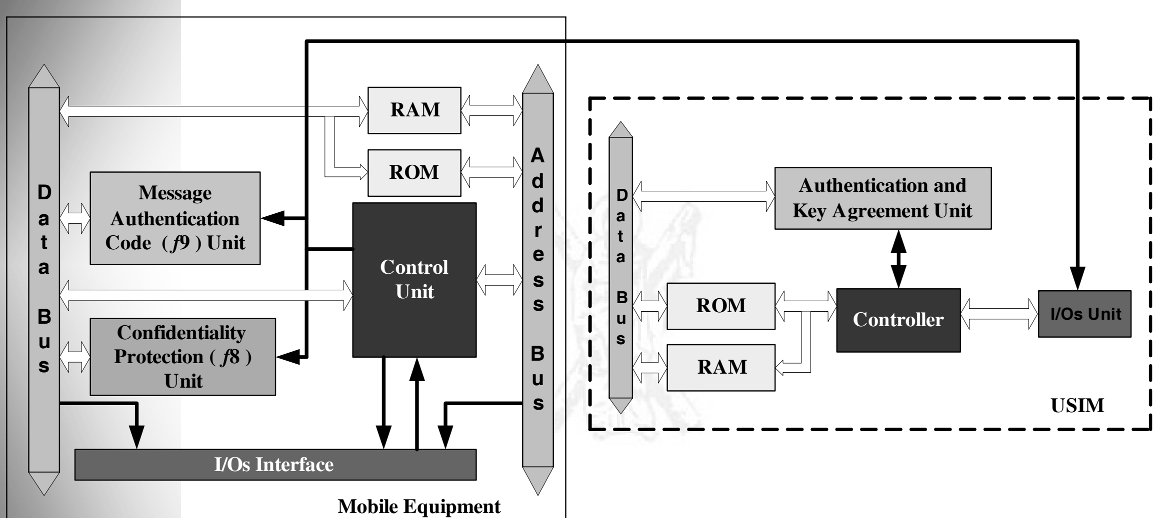
Wireless Encryption Privacy (WEP): CRC-32, RC4

31

IEEE 802.11 Security (continued)

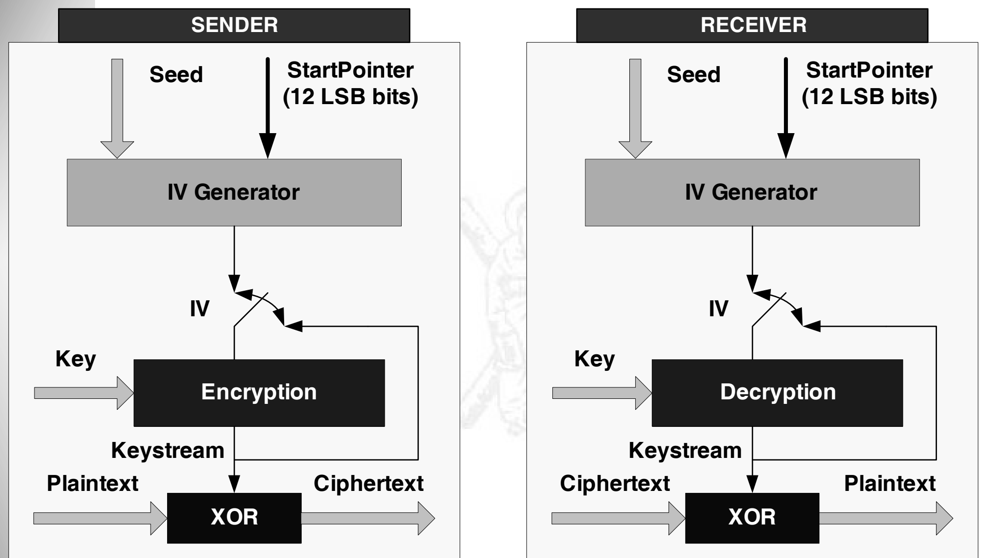
- *AES and CCM/OCB Authenticated Encryption*
  - **i) CCM mandatory**
  - **ii) OCB optional**
- *CCM: Counter Mode Encryption with CBC-MAC Origin Authentication*
  - **i) uses simple key**
  - **ii) 128-bit block cipher**
  - **iii) 802.11i uses AES as block cipher**
- *Advanced Encryption Standard (AES)*
  cipher with variable block and key length: 128, 192, 256-bit

