Share-slicing: Friend or Foe?

Si Gao $^{1,2}$  Ben Marshall $^1$  Dan Page $^1$  Elisabeth Oswald $^{1,2}$

$^1$University of Bristol, Bristol, UK

$^2$University of Klagenfurt, Klagenfurt, Austria

September 4, 2020
1. Intro
2. Evaluations in practice
3. Read pass the "headlines"...
4. Discussion
SCA

- Attacks based on information leakage
- Recover the secret key potentially within a few minutes

**Figure:** Side Channel Analysis
Masking: hardware masking

Academia
"Countermeasures to rescue!"
- Threshold implementation [NRR06]
- Domain-Oriented Masking [GMK16]
- Various schemes available!

Industry
"Oh no... what should I do?"
Masking: look-up table-based

Academia
"OK...that is trickier, but still do-able"
- Look-up table based approaches
- Global look-up table
- Re-computation method [Coron14]

Industry
"Emm... I can update codes, but not revoke all devices..."
Masking: bit-sliced

Academia

"How about bit-sliced masking?"
- Utilise small gadgets (eg. AND2)
- Moderate memory cost, flexible
- Difficult for chaining mode (eg. CBC-ENC)

Industry

"Actually my application is quite memory-tight. Any other suggestions?"
Academia

"OK. Here are some results:"

- ISW multiplication [ISW03]
- Multiplication in bounded-moment model [BDF+17]
- Proof + some codes on Github
- Performance on ARM [GR17, GJRS18]

Industry

"Fair enough. Let us do this!"
"But please be careful with your implementations:"

- Pitfalls (eg. bad randomness)
- Model v.s. Practice
  - "order reduction theorem" [BGGRS14]

"Brilliant! I will implement one of this."
Masking: code

Academia
On a code-level, a $d$-share scheme:
- is seldom $(d - 1)$-order secure
  - Few would do the full "diagnose-and-cure" cycle
- Even if it is $d - 1$-order secure...
  - Weak protection when $d$ is small

Industry
"Alright...I will keep that in mind."
A few days later...
Masking: theory to practice

Academia
"OK... you sure it is working properly?"

Industry
"Professor, I have implemented my 4-share secure AES!"
- Barthe et al.'s secure multiplication [BDF+17]
  - Parallel share processing —> efficiency
  - Share-slicing: all shares in one register
"Emm...maybe you are right?"

"Should be OK, I guess"
- Only claiming 1st order secure
- "order reduction theorem"
- Previous study said so [JS17]
- if we ignore physical coupling [CEM18, LBS19]
Academia

"Or is it really correct?"
Outline

1. Intro
2. Evaluations in practice
3. Read pass the "headlines"...
4. Discussion
Setups

ARM M3 (NXP LPC1313) & M0 (NXP LPC1114)
- Working at 12 MHz
- Scope sampling at 250 MSa/s
- Code written in Thumb assembly
- Unused bit-width
  - Constants: all 0-s (trivial yet waste)
  - Randomise: worst for the attacker (costly)
  - Repetition: same unshared value
Target secure AND2 tailored:

- Transition-leakage reduced to minimal

```plaintext
// rotation of b by 1
mov     r8, r8     // Clear HD
lsls    r6, r1, #1 // r6=b<<1
mov     r8, r8     // Clear HD

// The following computation has been commented out
// ands    r6, r5     // r6=(b<<1)&0xenenenenen
// lsrs    r7, r1, #3 // r7=(b>>3)
// bics    r7, r5
// eors    r6, r7     // r6=(b<<<<1)
// ...
```
Evaluations

Barthe et al’s multiplication on M3: 2-share version
- All other 30 bits random
- correct key(red)/incorrect keys(gray)
- 1st order $\approx$ 2nd order, not a big deal
Evaluations

Barthe et al’s multiplication on M3: 4-share version

- All other 28 bits random
- Correct key (red)/incorrect keys (gray)
- 2nd order is better/1st order still exists
Academia

"Have you checked the model assumptions?"

