DPA, Bitslicing and Masking at 1 GHz

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

KU Leuven ESAT / COSIC (Belgium)

CHES 2015

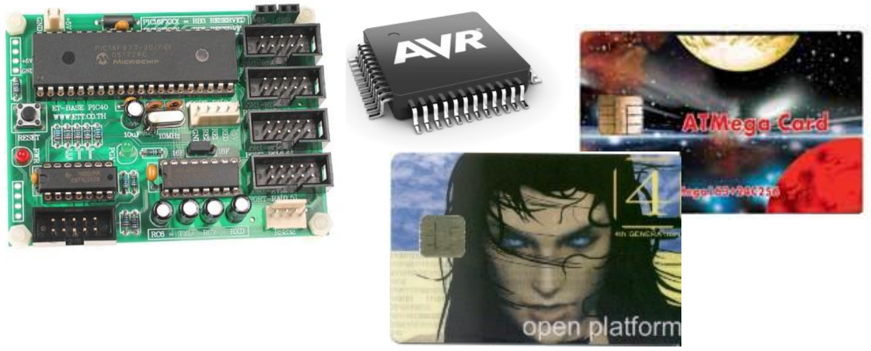
Saint-Malo, France

16 September 2015

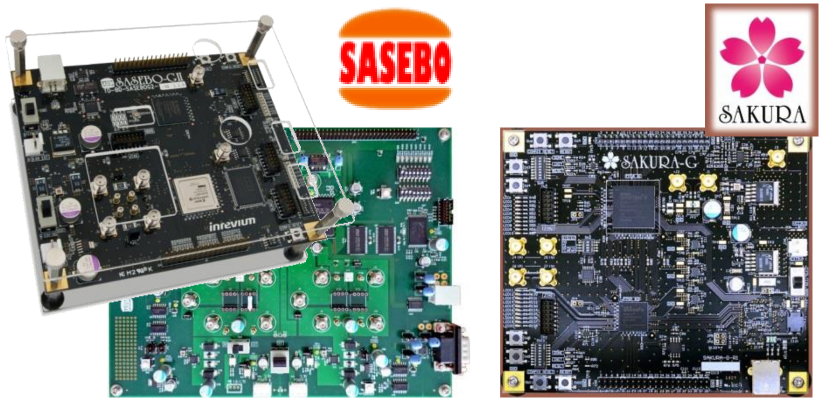
Motivation (I)

- 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

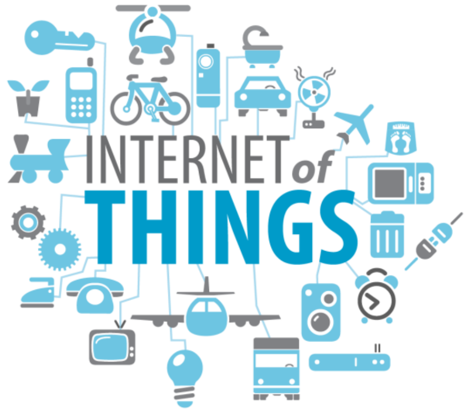
Motivation (II)

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

Mobile phones

source: http://www.cryptomathic.com

Internet of Things (IoT)



source: http://www.engineering.com

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

Related work

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

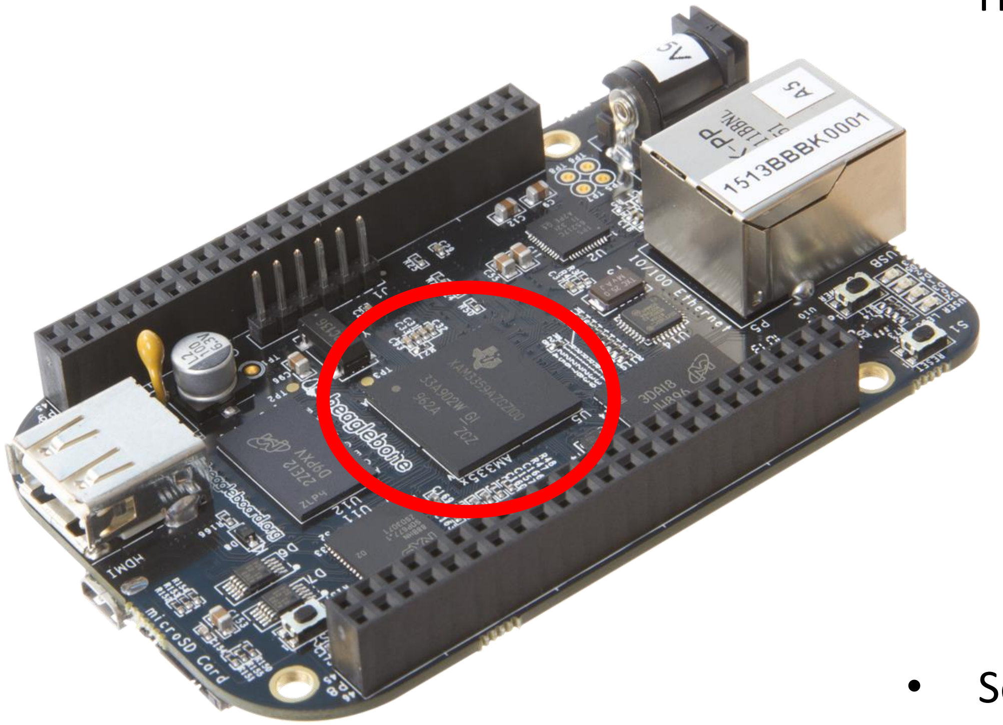
- Power or Electromagnetic attacks
  - Mostly SPA/SEMA on RSA or ECC
    - High clock frequency less important
    - Critical operations at much slower rate

Research challenge

Can we do DPA/DEMA on block ciphers running on high-end embedded processors?