UMTS Security



- **Kasumi, AES**

HIPERLAN Security



- DES, Triple-DES

34

Presentation Outline (Section II)

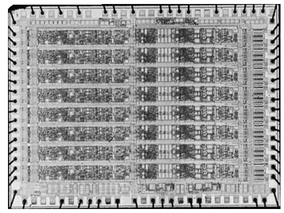
- Hardware Devices
- Implementation Parameters
- Operation Modes
- VLSI Architectures
- Hardware Implementation Performance
- New ciphers designs
- Conclusions

35

Hardware Devices

- **ASICs**

  Application Specific Integrated Circuits



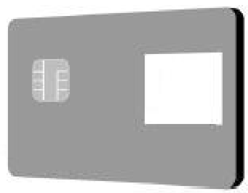
ASIC

- **FPGAs**

  Field Programmable Gate Arrays

FPGA Device (Xilinx)

- **Smart Cards**



Smart Card

Application Specific Integrated Circuits (ASICs)

- Expensive devices
- Time consuming fabrication
- Fixed implementation, no reconfiguration
- High speed performance
- Better performance.
- Smaller size.
- Higher reliability.
- Faster turnaround time.
- Tighter design security.

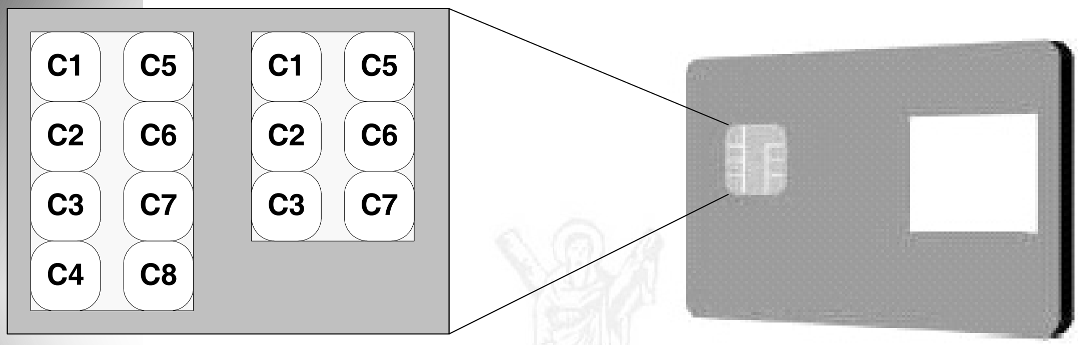
Field Programmable Gate Arrays (FPGAs)

- **Cheap devices**
- **No physical design (layout) needed**
- **Shorter design cycle and testing**
- **Reconfigured by designers**
- **Embedded RAM**
- **Customisable hardware**
- **Software-driven implementations**
- **Not suitable for large designs/functions**

Type of Cards

- **Embossed**
- **Magnetic Strip**
- **Smartcards**
- **Memory Cards**
- **Microprocessor cards**
- **Cryptographic coprocessor cards**
- **Contactless smartcards**
- **Optical memory cards**

Smartcard Electric Contacts



Contact	Abbreviation	Function
C1	Vcc	Supply Voltage
C2	Rst	Reset
C3	Clk	Clock
C4, C8	RFU	Future us
C5	Gnd	Ground
C6	Vpp	External Voltage
C7	I/O	Serial I/O

Smartcard Operating System

- **Data Transmission over bi-directional, serial terminal interface**
- **Loading, operating and management of applications**
- **Execution control and instruction processing**
- **Protected access to data**
- **Memory management**
- **File management**
- **Management and execution of cryptographic algorithms**

Parameters of Hardware Implementations

- Encryption (decryption) Throughput:

