Configurable Security Modules for SoC Designs

Ruby B. Lee and Zhijie Shi
Princeton University

Emerging Information Technology Conference,
Princeton, NJ, Nov. 1-2, 2002

Outline

• Motivation
• Past methods
• Structure of configurable security modules
• Function of configurable security modules
• Configuration procedures
• Other features
• Summary
Motivation

- Many types and levels of security needed
- Common security needs for many applications
  - Protect private data
  - Authenticate users to remote server
- Security is complex, and there are many past mistakes

Example of past mistakes

- X-Box
  - Allow only users who have paid for the game software to run it on the game machine
  - Allow safe distribution of game over Internet
  - Encrypt the game once (not for each user)
  - Broken!
    - Same symmetric key used to encrypt all games
    - Game key store in every X-Box
Past methods

- High-end secure co-processor
  - IBM 4758 secure co-processor in sealed PCI card
    - Expensive
- Smart cards
  - Limited computational power
  - Small memory
- System-on-Chip
  - IP-core for each cryptographic algorithm
    - Not flexible
    - No secure storage for secrets
    - No secure configuration

Configurable security module

- Programmable crypto processor
  - For any crypto algorithms
- Optional IP-cores to accelerate crypto algorithms
- ROM
- Non-volatile memory
  - Secure, e.g. for keys and some code
  - Non-secure
- Configuration registers
- Initialization, configuration and update protocol
Non-volatile memory in the security module

- ROM
  - Power-on Self-test code
  - Basic set of crypto algorithms

- Flash memory
  - Secrets
    - Keys
  - More crypto algorithms

- Battery-backed SRAM
  - Easy to destroy sensitive data
  - Need a battery attached to the system

Security functions

<table>
<thead>
<tr>
<th>Security Function</th>
<th>Crypto Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidential messages</td>
<td>Symmetric key algorithms</td>
</tr>
<tr>
<td></td>
<td>DES, 3DES, AES</td>
</tr>
<tr>
<td>Secret key exchange over public networks</td>
<td>Diffie-Hellman, RSA</td>
</tr>
<tr>
<td>User authentication</td>
<td>Public-key algorithms</td>
</tr>
<tr>
<td></td>
<td>RSA, ECC</td>
</tr>
<tr>
<td>Data integrity</td>
<td>Hash algorithms</td>
</tr>
<tr>
<td></td>
<td>SHA-1, MD5</td>
</tr>
</tbody>
</table>
Basic security functions implemented by the security module

- Symmetric key algorithms
  - 3DES/AES
- Public key algorithms
  - RSA
- Hash functions
  - SHA-1/MD5
- Key generation
  - Random number generator (RNG)
- Secure configuration
  - Secure initialization
  - Secure bootup
  - Secure configuration and update in the field

Configure the security module

- Configuration at the design time
- Factory initialization
- Owner initialization
- Reconfiguration or update in the field
Configuration of security module at design time

- Crypto algorithms in ROM
- Optional crypto accelerators
- Hardware memory protection mechanism
  - Memory lock
- Power-On Self-Test options
**Configure at factory initialization**

- The device may generate the keypair itself
  - More expensive

**Configure at Owner initialization**

- Establish ownership with a (public, private) keypair different from device’s keypair
  - By trusted third party (TTP) or PKI
  - By manufacturer
    - Keys are generated and saved in the device in factory
    - Keys are generated by owner
- Owner’s keypair must be certified
- Owner’s keypair are stored in secure non-volatile memory
  - Private key is encrypted with a passphrase
Authenticate owner to device

- What you know?
  - Shared secrets (what you know)
    - PIN or passphrase
- What you have?
  - Public key algorithms
    - RSA key in smart card (smart ring)
- Who you are?
  - Biometric
    - Thumbprint
    - Handprint
    - Speaker identification
    - Retina scan

Configure/update in the field

- Install/upgrade applications
- Reconfigure in the field
- Specify or change authentication method
- Change authentication data
  - PIN
  - Keypair
- Change the ownership
  - Give device to another person
  - Destroy all owner related secrets
Install applications

- The owner downloads the application into volatile memory
- The owner checks the hash value and the signature of the application

<table>
<thead>
<tr>
<th>Code</th>
<th>Signed Hash Value</th>
</tr>
</thead>
</table>

- If the owner trusts the signer, he issues the command to install the application
  Installation must be an atomic operation

Protected storage for many keys

Root key

- Key1
- Data1
- Key2
- ..........  
- Key21
- Data21
- Data22

Encrypted data can be stored anywhere
Secrets in the secure non-volatile memory

- Keypair of the device
- Keypair of the owner
  - Data may be shared with other devices
  - A user may have more than one device
  - The owner may recover the data even if the device key is destroyed
- Root key for “protected storage”
  - Used to encrypt data to be stored outside in SOC
    - Do not have to re-encrypt all data when user’s key changes
  - Generated inside the security module
    - Saved in secure memory, or protected by owner’s keypair

Hardware memory lock

- For further protection of memory
  - Optional
- Software may have bugs even if signed by a trusted party
- Use hardware memory lock to protect memory

Layer 0: Code in ROM
Layer 1: Software 1 (OS Loader)
Layer 2: Software 2 (OS)
Layer 3: Software 3 (Apps)
Summary of configuration options

- Configure at design time
  - Secure memory
  - Crypto accelerators
  - Hardware memory lock
- Configure at factory initialization
  - Generation of device master key
  - Choice of crypto algorithms
- Configure at owner initialization
  - Owner's information (e.g. owner's keypair)
- Configure in the field
  - Authentication method
  - Authentication data
  - Code upgrading
  - Change ownership

Conclusion

- SoC vendor can build in security functions without becoming security expert
- Security functions are used by third-party software/service providers
- We provide a security IP module with
  - Secure memory for secrets and important code
  - Basic set of crypto algorithms for important security functions
  - Reconfigurable options
  - Secure bootup
  - Scalable in cost, performance, size and power