

Side-channel Masking with Pseudo-Random Generator

Eurocrypt 2020

Jean-Sébastien Coron

Aurélien Greuet

Rina Zeitoun

University of Luxembourg & IDEMIA



11/05/2020



Motivation: side-channel attacks

High-order masking : randomness cost

- Number of randoms is high: can become a bottleneck
- **Our goal:** minimize number of calls to TRNG *and* remain secure in the probing model

	$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$
Rivain-Prouff [RP10]	2880	5760	9600	14400	20160
Belaïd <i>et. al</i> [BBP16]	2560	5120	8000	13120	18240
Faust <i>et. al</i> [FPS17]	1415	2530	6082	6699	20712
This paper	48	108	192	300	432

Table: number of bytes of true randomness to get t -th order security for AES.

Motivation: side-channel attacks

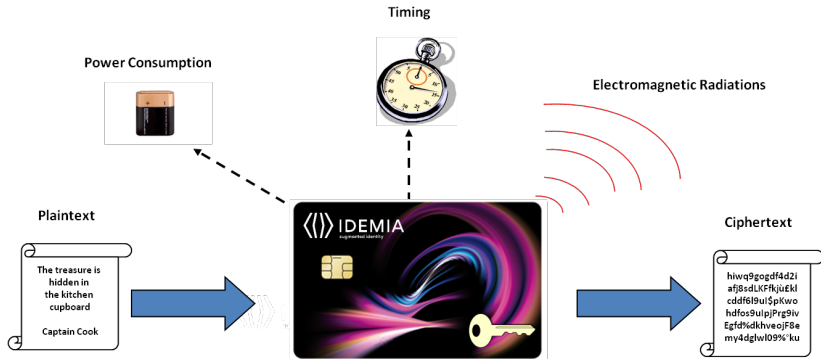
High-order masking : randomness cost

- Number of randoms is high: can become a bottleneck
- **Our goal:** minimize number of calls to TRNG *and* remain secure in the probing model

	$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$
Rivain-Prouff [RP10]	2880	5760	9600	14400	20160
Belaïd <i>et. al</i> [BBP16]	2560	5120	8000	13120	18240
Faust <i>et. al</i> [FPS17]	1415	2530	6082	6699	20712
This paper	48	108	192	300	432

Table: number of bytes of true randomness to get t -th order security for AES.

Side-Channel attacks

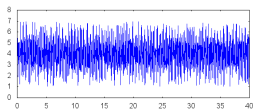


Differential Power Analysis [KJJ99]

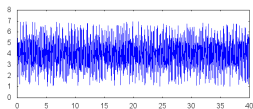
Group by predicted
SBox output bit

111

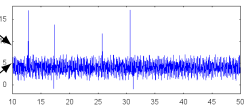
Average trace



000



Differential trace



Countermeasure

Masking countermeasure

Let x be a sensitive variable:

- Generate a random r (different for each execution)
- Mask x using r : $x' = x \oplus r$
- Manipulate x' (instead of x) and r independently

r is random $\Rightarrow x'$ is random \Rightarrow no information on x leaks

 True only in case of **one leakage point**

Countermeasure

Masking countermeasure

Let x be a sensitive variable:

- Generate a random r (different for each execution)
- Mask x using r : $x' = x \oplus r$
- Manipulate x' (instead of x) and r independently

r is random $\Rightarrow x'$ is random \Rightarrow no information on x leaks


 True only in case of **one leakage point**

Masking countermeasure

Let x be a sensitive variable:

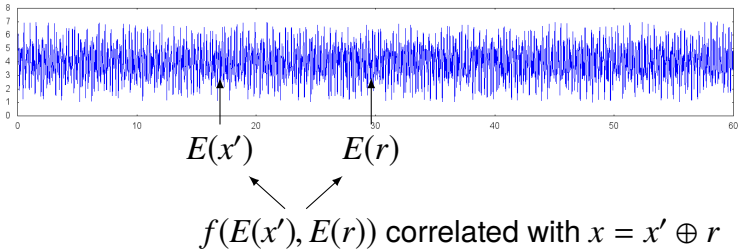
- Generate a random r (different for each execution)
- Mask x using r : $x' = x \oplus r$
- Manipulate x' (instead of x) and r independently

r is random $\Rightarrow x'$ is random \Rightarrow no information on x leaks

 True only in case of **one leakage point**

Differential Power Analysis (second-order)

