Understanding & Implementing Embedded System Security

May 25, 2016

Presentation objectives, audience & requirements

• Objectives
  • Learn the basics of security & related technologies applicable to MCU-based embedded systems
  • Learn how to address various security requirements using NXP products

• Target Audience
  • Anyone who is interested in security and how it applies to embedded systems

• Requirements
  • Basic embedded systems knowledge
Topics

• Security goals for embedded systems
  • Data Integrity, Code Integrity & Device Integrity

• Basics of cryptographic algorithms

• Details of each goal & various technologies available to achieve them

• How NXP LPC18S/43S MCUs and A7 secure element can address security challenges

• Design Examples of highly secure devices

WHY IS SECURITY NECESSARY?
Use case: A growing security threat from the cloud

25% of installed 802.15.4 edge nodes will be IPv-based (Thread, ZigBee & IPv6), thus exposed to cloud attacks in 2019

4M of mobile malware installation packs seen in 2014

Remote attacks on home devices (gateway or IP edge nodes) from cloud

Remote attacks on Cloud Service Provider

Physical attack (e.g., using side channel) on peripheral systems (e.g., door lock)

Vulnerable or compromised device from unknown origin compromising home network from inside

Attack from infected Smart phones/watch (rogue App, Pin code phishing, etc)

Decommissioned devices used as Trojan horse to compromise other networked devices

Why Security in Building Automation?

Lighting
- Serious injury if turned off
- Trip hazard
- Panic condition

Environmental
- Malicious hack of thermostat
- Building damages can cost millions

Health Issues
- Tampering with industrial coolers
- Many people can get sick

Brand Issues
- Bad publicity
- Building manager reluctant to use
Vulnerable IoT Devices (1/3)

Web Sites Publish Vulnerable Equipment

- Hackers know exactly where to start
- Anyone can view the equipment to see what is happening
- No user fee required to access the data

Vulnerable IoT Devices (2/3)

Access Points
- Lists company where located
- City
- Passwords to gain access

Products Hacked
- Lists manufacturers
- How many times it shows up in a location
Vulnerable IoT Devices (3/3)

**Well Known Names**

- Specific instructions on how to hack
- Locations posted

**Keyboard Sniffer**

**Published Reference Design**

- Disguised as a USB wall charger
- Captures all Bluetooth keyboard strokes
- Stores keystrokes locally or sends them over the internet
- Published on microcontroller blog site
- BOM cost is $10
IoT Is Being Recognized

IoT
• More than a buzzword
• Customers are recognizing it
• Many products being designed

Common Platforms
• Easy to develop with
• Commonality between vendors
• Also means hackers know vulnerabilities

Electronic Toaster

Published Article
• Attaches costs associated with not securing a device
• Any device connected to the Internet is vulnerable
• Misfortune Virus
  • Released in 2002
  • Fixed in 2005
  • As of 2015, still infects half of all public servers!
**Hardware Secure Element**

A robust, proven security Solution

NXP has shipped nearly 8 billion Secure Elements in Bank chip cards, transport card, ePassports. This same proven technology and authentication approach is used to secure and authenticate IoT devices.

---

**Levels of IoT Security**

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3</td>
<td>Mission Critical information Management</td>
</tr>
<tr>
<td></td>
<td>• L2 + Confidentiality + Audit</td>
</tr>
<tr>
<td></td>
<td>• Remote upgradeable, manage</td>
</tr>
<tr>
<td>L2</td>
<td>Essential information management</td>
</tr>
<tr>
<td></td>
<td>• L1 + data integrity + Availability + Authorization</td>
</tr>
<tr>
<td></td>
<td>• Field upgradable</td>
</tr>
<tr>
<td>L1</td>
<td>Non-essential information management</td>
</tr>
<tr>
<td></td>
<td>• Identity Management/Mutual Authentication</td>
</tr>
<tr>
<td></td>
<td>• Example Black box “disposable” devices</td>
</tr>
<tr>
<td>L0</td>
<td>No Security</td>
</tr>
</tbody>
</table>

**Six (6) Tenets of Security**

1. Identity/Authentication
2. Authorization
3. Audit
4. Confidentiality
5. Integrity
6. Availability
How much protection is enough?

NXP RECOMMEND
Never reflash the board
Flash new code only with secure connection
Flash new code over internet

Use case: Device commissioning with NFC

1. Node + NFC label
   - Tap the gateway or NFC phone
   - Send network key over Zigbee link

2. Node + NTAG
   - Tap the gateway or NFC phone
   - Send network key over NFC

3. (Node + NTAG) = secure key exchange
   - Tap the gateway or NFC phone
   - Send encrypted network key over NFC

4. (Node + NTAG) = secure key exchange + node authentication
   - Tap the gateway or NFC phone
   - Authenticate end-node
   - Derive session key
   - Send encrypted network key over NFC
Security goals for embedded systems

Data Integrity
- Prevent data snooping
- Detect data alterations

Code Integrity
- Prevent code theft
- Detect code alterations
- Allow authorized code changes only

Device Integrity
- Protect cryptographic keys
- Prevent product counterfeits

Cost & Complexity
Cryptography

How do cryptographic algorithms work?