Industry

"Wait...how can it be?"
Masking: theory to practice

Academia

"Err..."

Industry

"I only checked the 'implementation defaults' section:"
- Mostly hardware perspective
- What does it mean in software?
Independent assumption

"Each share leaks independently": specifically,

- Each share has its own leakage function
- No interaction/cross-talk

This larger power consumption is the following linear model:

$$L_c = \alpha^1_c \cdot G^1_c (Y_c(1)) + \alpha^2_c \cdot G^2_c (Y_c(2)) + \ldots + \alpha^{n_c}_c \cdot G^{n_c}_c (Y_c(n_c)) + R_c. \quad (2)$$

with all $\alpha^j_c$'s $\in \mathbb{R}$. Contrary to the additive noise assumption that is only used
Independent assumption: in hardware

In hardware masking

Such assumption is usually supported by:

- Parallel separated sub-circuits (motivated by MPC)
  - No logical crosstalk
- "Keep Hierarchy"
  - No cross-talk from the synthesiser

This larger power consumption is the following linear model:

$$L_c = \alpha_c^1 \cdot G_c^1 (Y_c(1)) + \alpha_c^2 \cdot G_c^2 (Y_c(2)) + \ldots + \alpha_c^{n_c} \cdot G_c^{n_c} (Y_c(n_c)) + R_c.$$  \hspace{1cm} (2)

with all $\alpha_c^i$'s $\in \mathbb{R}$. Contrary to the additive noise assumption that is only used
Independent assumption: in software

Software with share-slicing

Independent assumption becomes

- following the same level of scrutiny
- each gate in the ALU connects with only 1 bit of the register
Independent assumption: in software

Software with share-slicing
But is that even possible?
Independent assumption: in software

**Zoom into the shifter**

Shifter can be our first headache:

Other parts of the ALU (eg. adders) can also contribute
Verifying independent assumption

Testing on shift alone

Already illustrates the issue:

**Figure 6:** Instruction-wise leakage analysis: LSL/LSR
Verifying independent assumption

Academia

"Well..."

Industry

"But did not the previous study verify this already? "

www.aau.at
Let us read pass the "headlines"

"Very high order masking: Efficient implementation and security evaluation"[JS17]:

- TVLA on one specific instance, NOT the assumption itself
  - Only 2/4 bits are used
  - Conservative interpretation: assuming $d/2 = 15$ order security
- Fair for their purpose, but should not be taken out of the context
Verifying independent assumption

Academia

"Well..."

Industry

"O.K...then how about the order reduction theorem? "

www.aau.at
Let us read pass the "headlines"

"On the cost of lazy engineering for masked software implementations" [BGGRS14]:

- Security reduction for "transition-based leakage"
  - Implicitly assumed shares stores in different registers
- Do not apply at the first place
  - Which has been said in [JS17]

**Theorem 1.** An $d$th-order secure implementation against value-based leakage functions is $\left\lceil \frac{d}{2} \right\rceil$th-order secure against transition-based leakage functions.
Our results suggest...

- Independent assumption should not be taken for granted on software platforms

They do not suggest...

- Share-slicing should be forbidden
  - A weaker assumption (say, SNR-based)?
  - Proof does not guarantee everything...
  - Platform-dependent

- Shifter is the (only) source of interaction
  - Various components can contribute
  - Cannot locate the exact source (unless the CPU is completely open-sourced)
What does model assumption mean in practice...

**Academia**
- Offer schemes in security model

**Industry**
- Needs the connecting to practice

Who should be the "interpreter"?
Questions?

Thank you!
NRR06  Nikova, S., Rechberger C., Rijmen V.: Threshold Implementations Against Side-Channel Attacks and Glitches. Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006


GJRS18  Dahmun Goudarzi, Anthony Journault, Matthieu Rivain, and François-Xavier Standaert. Secure multiplication for bitslice higher-order masking: Optimisation and comparison. COSADE 2018

GR17  Dahmun Goudarzi and Matthieu Rivain. How fast can higher-order masking be in software? Advances in Cryptology – EUROCRYPT 2017