- Manipulation of $x' = x \oplus r$



- Second-order attack
 - requires more curves but can be practical

Solution: Higher-Order Boolean Masking

Basic principle

Each sensitive variable x is shared into n variables:

$$x = x_1 \oplus x_2 \oplus \dots \oplus x_n$$

- Generate $n - 1$ random variables x_1, x_2, \dots, x_{n-1}
- Initially let $x_n = x \oplus x_1 \oplus x_2 \oplus \dots \oplus x_{n-1}$

Security against DPA attack of order $n - 1$

- Any subset of $n - 1$ shares is uniformly and independently distributed
- ⇒ If we probe at most $n - 1$ shares x_i , we learn nothing about x

Solution: Higher-Order Boolean Masking

Basic principle

Each sensitive variable x is shared into n variables:

$$x = x_1 \oplus x_2 \oplus \dots \oplus x_n$$

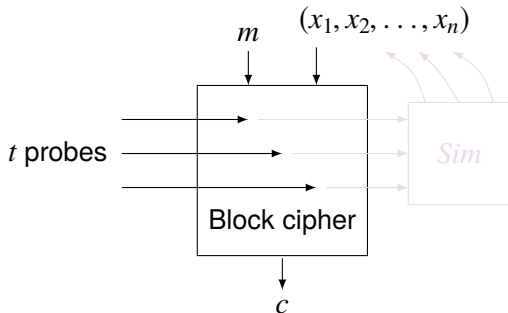
- Generate $n - 1$ random variables x_1, x_2, \dots, x_{n-1}
- Initially let $x_n = x \oplus x_1 \oplus x_2 \oplus \dots \oplus x_{n-1}$

Security against DPA attack of order $n - 1$

- Any subset of $n - 1$ shares is uniformly and independently distributed
 \Rightarrow If we probe at most $n - 1$ shares x_i , we learn nothing about x

ISW Security Model

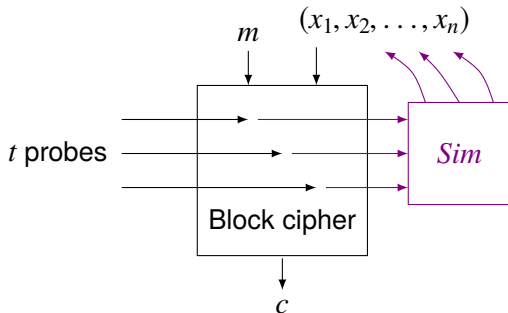
- Proof of security in the Probing Model [ISW03]:



- Show that any $t < n$ probes can be perfectly simulated from at most $n - 1$ of the x_i 's.
- Those $n - 1$ shares x_i are uniformly and independently distributed.
- \Rightarrow The adversary learns nothing from the t probes

ISW Security Model

- Proof of security in the Probing Model [ISW03]:



- Show that any $t < n$ probes can be perfectly simulated from at most $n - 1$ of the x_i 's.
- Those $n - 1$ shares x_i are uniformly and independently distributed.
- \Rightarrow The adversary learns nothing from the t probes

Linear Operations

Computation of $a \oplus b$

- **Inputs:** $(a_i)_i$ and $(b_i)_i$ such that
 - $a_1 \oplus a_2 \oplus \dots \oplus a_n = a$
 - $b_1 \oplus b_2 \oplus \dots \oplus b_n = b$
- **Output:** $(c_i)_i$ such that
 - $(a_1 \oplus b_1) \oplus (a_2 \oplus b_2) \oplus \dots \oplus (a_n \oplus b_n) = a \oplus b \Rightarrow$
 $c_1 \oplus c_2 \oplus \dots \oplus c_n = a \oplus b$

Computation of a^2 in \mathbb{F}_{2^k}

- **Inputs:** $(a_i)_i$ such that
 - $a_1 \oplus a_2 \oplus \dots \oplus a_n = a$
- **Output:** $(c_i)_i$ such that
 - $(a_1^2) \oplus (a_2^2) \oplus \dots \oplus (a_n^2) = a^2 \Rightarrow$
 $c_1 \oplus c_2 \oplus \dots \oplus c_n = a^2$

Linear Operations

Computation of $a \oplus b$

- **Inputs:** $(a_i)_i$ and $(b_i)_i$ such that

- $a_1 \oplus a_2 \oplus \cdots \oplus a_n = a$

- $b_1 \oplus b_2 \oplus \cdots \oplus b_n = b$

- **Output:** $(c_i)_i$ such that

- $(a_1 \oplus b_1) \oplus (a_2 \oplus b_2) \oplus \cdots \oplus (a_n \oplus b_n) = a \oplus b \Rightarrow$