- **Objective**: Scramble data so that only select entities can decipher it

- **Overview**

  - Original data (aka Plain-text) ("Social Security: 123-45-3458")
  - Algorithm
  - Secret ("ab123456c90x1f")
  - Key
  - Scrambled Data (aka Cypher-text) ("$1c^213*2!"")

- **Usage Notes**
  - Can be implemented in software or hardware
    - Hardware implementations can save performance & power
  - Two basic types of cryptography algorithms
    - Symmetric – same key can encrypt & decrypt
    - Asymmetric – different key required for reverse operation

Symmetric cryptography

- DES, 3DES, Blowfish, AES are symmetric cryptographic algorithms
  - AES is the most popular due to its strength – no published reports of successful hack
- 128-, 192- & 256-bits are frequently-used key lengths for AES algorithm
- Pros: Faster than asymmetric cryptography
- Cons: Difficult to distribute & protect the shared secret key securely
Asymmetric (Public-key) cryptography

- Public & Private Key pair per system
- Only the Private key is kept secret
- Keys operate “one-way”
  - Public key encrypts -> Private key decrypts
  - Private key encrypts -> Public key decrypts

Sender uses Receiver’s Public key
Receiver uses own Private key

Encrypt using Public2
Decrypt using Private2

Asymmetric (Public-key) cryptography basics

- Commonly used algorithms: RSA & ECC
- Keys can be up to 2048-bits or longer

Pros
- Easy to manage, scalable

Cons
- Slower than symmetric cryptography
SECURITY GOALS

Data Integrity - why is it important?

- **Unencrypted messages**
  - Hackers can snoop messages
  - Hackers can substitute, replay or monitor messages & create havoc

- **Encrypted messages**
  - Message cannot be read without key
  - Cons: Enabling encryption reduces payload size
  - Cons: Takes time to decrypt the message before it can be used
Data integrity – component & solutions

Prevent data snooping
- Symmetric cryptography
- Asymmetric cryptography

Detect data alterations
- Hash
- Digital signature

Trusting device identity
- Certificates
- Certificate authorities

Objective: Scramble data so that only intended devices can unscramble

Overview

Usage Notes
- AES is used to encrypt/decrypt large set of data due to its speed
  - aka bulk encryption & decryption
- Must share & manage shared secret key(s) with other devices
**Preventing data snooping using asymmetric crypto**

- **Objective:** Scramble data so that only intended devices can unscramble
- **Overview**
  - Use AES to bulk encrypt/decrypt
  - Store in RAM to lose transformation

- **Usage Notes**
  - Public and session key exchange details in later slides
  - Only used to encrypt/decrypt small initial set of data, due to slower speed
  - To exchange certificates, establish session key etc.
  - AES (symmetric) cryptography is used to encrypt/decrypt remaining data
  - Public- and session- key exchange details in later slides

**Detecting data alterations using hash functions**

- **Objective:** Create a shorter representation of data to ascertain original data vs altered data
- **Overview**
  - **One-way:** Hash value cannot yield original data

- **Usage Notes**
  - Sender includes hash value with the data
  - Receiver recalculates hash value of received data and compares with the sender’s hash value
    - Hash value match = Data not altered
Types of hash functions

1. **Non-cryptographic**: CRC, checksum
   - Pros – easy, fast
   - Cons – anyone can recreate if the type of hash function is known

2. **Cryptographic**: MD2/3/4/5/6, SHA-0/1/2/3
   - Pros – only entities with correct key can recreate
   - Cons – more complex than non-cryptographic hash
   - Cons – anyone can recreate if the type of hash function is known
   - Pros – easy, fast
   - MD5 (message digest 5) is not recommended due to successful attacks
   - SHA-2 (secure hash algorithm 2) is certified by U.S. government
   - SHA-2 algorithms differ in bit length of hash value
     - SHA-224, SHA-256, SHA-384, SHA-512 are some examples

Detecting alterations using hash & symmetric crypto

- **MAC** = Message Authentication Code
  - Node 1
    - Data
    - Hash
    - Encrypt
    - MAC
    - Cipher-text
  - Node 2
    - MAC
    - Cipher-text
    - Decrypt
    - Hash
    - Unaltered
  - Also authenticates the sender
    - Only trusted sender can generate the MAC using its copy of shared key
Detecting alterations using hash & asymmetric crypto

- **Digital signature**
  - Signature is uniquely tied to its owner
    - Only the owner has the matching private key to create it

Hardware Flow for Hash & Asymmetric Crypto

- Output is data that can not be snooped (encrypted with asymmetric key) and has a signature (hash) so other node can tell if data was tampered with.
Trusting device identity when using symmetric crypto

• **Objective:** Identify a device using a secret information that only trusted device is expected to possess

• **Overview:**
  - Symmetric cryptography requires a shared secret key
  - Any device with access to the shared secret key is trusted
    - But it is not that easy with asymmetric cryptography…
    - Cons: Counterfeit devices can be made if secret key is known

Transferring Symmetric Key

• **Methods to transfer the key**
NFC (nTag) Tap the gateway

- No key provisioning required during production
- No encryption, protected by short range of NFC
- Out of band (NFC only) instant key and network parameters exchange
  - Faster commissioning
- Possibility to issue commands: reset, decommission, etc.

