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
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

Introduction to Cryptography

CRYPTOLOGY

- Cryptanalysis
- Cryptography
  - Hash Functions
  - Public Key
  - Private Key
    - Block Cipher
    - Stream Cipher
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)

\[ C = S(P) \]
\[ S(\ ) = \text{Encryption function} \]
\[ P = H(C) \]
\[ H(\ ) = \text{Decryption function} \]

• 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?

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)
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
  1) Block Ciphers - DES, RIJNDAEL, KASUMI
  2) Stream Ciphers - RC4

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.
Feistel Cipher

- Plaintext must be even number of bits, 2n.
- Plaintext, m, is split into 2 halves m=(m_0, m_1).
- Key has sub-keys (K_1, K_2, ..., 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.

Data Encryption Standard (DES) (continued)
KASUMI

- Adopted by 3GPP as a basic component of the confidentiality algorithm $f_8$, and the integrity algorithm $f_9$.
- 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.
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)

But the $S_i$ is constructed as:

\[
i = (i + 1) \mod 256
\]

\[
j = (j + S_i) \mod 256
\]

For $i = 0$ to $255$;

\[
\text{swap } S_i \text{ and } S_j
\]

\[
t = (S_i + S_j) \mod 256
\]

\[
K = S_i
\]

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
RSA

• Public key: Choose two integers $h$ and $n$
• Plaintext: message $m$
• Encryption: $c = m^h \mod (n)$
• Decryption: $m = c^d \mod (n)$

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 $\text{GCD}(d,\varphi(n))=1$
$\varphi(n) = (p-1)(q-1)$
$\varphi \sim \text{Euler’s Function}$
Diffie_Hellman Key Exchange

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

- **Eo**: Encryption Decryption Scheme

Bluetooth Security (continued)

- **Safer+**
- **Authentication**
- **Encryption Key Generation**
Wireless Application Protocol (WAP)

- Wireless Transport Layer Security (WTLS)

<table>
<thead>
<tr>
<th>Internet</th>
<th>Wireless Application Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML Javascript</td>
<td>Wireless Application Environment (WAE)</td>
</tr>
<tr>
<td>HTTP</td>
<td>Wireless Session Protocol (WSP)</td>
</tr>
<tr>
<td>TLS-SSL</td>
<td>Wireless Transaction Protocol (WTP)</td>
</tr>
<tr>
<td>TCP/IP UDP/IP</td>
<td>Wireless Transport Layer Security (WTLS)</td>
</tr>
<tr>
<td></td>
<td>Wireless Datagram Protocol (WDP)</td>
</tr>
<tr>
<td></td>
<td>User Datagram Protocol (UDP)</td>
</tr>
</tbody>
</table>

Bearers:
- SMS
- USSD
- GPRS
- CSD
- CDPD
- RDATA
- ETC...

Wireless Transport Layer Security (WTLS)

- Data Modify Units
- Data Compression Unit
- ENCRYPTION DECRYPTION UNIT
  - Message Authentication Unit
  - Bulk Encryption Unit
  - Data Integrity Unit
- Handshake Protocol
- Secure Parameters Unit
- Alert Unit
- Change Cipher Spec
- Random Number Generator
- Application Data
- Client Random Values
- Client Random Values
Wireless Transport Layer Security (WTLS)

- **Bulk Encryption:**
  - DES, IDEA, RC5

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

- **Data Integrity:**
  - SHA-1, MD5

IEEE 802.11 Security

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

Presentation Outline (Section II)

• Hardware Devices

• Implementation Parameters

• Operation Modes

• VLSI Architectures

• Hardware Implementation Performance

• New ciphers designs

• Conclusions

• DES, Triple-DES
Hardware Devices

• **ASICS**
  Application Specific Integrated Circuits

• **FPGAs**
  Field Programmable Gate Arrays

• **Smart Cards**

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

<table>
<thead>
<tr>
<th>Contact</th>
<th>Abbreviation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Vcc</td>
<td>Supply Voltage</td>
</tr>
<tr>
<td>C2</td>
<td>Rst</td>
<td>Reset</td>
</tr>
<tr>
<td>C3</td>
<td>Clk</td>
<td>Clock</td>
</tr>
<tr>
<td>C4, C8</td>
<td>RFU</td>
<td>Future us</td>
</tr>
<tr>
<td>C5</td>
<td>Gnd</td>
<td>Ground</td>
</tr>
<tr>
<td>C6</td>
<td>Vpp</td>
<td>External Voltage</td>
</tr>
<tr>
<td>C7</td>
<td>I/O</td>
<td>Serial I/O</td>
</tr>
</tbody>
</table>

### 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:
  \[ T = (n\text{-bit data}) \times F / k \text{ clock cycles (Mbps)} \]

- Encryption (decryption) Latency:
  \[ L = (n\text{-bit data}) \times \text{number of blocks}/T \text{ (nsec)} \]

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

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
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

VLSI Architecture B

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

Other Architectures for VLSI Integration

A) Partial Loop Rolling
B) Typical microprocessor structure
### Hardware Performance (I)

