

---

# Non-profiled Mask Recovery: the impact of Independent Component Analysis

Si Gao<sup>1</sup> Elisabeth Oswald<sup>1</sup> Hua Chen<sup>2</sup> and Wei Xi<sup>3</sup>

<sup>1</sup>University of Bristol

<sup>2</sup>Institute of Software, Chinese Academy of Sciences

<sup>3</sup>Southern Power Grid Science Research Institute

---

# Outline

Introduction

Independent Component Analysis in SCA

Case I: Table Re-computation Schemes

Case II: DPAContest v4.2

Conclusion

---

## Side Channel Analysis

### SCA

- ⚡ Attacks based on information leakage (timing, power consumption, electromagnetic emission, etc.)
- ⚡ Recover the secret key within a few minutes (1 — several million traces)

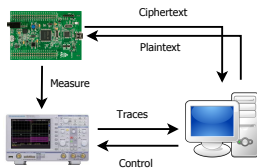


Figure: Side Channel Analysis

---

## Conventional (non-profiled) SCA

A typical “guess-and-determine” procedure

- ✂ Take a guess on a fixed secret (eg. the partial key  $k$ )
- ✂ Statistic calculations (distinguishers) on the leakages
- ✂ Find the most likely key guess  $k$

## Conventional (non-profiled) SCA

Eg. a typical HW-CPA on AES, with  $T$  power traces

🔥 Step 1: Take a guess on one key byte  $k$

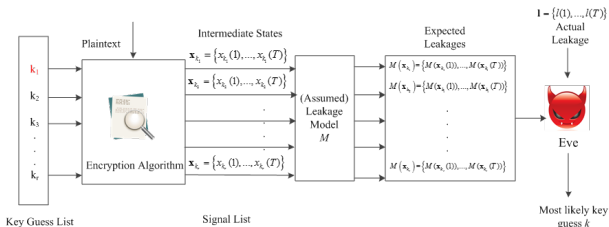


Figure: Conventional SCA: Step 1

## Conventional (non-profiled) SCA

Eg. a typical HW-CPA on AES, with  $T$  power traces

🔥 Step 2: Compute the intermediate states from the plaintexts and  $k$

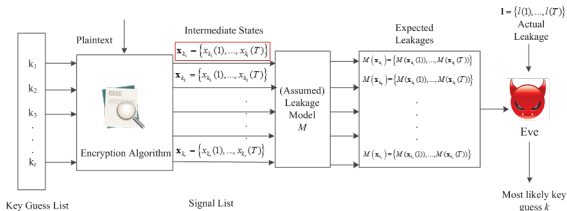


Figure: Conventional SCA: Step 2

## Conventional (non-profiled) SCA

Eg. a typical HW-CPA on AES, with  $T$  power traces

🔥 Step 3: Compute the expected leakages as  $M(x) = HW(x)$

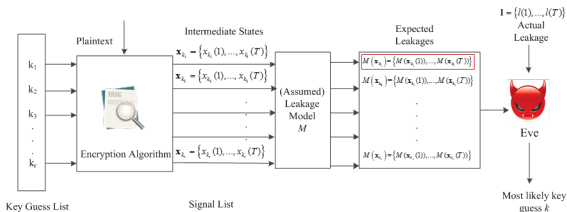


Figure: Conventional SCA: Step 3

## Conventional (non-profiled) SCA

Eg. a typical HW-CPA on AES, with  $T$  power traces

🔥 Step 4: Find out the key guess with the highest correlation coefficient

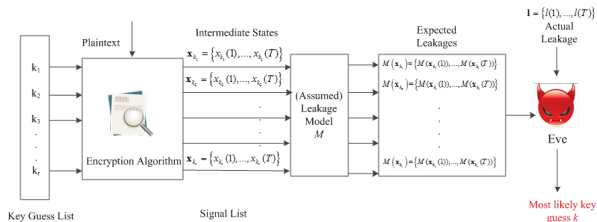


Figure: Conventional SCA: Step 4



---

## Countermeasure: random masking

A 1st-order masking scheme splits the intermediate state  $x$  into  $(x_m, m)$ , where

