DPA, Bitslicing and Masking at 1 GHz

Josep Balasch, Benedikt Gierlichs, Oscar Reparaz, and Ingrid Verbauwhede

KU Leuven ESAT / COSIC (Belgium)

CHES 2015
Saint-Malo, France
16 September 2015
Motivation (I)

• Typical targets of side channel related publications

  *Smart card, microcontrollers*
  *Cryptographic coprocessors*

• Good targets for side channel analysis
  - Not very complex, slow frequencies
  - Common evaluation platforms
Motivation (II)

- Paradigm shift: cryptography is moving to software in the main processor

  *Mobile phones*

  *Internet of Things (IoT)*

- Does the research on side channel analysis apply to more complex processors that operate at gigahertz frequency?

  *source: http://www.cryptomathic.com*

  *source: http://www.engineering.com*
Related work

• Timing attacks
  o Leakage through caches, branches, HPC, etc.
    • No lookup tables with secret indexes
    • No branches on secret values
    • Not easy
  o NaCl cryptographic software library

• Power or Electromagnetic attacks
  o Mostly SPA/SEMA on RSA or ECC
    • High clock frequency less important
    • Critical operations at much slower rate
Can we do DPA/DEMA on block ciphers running on high-end embedded processors?
Platform

- BeagleBone Black single board computer

- Hardware: Texas Instruments Sitara SOC
  - DDR3 memory controller, 3D graphics, HDMI, ...
  - USB, Ethernet
  - ARM Cortex-A8 processor
    - Apple Iphone4, Samsung Galaxy S, ...
    - 32-bit processor
    - 13 stage pipeline
    - Dynamic branch prediction
    - L1 and L2 cache
    - Up to 1 GHz clock frequency

- Software: Complete Linux distribution
  - OS image on embedded MMC
  - 102 processes incl. X, SSH, Apache2, etc.
AES software implementation

- Bitslicing
  - Describe algorithm as sequence of Boolean operations
  - Suitable for hardware: circuit description
  - But also for software: SIMD instructions

```
<table>
<thead>
<tr>
<th>A1</th>
<th>A5</th>
<th>A9</th>
<th>A13</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2</td>
<td>A6</td>
<td>A10</td>
<td>A14</td>
</tr>
<tr>
<td>A3</td>
<td>A7</td>
<td>A11</td>
<td>A15</td>
</tr>
<tr>
<td>A4</td>
<td>A8</td>
<td>A12</td>
<td>A16</td>
</tr>
<tr>
<td>B1</td>
<td>B5</td>
<td>B9</td>
<td>B13</td>
</tr>
<tr>
<td>B2</td>
<td>B6</td>
<td>B10</td>
<td>B14</td>
</tr>
<tr>
<td>B3</td>
<td>B7</td>
<td>B11</td>
<td>B15</td>
</tr>
<tr>
<td>B4</td>
<td>B8</td>
<td>B12</td>
<td>B16</td>
</tr>
</tbody>
</table>

\[ \mathcal{R}_1 = \begin{array}{cccccccc}
    a_1^1 & b_1^1 & a_1^5 & b_1^5 & a_1^9 & b_1^9 & a_1^{13} & b_1^{13} \\
    a_1^{12} & b_1^{12} & a_1^{16} & b_1^{16} \\
\end{array} \]

\[ \mathcal{R}_8 = \begin{array}{cccccccc}
    a_8^1 & b_8^1 & a_8^5 & b_8^5 & a_8^9 & b_8^9 & a_8^{13} & b_8^{13} \\
    a_{8}^{12} & b_{8}^{12} & a_{8}^{16} & b_{8}^{16} \\
\end{array} \]

Bitsliced representation
AES software implementation

• Bitslicing
  o Written in C language
  o Hardware gates $\rightarrow$ software macros

```c
#define XOR(c,a,b) c = a ^ b;
#define AND(c,a,b) c = a & b;
#define NOT(c,a) c = ~ a;
#define MOV(c,a) c = a;
#define ROTL(c,a,l) c = (a << l) | (a >> (32 - l));
```
How to measure side channel leakage of ARM core?

- Contactless power measurement (EM of decoupling capacitors)

Type of probe, position and orientation are important

... while (1) {
    SLEEP
    DO_SOMETHING
    SLEEP
}

Magnetic near field probe
(30 MHz to 3 GHz)
Developing an attack

• How to keep antenna in good position?
Developing an attack

- Challenges: timing and triggering
- Execute bitsliced AES and search for good trigger
- Reduced sampling rate
Developing an attack

• More zoom
Developing an attack

- More, more zoom
Developing an attack

• Practical issues:
  o Trigger is quite stable, but measurements are desynchronized
  o Bad measurements $\rightarrow$ filtered out
  o Good measurements $\rightarrow$ aligned

  o Post-processing is costly!
    • 7x more time than measurement
Attack on unprotected implementation

- First order CPA, 10,000 measurements
- Divide and conquer, attack byte-by-byte as usual
- Predictions specific to bitsliced implementation
  - Hamming weight of 2 bits out of 32

99% confidence interval for zero
Research challenge

• This attack is surprisingly easy

• How can we protect our bitsliced software implementation?
Masked bitsliced implementation

- Apply (hardware) gate-level masking
- Substitute 5 macros with secure versions
  - SXOR, SMOV, SROTL, SNOT: trivial
  - SAND: secure AND gate by Trichina
- Fetch randomness from /dev/urandom

```c
#define SAND(c, a, b)
    t0 = a[0] & b[0];
    t1 = a[0] & b[1];
    c[0] = RND();
    t0 = t0 ^ c[0];
    t0 = t0 ^ t1;
    t1 = a[1] & b[0];
    t0 = t0 ^ t1;
    t1 = a[1] & b[1];
    t0 = t0 ^ t1;
    c[1] = t0;
```

J. Balasch, B. Gierlichs, O. Reparaz, I. Verbauwhede
Attack on protected implementation

- First attack with masks equal zero (fetch from /dev/zero)
  - First order CPA should work
- Code is different → traces are different
  - Find new pattern for alignment
  - Tune parameters for filtering out traces and alignment
Attack on protected implementation

• Second attack with random masks

• What do we expect?
  o Masking in software is difficult
  o Processor is complex
  o This is our first attempt
  o We write C code

• ... probably not secure

• Collect 2 million measurements, keep 1.2 million

• Apply same attack as before
Attack on protected implementation
Attack on protected implementation

- Result differs for different key bytes and register
- Full key extraction possible with 1.2 million traces
- Masked implementation is surprisingly resistant 😊

- Second-order analysis
- Combine all possible pairs of time samples
  - Absolute difference
  - Centered product
- Apply same attack as before
Attack on protected implementation

- Attack with centered product combination did not work
- Absolute difference combination
  - If we know already which time samples to combine

- Real attack requires more effort
Conclusion

- Side channel analysis of complex & high-performance processor, operating at the gigahertz range, and running a complex OS.

- We show that DPA / DEMA attacks can be mounted
  - Attacks are surprisingly easy
  - But triggering and alignment are difficult

- We show that gate-level masking can be used to protect bitsliced software
Thanks for your attention!

QUESTIONS

extended version: https://eprint.iacr.org/2015/727
• FFT shows main frequency component at 1 GHz