- $c_1 \oplus c_2 \oplus \cdots \oplus c_n = a \oplus b$

Computation of a^2 in \mathbb{F}_{2^k}

- **Inputs:** $(a_i)_i$ such that

- $a_1 \oplus a_2 \oplus \cdots \oplus a_n = a$

- **Output:** $(c_i)_i$ such that

- $(a_1^2) \oplus (a_2^2) \oplus \cdots \oplus (a_n^2) = a^2 \Rightarrow$

- $c_1 \oplus c_2 \oplus \cdots \oplus c_n = a^2$

Secure Multiplication in High-Order Masking Schemes

Secure Computation of $a \times b$

- **Inputs:** $(a_i)_i$ and $(b_i)_i$ such that
 - $a_1 \oplus a_2 \oplus \dots \oplus a_n = a$
 - $b_1 \oplus b_2 \oplus \dots \oplus b_n = b$
- **Output:** $(c_i)_i$ such that
 - $c_1 \oplus c_2 \oplus c_2 \oplus \dots \oplus c_n = a \times b$

Ishai-Sahai-Wagner private circuit [ISW03]

- Secure against t probes for $n = 2t + 1$ shares.
- Number of operations: $O(t^2)$
- Requires $O(t^2)$ randoms per multiplication.

Secure Multiplication in High-Order Masking Schemes

Secure Computation of $a \times b$

- **Inputs:** $(a_i)_i$ and $(b_i)_i$ such that
 - $a_1 \oplus a_2 \oplus \dots \oplus a_n = a$
 - $b_1 \oplus b_2 \oplus \dots \oplus b_n = b$
- **Output:** $(c_i)_i$ such that
 - $c_1 \oplus c_2 \oplus c_2 \oplus \dots \oplus c_n = a \times b$

Ishai-Sahai-Wagner private circuit [ISW03]

- Secure against t probes for $n = 2t + 1$ shares.
- Number of operations: $O(t^2)$
- Requires $O(t^2)$ randoms per multiplication.

Ishai-Sahai-Wagner (ISW) Scheme

Decomposition of the c_i

$$\bigoplus_i c_i = \left(\bigoplus_i a_i \right) \left(\bigoplus_i b_i \right) = \bigoplus_{ij} a_i b_j$$

Example for $n = 3$

For n shares: requires $n(n-1)/2$
fresh random values

Ishai-Sahai-Wagner (ISW) Scheme

Decomposition of the c_i

$$\bigoplus_i c_i = \left(\bigoplus_i a_i \right) \left(\bigoplus_i b_i \right) = \bigoplus_{i,j} a_i b_j$$

Example for $n = 3$

$$\begin{pmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 \\ a_2 b_1 & a_2 b_2 & a_2 b_3 \\ a_3 b_1 & a_3 b_2 & a_3 b_3 \end{pmatrix}$$

👉 For n shares: requires $n(n-1)/2$
fresh random values

Ishai-Sahai-Wagner (ISW) Scheme

Decomposition of the c_i

$$\bigoplus_i c_i = \left(\bigoplus_i a_i \right) \left(\bigoplus_i b_i \right) = \bigoplus_{ij} a_i b_j$$

Example for $n = 3$

$$\begin{pmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 \\ a_2 b_1 & a_2 b_2 & a_2 b_3 \\ a_3 b_1 & a_3 b_2 & a_3 b_3 \end{pmatrix} \begin{array}{l} \rightarrow c_1 \\ \rightarrow c_2 \\ \rightarrow c_3 \end{array}$$

For n shares: requires $n(n-1)/2$
fresh random values

Ishai-Sahai-Wagner (ISW) Scheme

Decomposition of the c_i

$$\bigoplus_i c_i = \left(\bigoplus_i a_i \right) \left(\bigoplus_i b_i \right) = \bigoplus_{ij} a_i b_j$$

Example for $n = 3$

$$\begin{pmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 \\ a_2 b_1 & a_2 b_2 & a_2 b_3 \\ a_3 b_1 & a_3 b_2 & a_3 b_3 \end{pmatrix} \begin{array}{l} \rightarrow c_1 \\ \rightarrow c_2 \\ \rightarrow c_3 \end{array}$$

