SMART: Secure and Minimal Architecture for Establishing a Dynamic Root of Trust

Karim El Defrawy, Aurelien Francillon, Daniele Perito, Gene Tsudik
UCI ETH INRIA UCI

Feb 8 2012
Low-end Embedded Devices

- Low cost, low power devices
- Built around a Micro-controller Unit (MCU)
- Limited:
  - Memory
    - 4 KB Data Memory (SRAM)
    - 128KB Program Memory (Flash)
  - Power
  - computation capabilities
- For example
  - MSP430
  - AVR
“A cyber-physical system (CPS) is a system where there is tight coordination of the system’s computational and physical elements, though sensors and actuators”
Why Security Now?

■ Cyber-physical systems are built to be reliable
■ Security was treated as an afterthought
■ Acceptable with very limited connectivity

And

■ Ease of management is pushing wireless connectivity
  • Implantable medical devices can be accessed via home readers through an RF channel
  • In car systems are connected via wireless
■ Indirectly connected to the Internet
Recent Attacks

- **Stuxnet** [1]
  - Infected controlling windows machines
  - Changed parameters of the PLC of the centrifuges of Iranian nuclear reactors

- ** Attacks against automotive controllers** [2]
  - Internal controller-area network (CAN)
  - Exploiting one subsystem (e.g., bluetooth) allows access to critical subsystems (e.g., braking)

- **Medical devices**
  - Insulin pumps hack [3]
  - Implantable cardioverter defibrillator [4]

---

Remote Attestation

Definitions
- Two party protocol between trusted verifier and untrusted prover
- Remotely verify the internal state of the prover

Where
- Prover is the untrusted embedded device
- Verifier is the trusted reader/controller/base station
- Internal state is composed
  - Code
  - Registers
  - Data Memory
  - I/O

Two types of attestation:
- Secure Hardware supported (e.g., TPM)
- Software attestation
  - Does not support multi-hop communication
Remote attestation

- Malicious software will lie about the software state of the prover
- Need to have guarantees that the device is not lying
SMART: Secure and Minimal Architecture for a Root of Trust

Motivation:
- Existing solutions (TPM) are expensive for embedded devices
- What is a minimal set of architectural features to achieve remote attestation?

Desirable features:
- Minimal modifications to existing platforms
  - Fewest additional gates
- Security under a strong attacker model
- Portable to multiple platforms
  - Implemented on AVR and MSP430
Security Goals

Establish a dynamic root of trust on the prover

- “Guarantee untampered execution of a target piece of code, even in the presence of a corrupted platform”

In particular

- Prover authentication
  - Are we are talking with the right prover?
- External verification
  - Do we know the internal state of the prover?
- Guaranteed execution
  - Do we know the execution state?

No tamper resistance/no hardware attacks
Great. How do we do that?
Building Blocks

- Secure Key Storage
  - Required for multi-hop authentication
  - Provides prover authentication

- Trusted ROM code memory region
  - Read-only means integrity
  - Accesses and operates on key

- MCU access controls
  - Grants access to key to Trusted ROM
Key storage

- Provides remote prover authentication
- The key cannot be stored in normal memory
  - Malware would steal it
- Need to protect key access

Our approach

- Only a trusted code region can access the key
Trusted code region

- Low-end embedded devices do not have support for rings to restrict access to memory
- Adding those would require significant complexity

Our approach

- Restrict access to a read-only trusted code region
- Access control can be implemented easily
Memory access control

- Only ROM code must be able to access the key
- Control the program counter value
The complete protocol

Verifier

(nonce, target code)

Prover

HMAC result
Problems to solve

Trusted ROM code and malware share the same resources

- Malware can set up the environment of the execution to compromise trusted code and extract the key
- Interrupts can asynchronously execute while a copy of the key is in main memory
- Malware can use code gadgets in ROM to access the key
  - Return oriented programming
- ROM code might leave traces of the key in memory after execution
Counter Measures

■ Atomic ROM code execution
  • Enforce in hardware
  • Enter at first instruction
  • Exit at last instruction

■ ROM code is instrumented to check for memory safety
  • Upon detecting error reboot and clean memory

■ Interrupts are disabled immediately
  • Before key usage

■ Erase key material before end of execution
Schematics
Cost of adding ROM and access control

- Implemented on two common MCU platforms
  - AVR
  - MSP430

<table>
<thead>
<tr>
<th>Component</th>
<th>Original Size in kGE</th>
<th>Changed Size in kGE</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AVR MCU</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core</td>
<td>11.3</td>
<td>11.6</td>
<td>2.6%</td>
</tr>
<tr>
<td>Sram</td>
<td>26.6</td>
<td>26.6</td>
<td>0%</td>
</tr>
<tr>
<td>Flash</td>
<td>65</td>
<td>65</td>
<td>0%</td>
</tr>
<tr>
<td>ROM</td>
<td>6 kB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>MSP430 MCU</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core</td>
<td>7.6</td>
<td>8.3</td>
<td>9.2%</td>
</tr>
<tr>
<td>Sram</td>
<td>55.4</td>
<td>55.4</td>
<td>0%</td>
</tr>
<tr>
<td>Flash</td>
<td>65</td>
<td>65</td>
<td>0%</td>
</tr>
<tr>
<td>ROM</td>
<td>4 kB</td>
<td>12.7</td>
<td>-</td>
</tr>
</tbody>
</table>
Considerations on SMART

- SMART provides an efficient hardware attestation solution for embedded devices
- Low additional gates required
- No run-time cost
Thanks for your time

Questions?
Low-end Embedded Devices

- Memory for program and data
- CPU
- Integrated clock
- In addition to
  - Communication interfaces (USB, CAN, Serial, Ethernet, etc.)
  - Analog to digital converters
  - ...