Platform

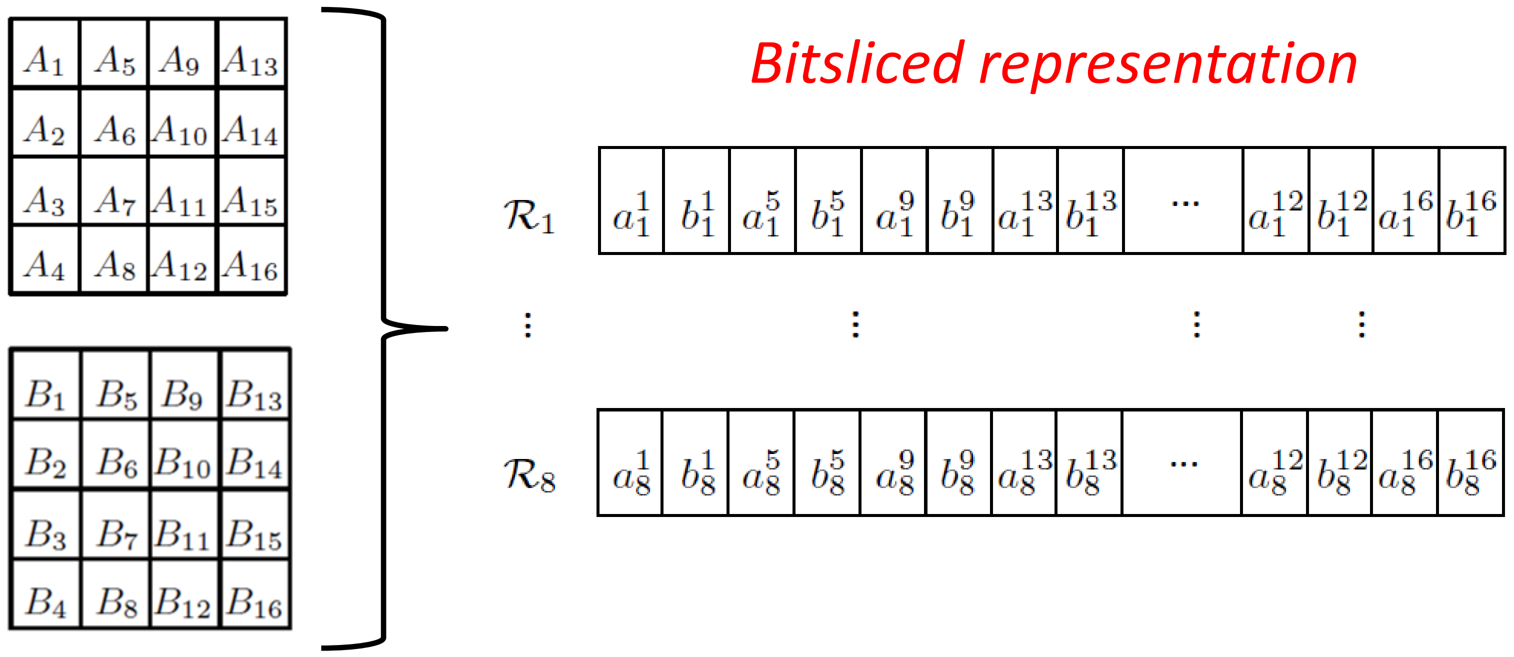
- BeagleBone Black single board computer



- 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


A_1 A_5 A_9 A_{13}
A_2 A_6 A_{10} A_{14}
A_3 A_7 A_{11} A_{15}
A_4 A_8 A_{12} A_{16}
B_1 B_5 B_9 B_{13}
B_2 B_6 B_{10} B_{14}
B_3 B_7 B_{11} B_{15}
B_4 B_8 B_{12} B_{16}
Bitsliced representation
$\mathcal{R}_1$: 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}
$\mathcal{R}_8$: 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}

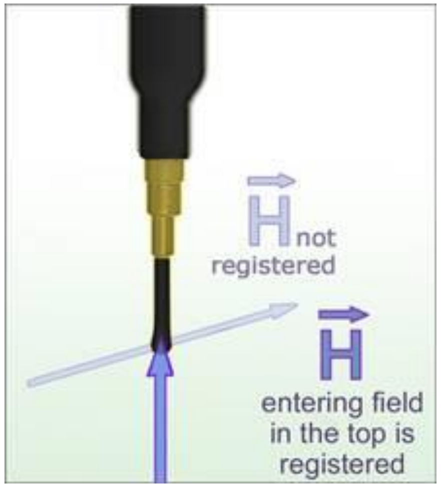
AES software implementation

- Bitslicing
  - Written in C language
  - Hardware gates → software macros

```
#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));
```

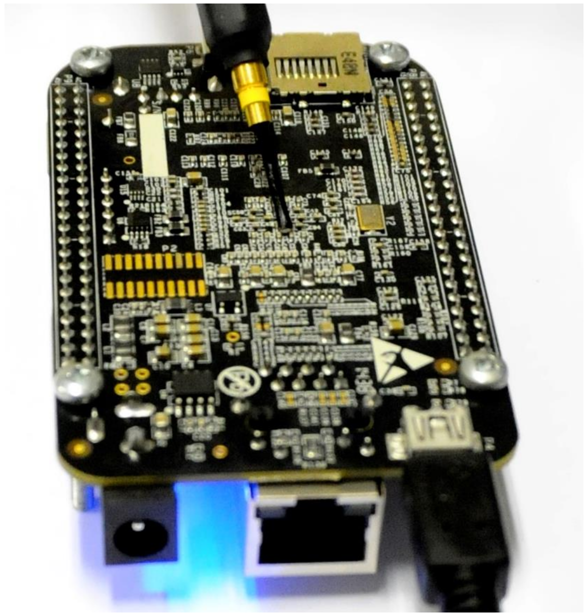
Developing an attack

- How to measure side channel leakage of ARM core?
    - Contactless power measurement (EM of decoupling capacitors)
- Type of probe, position and orientation are important



