Black-box Laser Fault Injection on a Secure Memory

Olivier Hériveaux
Secret protection in embedded systems

Microcontrollers
FLASH memory, basic readout protection fuses
Low-cost
Low resistance against hardware attacks

Secure Elements
Physical attacks counter-measures
Evaluated by accredited labs
Restricted access (JCVM, NDA, ...)

Microchip ATECC508A
Secure memory
IoT applications
Easy access, no NDA
Is this secure?
Coldcard Wallet

Bitcoin hardware wallet
Version Mk2 studied

STM32L4 Microcontroller
Main firmware
ATECC508A
Stores the "Seed" (private key)
Protected with authentication
ATECC508A

Reduced software attack surface

Confidential firmware

Voltage glitch sensors

Top-metal shield

Internal clock generator

No laser counter-measures
Device Under Test
Scaffold control board
https://github.com/Ledger-Donjon/scaffold

Motorised XYZ stage

Laser source

Microscope objective lens 50X

Optical fiber
Silicon is transparent to infra-red light

Integrated circuits are photosensitive

Light can enable transistors conduction...

... hence introducing computation errors!

Laser is a powerful and semi-invasive tool
What’s the plan?

Identify assets and seek an attack path

Prepare and instrument the sample

Target

Test
ATECC508A Memory Layout
ATECC508A Memory Layout

<table>
<thead>
<tr>
<th>CONFIG</th>
</tr>
</thead>
<tbody>
<tr>
<td>#0 · 36 bytes</td>
</tr>
<tr>
<td>#2 · 36 bytes</td>
</tr>
<tr>
<td>#4 · 36 bytes</td>
</tr>
<tr>
<td>#6 · 36 bytes</td>
</tr>
<tr>
<td>#8 · 416 bytes</td>
</tr>
<tr>
<td>#9 · 72 bytes</td>
</tr>
<tr>
<td>#10 · 72 bytes</td>
</tr>
<tr>
<td>#11 · 72 bytes</td>
</tr>
<tr>
<td>#12 · 72 bytes</td>
</tr>
<tr>
<td>#13 · 72 bytes</td>
</tr>
<tr>
<td>#14 · 72 bytes</td>
</tr>
<tr>
<td>#15 · 72 bytes</td>
</tr>
<tr>
<td>OTP</td>
</tr>
</tbody>
</table>
# ATECC508A Memory Layout

<table>
<thead>
<tr>
<th>CONFIG</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Unused</td>
<td>Pairing secret</td>
<td>PIN1 hash</td>
</tr>
<tr>
<td>Anti-phishing</td>
<td>PIN1 try counter</td>
<td>PIN3</td>
</tr>
<tr>
<td>PIN2</td>
<td>PIN1 try counter</td>
<td>PIN3</td>
</tr>
<tr>
<td>PIN2 try counter</td>
<td>PIN3</td>
<td>PIN4</td>
</tr>
</tbody>
</table>

| PIN4            |                |                |
| Seed1           |                |                |
| Seed2           |                |                |
| Seed3           |                |                |
| Seed4           |                |                |
| BrickMe         |                |                |
| Firmware hash   |                |                |
| Unused          |                |                |
| OTP             |                |                |
Accessing data slots

**ReadMemory command:**

<table>
<thead>
<tr>
<th>Command</th>
<th>Length</th>
<th>OpCode</th>
<th>DATA zone</th>
<th>Adresse</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>03</td>
<td>07</td>
<td>02</td>
<td>82</td>
<td>1800</td>
<td>0a78</td>
</tr>
</tbody>
</table>

Response when access granted:

<table>
<thead>
<tr>
<th>Length</th>
<th>Data (32 bytes)</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>303132333435363738396162636465666768696a6b6c6d6e6f70717273747576</td>
<td>384a</td>
</tr>
</tbody>
</table>

(Note: The data bytes and CRC values are in hexadecimal format.)
Accessing data slots

*ReadMemory* command:

```
03 07 02 82 1800 0a78
```

Response when access denied:

```
1 10 384a
```

<table>
<thead>
<tr>
<th>Length</th>
<th>Error code</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EXECUTION_ERROR</td>
<td></td>
</tr>
</tbody>
</table>
PIN1 data slot configuration

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw</td>
<td>0x8f43</td>
</tr>
<tr>
<td>Write config</td>
<td>Encrypt</td>
</tr>
<tr>
<td>Write key</td>
<td>3</td>
</tr>
<tr>
<td>Read key</td>
<td>15</td>
</tr>
<tr>
<td>Is secret</td>
<td>Yes</td>
</tr>
<tr>
<td>Encrypt read</td>
<td>No</td>
</tr>
<tr>
<td>Limited use</td>
<td>No</td>
</tr>
<tr>
<td>No MAC</td>
<td>No</td>
</tr>
</tbody>
</table>
PIN1 data slot configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw</td>
<td>0x8f43</td>
</tr>
<tr>
<td>Write config</td>
<td>Encrypt</td>
</tr>
<tr>
<td>Write key</td>
<td>3</td>
</tr>
<tr>
<td>Read key</td>
<td>15</td>
</tr>
<tr>
<td>Is secret</td>
<td>No</td>
</tr>
<tr>
<td>Encrypt read</td>
<td>No</td>
</tr>
<tr>
<td>Limited use</td>
<td>No</td>
</tr>
<tr>
<td>No MAC</td>
<td>No</td>
</tr>
</tbody>
</table>
Code hypothesis

```c
1 config_address = get_config_address(slot);
2 config = eeprom_read(config_address);
3
4 if (!config.is_secret){
5   data_address = get_data_address(slot);
6   data = eeprom_read(data_address);
7
8   if (config.encrypt_read)
9      encrypt(data);
10
11   i2c_send(data);
12 } else {
13   i2c_send(EXECUTION_ERROR);
14 }
```
config_address = get_config_address(slot);
config = eeprom_read(config_address);