  *T= (n-bit data) * F / k clock cycles (Mbps)*

- Encryption (decryption) Latency:

  *L= (n-bit data) * number of blocks/ T (nsec)*

- Area:
  - ASIC (gates, um^2, sqmil, transistors)
  - FPGA (CLBs, equivalent logic gates)

42

Modes of Operation

- **Cryptography Operation Modes**

  1) *Electronic Codebook* (ECB)

  2) *Cipher Block Chaining Mode* (CBC)

  3) *Cipher Feedback Mode* (CFB)

  4) *Output Feedback Mode* (OFB)

  5) *Counter Mode* (CTR)

  6) *Cipher Block Chaining-Message Authentication Code Mode* (CBC-MAC)

- **Implementation Operation Modes**

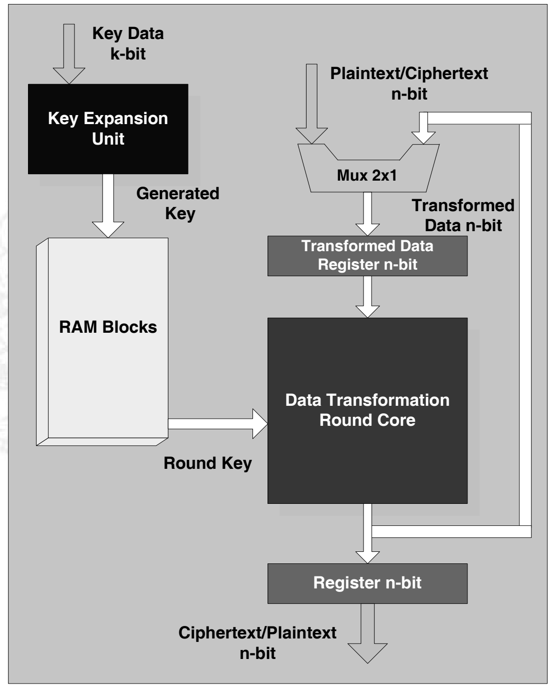
  I) *Non Feedback*: ECB and CTR

  II) *Feedback*: CBC, CFB, OFB, CBC-MAC

43

VLSI Architecture A

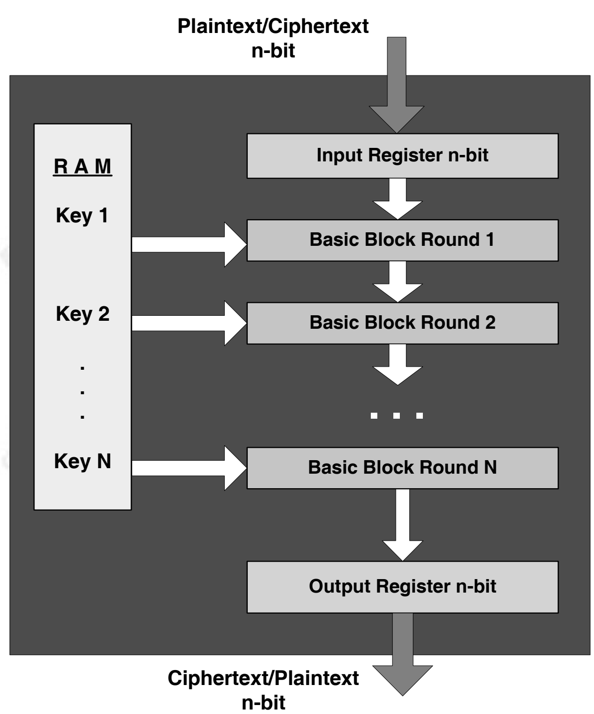
- **Full Rolling**
- **Encryption/Decryption Process**
- **Core with resources sharing**
- **One process at a time**
- **One operation mode**
- **On the fly key generation**
- **RAM for keys storage**
- **Area VS Performance**