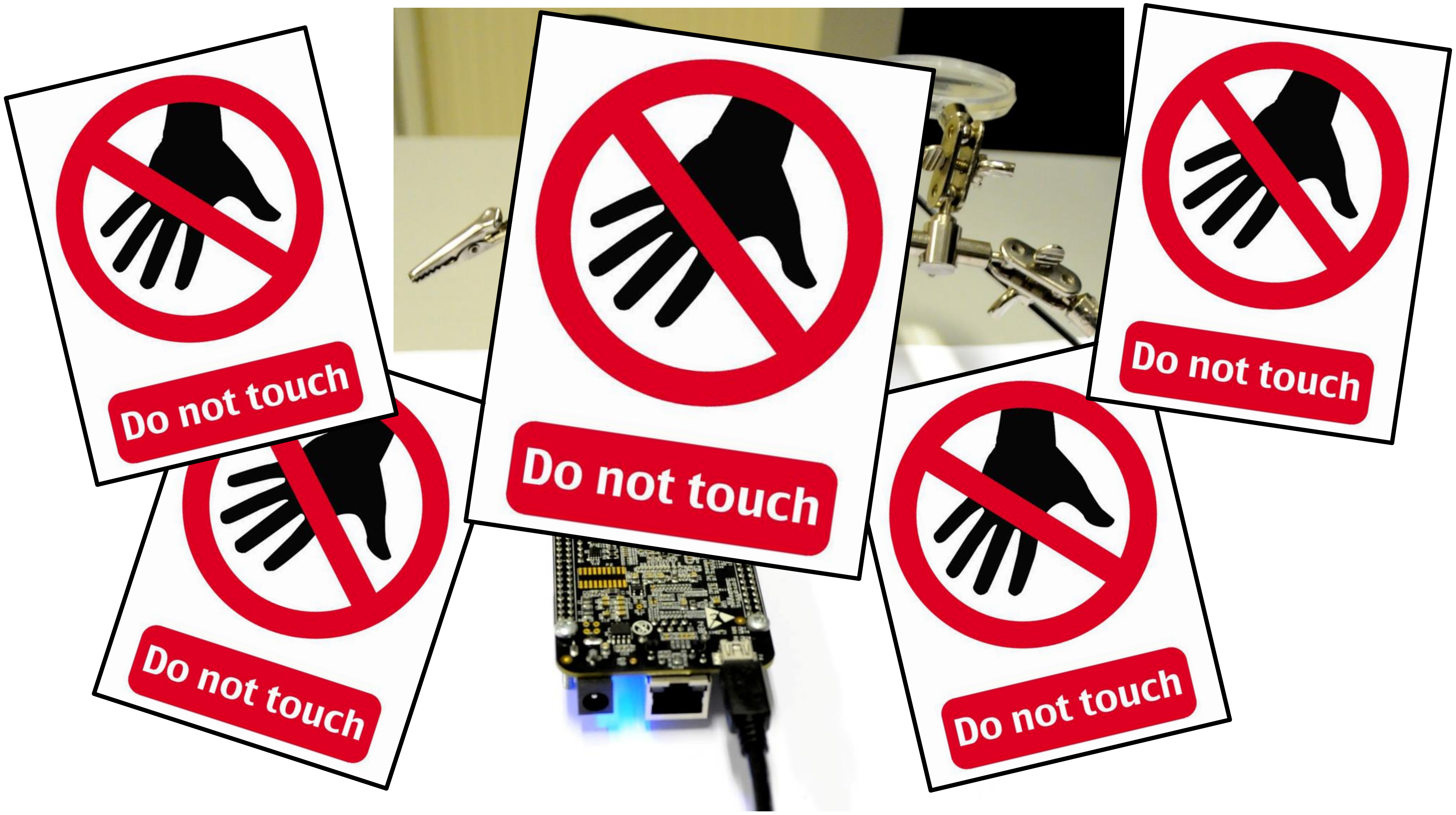
Magnetic near field probe (30 MHz to 3 GHz)

```
…
while (1) {
   SLEEP
   DO_SOMETHING
   SLEEP
}
…
```



