Reviving smart card analysis

Christopher Tarnovsky  chris@flylogic.net
Karsten Nohl          nohl@srlabs.de
Executive summary – Modern smart cards should be analyzed

1. Smart card chips provide the trust base for various applications from banking to ID cards to hardware encryption

2. The secured chips not only protect secret keys but also shield software and protocols from independent analysis. Vulnerability analysis by manufacturers and contracted labs has overlooked bugs numerous times, so independent analysis is needed for software protected by smart cards

3. This talk describes a method for extracting smart card software for analysis through semi-automated reverse-engineering of circuits and optical memory read-out

4. The potential for code read-out affects most modern smart cards. Mitigating it requires stronger memory encryption combined with smarter key storage
Agenda

- Smart card basics
- Reverse-engineering memory encryption
- Mitigation measures
A few smart cards chips cover numerous security domains

**Smart card applications**

- Payment cards, electronic IO cards, access badges
- Trusted Platform Modules (TPM)
- Device and accessory identification
- SIM cards, NFC secure elements
- Content protection

**Bulk risk**

- Only a handful of manufacturers provide smart cards
- Therefore, every card compromise affects numerous applications
Unencrypted ROM memory can be read optically

Metal-masked ROM

Ion-implanted ROM (after staining)
Smart cards protect code and data with memory encryption

Newer smart cards hide the decryption in the CPU
Recent design iterations moved cryptography deeper into chips

**Attack**

1. Optically read code and data from memories

2. Probe wires after MED that carry unencrypted data

3. Reverse-engineer (part of) CPU to find unencrypted wires

**Mitigation**

1. Encrypt memories on the fly in MED

2. Move encryption into CPU

3. Fully encrypted CPU (possible?)
Smart card with memory encryption in CPU core
Protection meshes create additional complexity for invasive attacks

- Meshes increase effort of invasive data extraction
- This talk introduces method for extracting ROM code not hindered by meshes
Agenda

- Smart card basics
- Reverse-engineering memory encryption
- Mitigation measures
Researchers need better tools for finding algorithms in hardware

Software Domain

- Algorithm / Source Code
- Compiler
- Disassembler
- Assembly Instructions
- Bits

Hardware Domain

- Algorithm / HDL code
- Route & Place Tool
- Reverse-Eng.
- Logic Gates
- Transistors

Focus of this talk (using the Degate software)
Algorithms can be extracted from chips in a 3-step process

Silicon disassembling process

Image chips
- Polishing
- Microscoping
- Stitching

Recognize structures
- Pattern recog.
- Wire tracing

Interpret structures
- Gate simulation

Algorithm

Silicon Chip
Chip images
Netlist
Annotated netlist
Demo: Reverse-engineering with Degate
Degate outputs synthesizable code that can be visualized with standard tools.
Memory is ciphered with 64 bit key
Memory encryption includes non-deterministic and obfuscated cells

- 4 bit ROM key
- 4 bit Flash key
- 4 bit RAM key randomly generated at “boot”
- 2 bit random number generator

MUX designs:
Agenda

- Smart card basics
- Reverse-engineering memory encryption
- Mitigation measures
Hardening smart card systems requires actions from manufacturers and system designers

<table>
<thead>
<tr>
<th>Mitigation step</th>
<th>Responsible</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong> Increase complexity of reverse-engineering</td>
<td>Smart card manufacturer</td>
</tr>
<tr>
<td><strong>B</strong> Harden cipher against cryptanalysis</td>
<td></td>
</tr>
<tr>
<td><strong>C</strong> Keep smart card attack value low</td>
<td>System designer</td>
</tr>
</tbody>
</table>
Best defense against reverse-engineering –
Increase complexity, decrease feature size

- Use more random logic, separated in fewer blocks
- Make it harder to image chip (feature size <180 nm)
ROM code could be protected through better cryptography

**Outlined attack**

1. Optical ROM read out (including decryption key!)
2. Reverse-engineering of proprietary decryption function

**Possible mitigation**

- Store ROM key in hard-to-read memory (Flash, PUF)
  - **Effort: Easy** to implement on modern chips that have Flash
- Use well-designed cryptography to prevent key cracking
  - **Effort: High**, research into lightweight crypto needed!
Technology risks vary widely with use case

Example: Nationwide micro-payment scheme

Payment card

Extracting secret keys allows cloning **one card**

Payment terminal

Extracting secret keys allows cloning **all cards**

Same chip, same protection, but different security level
Key distribution should follow need-to-know principle to limit attack surface.

<table>
<thead>
<tr>
<th>Function</th>
<th>Card creation</th>
<th>Writing</th>
<th>Reading</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤</td>
<td>≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤</td>
<td>≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤≤</td>
</tr>
</tbody>
</table>

Each card holds unique keys.

Reading terminals only hold the keys of the functions they perform.
Smart card can provide large but not infinite level of protection

All smart cards can be reverse-engineered or broken with intrusive attacks at some cost

Software stored in ROM should be extractable for analysis from most modern smart cards

Smart cards are the most secure hardware, but system designers must limit attack incentives and not expect miracles from chips

**Degate software**, degate.org  
**Silicon gate examples**, siliconzoo.org

**Questions?**  
Karsten Nohl, nohl@srlabs.de  
Chris Tarnovsky, chris@flylogic.net