For n shares: requires $n(n-1)/2$ fresh random values

Ishai-Sahai-Wagner (ISW) Scheme

Decomposition of the c_i

$$\bigoplus_i c_i = \left(\bigoplus_i a_i \right) \left(\bigoplus_i b_i \right) = \bigoplus_{i,j} a_i b_j$$

Example for $n = 3$

$$\begin{pmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 \\ 0 & a_2 b_2 & a_2 b_3 \\ 0 & 0 & a_3 b_3 \end{pmatrix} \oplus \begin{pmatrix} 0 & 0 & 0 \\ a_2 b_1 & 0 & 0 \\ a_3 b_1 & a_3 b_2 & 0 \end{pmatrix}$$

For n shares: requires $n(n-1)/2$
fresh random values

Ishai-Sahai-Wagner (ISW) Scheme

Decomposition of the c_i

$$\bigoplus_i c_i = \left(\bigoplus_i a_i \right) \left(\bigoplus_i b_i \right) = \bigoplus_{i,j} a_i b_j$$

Example for $n = 3$

$$\begin{pmatrix} a_0 b_1 & a_1 b_2 & a_1 b_3 \\ 0 & a_2 b_2 & a_2 b_3 \\ 0 & 0 & a_3 b_3 \end{pmatrix} \oplus \begin{pmatrix} 0 & a_2 b_1 & a_3 b_1 \\ 0 & 0 & a_3 b_2 \\ 0 & 0 & 0 \end{pmatrix}$$

For n shares: requires $n(n-1)/2$
fresh random values

Ishai-Sahai-Wagner (ISW) Scheme

Decomposition of the c_i

$$\bigoplus_i c_i = \left(\bigoplus_i a_i \right) \left(\bigoplus_i b_i \right) = \bigoplus_{i,j} a_i b_j$$

Example for $n = 3$

$$\begin{pmatrix} a_1 b_1 & a_1 b_2 \oplus a_2 b_1 & a_1 b_3 \oplus a_3 b_1 \\ 0 & a_2 b_2 & a_2 b_3 \oplus a_3 b_2 \\ 0 & 0 & a_3 b_3 \end{pmatrix}$$

👉 For n shares: requires $n(n-1)/2$
fresh random values

Ishai-Sahai-Wagner (ISW) Scheme

Decomposition of the c_i

$$\bigoplus_i c_i = \left(\bigoplus_i a_i \right) \left(\bigoplus_i b_i \right) = \bigoplus_{i,j} a_i b_j$$

Example for $n = 3$

$$\begin{pmatrix} a_1 b_1 & a_1 b_2 \oplus a_2 b_1 & a_1 b_3 \oplus a_3 b_1 \\ 0 & a_2 b_2 & a_2 b_3 \oplus a_3 b_2 \\ 0 & 0 & a_3 b_3 \end{pmatrix}$$

For n shares: requires $n(n-1)/2$
fresh random values

Ishai-Sahai-Wagner (ISW) Scheme

Decomposition of the c_i

$$\bigoplus_i c_i = \left(\bigoplus_i a_i \right) \left(\bigoplus_i b_i \right) = \bigoplus_{i,j} a_i b_j$$

Example for $n = 3$

$$\begin{pmatrix} a_1 b_1 & a_1 b_2 \oplus a_2 b_1 & a_1 b_3 \oplus a_3 b_1 \\ 0 & a_2 b_2 & a_2 b_3 \oplus a_3 b_2 \\ 0 & 0 & a_3 b_3 \end{pmatrix} \oplus \begin{pmatrix} 0 & r_{1,2} & r_{1,3} \\ 0 & 0 & r_{2,3} \\ 0 & 0 & 0 \end{pmatrix}$$

For n shares: requires $n(n-1)/2$
fresh random values

Ishai-Sahai-Wagner (ISW) Scheme

Decomposition of the c_i

$$\bigoplus_i c_i = \left(\bigoplus_i a_i \right) \left(\bigoplus_i b_i \right) = \bigoplus_{i,j} a_i b_j$$

Example for $n = 3$

$$\begin{pmatrix} a_1 b_1 & a_1 b_2 \oplus a_2 b_1 & a_1 b_3 \oplus a_3 b_1 \\ 0 & a_2 b_2 & a_2 b_3 \oplus a_3 b_2 \\ 0 & 0 & a_3 b_3 \end{pmatrix} \oplus \begin{pmatrix} 0 & r_{1,2} & r_{1,3} \\ r_{1,2} & 0 & r_{2,3} \\ r_{1,3} & r_{2,3} & 0 \end{pmatrix}$$