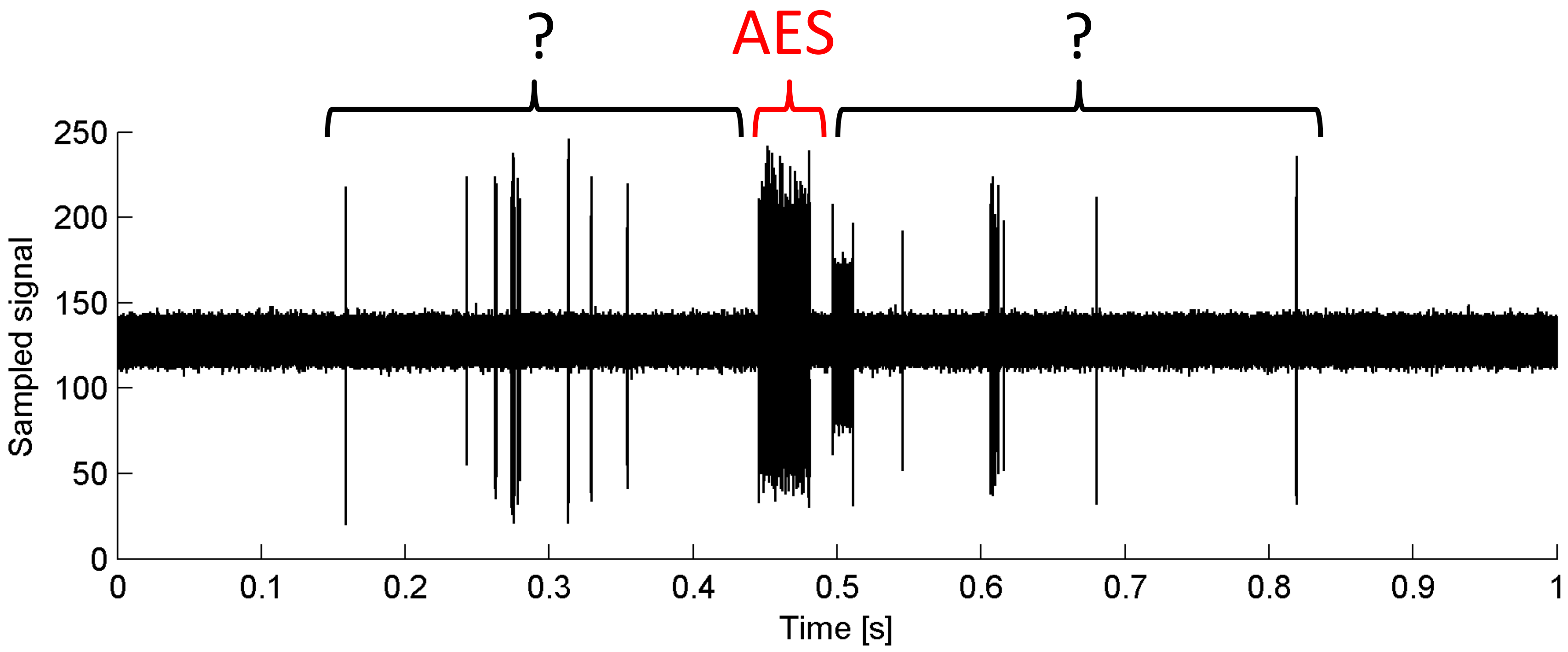
Developing an attack

- How to keep antenna in good position?