44

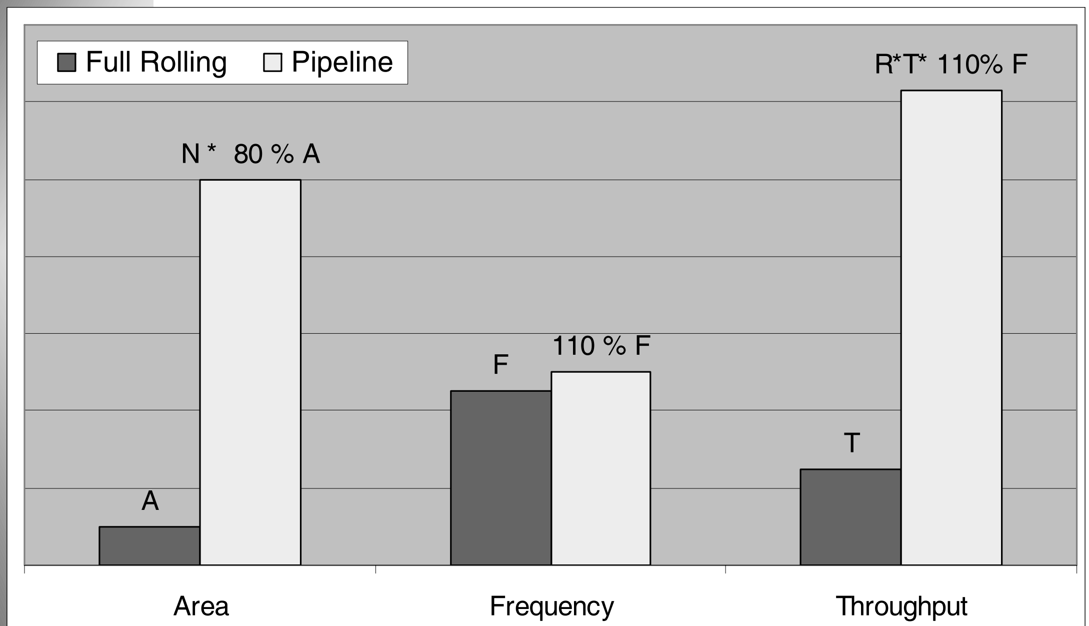
VLSI Architecture B

- **Pipeline Architecture**
- **N cascade stages**
- **(N+1) registers n-bit**
- **Precomputed keys**
- **RAM for key storage**
- **High Speed VS Area**