For n shares: requires $n(n-1)/2$
fresh random values

Ishai-Sahai-Wagner (ISW) Scheme

Decomposition of the c_i

$$\bigoplus_i c_i = \left(\bigoplus_i a_i \right) \left(\bigoplus_i b_i \right) = \bigoplus_{i,j} a_i b_j$$

Example for $n = 3$

$$\begin{pmatrix} a_1 b_1 & (r_{1,2} \oplus a_1 b_2) \oplus a_2 b_1 & (r_{1,3} \oplus a_1 b_3) \oplus a_3 b_1 \\ r_{1,2} & a_2 b_2 & (r_{2,3} \oplus a_2 b_3) \oplus a_3 b_2 \\ r_{1,3} & r_{2,3} & a_3 b_3 \end{pmatrix}$$

For n shares: requires $n(n-1)/2$ fresh random values

Ishai-Sahai-Wagner (ISW) Scheme

Decomposition of the c_i

$$\bigoplus_i c_i = \left(\bigoplus_i a_i \right) \left(\bigoplus_i b_i \right) = \bigoplus_{i,j} a_i b_j$$

Example for $n = 3$

$$\begin{pmatrix} a_1 b_1 & (r_{1,2} \oplus a_1 b_2) \oplus a_2 b_1 & (r_{1,3} \oplus a_1 b_3) \oplus a_3 b_1 \\ r_{1,2} & a_2 b_2 & (r_{2,3} \oplus a_2 b_3) \oplus a_3 b_2 \\ r_{1,3} & r_{2,3} & a_3 b_3 \end{pmatrix} \begin{array}{l} \rightarrow c_1 \\ \rightarrow c_2 \\ \rightarrow c_3 \end{array}$$

For n shares: requires $n(n-1)/2$ fresh random values

Ishai-Sahai-Wagner (ISW) Scheme

Decomposition of the c_i

$$\bigoplus_i c_i = \left(\bigoplus_i a_i \right) \left(\bigoplus_i b_i \right) = \bigoplus_{i,j} a_i b_j$$

Example for $n = 3$

$$\begin{pmatrix} a_1 b_1 & (r_{1,2} \oplus a_1 b_2) \oplus a_2 b_1 & (r_{1,3} \oplus a_1 b_3) \oplus a_3 b_1 \\ r_{1,2} & a_2 b_2 & (r_{2,3} \oplus a_2 b_3) \oplus a_3 b_2 \\ r_{1,3} & r_{2,3} & a_3 b_3 \end{pmatrix} \begin{array}{l} \rightarrow c_1 \\ \rightarrow c_2 \\ \rightarrow c_3 \end{array}$$

👉 For n shares: requires $n(n-1)/2$
fresh random values

Secure SBox Computation

Secure Computation of $S(x)$

- **Inputs:** $(x_i)_i$ such that
 - $x_1 \oplus x_2 \oplus \dots \oplus x_n = x$
- **Output:** $(y_i)_i$ such that
 - $y_1 \oplus y_2 \oplus \dots \oplus y_n = S(x)$

[RP10] countermeasure for AES: compute $S(x) = x^{254}$



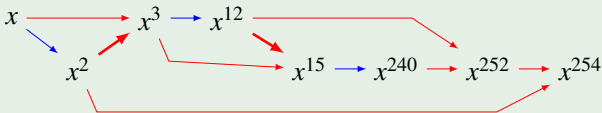
- 4 multiplications over \mathbb{F}_{2^8} with ISW
- 2 mask refreshings

Secure SBox Computation

Secure Computation of $S(x)$

- **Inputs:** $(x_i)_i$ such that
 - $x_1 \oplus x_2 \oplus \dots \oplus x_n = x$
- **Output:** $(y_i)_i$ such that
 - $y_1 \oplus y_2 \oplus \dots \oplus y_n = S(x)$

[RP10] countermeasure for AES: compute $S(x) = x^{254}$



- 4 multiplications over \mathbb{F}_{2^8} with ISW
- 2 mask refreshings

Reducing randomness complexity

Natural idea

- Use a **TRNG** to generate a seed
- Use a **PRG** to generate all needed randoms

👉 **Security:** The Pseudo-Random Generator (PRG) should also be secure against t -probing attacks

Notion of robust PRG [IKL+13]

- Original ISW: randomness complexity $O(t^2 |C|)$
- New randomness complexity $O(t^{3+\varepsilon} \log^k |C|)$
 - Exponential improvement in $|C|$

