On Efficient and Secure Code-based Masking: A Pragmatic Evaluation

Qianmei Wu[1], Wei Cheng[2,3], Sylvain Guilley[3,2], Fan Zhang[1], and Wei Fu[4]

[1] Zhejiang University, China

[2] LTCI, Télécom Paris, Institut Polytechnique de Paris, France

[3] Secure-IC S.A.S., France [4] Ant Group, China

wei.cheng@telecom-paris.fr

IACR Cryptographic Hardware and Embedded Systems
Sep 20, 2022

Side-Channel Attacks

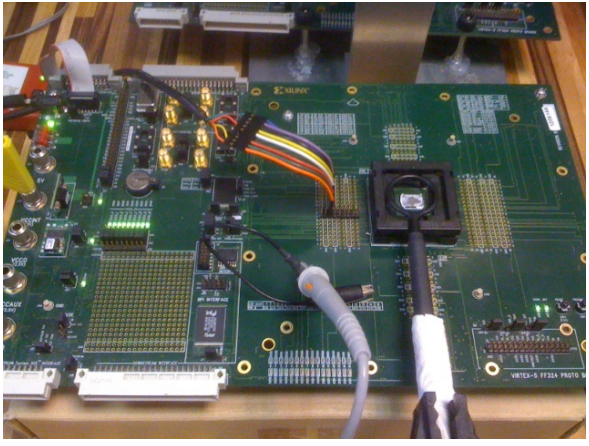


Figure 1: Observable leakages from the manipulation of intermediate variable

Masking against SCA

Masking

- **Security**: provable security [Ishai et al. Crypto'03], [Rivain et al. CHES'10]
- **Costs**: quadratically or cubically in security orders [Grosso et al. CHES'13]
- **Others**: algorithmic level

Boolean masking [Chari et al. Crypto'99]

Let $\mathbb{K} = \mathbb{F}_{2^l}$ be a finite field, for a Boolean masking with n shares:

$$Z = (Z_1, \cdots, Z_n) = (X + \sum_{i=1}^{n-1} Y_i, Y_1, Y_2, \cdots, Y_{n-1})$$

- $X \in \mathbb{K}$: sensitive variable
- $Y \in \mathbb{K}^{n-1}$: random masks
- $Z \in \mathbb{K}^n$: masked variables

Code-base Masking

Uniform representation

The encoding of code-based masking [Wang et al. CHES'20], [Cheng et al. CHES'21]:

$$Z = X\mathbf{G} + Y\mathbf{H}$$

- $X \in \mathbb{K}^k$: sensitive variables
- $Y \in \mathbb{K}^m$: random masks
- $Z \in \mathbb{K}^n$: masked variables
- $\mathbf{G} \in \mathbb{K}^{k\times n}$ and $\mathbf{H} \in \mathbb{K}^{m\times n}$: generator matrices of $\mathcal{C}$ and $\mathcal{D}$, resp.

Constraints and redundancy

- Condition for decoding: $\mathcal{C} \cap \mathcal{D} = \{\mathbf{0}\}$
- Without redundancy: $n = k + m$; with redundancy: $n > k + m$

Code-base Masking: Examples

Boolean masking [Chari et al. Crypto'99]

$$
\begin{aligned}
Z &= (Z_1, \ldots, Z_n) \\
&= \left(X + \sum_{i=1}^{n-1} Y_i,\ Y_1, Y_2, \ldots, Y_{n-1} \right) \\
&= X\mathbf{G} + Y\mathbf{H},
\end{aligned}
$$

where $\mathbf{G}$ and $\mathbf{H}$ are:

$$
\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \end{pmatrix} \quad \in \mathbb{K}^{1\times n}
$$

$$
\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \cdots & 1 \end{pmatrix} \quad \in \mathbb{K}^{t\times n}.
$$