Energy harvest option permits operating the device with just the NFC

---

Trusting device identity when using asymmetric crypto

- Asymmetric cryptography requires other device’s public-key

  What is your public key?
  Establish a session key
  Use AES to bulk encrypt/decrypt

  Hijacked?

- Problem
  - How do you trust other device’s identity & its public key
  - Much harder to distinguish because no effort is made to hide the key

- Solution
  - Digital certificates and certificate authorities
Digital certificates & certificate authorities (CA)

- **Objective**: Vouch for an entity’s identity & provide its public key
- **Obtaining a certificate**
  - All entities must obtain certificate from a CA and produce it on request

- VeriSign, DigCert, etc. are authorized third party CAs
- Organizations can also issue self-signed certificates
- Certificate authorities are like notary services
- Public key infrastructure (PKI) = HW, SW & process to manage certificate

Certificate usage

1. Devices may cache certificates to speed up or avoid Internet connectivity
2. Verification involves decrypting cert signature using CA’s public-key & matching it with self-calculated hash
Several algorithms – Diffie-Hellman discussed next

• Session key: A temporary, single-use, shared symmetric key used for the duration of a transaction – exact duration depends on use case
  • e.g., Web browser transactions last until an entire page is downloaded (~secs)
• Used in asymmetric & symmetric cryptographic connections
• Usually randomly selected to make the attacks more difficult
  • Must not be visible to man-in-the-middle even when using unsecured channel
• Several algorithms – Diffie-Hellman discussed next

Establishing a session key
Diffie-Hellman key exchange algorithm

- Common paint = Two large numbers (G, g)
- Alice’s secret color = Private key ‘a’
- Bob’s secret color = Private key ‘b’

- Alice’s mixture m1 = Math1(G, g, a)
- Bob’s mixture m2 = Math1(G, g, b)

- Common secret
  • Math2(G, g, b, m1) = Math2(G, g, a, m2)

- Hackers cannot create common secret without a matching private key

Source: https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange
Code integrity – component & solutions

- Protect firmware IP
  - Code read protect (CRP)
  - Secure boot

- Execute only trusted code
  - Secure boot

- Update only intended targets
  - Cryptographic authentication
  - Symmetric cryptography
  - Asymmetric cryptography

**Objective:** Prevent external devices from accessing the internal memory

**Overview**
- Enable protection level(s)
- OTP fuses OR Flash fuses OR Flash memory
- Disable JTAG/SWD
- Disable ISP
- Disable boot modes
- Write-protect boot memory, etc.

**Usage notes**
- **Code Read Protect** (CRP, NXP name) is enabled when the design is released
- Once enabled, may not be disabled
- Prevents hackers from looking for security loopholes
Protecting firmware when executing from external memory

**Objective:** Verify code’s trustworthiness before executing it – aka secure boot

- **Overview**

- **Usage Notes**
  - Secure boot may be implemented using on-chip ROM or Flash
    - ROM code increases reliability
  - May use symmetric or asymmetric cryptography
  - Usually available with external memory boot configuration only
Secure boot using asymmetric cryptography

Desktop post-build process

- Binary code
- Private key
- Sig. algorithm
- Signature
- Cipher-text
- External memory

Target boot process

- External memory
- Signature
- Cipher-text
- RSA
- Public Key
- Binary Code
- Sig. algorithm
- Signature
- Execute
- Internal RAM

Updating intended targets

- Objective: Verify target’s credentials before sending new firmware – known as secure firmware update

- Overview

  - Start
  - Authenticate
  - Download encrypted FW

- Usage Notes
  - Useful for both local & remote firmware updates
  - Authentication may use software or hardware techniques
  - May use symmetric or asymmetric cryptography
  - May use elaborate version control to enforce strict update policies
Authenticating a target using challenge-response

- Server & target must contain a shared secret key to authenticate

  Challenge (random number) → Hash (of random number)
  
  Calculate hash
  
  my hash = target hash?

- Shared key may reside in internal non-volatile memory (NVM) or in an external secure element

Download firmware securely to the intended target

- Re-encryption & secure boot steps are optional
Device integrity – component & solutions

Protect keys
- Restricted NVM storage
- RAM with tamper detect
- Certified secure storage

Prevent counterfeits
- Write-only key storage
- Secure boot
- Secure firmware update
- Certified secure storage

Protecting keys using NVM storage

- Two approaches

1. Plain-sight storage
   - Usually stored in internal or external Flash memory
   - Key fractions stored in multiple address locations
   - Key values transformed using a math & stored in whole or fractions

2. Hardware-assisted storage
   - Maybe stored in flash or OTP memory
   - Write-only
   - May be stored in scrambled form
Protecting keys in RAM with tamper detect

- Two approaches

  1. Software solution
     - Stored in RAM
     - Strategically placed enclosure tamper detect switches
     - Software cleared RAM upon tamper event

  2. Hardware tamper detect with battery-backed RAM
     - Stored in battery-backed RAM
     - Strategically placed enclosure tamper detect switches
     - Hardware cleared RAM upon tamper event

Protecting keys using certified, secured storage

- Secure element can provide multiple key storage for multiple purposes
  - e.g., private key, public key, master key, etc.