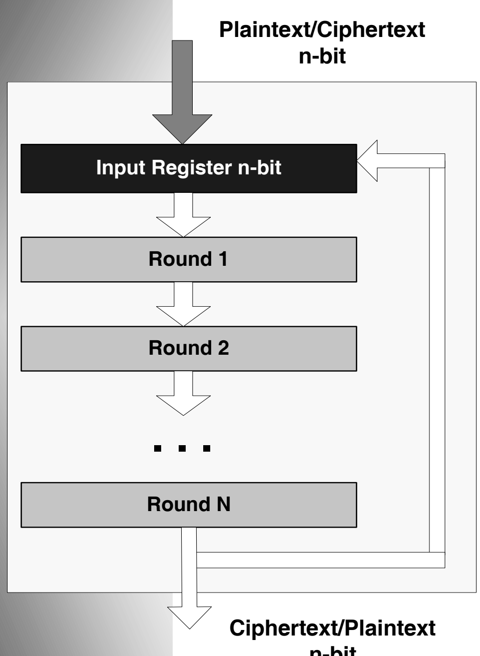


45

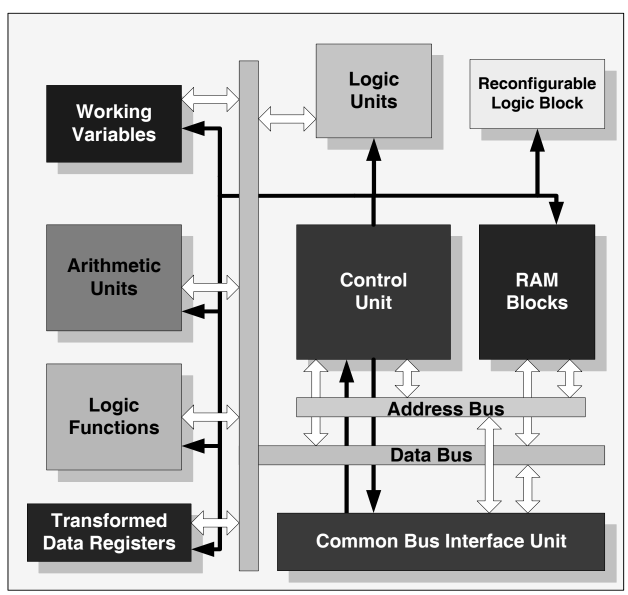
VLSI Architectures Comparisons



Other Architectures for VLSI Integration



A) Partial Loop Rolling



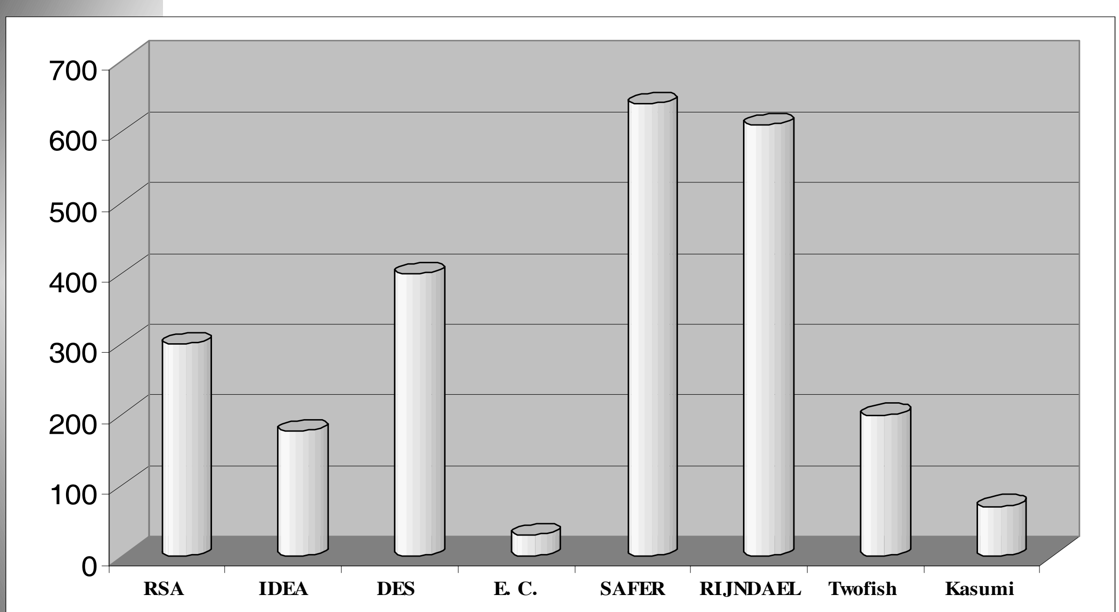
B) Typical microprocessor structure

Hardware Performance (I)

[redacted]				
RSA [1]	ASIC	47.61 mm^2	80	0.301
IDEA [2]	ASIC	50.01 mm^2	25	178
DES [3]	FPGA	741 CLBs	100	400
Elliptic Curve [4]	FPGA	1290 CLBs	45	0.031
SAFER + [5]	FPGA	6068 CLBs	50	640
Rijndael [6]	ASIC	32.50 mm^2	55	610
Triple-DES [7]	ASIC	1225 mm^2	105	6.7 Gbps
Twofish [8]	ASIC	35000 gates	66	200
Kasumi [9]	FPGA	749 CLBS	35.35	71

48

Hardware Performance (II)



49

Wireless Communications Demands

- **High Speed Performance**
- **Minimized Area Resources**
- **Limited Computing Power**
- **Low Power Consumption**
- **Bulk encryption, authentication, data integrity**
- **Selection between alternative ciphers**