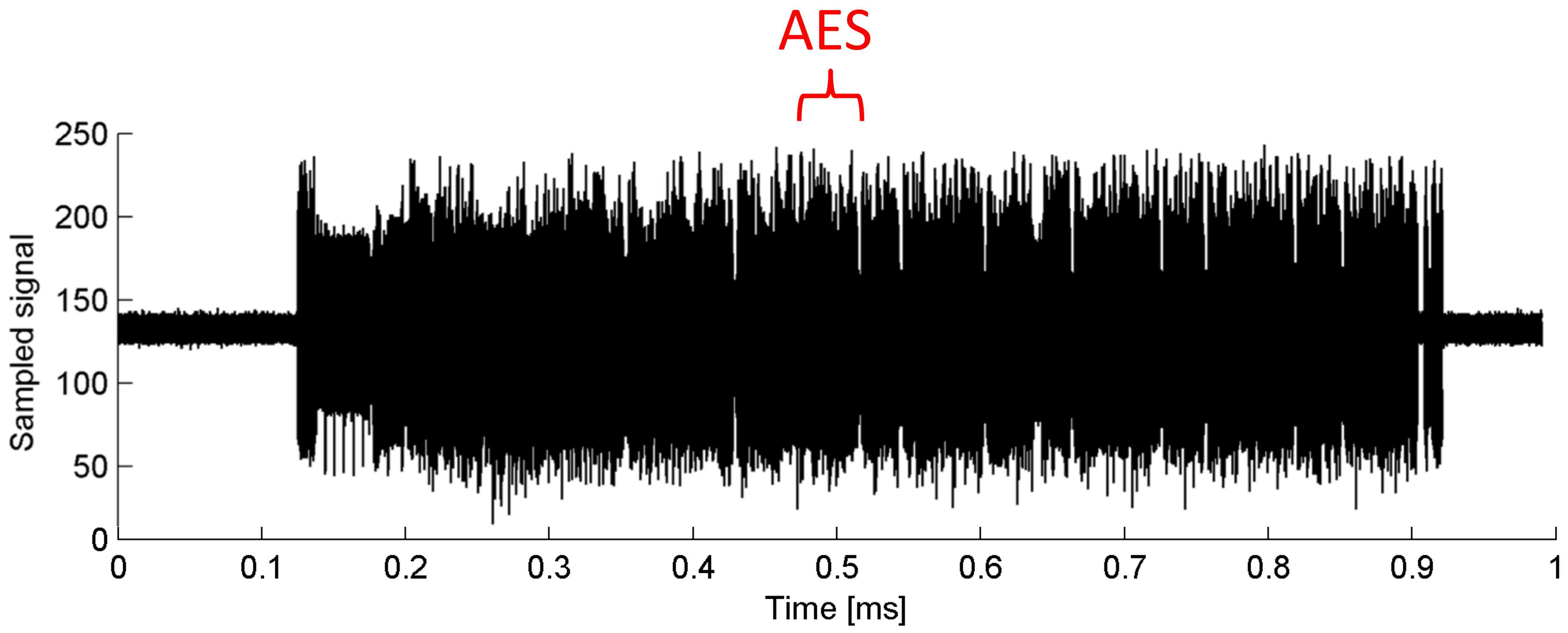
Developing an attack

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



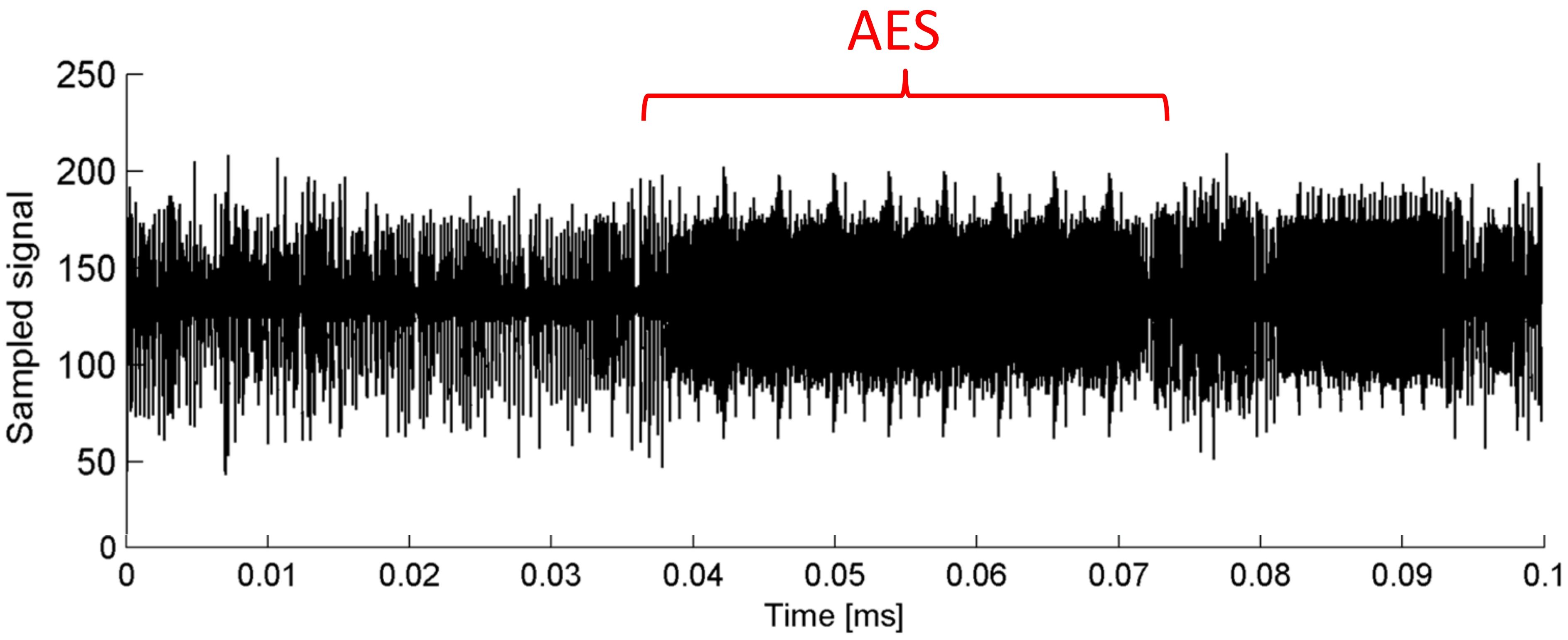
Developing an attack

- More zoom



Developing an attack

- More, more zoom

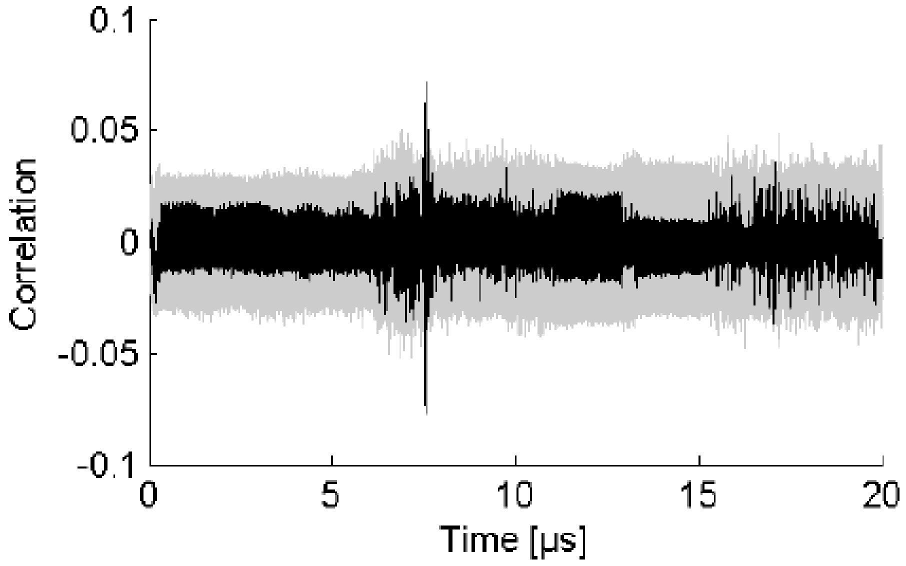


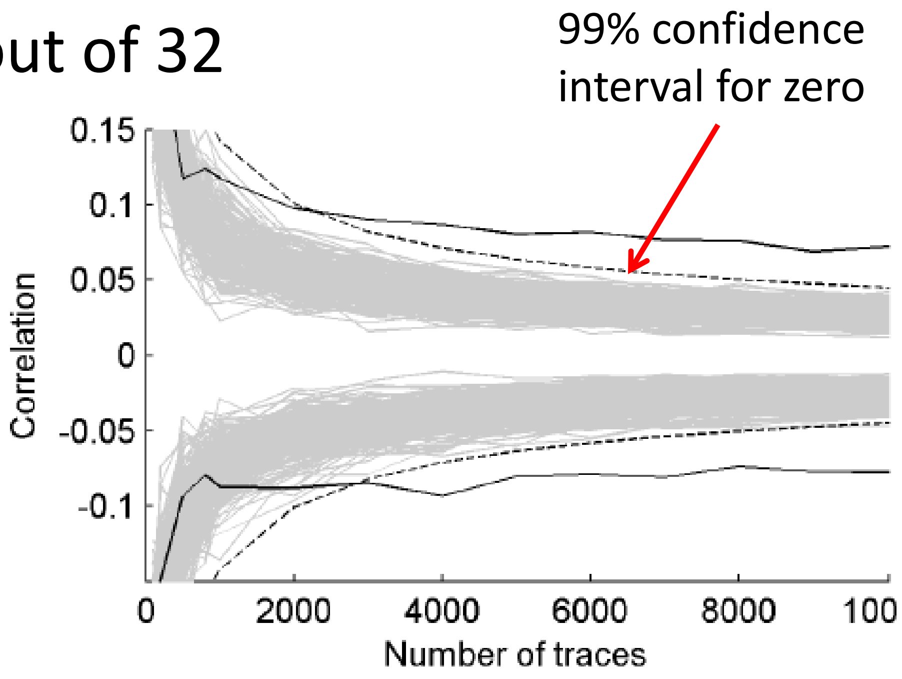
Developing an attack

- Practical issues:
  - Trigger is quite stable, but measurements are desynchronized
  - Bad measurements → filtered out
  - Good measurements → aligned
  - 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