- May provide additional functionalities such as symmetric and asymmetric crypto, and challenge-response authentication
Preventing Counterfeits

- Imperatives for preventing counterfeits (one or more maybe used)
  - Hardware component(s) that only the OEM can source
    - Pre-programmed MCUs with internal program memory & disabled JTAG/SWD & ISP
    - Flash-less MCUs pre-programmed with internal write-only key storage
    - Secure element programmed with OEM-specific keys
  - For connected devices, server may deny services to devices with invalid S/N or other identifying information
  - Secure boot with encrypted firmware
  - Secure firmware update only to genuine products
### Protecting embedded systems

- **MCU**
- **USB CAN Ethernet**
- **I²C/SPI UART**
- **Wireless**
- **Other external memories**
- **End product**

1. **Code copy, alterations & reverse engineering**
2. **Data snooping & alterations**
3. **Unauthorized product builds**
4. **Authenticity**
5. **FW update**

---

### Preventing code copy, alterations & reverse engr

- **LPC18S/43S**
- **Secure Boot**
- **AES**
- **CRP**

**MCUs booting from on-chip Flash memory**
- CRP to disable JTAG/SWD & ISP interfaces

**MCU booting from external Flash memory**
- CRP to disable JTAG/SWD & ISP interfaces
- Secure boot to execute only trusted encrypted code
2 Preventing data snooping & alterations

- AES to scramble the data (less complex, difficult to scale) OR
- Public-key software library to exchange key (complex, scalable)
  - AES to perform fast bulk encryption/decryption
  - TRNG to select unpredictable session key

3 Preventing unauthorized product builds

- Flashless MCU with secure boot required
- Use secure boot to decrypt and authenticate the FW
  - OTP key storage provides non-readable key storage
- Supply pre-programmed MCUs & encrypted firmware to CM
Protecting authenticity

- Requires a secret key stored in product to perform **challenge-response**
- Three options
  - Store key in internal/external Flash memory - vulnerable to software hacks
  - Store key in **OTP memory** of LPC18S/43S MCUs – not modifiable
  - Store key in NXP A7x **secure element** – certified, Flash-based storage

Secure firmware update

1. Authenticate the product using OTP key or secure element
2. Download firmware
   - **Symmetric**: Use OTP key to encrypt & download the firmware
   - **Asymmetric**: Use PKI to encrypt & decrypt using a temporary key
3. Program the memory
   - **MCU with Flash**: decrypt the firmware into second Flash bank & activate
   - **Flashless MCU**: encrypt using OTP key, program the Flash & switch using secondary bootloader
### What security problems does my system have?

<table>
<thead>
<tr>
<th>I want to:</th>
<th>Security Measure(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Integrity</strong></td>
<td></td>
</tr>
<tr>
<td>Prevent data alterations</td>
<td>Hash software functions</td>
</tr>
<tr>
<td>Prevent data snooping</td>
<td>Hardware AES or software PKI with hardware AES</td>
</tr>
<tr>
<td>Prevent debugging/ reprogramming in the field</td>
<td>Disable JTAG/SWD; enable secure boot Enable CRP</td>
</tr>
<tr>
<td>Prevent execution of unauthorized firmware changes</td>
<td>Enable secure boot Enable CRP</td>
</tr>
<tr>
<td>Protect firmware IP</td>
<td>Enable secure boot Enable CRP</td>
</tr>
<tr>
<td>Prevent unauthorized product builds</td>
<td>Pre-program the key &amp; enable secure boot Pre-program firmware &amp; enable CRP</td>
</tr>
<tr>
<td>Provide secure firmware updates</td>
<td>Use OTP or Secure Element to authenticate &amp; download FW using AES or PKI Use OTP or Secure Element to authenticate &amp; download FW using AES or PKI into 2nd Flash panel</td>
</tr>
<tr>
<td><strong>Code Integrity</strong></td>
<td></td>
</tr>
<tr>
<td>Prevent product impersonators</td>
<td>Use AES key from OTP storage or Secure Element</td>
</tr>
<tr>
<td>Provide individualized security to each product unit</td>
<td>Use companion NXP secure element to diversify keys</td>
</tr>
<tr>
<td>Implement certified security</td>
<td>Use companion NXP secure element.</td>
</tr>
<tr>
<td>Provide tamperproof storage of data &amp; hardened crypto</td>
<td>Use companion NXP secure element.</td>
</tr>
<tr>
<td><strong>Device Integrity</strong></td>
<td></td>
</tr>
</tbody>
</table>

### LPC18S/43S MCUs

![LPC18S/43S MCUs](image-url)
When is LPC18S/43S MCU the best selection for implementing secure processing?