✦  $x_m \oplus m = x$

✦ mask  $m$  changes after each encryption

Therefore, the attacker's options are:

✦ **Recovering  $m$  first**

✦ “Cancel”  $m$  by combining multiple leakages (a.k.a. higher order attack)

---

## Recovering random masks

### Profiled attacks:

- ✂ Match the leakage of  $m$  with pre-built templates
- ✂ One valid trace for each  $m$
- ✂ Possible, yet not as powerful as conventional key-recovery

### Non-profiled attacks:

- ✂ Cannot “guess-and-determine”: only one trace for each  $m$  (v.s. the fix secret  $k$ )
- ✂ Cannot recover  $m$  **in general**
- ✂ However, under certain circumstances...

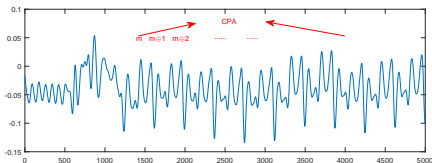
---

## Non-profiled mask recovery

“Horizontal attacks” on the “table re-computation” procedure:

- ✦ Use all  $2^n$  leakages of  $m, m \oplus 1, \dots, m \oplus (2^n - 1)$  ( $n$ : bit-width of the Sbox)
- ✦ Verify guesses of  $m$  with all the “horizontal” information on this trace
- ✦ Same as a vertical SCA with  $2^n$  traces

```
2:  for all  $u \in \{0, 1\}^n$  do
3:     $T'(u) = T(u \oplus x^{(i)})$ 
4:  end for
```



---

## Non-profiled mask recovery

Previous “Horizontal attacks”:

- ✦ AES on 8-bit microcontroller, 900 traces [WISA 09]
- ✦ AES on 8/32-bit microcontroller, with several hiding techniques [FSE 13]
- ✦ Need enough leakage samples for “guess-and-determine” ( $n$  is large enough)

Can we do better than this?

Introduction

**Independent Component Analysis in SCA**

Case I: Table Re-computation Schemes

Case II: DPAContest v4.2

Conclusion

---

## Independent Component Analysis (ICA)

A typical example of “Blind Source Separation (BSS)”:

- ✦ Blind sources  $\mathbf{S} = (s_1, s_2, \dots, s_n)$
- ✦ Linear mix matrix  $\mathbf{A}$
- ✦  $m$  observations  $\mathbf{Y} = (y_1, y_2, \dots, y_m)$
- ✦  $\mathbf{Y} = \mathbf{A} * \mathbf{S} + \mathbf{N}$  ( $\mathbf{N}$  represents the noise)

Goal: Learn  $\mathbf{S}$  from  $\mathbf{Y}$

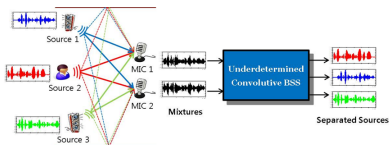


Figure: Blind Source Separation

---

## ICA in side channel analysis

Denoising:

- ✦ Idea/initial discussion [e-SMART 03]
- ✦ AES traces on software/hardware, with/without countermeasure [COSADE 18]

Other applications (all related to non-profiled state recovery):

- ✦ Side Channel Analysis based Reverse Engineering [CT-RSA 17]
- ✦ Middle round attack [CT-RSA 17]

---

## ICA & Random masks

### Rationale

- ✦ Unlike conventional SCA, ICA recovers the “changing secrets”
- ✦ The following key recovery should be easy

### Technical Issue

- ✦ Constructing multiple “ICA observations”
  - ▶ To recover an  $n$ -bit state  $x$ , ICA needs  $n$  different leakages  $(L_1(x), \dots, L_n(x))$



Introduction

Independent Component Analysis in SCA

**Case I: Table Re-computation Schemes**

Case II: DPAContest v4.2

Conclusion

---

## Table re-computation schemes

For a  $d$ -order masking scheme,

- ✂ Compute the masked table according to the first  $d - 1$  shares
- ✂ Look up the last share  $x^{(d)}$