50

The need for new flexible ciphers

- **Wide range of applications and usages**
- **Fast, and simple computation components (minimized area, high performance, low power)**
- **Alternative supported transformation block lengths**
- **Agile key expansion procedure**
- **Not precomputed round keys, high key refreshing**
- **Minimized RAM requirements**

51

Encryption Algorithms of the Future

1. Advanced Encryption Standard (AES)

128-, 192-, and 256-bit

2. SHA-2 Hash Family Standard

256-, 384-, and 512-bit message digest

3. DDP Ciphers: CIKS-1, SPECTR-H64

Data Depended Permutations, 64-bit

- **N.D. Goots, A.A. Moldovyan, N.A. Moldovyan,"Fast encryption algorithm SPECTR-H64", proceedings of the International workshop, Methods, Models, and Architectures for Network Security, Lecture Notes in Computer Science, Berlin, Springer-Verlag, 2001, vol. 2052, pp. 275-286.**
- **A.A Moldovyan and N.A. Moldovyan, "A Cipher Based On Data-Dependent Permutations", Journal of Cryptology, Vol. 15, pp 61-72, (2002).**

Conclusions and Outlook (Section II)

- **Hardware Devices**
- **Implementation Parameters: Wireless Protocols**
- **Operation Modes: Cryptography - Implementation**
- **Implementation Architectures**
- **VLSI Integration Performance and Cost**
- **Encryption Algorithms of the Future**

Presentation Outline (Section III)

- Introduction to Software Approach
- Which cryptographic software to use?
- Cryptography in Software
- General Optimization Principles
- Performance analysis cryptography software implementations
- Cryptography in Software – in the Future

54

Introduction to Software Approach

The needs to software approach are great

- Connection to the Internet
- E-mails
- Computer connection to a multitude of other computers through LANs.
- Encrypted UNIX, LINUX, WINDOWS client logins.
- Encrypting network traffic (Virtual Private Networks)
- Wireless Communications

55

Introduction to Software Approach

- Software can be distributed in two types
  -Binary Codes
  -Source Codes
- Both types

Binary Codes

- Run on a computer with a specific operating system.
- It is difficult to find out what it does and it can not be modified.
- Looks like this
  P^%^&^A^@^@^#øÞ^B^@øn^øn^¨^K^@^@ìÈ^@^@^F ^@^&^@^!^P^@^&^B^@^#^@$è^B^@$x^$x^Ð^*^@^ RÐ^$^@^$^F^@^!^*^D

Source Codes

- May be understandable.
- Can be easily changed.
- Needs a compiler software to translate it into binary code (for each type of machine).

58

Which cryptographic software to use?

- E-mail Encryption
  - PGP, Verising, Digital ID, GnuPG
- File Encryption
  - PGP, INFOSEC Products, GnuPG
- Drive Encryption
  - BestCrypt, PGP, Scramdisk
- Encrypted WWW Browsers
  - MSIE, Netscape, Opera

59

Which cryptographic software to use?

- IPsec Network
  - PGP, SafeNet/Soft-PK, F-Secure VPN+
- Other Networks
  - F-Secure SSH, OpenSSH, NSH

60

Cryptography in Software

The most complicate security systems are implemented in software, because….

- The rapidly known of software languages.
- The software compilers are cheap.
- The VLSI CAD tools are used by big companies.
- All security algorithms are implemented by Java, C++, and Assembly.

61

Cryptography in Software - Advantages

- New languages in order to implemented real time systems.
  Examples: ADA, Modula2, Occam
- New compilers in order to implemented real time systems.
  Example: Erlang
- There are many cryptography libraries
  Has no time delay for implementation and testing.

62

General Optimization Principles

- The knowledge of the CPU structure helps the programmers comprehend how the code will run easier.
- Careful design in order to execute more than one instructions per clock cycle.
- Avoid conditional jumps in the inner loop.
- Avoid intrinsically expensive instructions.
- Limit the number of variable.
- Allow parallelism.

63

Performance analysis cryptography software implementations

- The algorithms are implemented with Java and C++.
- The codes are executed in Pentium II/266 over LINUX operational system.
- 1 Mbyte data is used.