<table>
<thead>
<tr>
<th>Encryption Algorithm</th>
<th>Device Type</th>
<th>Area Cost</th>
<th>Frequency (MHz)</th>
<th>Throughput (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA [1]</td>
<td>ASIC</td>
<td>47.61 mm$^2$</td>
<td>80</td>
<td>0.301</td>
</tr>
<tr>
<td>IDEA [2]</td>
<td>ASIC</td>
<td>50.01 mm$^2$</td>
<td>25</td>
<td>178</td>
</tr>
<tr>
<td>DES [3]</td>
<td>FPGA</td>
<td>741 CLBs</td>
<td>100</td>
<td>400</td>
</tr>
<tr>
<td>Elliptic Curve [4]</td>
<td>FPGA</td>
<td>1290 CLBs</td>
<td>45</td>
<td>0.031</td>
</tr>
<tr>
<td>SAFER + [5]</td>
<td>FPGA</td>
<td>6068 CLBs</td>
<td>50</td>
<td>640</td>
</tr>
<tr>
<td>Rijndael [6]</td>
<td>ASIC</td>
<td>32.50 mm$^2$</td>
<td>55</td>
<td>610</td>
</tr>
<tr>
<td>Triple-DES [7]</td>
<td>ASIC</td>
<td>1225 mm$^2$</td>
<td>105</td>
<td>6.7 Gbps</td>
</tr>
<tr>
<td>Twofish [8]</td>
<td>ASIC</td>
<td>35000 gates</td>
<td>66</td>
<td>200</td>
</tr>
<tr>
<td>Kasumi [9]</td>
<td>FPGA</td>
<td>749 CLBS</td>
<td>35.35</td>
<td>71</td>
</tr>
</tbody>
</table>

### Hardware Performance (II)

![Graph showing comparison of encryption algorithms]
Wireless Communications Demands

• High Speed Performance

• Minimized Area Resources

• Limited Computing Power

• Low Power Consumption

• Bulk encryption, authentication, data integrity

• Selection between alternative ciphers

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
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


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

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
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
Source Codes

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

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
Which cryptographic software to use?

- **IPsec Network**
  - PGP, SafeNet/Soft-PK, F-Secure VPN+

- **Other Networks**
  - F-Secure SSH, OpenSSH, NSH

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.
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.

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.
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.

Performance analysis cryptography software implementations in Java

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Encryption Time (msec)</th>
<th>Rate (Kbit/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDEA</td>
<td>43409</td>
<td>193</td>
</tr>
<tr>
<td>SAFER</td>
<td>41442</td>
<td>202</td>
</tr>
<tr>
<td>Blowfish</td>
<td>20506</td>
<td>409</td>
</tr>
<tr>
<td>Triple-DES</td>
<td>160807</td>
<td>52</td>
</tr>
<tr>
<td>Loki92</td>
<td>31071</td>
<td>269</td>
</tr>
<tr>
<td>RC2</td>
<td>43329</td>
<td>193</td>
</tr>
</tbody>
</table>
### Performance analysis cryptography software implementations in Java

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Encryption Time (msec)</th>
<th>Rate (Kbit/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>29610</td>
<td>283</td>
</tr>
<tr>
<td>RC4</td>
<td>12945</td>
<td>648</td>
</tr>
<tr>
<td>DES</td>
<td>48629</td>
<td>172</td>
</tr>
<tr>
<td>CAST-5</td>
<td>23772</td>
<td>352</td>
</tr>
<tr>
<td>SHA-1</td>
<td>-</td>
<td>4.23 Mbit/s</td>
</tr>
<tr>
<td>SAFER+</td>
<td>-</td>
<td>25.6</td>
</tr>
</tbody>
</table>

### Performance analysis cryptography software implementations in C++

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Clocks/ Round</th>
<th>Number of Rounds</th>
<th>Clocks/Byte of Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC4</td>
<td>-</td>
<td>-</td>
<td>7</td>
</tr>
<tr>
<td>SEAL</td>
<td>-</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>Blowfish</td>
<td>9</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td>Knufu/Khafre</td>
<td>5</td>
<td>32</td>
<td>20</td>
</tr>
<tr>
<td>RC5</td>
<td>12</td>
<td>16</td>
<td>23</td>
</tr>
</tbody>
</table>
Performance analysis cryptography software implementations in C++

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Clocks/ Round</th>
<th>Number of Rounds</th>
<th>Clocks/Byte of Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES</td>
<td>18</td>
<td>16</td>
<td>45</td>
</tr>
<tr>
<td>IDEA</td>
<td>50</td>
<td>8</td>
<td>50</td>
</tr>
<tr>
<td>Triple-DES</td>
<td>18</td>
<td>48</td>
<td>108</td>
</tr>
<tr>
<td>Rijndael</td>
<td>369</td>
<td>11</td>
<td>32</td>
</tr>
</tbody>
</table>

Another Comparison

![Graph showing encryption algorithms comparison](image_url)
Evaluation

- Software implementation usually satisfies user requirements.
- 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.

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
Thank you for your attention !!!!