Code-base Masking: Examples

Boolean masking [Chari et al. Crypto'99]

$$
\begin{aligned}
Z &= (Z_1, \ldots, Z_n) \\
&= \left(X + \sum_{i=1}^{n-1} Y_i,\ Y_1, Y_2, \ldots, Y_{n-1} \right) \\
&= X\mathbf{G} + Y\mathbf{H},
\end{aligned}
$$

where $\mathbf{G}$ and $\mathbf{H}$ are:

$$
\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \end{pmatrix} \quad \in \mathbb{K}^{1\times n}
$$

$$
\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \cdots & 1 \end{pmatrix} \quad \in \mathbb{K}^{t\times n}.
$$

Inner Product masking [Balasch et al. EC'15]

$$
\begin{aligned}
Z &= (Z_1, \ldots, Z_n) \\
&= \left(X + \sum_{i=1}^{n-1} a_i Y_i,\ Y_1, Y_2, \ldots, Y_{n-1} \right) \\
&= X\mathbf{G} + Y\mathbf{H},
\end{aligned}
$$

where $\mathbf{G}$ and $\mathbf{H}$ are:

$$
\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \end{pmatrix} \quad \in \mathbb{K}^{1\times n}
$$

$$
\mathbf{H} = \begin{pmatrix} a_1 & 1 & 0 & \cdots & 0 \\ a_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_t & 0 & 0 & \cdots & 1 \end{pmatrix} \quad \in \mathbb{K}^{t\times n}.
$$

Code-based Masking: two issues

High computational overhead

- Improved IPM: $\approx \mathbf{1.5}$ times to Boolean one [Balasch et al. EC'15 & AC'17]
- Cost amortization works better for large n, e.g., $\boldsymbol{n \geq 5}$ [Wang et al. CHES'20]

Code-based Masking: two issues

High computational overhead

- Improved IPM: $\approx \mathbf{1.5}$ times to Boolean one [Balasch et al. EC'15 & AC'17]
- Cost amortization works better for large n, e.g., $\boldsymbol{n \geq 5}$ [Wang et al. CHES'20]

Lack of real-world attack-based evaluations

- Most of works are based on theoretical analysis and/or simulations
- A few works are with *t-test* for leakage detection

Practical Encoder

$$\mathbf{A} = \begin{pmatrix} \mathbf{G} \\ \mathbf{H} \end{pmatrix} = \underbrace{\left.\begin{pmatrix} 1 & & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ & \ddots & & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ a_{11} & \cdots & a_{1k} & a_{1(k+1)} & \cdots & a_{1(n-m)} & 1 & & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & & \ddots & \\ a_{m1} & \cdots & a_{mk} & a_{m(k+1)} & \cdots & a_{m(n-m)} & 0 & & 1 \end{pmatrix}\right\} k+m}_{n}$$

where

- $\mathbf{G} = [\mathbf{I}_k, \mathbf{O}^{k\times(n-k)}]$
- $\mathbf{H} = [\mathbf{R}, \mathbf{I}_m]$ in which $\mathbf{R}$ is an $m \times (n-m)$ matrix with $a_{ij} \in \mathbb{F}_q$ for designers

Practical Encoder

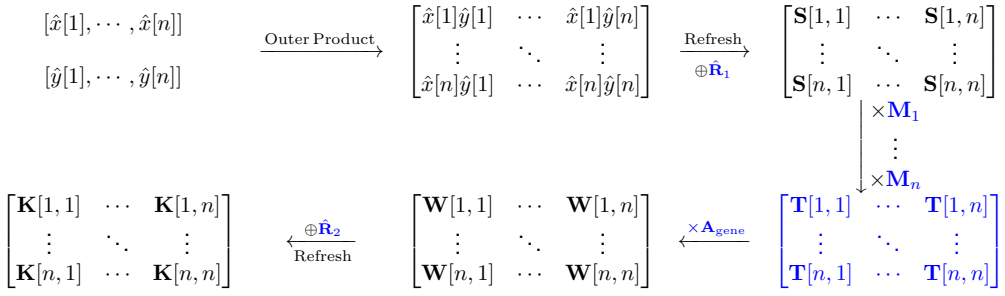
$$\mathbf{A} = \begin{pmatrix} \mathbf{G} \\ \mathbf{H} \end{pmatrix} = \underbrace{\left.\begin{pmatrix} 1 & & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ & \ddots & & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ a_{11} & \cdots & a_{1k} & a_{1(k+1)} & \cdots & a_{1(n-m)} & 1 & & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & & \ddots & \\ a_{m1} & \cdots & a_{mk} & a_{m(k+1)} & \cdots & a_{m(n-m)} & 0 & & 1 \end{pmatrix}\right\} k+m}_{n}$$