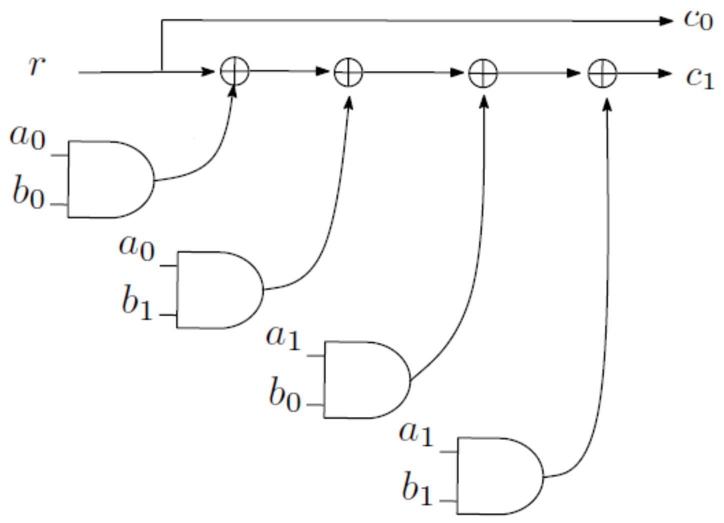


Research challenge

- 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

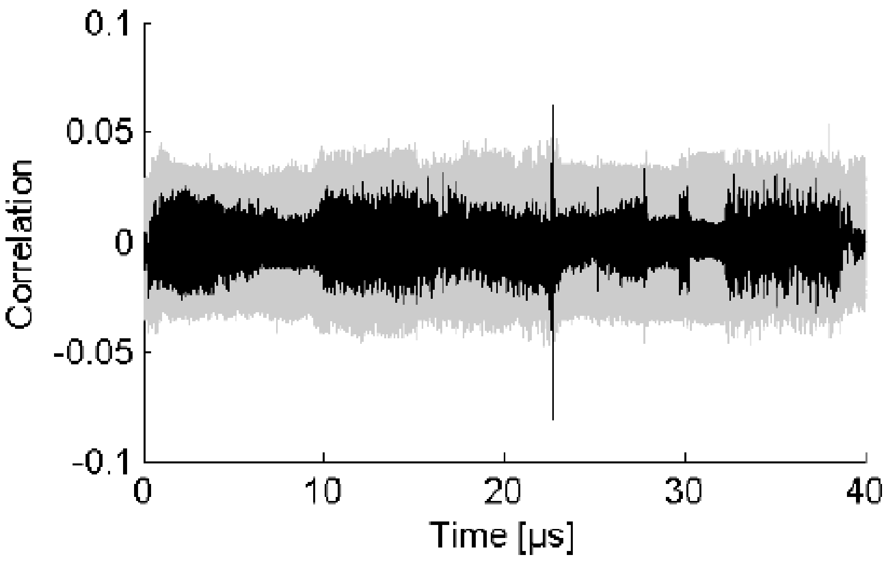


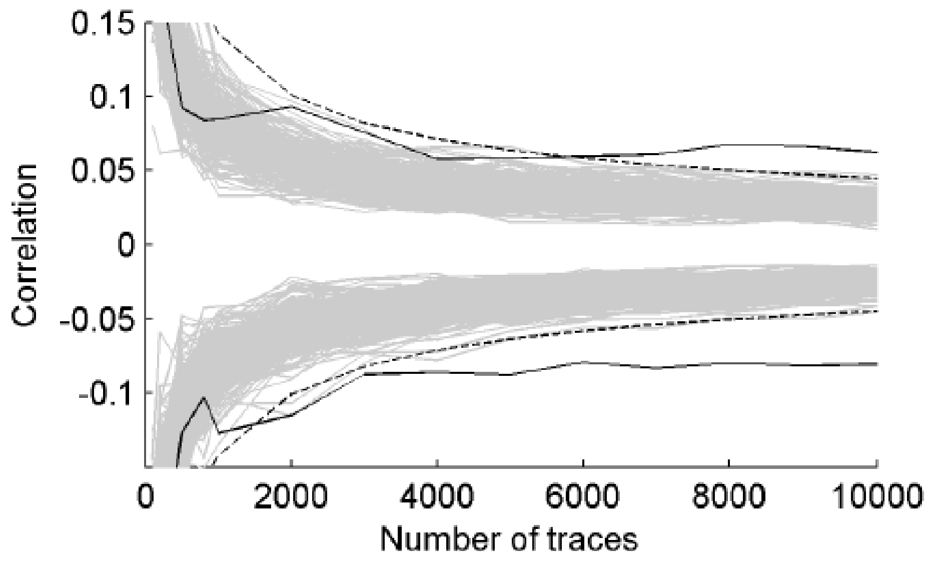
```
#define SAND(c, a, b){
  t0 = a[0] & b[0];
  t1 = a[0] & b[1];
  c[0] = RAND();
  t0 = t0 ^ c[0];
  t0 = t0 ^ t1;
  t1 = a[1] & b[0];
  t0 = t0 ^ t1;
  t1 = a[1] & b[1];
  t0 = t0 ^ t1;
  c[1] = t0;}
```