if (!config.is_secret){
data_address = get_data_address(slot);
data = eeprom_read(data_address);

if (config.encrypt_read)
encrypt(data);

i2c_send(data);
} else {
i2c_send(EXECUTION_ERROR);
}
config_address = get_config_address(slot);
config = eeprom_read(config_address);

if (!config.is_secret){
    data_address = get_data_address(slot);
    data = eeprom_read(data_address);

    if (config.encrypt_read)
        encrypt(data);

    i2c_send(data);
} else {
    i2c_send(EXECUTION_ERROR);
}
config_address = get_config_address(slot);
config = eeprom_read(config_address);

if (!config.is_secret){
    data_address = get_data_address(slot);
    data = eeprom_read(data_address);

    if (config.encrypt_read)
        encrypt(data);

    i2c_send(data);
} else {
    i2c_send(EXECUTION_ERROR);
}
When?
Power analysis

Circuit processing activity can be observed on the power trace.
Power analysis

Reading a granted data slot

I2C SDA signal

Electrical current

← Processing begin  Processing end →
Power analysis

Reading a denied data slot

I2C SDA signal

Electrical current

Processing stops earlier →
Power analysis

Reading a denied data slot

I2C SDA signal

Electrical current

Coarse time frame
Power analysis

Comparison of averaged traces

Transfer of $8 \times 4$ bytes
EEPROM $\rightarrow$ RAM

Divergence

123 456 78

Denied

Granted
Power analysis

Comparison of averaged traces

Transfer of 8 x 4 bytes
EEPROM → RAM

12345678

Denied

Granted

Divergence
Where?
Circuit Dissection
Circuit Dissection
Backside decapsulation
Backside decapsulation
Backside decapsulation
Backside decapsulation
Backside decapsulation
Backside decapsulation
Backside decapsulation
Backside decapsulation
Backside decapsulation
Backside decapsulation
Backside decapsulation
Backside decapsulation
Backside decapsulation
Backside decapsulation
Backside decapsulation
Backside decapsulation
Backside decapsulation
Backside decapsulation

1.5 mm
(60 mils)
Infrared imaging
Infrared imaging

- EEPROM
- ROM
- RAM
- Logic gates, CPU, Peripherals...
- Analog circuitry?
Infrared imaging
Targeting
Targeting
Targeting
Targeting
Targeting
Targeting
Testing campaign

Known data is loaded prior to testing:

303132333435363738396162636465666768696a6b6c6d6e6f70717273747576
Testing campaign

For each test:

1. Laser shooting time configuration
2. Laser beam displacement
3. Power-on
4. Initialization
5. Laser activation
6. *ReadMemory* command + Laser shoot
7. Laser deactivation
8. Response readout
9. Power-off
10. Result and parameters logging
Test campaign

Test #1:
  EXECUTION_ERROR
Test #2:
  Timeout, no response received!
Test #3:
  PARSE_ERROR
Test #4:
  OK 09c8420000000000000000000000000000000000000000000000000000000000
Test #5:
  OK 41e0f633a019cd625920691b11400c9387009e68d0b13e53d73257216a4c0ce8
Test #6:
  UNKNOWN_ERROR 0xFE
Test #7:
  EXECUTION_ERROR
Test #8:
  OK
Test #9:
  OK 2ffef9424c7e67d31b519d3d4ea96444265a5189aada8ab27624ca34c2fdf27
Testing campaign