- Customer security needs
  - Firmware intellectual protection
  - Secure communication with other systems
  - Controlled product builds
  - Authenticated products
  - Secure firmware updates
- AND
  - High performance
  - Up to 164 GPIOs
  - Optional Hi-Speed USB, Ethernet & CAN interfaces
  - Optional 1024x768 graphics LCD controller
- **Applications**: smart home, IoT gateways, industrial controls, HMI, building automation, office automation and others
- Pair with NXP secure element for certified, tamperproof storage and crypto accelerator

---

**LPC18Sxx and LPC43Sxx MCU families**

- **Same LPC18xx & LPC43xx features**
  - High-performance ARM Cortex-M cores
  - Large internal memories and support for external memory expansion
  - Multiple high-speed connectivity and display
- **Plus features for protecting data communications & application code**
  - Hardware-accelerated AES-128 encryption engine for fast bulk encryption
  - Two 128-bit non-volatile OTP memories for storage of write-only AES keys to prevent readouts
  - True random number generator for unique key creation
  - Boot ROM drivers supporting secure boot of authenticated, encrypted firmware image
  - Code read protection (CRP) prevents unauthorized access to internal Flash
Development tools
LPCXpresso43S37 & 18S37 evaluation boards

LPC18Sxx ordering information: parts & tools

<table>
<thead>
<tr>
<th>ORDERABLE PART NO</th>
<th>12NC</th>
<th>AVAILABILITY</th>
<th>FLASH [KB]</th>
<th>RAM [KB]</th>
<th>PACKAGE</th>
<th>WEB/INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPC18S30FBD128E</td>
<td>935303212551</td>
<td>Now</td>
<td>2</td>
<td>136</td>
<td>LQFP144</td>
<td>LPC18S30FBD128E</td>
</tr>
<tr>
<td>LPC18S30FET128E</td>
<td>935303217551</td>
<td>Now</td>
<td>2</td>
<td>136</td>
<td>LQFP144</td>
<td>LPC18S30FET128E</td>
</tr>
<tr>
<td>LPC18S30FRT128E</td>
<td>935303217551</td>
<td>Now</td>
<td>2</td>
<td>136</td>
<td>LQFP144</td>
<td>LPC18S30FRT128E</td>
</tr>
<tr>
<td>LPC18S30JBD128E</td>
<td>935303217551</td>
<td>Now</td>
<td>2</td>
<td>136</td>
<td>LQFP144</td>
<td>LPC18S30JBD128E</td>
</tr>
<tr>
<td>LPC18S30FBD208E</td>
<td>935303217551</td>
<td>Now</td>
<td>2</td>
<td>136</td>
<td>LQFP144</td>
<td>LPC18S30FBD208E</td>
</tr>
<tr>
<td>LPC18S30FET208E</td>
<td>935303217551</td>
<td>Now</td>
<td>2</td>
<td>136</td>
<td>LQFP144</td>
<td>LPC18S30FET208E</td>
</tr>
<tr>
<td>LPC18S30JBD208E</td>
<td>935303217551</td>
<td>Now</td>
<td>2</td>
<td>136</td>
<td>LQFP144</td>
<td>LPC18S30JBD208E</td>
</tr>
<tr>
<td>LPC18S30FBD100E</td>
<td>935303217551</td>
<td>Now</td>
<td>2</td>
<td>136</td>
<td>LQFP144</td>
<td>LPC18S30FBD100E</td>
</tr>
<tr>
<td>LPC18S30FET100E</td>
<td>935303217551</td>
<td>Now</td>
<td>2</td>
<td>136</td>
<td>LQFP144</td>
<td>LPC18S30FET100E</td>
</tr>
<tr>
<td>LPC18S30JBD100E</td>
<td>935303217551</td>
<td>Now</td>
<td>2</td>
<td>136</td>
<td>LQFP144</td>
<td>LPC18S30JBD100E</td>
</tr>
<tr>
<td>LPC18S30FBD144E</td>
<td>935303217551</td>
<td>Now</td>
<td>2</td>
<td>136</td>
<td>LQFP144</td>
<td>LPC18S30FBD144E</td>
</tr>
<tr>
<td>LPC18S30FET144E</td>
<td>935303217551</td>
<td>Now</td>
<td>2</td>
<td>136</td>
<td>LQFP144</td>
<td>LPC18S30FET144E</td>
</tr>
<tr>
<td>LPC18S30JBD144E</td>
<td>935303217551</td>
<td>Now</td>
<td>2</td>
<td>136</td>
<td>LQFP144</td>
<td>LPC18S30JBD144E</td>
</tr>
<tr>
<td>LPC18S30FBD256E</td>
<td>935303217551</td>
<td>Now</td>
<td>2</td>
<td>136</td>
<td>LQFP144</td>
<td>LPC18S30FBD256E</td>
</tr>
<tr>
<td>LPC18S30FET256E</td>
<td>935303217551</td>
<td>Now</td>
<td>2</td>
<td>136</td>
<td>LQFP144</td>
<td>LPC18S30FET256E</td>
</tr>
<tr>
<td>LPC18S30JBD256E</td>
<td>935303217551</td>
<td>Now</td>
<td>2</td>
<td>136</td>
<td>LQFP144</td>
<td>LPC18S30JBD256E</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEVELOPMENT &amp; DEMO TOOLS</th>
<th>ORDERABLE PART NO</th>
<th>AVAILABILITY</th>
<th>DESCRIPTION</th>
<th>WEB / INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPCXpresso18S37</td>
<td>OM13076</td>
<td>Now</td>
<td>Rapid prototyping and evaluation board</td>
<td><a href="http://www.nxp.com/lpcxpresso/home">www.nxp.com/lpcxpresso/home</a></td>
</tr>
<tr>
<td>LPCXpresso IDE</td>
<td>LPCXpresso IDE (V1.7.2)</td>
<td>Now</td>
<td>Cross-platform GNU 32-bit LPC IDE</td>
<td><a href="http://www.nxp.com/lpcxpresso/home">www.nxp.com/lpcxpresso/home</a></td>
</tr>
</tbody>
</table>
### LPC43Sxx ordering information: parts & tools