where

- $\mathbf{G} = [\mathbf{I}_k, \mathbf{O}^{k\times(n-k)}]$
- $\mathbf{H} = [\mathbf{R}, \mathbf{I}_m]$ in which $\mathbf{R}$ is an $m \times (n-m)$ matrix with $a_{ij} \in \mathbb{F}_q$ for designers

- **Validity**: $\mathbf{G}$ and $\mathbf{H}$ are both full-rank and $\mathcal{C}_{\mathbf{G}} \cap \mathcal{C}_{\mathbf{H}} = \{\mathbf{0}\}$.
- **Generality**: generic encoder $\mathbf{A}_{\text{gene}}$ in [Wang et al. CHES'20] shall be set as $\mathbf{A}$.
- **Sparsity**: more computationally friendly as multiplications can be omitted if as many as $a_{i,j}$ are 0 or 1.

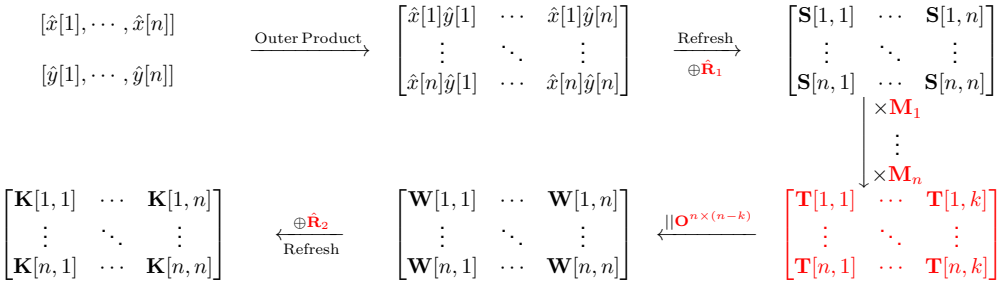
Multiplication Gadget of [Wang et al. CHES'20]



Our improvements:

- Simplifying refresh variables $\hat{\mathbf{R}}_1$ and $\hat{\mathbf{R}}_2$:
  $\hat{\mathbf{R}}_1 = ([\mathbf{O}^{n\times k}, \mathbf{R}_1]\mathbf{A})^T \longrightarrow \hat{\mathbf{R}}_1 = (\mathbf{R}_1 \times \mathbf{H})^T$
  $\hat{\mathbf{R}}_2 = [\mathbf{O}^{n\times k}, \mathbf{R}_2]\mathbf{A} \longrightarrow \hat{\mathbf{R}}_2 = \mathbf{R}_2 \times \mathbf{H}$
- Reducing Internal Computation: $\mathbf{M}_i = \mathbf{M}_i^*[*, 1:k]$
- Removing Re-encoding: $\mathbf{W} = \mathbf{T}^{n\times(k+m)}\mathbf{A}_{\text{gene}} \longrightarrow \mathbf{W} = [\mathbf{T}^{n\times k}, \mathbf{O}^{n\times(n-k)}]$

Improved Multiplication Gadget



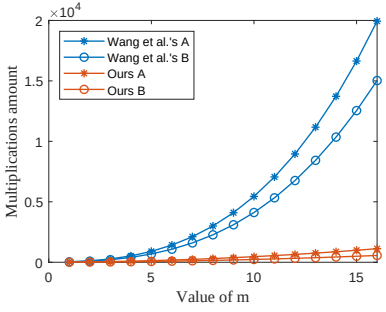
Our improvements:

- Simplifying refresh variables $\hat{\mathbf{R}}_1$ and $\hat{\mathbf{R}}_2$:
  $\hat{\mathbf{R}}_1 = ([\mathbf{O}^{n\times k}, \mathbf{R}_1]\mathbf{A})^T \longrightarrow \hat{\mathbf{R}}_1 = (\mathbf{R}_1 \times \mathbf{H})^T$
  $\hat{\mathbf{R}}_2 = [\mathbf{O}^{n\times k}, \mathbf{R}_2]\mathbf{A} \longrightarrow \hat{\mathbf{R}}_2 = \mathbf{R}_2 \times \mathbf{H}$
- Reducing Internal Computation: $\mathbf{M}_i = \mathbf{M}_i^*[*, 1:k]$
- Removing Re-encoding: $\mathbf{W} = \mathbf{T}^{n\times(k+m)}\mathbf{A}_{\text{gene}} \longrightarrow \mathbf{W} = [\mathbf{T}^{n\times k}, \mathbf{O}^{n\times(n-k)}]$

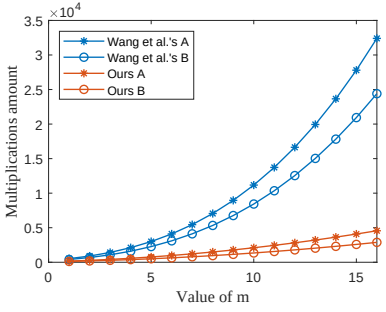
Computational Complexity and Comparisons

Table 1: Comparison of the number of field multiplications in different components.

Component	Scheme in [Wang et al. CHES'20]	Our improved scheme
Encoding	$n(k+m)$	$m(n-m)$
Multiplication Gadget (A)	$n^2(4k+4m+1)$	$n^2(k+1)+2nm(n-m)$
L Gadget (B)	$n^2(3k+3m+1)$	$n^2k+nm(n-m)$
Decoding	nk	$k(n-k)$



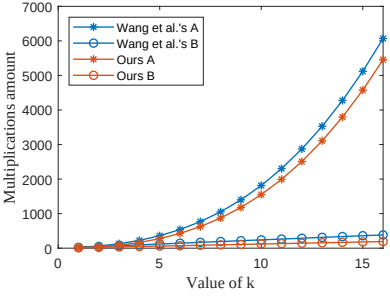
(a) $k = 1$



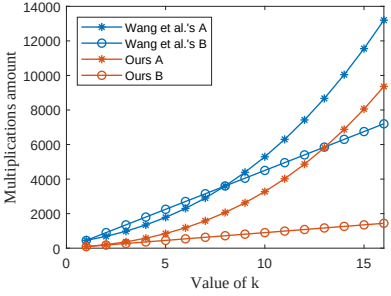
(b) $k = 4$

Figure 2: Comparison of the number of multiplications with increasing m

Cost Amortization and Comparisons



(a) $m = 1$



(b) $m = 4$

Figure 3: Comparison of the number of multiplications with increasing k for the cost amortized operation (A) and k repeated computations (B).

Observations:

- when m is small, cost amortized operation shows a negative effect
- If m increases (e.g., $m \geq 5$), the cost amortization could be effective
- In our case, the potential advantage of the cost amortized operation vanishes

Implementation Results

Table 2: Implementation platforms.

	Target board	Operating frequency	Number of registers	Barrel shift
[Balasch et al. AC'17]	AVR ATMega163	8MHz	32	No
This work	ARM STM32F407	168MHz	16	Yes

Table 3: Performance comparison by clock cycles for implementations.

	BM in [Balasch et al. AC'17]	IPM in [Balasch et al. AC'17]	IPM(C)	Our BM	Our IPM
2-share	110569	157196	812314	155062	193765
3-share	230221	372225	1730163	285025	334983

Evaluation Objects

Three types

- **Non-redundant type**: $n = k + m$ and $k = 1$, we take $m \in \{1, 2\}$
- **Amortized type**: $n = k + m$ and $k > 1$, we take $m = 1$ and $k \in \{1, 2, 4\}$
- **Redundant type**: $n > k + m$, we take $k = m = 1$ and $n \in \{2, 3, 4\}$

Targets

A masking scheme is composed of *encoding function* and *private computation*.

