Asymmetric cryptography for SEcube

Cybersecurity for Embedded Systems

Brignone Giovanni Castagneri Dario Licastro Dario

September 13, 2021

Politecnico di Torino

Outline

1. Introduction
2. Development
3. Demo

Introduction

Cryptography

- Science behind multiple aspects of information security
- Central operations:
    - Encryption
    - Decryption
- Two approaches:
    - Symmetric
    - Asymmetric

RSA algorithm

- Asymmetric key algorithm published in 1977
- Keys are derived from two prime numbers
- Security relies on the difficulty of factorizing large numbers
    - Increase key size to improve encryption strength
- Applications:
    - Key distribution
    - Digital signature

Digital certificates

- Check identity and guarantee secure communications
- Signed by a certificate authority or self-signed
- X.509 is a standard format for public key certificates

Goal of the project

- Extends the SEcube SDK
    - RSA-based asymmetric cryptosystem
        - Key storage
        - Symmetric key distribution
        - Digital signature
    - Digital certificates based on X.509 format

Development

HW/SW partitioning

- **Initial idea**:
  - HW: RSA functionality
  - SW: Drivers and APIs
- **Issue**: Resource constraints: small FPGA (7000 LUTs) and long keys (1024+ bits)
- **Design exploration**: Map to HW most critical parts of design only: Encryption/Decryption → Modular exponential → $\underline{\text{Modular multiplication}}$

HW/SW partitioning

- **Initial idea**:
    - HW: RSA functionality
    - SW: Drivers and APIs
- **Issue**: Resource constraints: small FPGA (7000 LUTs) and long keys (1024+ bits)
- **Design exploration**: Map to HW most critical parts of design only: Encryption/Decryption → Modular exponential → <u>Modular multiplication</u>
- **Solution**: No suitable architecture in literature ⇒ Full SW implementation

Firmware side: RSA and X.509 library

- Compatible with STM32F4429 micro controller:
  - Written in C
  - Low resources usage
- Compatible with SEcube Open Source project:
  - Permissive license
- Secure:
  - Reliable developer
  - Widely used

Firmware side: RSA and X.509 library

- Compatible with STM32F4429 micro controller:
  - Written in `C`
  - Low resources usage
- Compatible with SEcube Open Source project:
  - Permissive license
- Secure:
  - Reliable developer
  - Widely used

⇒ `mbedtls` by ARM

Firmware side: Flash memory

RSA keys:

- Problem: Share same ID space of symmetric keys
- Solution:
  - Pack RSA keys (multiple fields) into symmetric keys nodes (single field)
  - Reuse symmetric keys code

Firmware side: Flash memory

RSA keys:

- Problem: Share same ID space of symmetric keys
- Solution:
  - Pack RSA keys (multiple fields) into symmetric keys nodes (single field)
  - Reuse symmetric keys code

X.509 certificates:

- Problem: No previous support for any kind of certificate
- Solution: Dedicated node type and functions

Firmware side: Dispatcher

- **Problem**:
  Manage RSA/X.509 requests from host without cluttering the *Dispatcher Core*
- **Solution**:
  RSA Dispatcher receives all the related requests and calls specific functions

Firmware side: Dispatcher

- **Problem**:
  Manage RSA/X.509 requests from host without cluttering the *Dispatcher Core*
- **Solution**:
  RSA Dispatcher receives all the related requests and calls specific functions
- Security enhancement:
  - Each key has a type (generic, crypto-only, sign-only)
  - Forbidden operations are blocked before execution

Host side: L1 API

- **Problem**:
  - Expose RSA and X.509 functionalities through APIs
  - Integrate new APIs with pre-existing ones
- **Solution**:
  - Reuse existing APIs (e.g. `L1FindKey`)
  - Extend existing APIs (e.g. `L1KeyEdit`, `L1Encrypt`, `L1Decrypt`...)
  - Add new APIs (e.g. `L1Sign`, `L1Verify`...)

Demo