<table>
<thead>
<tr>
<th>ORDERABLE PART NUMBER</th>
<th>ORDERABLE PART NO</th>
<th>AVAILABILITY</th>
<th>FLASH (KB)</th>
<th>RAM (KB)</th>
<th>PACKAGE</th>
<th>WEB/INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPC43S20FBD144E</td>
<td>935304225551</td>
<td>Now</td>
<td>200</td>
<td>LQFP144</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPC43S20FET144E</td>
<td>935305913551</td>
<td>Now</td>
<td>264</td>
<td>BGA180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPC43S30FBD144E</td>
<td>935304535551</td>
<td>Now</td>
<td>200</td>
<td>LQFP144</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPC43S30FBD145E</td>
<td>9353045914551</td>
<td>Now</td>
<td>1024</td>
<td>BGA180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPC43S37JBD144E</td>
<td>935305914551</td>
<td>Now</td>
<td>1024</td>
<td>BGA180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPC43S37JBD144E</td>
<td>935304409551</td>
<td>Now</td>
<td>1024</td>
<td>BGA180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPC43S37JET100E</td>
<td>935305915551</td>
<td>Now</td>
<td>264</td>
<td>BGA180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPC43S37JET100E</td>
<td>935304409551</td>
<td>Now</td>
<td>264</td>
<td>BGA180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPC43S50FET180E</td>
<td>935305917551</td>
<td>Now</td>
<td>282</td>
<td>BGA180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPC43S50FET256E</td>
<td>935306003551</td>
<td>Now</td>
<td>282</td>
<td>BGA180</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### DEVELOPMENT & DEMO TOOLS

<table>
<thead>
<tr>
<th>DEVELOPMENT &amp; DEMO TOOLS</th>
<th>ORDERABLE PART NO</th>
<th>AVAILABILITY</th>
<th>DESCRIPTION</th>
<th>WEB / INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPCXpresso43S37</td>
<td>OM13073</td>
<td>Now</td>
<td>Rapid prototyping and evaluation board</td>
<td>LPCXpresso43S37 Development Board</td>
</tr>
<tr>
<td>LPCXpresso IDE</td>
<td>LPCXpresso IDE v7.7.2+</td>
<td>Now</td>
<td>Cross platform C/C++ development suite (supports all 32-bit LPC MCUs)</td>
<td><a href="http://www.nxp.com/lpcxpresso/home">www.nxp.com/lpcxpresso/home</a></td>
</tr>
</tbody>
</table>

**SECURE ELEMENT**
LPC43S Protects Home Automation Systems

Secure IoT Home Automation
- Secure boot protects code in QSPI
- A7 Series secure element protects meter against physical attacks attempting to extract or determine keys
- High-speed encrypted connectivity via Ethernet or WiFi with hardware AES
- Secure boot protects software IP
- TRNG for secure session keys
- SDIO for high speed WiFi data transfer
- Bluetooth connectivity

Secure Element use cases
- Allow the establishment of a secure authenticated connection to Cloud Services
  - Support of Measured boot – Checking device integrity
  - Handle device identity relationship management
  - Secure Account management
    - Admin & access tokenization
  - Ease Device maintenance - firmware updates
    - August remote access.
  - Prevent Man-in-The-Middle Attack (DNS attack…)
- Securely store encryption keys and network credentia
  - Non – Spoofable (Anti-cloning / Anti-counterfeiting)
  - Non – Extractable (home network and device integrity)
- Ease Apple HAP compliance
  - Customized Certificates and unique public/private key pairs generated at NXP secure factory
  - PKI based crypto to support SRP for initial setup codes
Use case #1: Long term keys storage

A7 holds HAP Long term keys signs FW hash, Supports Secure Remote protocol for setup codes.