- **Encoding**: the output of the first SubBytes transformation in the first AES round
- **Computation**: the matrix $\mathbf{T}$ of L Gadget (the weakest part in theory)

Weakest Part of Computation

$$\overset{\mathbf{1}}{\begin{bmatrix} \hat{x}[1]\hat{y}[1] & \cdots & \hat{x}[1]\hat{y}[n] \\ \vdots & \ddots & \vdots \\ \hat{x}[n]\hat{y}[1] & \cdots & \hat{x}[n]\hat{y}[n] \end{bmatrix}} \rightarrow \begin{matrix} \times\mathbf{M}_1 \\ \vdots \\ \times\mathbf{M}_n \end{matrix} \rightarrow \overset{\mathbf{2}}{\begin{bmatrix} \mathbf{T}[1,1] & \cdots & \mathbf{T}[1,k] \\ \vdots & \ddots & \vdots \\ \mathbf{T}[n,1] & \cdots & \mathbf{T}[n,k] \end{bmatrix}} \xrightarrow{||\mathbf{O}^{n\times(n-k)}} \overset{\mathbf{3}}{\begin{bmatrix} \mathbf{W}[1,1] & \cdots & \mathbf{W}[1,n] \\ \vdots & \ddots & \vdots \\ \mathbf{W}[n,1] & \cdots & \mathbf{W}[n,n] \end{bmatrix}}$$

A back-and-forth Transformation

- **1** $\rightarrow$ **2**: Code-based masking to Boolean masking
- **2** $\rightarrow$ **3**: Boolean masking to Code-based masking

Matrix $\mathbf{T}$: the weakest part

- Matrix $\mathbf{T}$ is the additive sharing of the k unmasked input sensitive variables: $x_j = \sum_{i=1}^{n} \mathbf{T}[*, j], 1 \leq j \leq k$.
- There is no refreshing operation (like $\oplus\hat{\mathbf{R}}_1$ in multiplication gadget) in L Gadget

Evaluation Strategy and Experimental Setup

Evaluation strategy

- **Leakage assessment**: TVLA (*t-test*) as in [Balasch et al. AC'17]
- **Attacks**: 2nd-order CPA and template attacks (TA)
- **Metrics**: Success Rate (SR) and Guessing Entropy (GE)

Experimental setup

- **Target**: Legacy STM32F407
- **Acquisition**: Electromagnetic (EM) measurements sampled by a Keysight InfiniiVision DSOX3034T oscilloscope.

Security Evaluation on Non-Redundant Type

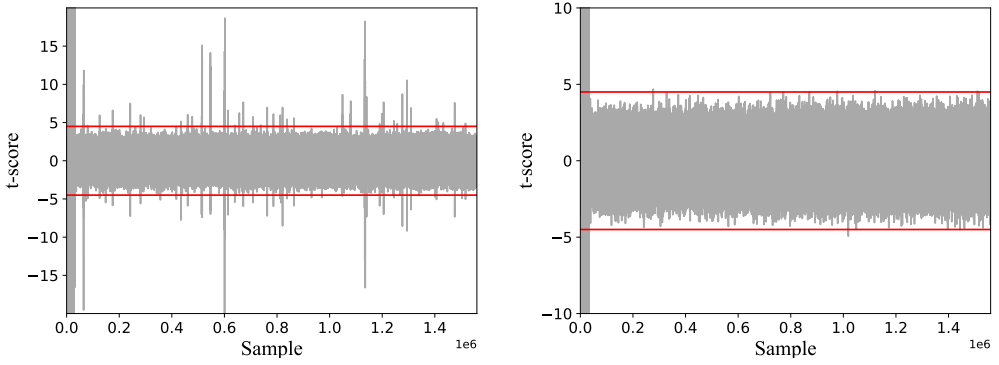
Table 4: Various choices of $\mathbf{A}$ for IPM instances with $n = k + m$, $k = 1$.