---

**Algorithm 1** A  $d$ -shares table re-computation for an  $n$ -bit Sbox

---

**Input:**  $x^{(1)}, \dots, x^{(d)}$  such that  $x = x^{(1)} \oplus x^{(2)} \oplus \dots \oplus x^{(d)}$   
 Shared table  $T$  such that  $\bigoplus_i T(u)^{(i)} = S(u)$

**Output:** Shared table  $T$  such that  $\bigoplus T(x^{(d)})^{(i)} = S(x)$

```

1: for  $i = 1$  to  $d - 1$  do
2:   for all  $u \in \{0, 1\}^n$  do
3:      $T'(u) = T(u \oplus x^{(i)})$ 
4:   end for
5:    $T = T'$ 
6: end for
  
```

Generate masked table

2^n table look-ups for each share

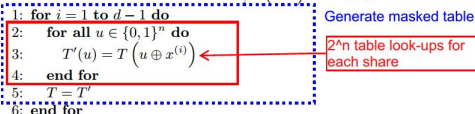


Figure: A  $d$ -order table re-computation scheme

---

## Utilising the $2^n$ table look-ups

Without loss of generality, let us assume the leakage for look up  $x$  is  $L(x)$ :

- ✂  $2^n$  leakage samples available:  $(L(x), L(x \oplus 1), \dots, L(x \oplus 2^n - 1))$
- ✂ Horizontal CPA[WISA 09, FSE 13]:
  - ▶ Guess the mask share  $x$
  - ▶ Recover  $x$  with conventional HW-CPA
- ✂ ICA-based recovery
  - ▶ If  $L$  is linear,  $L(x \oplus c)$  can be regarded as  $L_c(x)$  [CT-RSA 17]
  - ▶ ICA takes  $n$  constants  $c$ -s to recover  $x$

---

## Horizontal attacks v.s. ICA

Horizontal attacks are mainly restricted by  $n$

### ✂ Leakage model

- ▶ Horizontal attacks:  $L$  can be anything, but most likely HW
- ▶ ICA:  $L$  must be linear (i.e. weighted HW)

### ✂ Sample size (“the number of traces”)

- ▶ Horizontal attacks:  $2^n$  available, enough for  $n = 8$  [WISA 09, FSE 13]
- ▶ ICA:  $2^n$  available, in theory only takes  $n$

---

## Experimental Validation

### Acquisition Setup:

- ✦ Target: IC card with 8-bit microprocessor (Atmega163)
- ✦ PicoScope 3206D, running at 1GSa/s
- ✦ 2-shares implementation of the Sbox of PRESENT ( $n = 4$ )
- ✦ 200 traces, with 2M samples on the trace
- ✦ Testing attacks
  - ▶ Horizontal CPA with HW model
  - ▶ ICA-based mask recovery

---

## Experimental Validation

- ✿ Lack of samples for horizontal CPA ( $\sim 30\%$  mask recovery)
- ✿ ICA-based mask recovery works well ( $\sim 20$  traces for key recovery)

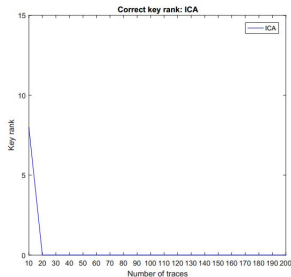
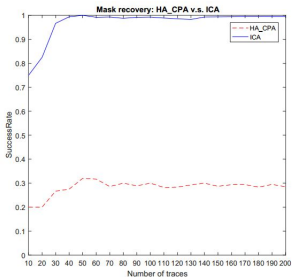


Figure: Horizontal CPA v.s. ICA

---

Introduction

Independent Component Analysis in SCA

Case I: Table Re-computation Schemes

**Case II: DPAContest v4.2**

Conclusion

---

## The Rotating Sbox Masking Scheme (RSM)

A type of Low-Entropy Masking Scheme (LEMS)

- ✂ A fix 16B mask set  $M$
- ✂ Choose a 4-bit random offset  $x$  and use  $M[x]$  as the 1st round mask
- ✂ 16 **pre-computed** masked tables (mask  $M[x]$  to mask  $M[x + 1]$ )
- ✂ The  $(r + 1)$ -th round is  $M[(x + r) \bmod 16]$