64

Performance analysis cryptography software implementations in Java

Algorithm	Encryption Time (msec)	Rate (Kbit/s)
IDEA	43409	193
SAFER	41442	202
Blowfish	20506	409
Triple-DES	160807	52
Loki92	31071	269
RC2	43329	193

65

Performance analysis cryptography software implementations in Java

Algorithm	Encryption Time (msec)	Rate (Kbit/s)
Square	29610	283
RC4	12945	648
DES	48629	172
CAST-5	23772	352
SHA-1	-	4.23 Mbit/s
SAFER+	-	25.6

66

Performance analysis cryptography software implementations in C++

Algorithm	Clocks/ Round	Number of Rounds	Clocks/Byte of Output
RC4	-	-	7
SEAL	-	-	4
Blowfish	9	16	18
Knufu/Khafre	5	32	20
RC5	12	16	23

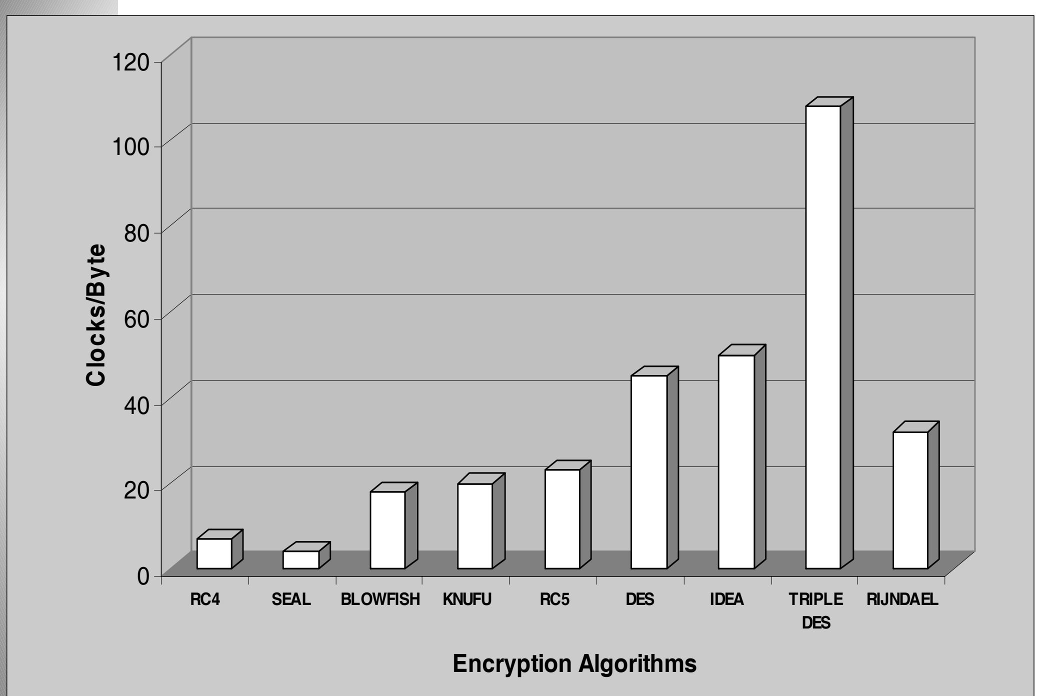
67

Performance analysis cryptography software implementations in C++

Algorithm	Clocks/ Round	Number of Rounds	Clocks/Byte of Output
DES	18	16	45
IDEA	50	8	50
Triple-DES	18	48	108
Rijndael	369	11	32

68

Another Comparison



69

Evaluation

- Software implementation usually satisfies user requirements.

  But…..
- Software is slow.
- The cryptography algorithms demands high processing capabilities.
- The implementation lead to a large code size.
- In the time critical application and processing constrained devices, the software is not preferable.

70

Cryptography in Software – in the Future

- All new algorithms are implemented in software.

  Examples: AES, SHA-2, KASUMI
- Java implementation is preferable.

  But…..
- The software implementation is easy to be broken or attacked

71


Thank you for
your attention !!!!
72