Experimental group	BM1 baseline	IPM2	BM2 baseline	IPM3	IPM14	IPM23
Practical encoder $\mathbf{A}$	$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 3 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 14 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 23 & 1 \end{pmatrix}$

Predictions by [Cheng et al. CHES'21]

- Security level: BM1 $<$ IPM2 $<$ BM2 $<$ IPM3 $<$ IPM14 $<$ IPM23
- IPM23 is one of the optimal encoders for 2-share IPM.

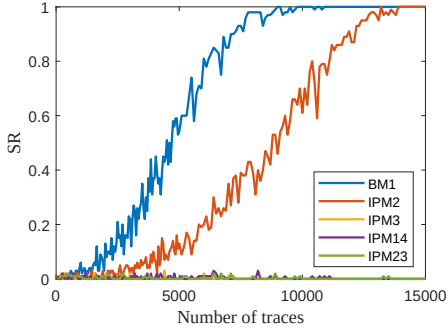
Non-Redundant Type: Leakage Assessment



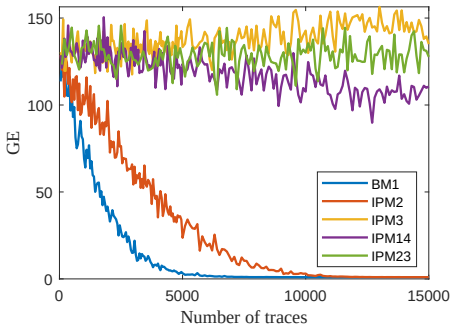
(a) BM1 (b) IPM23

Figure 4: T-test results with TRNG activated and sampling rate at 156 MHz, and EM covers the first 2.5 AES rounds.

Non-Redundant Type: Attack-based Evaluation



(a) SR of 2nd-order CPA

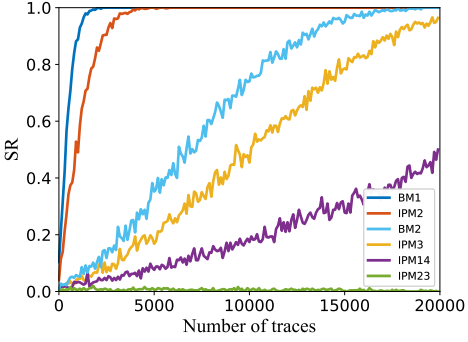


(b) GE of 2nd-order CPA

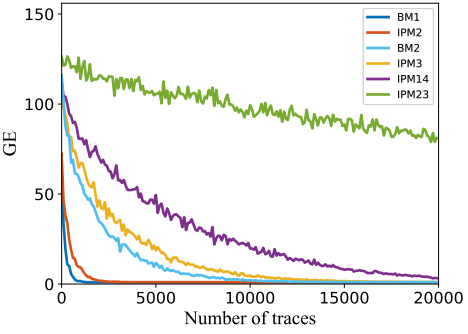
Observations:

- **Security order amplification**: three 2-share IPM instances are secure against 2nd-order CPA

Non-Redundant Type: Attack-based Evaluation



(c) SR of TA

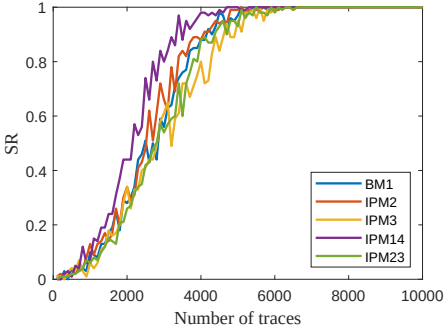


(d) GE of TA

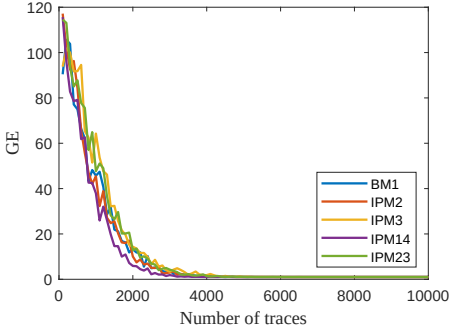
Observations:

- **Practical verification of** [Cheng et al. CHES'21]: the security levels of IPM instances are consistent with the predictions
- **Security order amplification**: even a 2-share IPM can have better resistance against template attacks than the 3-share BM

Non-Redundant Type: Targeting Matrix $\mathbf{T}$



(e) SR of 2nd-order CPA



(f) GE of 2nd-order CPA

Observations:

- All IPM instances can be easily attacked by 2nd-order CPA
- All IPM instances have a similar security level to BM1, losing the feature of "security order amplification"
- We identify a *security bottleneck*: the back-and-forth transformation

Security Evaluation on Amortized Type

Table 5: Various choices of generator matrices $\mathbf{A}$ over $\mathbb{F}_{2^8}^{(k+m)\times n}$ for BM instances in packed code-based masking with $n = k + m$ and $m = 1$.

Experimental group	$\mathrm{BM}^{k=1}$: $k = 1,\ m = 1$	$\mathrm{BM}^{k=2}$: $k = 2,\ m = 1$	$\mathrm{BM}^{k=4}$: $k = 4,\ m = 1$
Practical encoder $\mathbf{A}$	$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$
Security order	1	1	1

- k sensitive variables with one common mask
- Reusing the same random number might leak more information
- Predicted by [Cheng et al. CHES'21]: $\mathrm{BM}^{\mathrm{k=1}} > \mathrm{BM}^{\mathrm{k=2}} > \mathrm{BM}^{\mathrm{k=4}}$

Amortized Type: Leakage Assessment

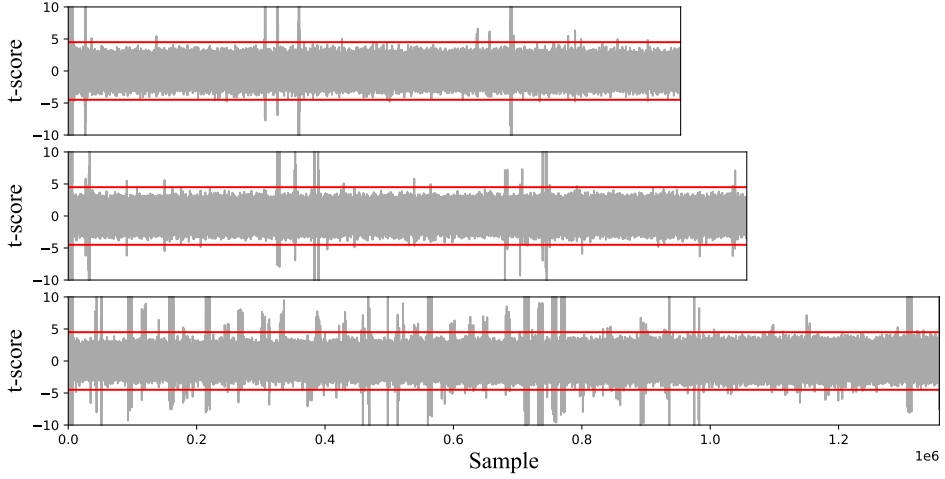
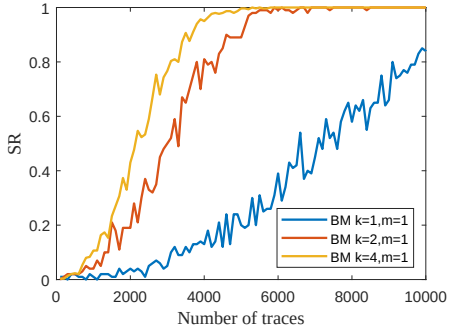
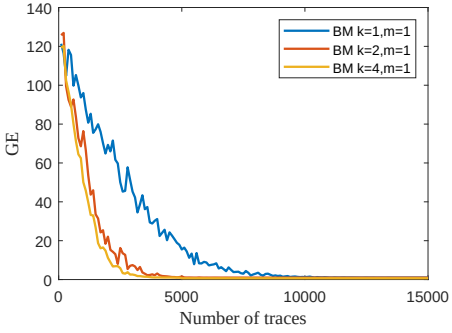


Figure 5: T-test results for $\mathrm{BM}^{k=1}$ (top), $\mathrm{BM}^{k=2}$ (middle) and $\mathrm{BM}^{k=4}$ (bottom) with a sampling rate of 100 MHz. EM covers the first 2.5 AES rounds.