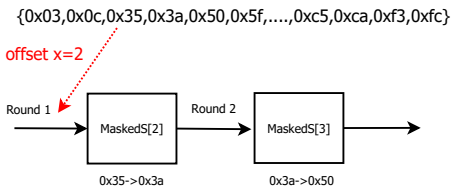


Figure: RSM scheme in DPAContest v4.2



---

## The Rotating Sbox Masking Scheme (RSM)

No table re-computation, but:

- ✂ In each round, the random offset  $x$  must be loaded from memory
- ✂ Leakage of  $(x + r) \bmod 16$

---

**Algorithm 3** ASM codes of the masked Sbox computation in DPAContest v4.2

1: ldi YH,hi8(_offset_)	▷ point to the offset array location
2: ldi YL,0x00	
3: ld offset, Y	▷ load offset $x$
4: ldi ZH, hi8(aes.sbox0)	Generate the leakage
5: add offset,I2	▷ $x = x + r$
6: andi offset,0x0F	▷ $x = x \bmod 16$
7: add ZH, offset	▷ Determine the masked table
8: clr ZL	
9: mov ZL, ST11	▷ Table look up
10: clr ST11	
11: lpm ST11, Z	

---

---

## Utilising such leakage...

Same strategy as before

✦ 10 leakage samples available

$$(L(x), L((x + 1) \bmod 16), \dots, L((x + 9) \bmod 16))$$

✦ Horizontal CPA

- ▶ Guess the mask share  $x$
- ▶ Recover  $x$  with conventional HW-CPA (only 10 samples!)

✦ ICA-based recovery

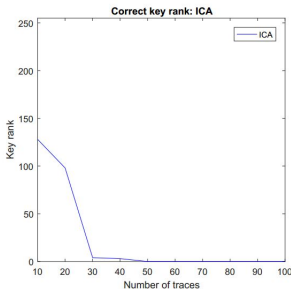
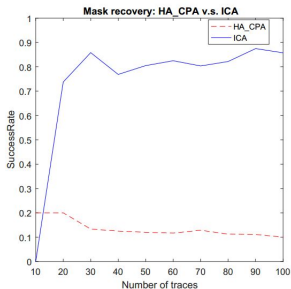
- ▶  $L((x + r) \bmod 16)$  cannot be regarded as  $L_r(x)$
- ▶ For certain  $r$ -s (1,2,4,8), can be approximated by  $L(x \oplus r)$
- ▶ Price to pay: adding more algorithmic noise

---

## Experimental Validation

EM traces from DPAContest V4.2,

- ✂ Lack of samples for horizontal CPA ( $\sim 15\%$  mask recovery)
- ✂ ICA-based mask recovery works well ( $\sim 30$  traces for key recovery)



---

Figure: Horizontal CPA v.s. ICA

---

## Other attacks

Compared with the hall of fame of DPAContest v4.2:

- ✦ Not the most efficient attack
  - ▶ Non-profiled: 14 traces
- ✦ Not really a fair comparison
  - ▶ ICA does not utilise the “Implementation-specific” information (eg. target register, bus, etc.)
  - ▶ Hard to fix by cautious implementations

A straightforward way to avoid accidental computation of the form  $m_i \oplus m_{i+1}$  in the implementation flow, is to rewrite the complete code in assembly language. However writing assembly code is a tedious and error-prone task. A common practice is to write only the sensitive modules of the code in assembly. This is considered as best practice to avoid any surprises from compilation. Another precaution which must be taken at this stage is register precharge. If we precharge every general purpose register to '0' value before writing in a new value, we can avoid all leakages of form  $m_i \oplus m_{i+1}$ . By ensuring these two

Introduction

Independent Component Analysis in SCA

Case I: Table Re-computation Schemes

Case II: DPAContest v4.2

Conclusion

---

---

## Conclusion

### Take home message

- ✦ ICA-based mask recovery works for some masking schemes
- ✦ A “pre-trace” changing secret is not always intrinsically secure
  - ▶ Profiled: template attacks suppose the masks can be profiled
  - ▶ Non-profiled—horizontal: having a lot leakages of the same secret helps
  - ▶ Non-profiled—ICA: the secret can possibly be recovered even if there are only **a few** leakage samples available

---

# Acknowledgement



Questions?