343617 faults injected

Many days of testing

1546 responses received

No success observed...
## Top 20

<table>
<thead>
<tr>
<th>#</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a712c6137b0b50b401d8def8b08b3b8e5f2b01e078d4e6aaeb6bbe589220274</td>
</tr>
<tr>
<td>2</td>
<td>a092cc6943e6c408bd924e9e0b8c895ddac03d2ada707088cace9d9cb803a</td>
</tr>
<tr>
<td>3</td>
<td>00000000</td>
</tr>
<tr>
<td>4</td>
<td>a1ff80fa7028066d4dca823f23e2ec6b79864aa8b6e979e1d63c0f0577eebeb7</td>
</tr>
<tr>
<td>5</td>
<td>41e0f633a019cd62590b611409c9387009e68db01e3635373257216a4c0ce8</td>
</tr>
<tr>
<td>6</td>
<td>929b86e3d0f0eab1d23183cf0c4bf5872b329db260cf012a7c00d40acac19c1</td>
</tr>
<tr>
<td>7</td>
<td>4e92d8096ba78254581b5f9b987e60337e4f9860f92a2615581676e896854dd</td>
</tr>
<tr>
<td>8</td>
<td>011ff4b459e81f8a7f42cd6626fc6117cad15cb99155e72ed6b7621106e22</td>
</tr>
<tr>
<td>9</td>
<td>09c84200000000000000000000000000000000000000000000000000000000000000000000...</td>
</tr>
<tr>
<td>10</td>
<td>9dbf7427f5098f2cb7081745875896f7294629a30049f5aa825dfaa5b7cb3c29</td>
</tr>
<tr>
<td>11</td>
<td>f6feced81f528d1e8b01f05bd59e8f8d84839dbbc01a9614be3a13351089b107</td>
</tr>
<tr>
<td>12</td>
<td>8f8a22572321aba4f803587d84e0eece928e7754d966b054f04f02e5d825999bc6</td>
</tr>
<tr>
<td>13</td>
<td>09ff7317d7351454477e8b6d63f97f27dd3c7c2b1fb41b4e8e54a9e41e9f6c6c</td>
</tr>
<tr>
<td>14</td>
<td>c776a370a55d031685d2abf76672ba5d23187ca07ce42b66286888be89ca02d3</td>
</tr>
<tr>
<td>15</td>
<td>01000000</td>
</tr>
<tr>
<td>16</td>
<td>89f3c21a72ebbb69f1b80100e3c0a3a6ebbb813563373e2a7024024d40ba371</td>
</tr>
<tr>
<td>17</td>
<td>2132e13ce836ed1ab62fc3c9b87345da28616d792e0eb3c7b8a5864c089e80</td>
</tr>
<tr>
<td>18</td>
<td>07f2bba24eebd721e76b9e08e8b2b8431679a147f0562a8565cb382b5ac2e1</td>
</tr>
<tr>
<td>19</td>
<td>e7edcd6b9e38c12eef387f592bc29cb7c7cfcfe14ed4195d251a57525ba6f2b870be</td>
</tr>
<tr>
<td>20</td>
<td>1c60381c211156de7b200149b12bc72ee416bd9d1db927d4fe0ab008d0349a</td>
</tr>
</tbody>
</table>
Analysis

Data overwrite

```
<table>
<thead>
<tr>
<th>Hexadecimal</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a712c613...</td>
<td></td>
</tr>
<tr>
<td>9dbf7427...</td>
<td></td>
</tr>
<tr>
<td>fea48df3...</td>
<td></td>
</tr>
<tr>
<td>01ffbd4b...</td>
<td></td>
</tr>
<tr>
<td>a1ff80fa...</td>
<td></td>
</tr>
<tr>
<td>07f2bb2a...</td>
<td></td>
</tr>
<tr>
<td>9298b6e3...</td>
<td></td>
</tr>
<tr>
<td>3496dbb...</td>
<td></td>
</tr>
<tr>
<td>1c60381c...</td>
<td></td>
</tr>
<tr>
<td>8f8a2257...</td>
<td></td>
</tr>
<tr>
<td>89f3c21a...</td>
<td></td>
</tr>
<tr>
<td>487ce193...</td>
<td></td>
</tr>
<tr>
<td>f6fecd81...</td>
<td></td>
</tr>
<tr>
<td>50f3f6d9...</td>
<td></td>
</tr>
<tr>
<td>c776a730...</td>
<td></td>
</tr>
<tr>
<td>41e0f633...</td>
<td></td>
</tr>
<tr>
<td>e7edcd6b...</td>
<td></td>
</tr>
<tr>
<td>2132c13c...</td>
<td></td>
</tr>
<tr>
<td>4e92d809...</td>
<td></td>
</tr>
<tr>
<td>e89fe351...</td>
<td></td>
</tr>
<tr>
<td>a92e69...</td>
<td></td>
</tr>
<tr>
<td>069ff731...</td>
<td></td>
</tr>
</tbody>
</table>
```


Oh wait!

The attack seems to work!

Can we do it without losing data?
Attack refinement

Optimal parameters identification

New sample preparation and programming

Test run
Success!

Two minutes of testing only

PIN1 and pairing secret data slots can be revealed
Grant's access to Seed1 data slot

Coldcard Mk2 vulnerable

Realistic attack
Did we killed chips?

Yes!

Misconfiguration due to misunderstanding

Failed sample preparation

Data corruption with bad EEPROM write
Possible software mitigations

Double checking

Sensitive constants value

Kill-chip
Possible hardware mitigations

- Light sensors for laser detection
- Power trace jamming
- CPU clock frequency randomization
- Error-Detection-Codes on memories
Cost of mitigations

Implementing them correctly is difficult.

More counter-measures requires more silicon area.

Power and performance is impacted.

Counter-measures may be patent protected.

Security is expensive!
Conclusion

High potential attack
Very expensive equipment

Specific configuration
P-256 keys are not affected

Less resistant than a Secure Element

ATECC508A now deprecated
Superseeded by ATECC608A
Thank you!