Might be tempting to use this AES coprocessor rather than micro based AES

Use case #2: Secure FW update & long term key storage

- MCU accelerates HAP, lowers overall power consumption
- A70x holds HAP Long term keys and Memory encryption keys, gates Ext-Flash Writes, signs FW hash, establishes session-based AES key with SoC, Supports Secure Remote protocol for setup codes
Secure Element Key Storage

Key ID

<table>
<thead>
<tr>
<th>Object type/purpose</th>
<th>NXP Provisioning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECC/RSA: public/private key pair for Device Authentication (TLs)</td>
<td>Created by NXP and injected by NXP after Wake-Up</td>
</tr>
<tr>
<td>ECC/RSA: public/private key pair for Device Authentication (TLS)</td>
<td>-</td>
</tr>
<tr>
<td>2 certificates for Device Authentication corresponding to Dev ID1 and Dev ID2</td>
<td>Optional: creation and injection by NXP</td>
</tr>
<tr>
<td>2 Public key (ECC/RSA) Source/recipient certificate checking</td>
<td>-</td>
</tr>
<tr>
<td>AES Key store: default AES 128 bits key set (triplet) (*)</td>
<td>Initialized by NXP to Random</td>
</tr>
<tr>
<td>AES Key store: default AES 256 bits key set (triplet) (*)</td>
<td>Initialized by NXP to Random</td>
</tr>
<tr>
<td>AES Key store: AES key Set 1 (triplet) (*)</td>
<td>Initialized by NXP to Random</td>
</tr>
<tr>
<td>AES Key store: AES key Set 24 (triplet) (*)</td>
<td>Initialized by NXP to Random</td>
</tr>
<tr>
<td>Public key (ECC/RSA) Remote key certificate empt (access control)</td>
<td>-</td>
</tr>
<tr>
<td>AES-128 key</td>
<td>-</td>
</tr>
<tr>
<td>AES-256 key</td>
<td>-</td>
</tr>
<tr>
<td>AES Key for Secure Module Upgrade (Card Manager Key)</td>
<td>Unique by Secure Element Available through NXP Key Delivery Service (KDS)</td>
</tr>
</tbody>
</table>

Note: Device = OEM product (*) Encryption keys (K_{MU}, K_{MP} or Micro keys (K_{MU}, K_{MP})

Key Wrapping

- Data transfer
  - Not secure between A70x and secure MCU
  - Need to wrap the key
- Key-Wrapping Key
  - Symmetric key
  - Stored in A70CM
  - Stored in OTP memory of secure micro
- AES key request
  - Secure micro requests AES key from A7
  - A7 Key-Wraps AES key & sends to Secure Micro
- Processing on Secure Micro
  - Secure micro decrypts AES key with Key-Wrapping Key in OTP
  - AES engine uses decrypted AES key to encrypt/decrypt
Private Key Storage Review

- **SRAM of Micro**
  - Not the best idea
  - Okay if power cycled before opening unit
  - Use tamper resistance methods

- **NVM of Micro**
  - Probably the worst idea
  - Easy to retrieve data after power off cycle

- **Secure off chip storage**
  - Use tamper resistant device
  - Excellent protection

- **Summary**
  - No really good places in the micro to store private keys
  - Use off chip storage for best system architecture

---

Time to harden your Devices
Embedding Strong Device ID!

- **Strong Authentication**
  is a key requirement for Reliable and Trusted Infrastructures & Networks

- This drives the need for a
  **Security IC**, decoupled from host application SW and its upgrades, and protecting authentication credentials

---
NXP A-Series Cyber Security Solutions

Key Features

- Advanced SmartMX™ Microcontroller Architecture
- More than 100 security features including
  - NXP Glue Logic Technology
  - Secure Fetch Technology
  - Asynchronous self-timed Handshake
  - Active Shielding Technology
- High performance PKI (RSA/ECC), AES and triple-DES crypto-coprocessors, TRNG
- -40°C...+90°C Operational Ambient Temperature
- Standalone IC with on-chip EEPROM, RAM and ROM
- High reliable EEPROM for both data storage and program execution
  - 25 years minimum data retention
  - 500,000 cycles minimum endurance
- ISO7816, I2C, SPI, ISO14443 (contactless) interfaces
- Factory Key/Certificate pre-injection in certified (Common Criteria) secure environment

Key Applications

A-Series Security ICs

- **Energy Management / Smart Grid**
  - Smart Home Appliances, smart plugs
  - Residential & Industrial Meters
  - Metering Gateways
  - Home & Building Automation systems
  - Grid Automation
  - Data concentrators, routers
  - Electrical & Hybrid Vehicles
  - (P)EV Charging Stations, batteries
  - Street Lighting, Solar panels
- **Industrial**
  - PLC, RTUs, IED, industrial equipment & parts, remote monitoring systems
- **Medical & Healthcare**
  - Home care/monitoring Gateways
  - Medical Devices
  - Traceability solutions
- **Transport**
  - ITS (car/scan), Telematics
  - Infrastructure networks, Tolling
- **Vending**
  - PoS terminals, ATMs
- **Smart Applications/Services**
  - IP cameras, sensors, Smart Cities, Smart Homes, Automation systems, etc
- **Security Systems**
  - Authentication tokens, access control systems, biometric controls, etc
SECURITY DESIGN EXAMPLES