- Fetch randomness from /dev/urandom

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



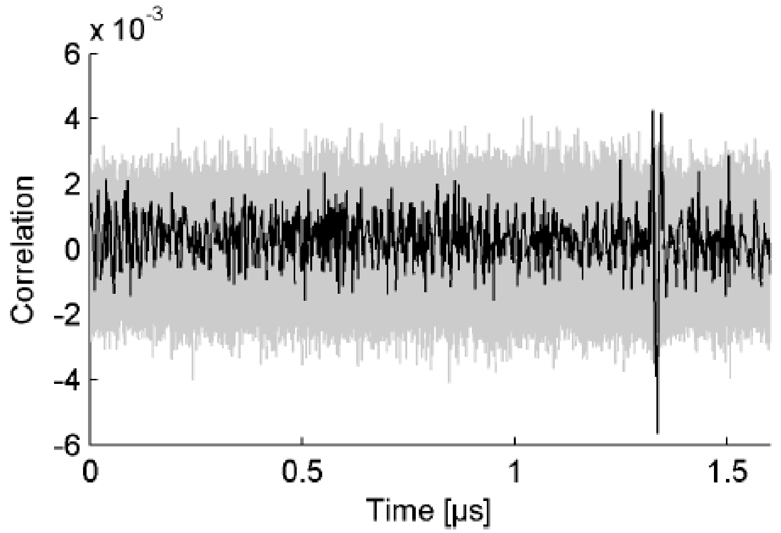


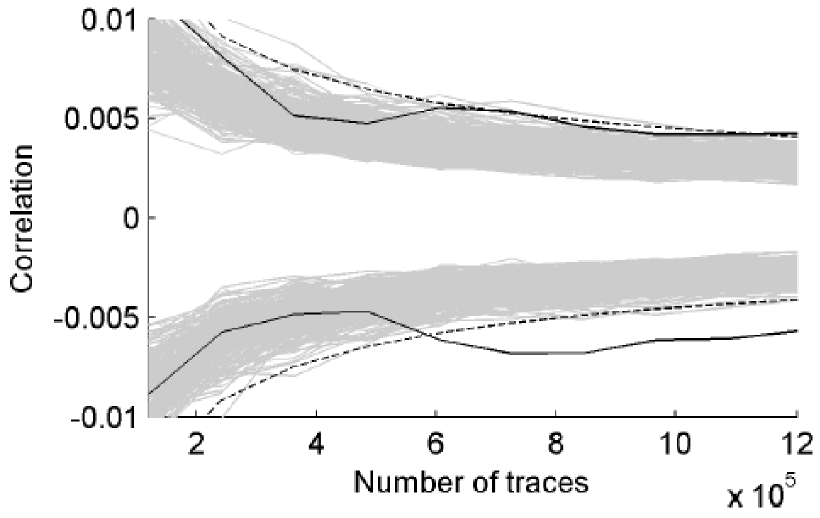
Attack on protected implementation

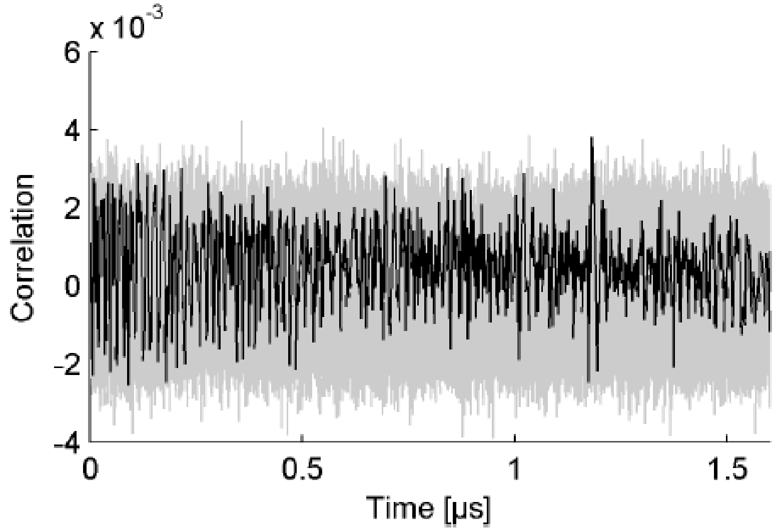
- Second attack with random masks
- What do we expect?
    - Masking in software is difficult
    - Processor is complex
    - This is our first attempt
    - We write C code
- … probably not secure

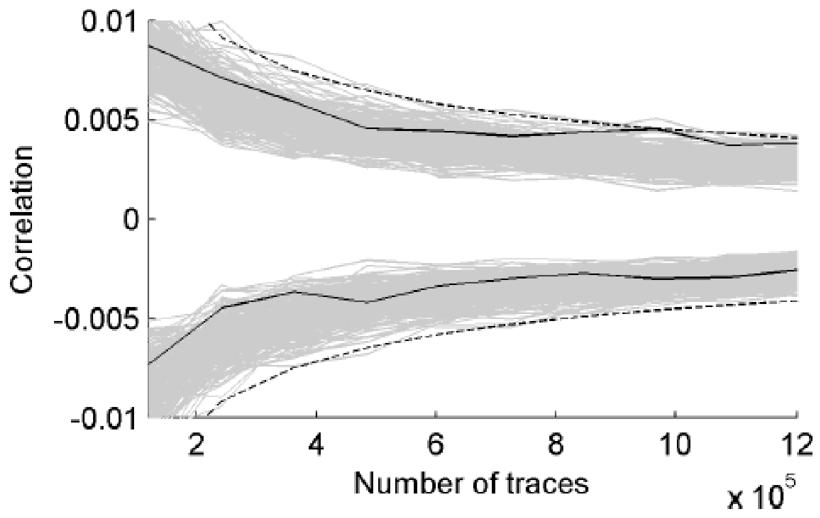
- Collect 2 million measurements, keep 1.2 million
- Apply same attack as before