Amortized Type: Attack-based Evaluation



(a) SR of 2nd-order CPA



(b) GE of 2nd-order CPA

Observations:

- The side-channel resistance is as expected
- The value of k also implies the number of subkeys adversaries can rebuild by knowing only one mask

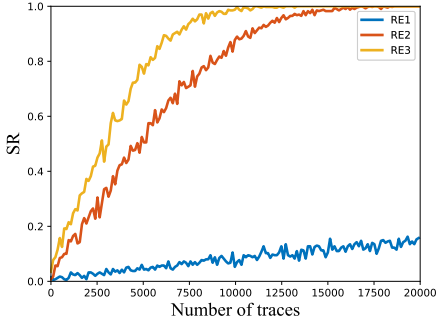
Security Evaluation on Redundant Type

Table 6: Various choices of $\mathbf{A}$ over $\mathbb{F}_{2^8}^{(k+m)\times n}$ in redundant cases with $k = 1$ and $m = 1$.

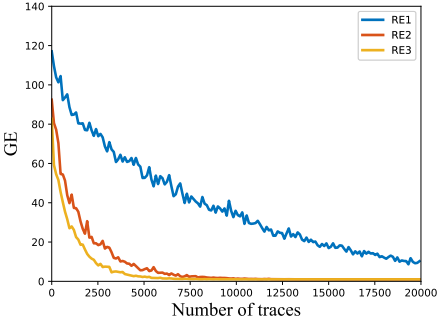
Experimental group	RE1: $n = 2$ Non-redundant	RE2: $n = 3$ Redundant	RE3: $n = 4$ Redundant
Practical encoder $\mathbf{A}$	$\begin{pmatrix} 1 & 0 \\ 23 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 23 & 29 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 23 & 29 & 51 & 1 \end{pmatrix}$
Security order	1	1	1

- Taking $L_2 \in \{23, 29, 51\}$ leads to 2-share IPM instances with the maximized dual distance $d_{\mathcal{D}}^{\perp} = 4$
- The dual distances of the corresponding codes in RE1, RE2 and RE3 are decreasing that $d_{\mathcal{D}}^{\perp} \in \{4, 3, 2\}$
- Security level predicted by [Cheng et al. CHES'21]: RE1 $>$ RE2 $>$ RE3

Redundant Type: Attack-based Evaluation



(c) SR of TA



(d) GE of TA

Observations:

- The redundancy indeed leads to a practical security decrease as expected
- More redundancies have more leakage that shall be exploited

Conclusions and Perspectives

Conclusions

Efficient implementation and practical security evaluation for code-based masking

- Improved constructions for computationally-friendly gadgets and implementations
- Security evaluations on three representative types of code-based masking (non-redundant, amortized, redundant)
- Identified a security bottleneck

Conclusions and Perspectives

Conclusions

Efficient implementation and practical security evaluation for code-based masking

- Improved constructions for computationally-friendly gadgets and implementations
- Security evaluations on three representative types of code-based masking (non-redundant, amortized, redundant)
- Identified a security bottleneck

Future work

- Construction of fully encoded computational framework for code-based masking by addressing the back-and-forth transformation
- Application to lightweight crypto or post-quantum crypto implementations

On Efficient and Secure Code-based Masking: A Pragmatic Evaluation

Thanks for your attention!

Qianmei Wu[1], Wei Cheng[2,3], Sylvain Guilley[3,2], Fan Zhang[1], and Wei Fu[4]

[1] Zhejiang University, China

[2] LTCI, Télécom Paris, Institut Polytechnique de Paris, France

[3] Secure-IC S.A.S., France [4] Ant Group, China

wei.cheng@telecom-paris.fr

IACR Cryptographic Hardware and Embedded Systems
Sep 20, 2022