NXP Gateway Reference Design

- Wireless Transceiver JN5188
- Ethernet PHY
- USB OTG PHY
- Ethernet PHY
- USB Host
- Power Switch & Monitor
- EEPROM 256KB
- DDR SDRAM
- NAND Flash
- 1GB
- Reader
- 1Gbit Ethernet
- Power Supply
- 7-18V DC
- I2C
- SPI
- NFC Reader
- Gainspan GS1011 WiFi
- 2.4GHz RF
- I2C or SPI
- UART
Secure Touch PINPad Reference Solution Overview

- Fully PCI 4.x certified POS PIN Pad Reference Design for customers seeking Payment Card Industry certifications
- Hardware and software, including all drivers, cryptographic libraries, NXP Secure Kinetis K81/KL81 MCUs - Pin to pin compatible, covering range of performance and price targets
- Chip-and-PIN keypad based on Cirque® SecureSense™ technology (PCI PTS compliant without requiring physical protection for touch sensor)

**Target Applications:**
- Point of Sales Terminals (secure pin entry for any terminal from mPOS to ePOS)
- Automatic Teller Machine Pin Pad
- Building and Home Automation, Secure Access Control

**Certifications & Testing:**
- TWIP-POS-K31 PCI 4.1 Certified as Pin Pad
- PCI silicon pre-certification
- Side channel attack testing
- CAVP (crypto assurance validation program) certified
- TRNG entropy evaluation

---

Many security features are implemented but require NDA and secure document transfer to the customer

Tamper resistance

Foreign Object Detect
Secure Touch PINPad Reference Solution Schematic

Secure Card Reader Reference Solution Overview

- **Fully PCI4.x certified POS PIN Pad Reference Design** for customers seeking Payment Card Industry certifications
- **Hardware and software**, including all drivers, cryptographic libraries, NXP Secure Kinetis K81/KL81 MCUs - Pin-to-pin compatible, covering range of performance and price targets
- **NXP PN5180 Contact & Contactless card reader module** with KSDK driver support
- Chip-and-PIN keypad based on Cirque® SecureSense™ technology (PCI PTS compliant without requiring physical protection for touch sensor)

**Target Applications:**
- Point of Sales Terminals, Contact & Contactless
- Automatic Teller Machine Pin Pad + Reader
- Building and Home Automation, Secure Access Control

**Certifications & Testing:**
- TWR-POS-K81 PCI 4.1 Certified as Pin Pad
- PCI silicon pre-certification
- Side channel attack testing
- CAVP (crypto assurance validation program) certified
- TRNG entropy evaluation
- EMVCo L1 CT/CL pre-certified
Secure Card Reader Reference Solution Overview

- **Certifications & Testing:**
  - TWR-POS-K81 PCI 4.1 Certified as Pin Pad
  - PCI silicon pre-certification
  - Side channel attack testing
  - CAVP (crypto assurance validation program) certified
  - TRNG entropy evaluation
  - EMVCo L1 CT/CL pre-certified

**Card Reader Reference Solution Schematic**

*mPOS*
Traditional & Smart Mobile POS Solution Enablement

- Quick Start Guide
- User Manual
- Software
  - EMVCo L1 CT/CL Library integrated into PN7462
  - Linux Drivers for PN7462
  - EMVCo L2 CT/CL Library integrated from 3rd parties
  - Trusted Execution Environment (TEE) leveraging ARM Trustzone
- Application Notes
  - ANxxxx - Using i.MX6UL/i.MX7 Security Features
- Certifications
  - Infogard PCI Silicon Pre-cert report

Card Reference Solution Overview
Leveraging Ultra-thin MCUs

- NXP Smart Card IC, Ultra-thin (0.34mm)
  Secure Kinetis KL81 MCU
- Multi-factor identification, PIN + Biometric
- Physically secure and encrypted key (user fingerprint)
  - No PIN entry (optional)
  - Fingerprint processed and transacts <1 second
- Supports contact and contactless transactions
- Target Applications:
  - Physical access control
  - Logical access control
  - Identification
  - Banking / Payment
SUMMARY
Summary

- There are three primary challenges to secure devices
  - Data integrity, code integrity & device integrity
- Each challenge requires multiple cryptographic solutions
- All cryptographic algorithms depend on a secret key
- Secret key must stored securely to protect against hacking
- HW-based key storage can provide the maximum protection

- Secure MCUs offers a number of features to enhance system security
  - Integrated hardware-accelerated AES to speed up encryption and decryption
  - ROM-based secure boot to protect and verify the firmware integrity before executing it
  - Read-only, scrambled OTP key storage memory to help protect keys from hackers
  - Can be coupled with the A7x secure elements from NXP to implement tamperproof secure storage and certified hardware accelerated authentication schemes

---

**secure connections for a smarter world**