Attack on protected implementation









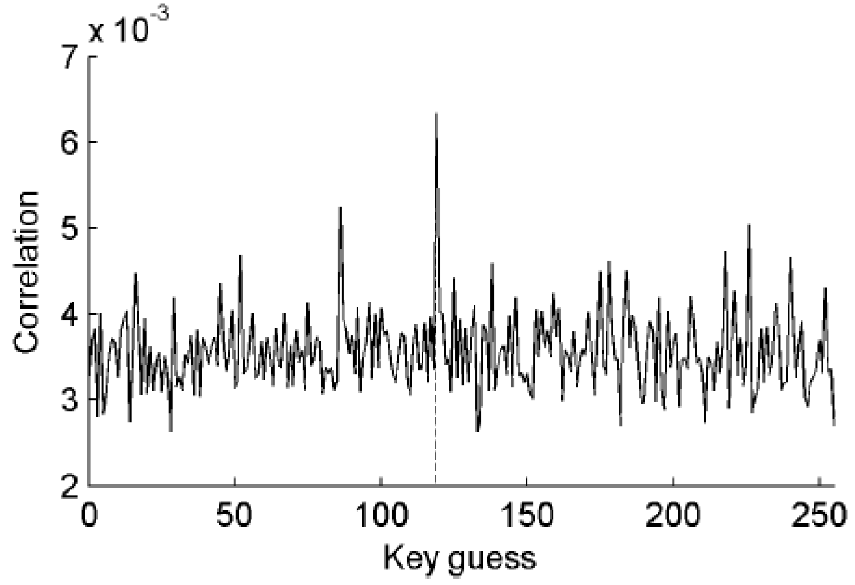
Attack on protected implementation

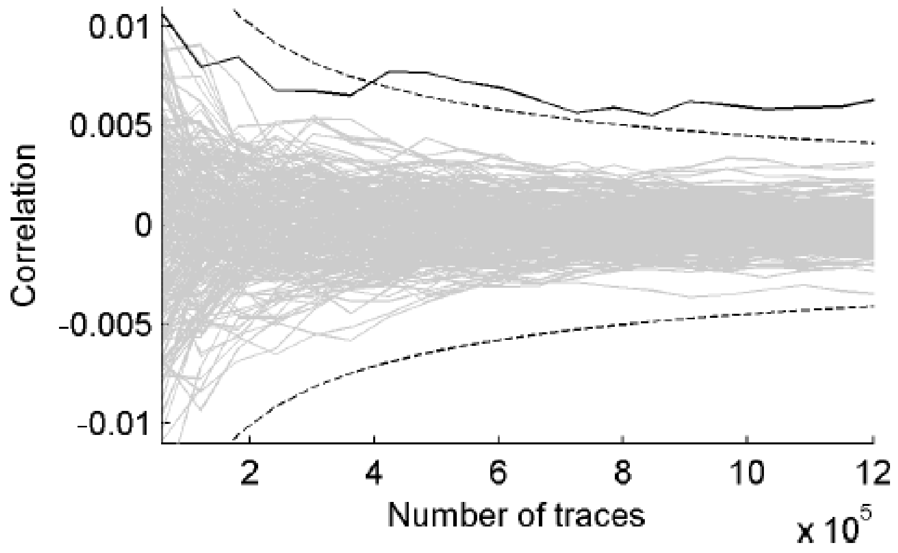
- Result differs for different key bytes and register
- Full key extraction possible with 1.2 million traces
- Masked implementation is surprisingly resistant ☺

- 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

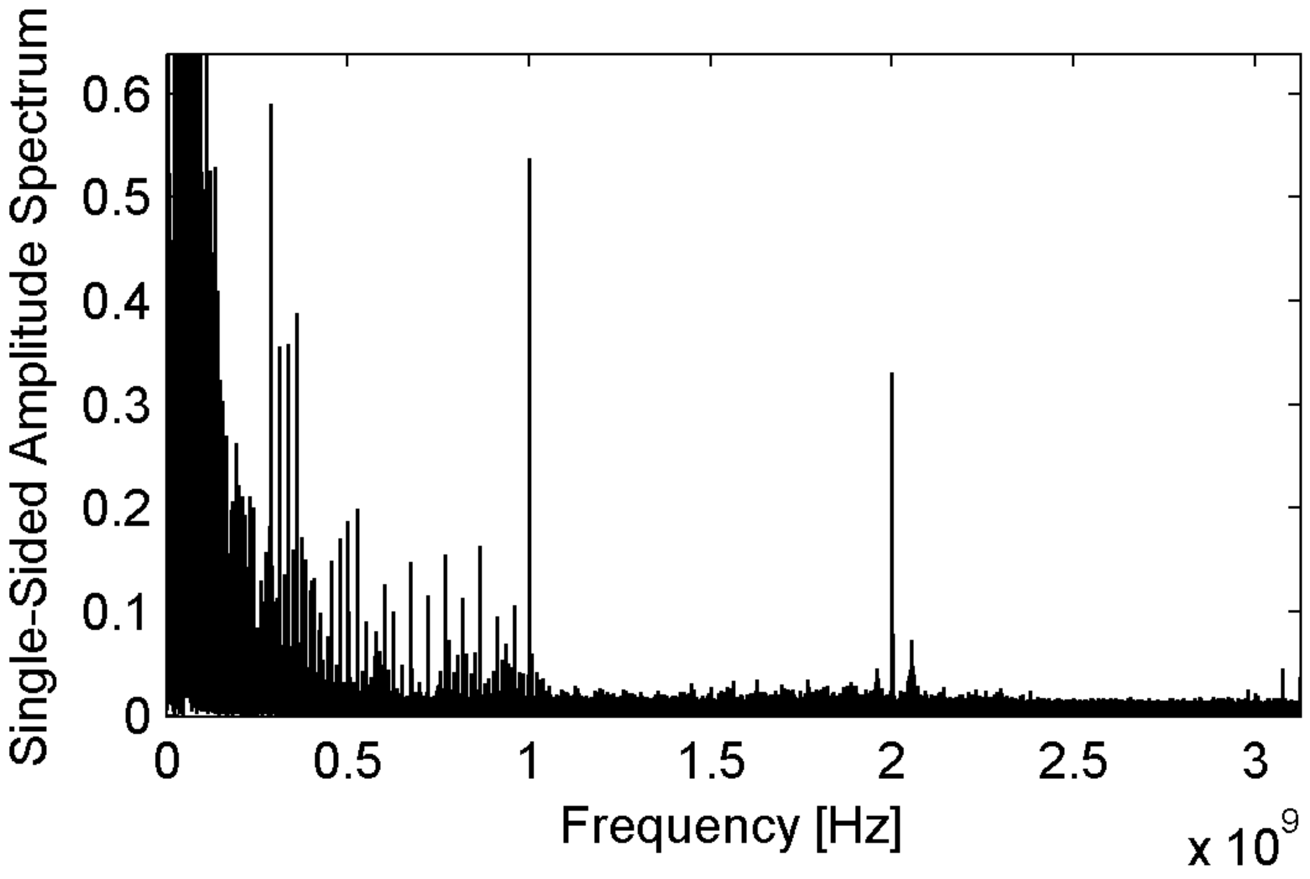
- We show that gate-level masking can be used to protect bitsliced software

Thanks for your attention!

QUESTIONS ?

extended version: https://eprint.iacr.org/2015/727

- FFT shows main frequency component at 1 GHz


Single-Sided Amplitude Spectrum
Frequency [Hz]
x 10^9