Reducing randomness complexity

Natural idea

- Use a **TRNG** to generate a seed
- Use a **PRG** to generate all needed randoms

👉 **Security:** The Pseudo-Random Generator (PRG) should also be secure against t -probing attacks

Notion of robust PRG [IKL+13]

- Original ISW: randomness complexity $\mathcal{O}(t^2 |C|)$
- New randomness complexity $\mathcal{O}(t^{3+\varepsilon} \log^k |C|)$
 - Exponential improvement in $|C|$

Notion of robust PRG [IKL+13]

- Original ISW: randomness complexity $O(t^2 |C|)$
- New randomness complexity $O(t^{3+\varepsilon} \log^k |C|)$
- Based on bipartite expander graphs; unpractical.

Our construction

- New randomness complexity $O(t^2(\log t + \log |C|))$
- Without expander graphs
- No need for robust PRG: a simple r -wise independent PRG is sufficient
- Practical implementation with AES

Notion of robust PRG [IKL+13]

- Original ISW: randomness complexity $O(t^2 |C|)$
- New randomness complexity $O(t^{3+\varepsilon} \log^k |C|)$
- Based on bipartite expander graphs; unpractical.

Our construction

- New randomness complexity $O(t^2(\log t + \log |C|))$
- Without expander graphs
- No need for robust PRG: a simple r -wise independent PRG is sufficient
- Practical implementation with AES

r -wise independent PRG

Definition: r -wise independent PRG

- $G : \{0, 1\}^n \rightarrow \{0, 1\}^m$ for $m \gg n$
- Any subset of r bits of $G(a)$ is uniformly and independently distributed, for $a \leftarrow U_n$.

Construction based on polynomial evaluation in \mathbb{F}

- Generate a random **seed** of r elements from TRNG:

$$\vec{a} = (a_0, \dots, a_{r-1}) \in \mathbb{F}^r$$

- Evaluate the polynomial

$$h_{\vec{a}}(x) = \sum_{i=0}^{r-1} a_i x^i \in \mathbb{F}$$

at fixed points $x_j \in \mathbb{F}$

- Interpolation theorem: any subset of r evaluations $h_{\vec{a}}(x_j)$ are iid in \mathbb{F} .

r -wise independent PRG

Definition: r -wise independent PRG

- $G : \{0, 1\}^n \rightarrow \{0, 1\}^m$ for $m \gg n$
- Any subset of r bits of $G(a)$ is uniformly and independently distributed, for $a \leftarrow U_n$.

Construction based on polynomial evaluation in \mathbb{F}

- Generate a random **seed** of r elements from TRNG:

$$\vec{a} = (a_0, \dots, a_{r-1}) \in \mathbb{F}^r$$

- Evaluate the polynomial

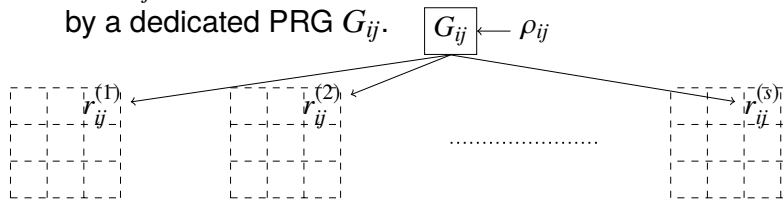
$$h_{\vec{a}}(x) = \sum_{i=0}^{r-1} a_i x^i \in \mathbb{F}$$

at fixed points $x_j \in \mathbb{F}$

- Interpolation theorem: any subset of r evaluations $h_{\vec{a}}(x_j)$ are iid in \mathbb{F} .

Our technique

- The r_{ij} 's in all ISW multiplication gadgets are generated by a dedicated PRG G_{ij} .

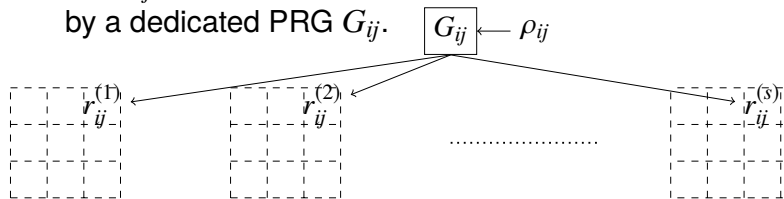


Security of ISW

- If the attacker probes a given r_{ij} , we can give to the attacker all other $r_{ij}^{(k)}$ for free.
- The attacker has no advantage in probing the PRG G_{ij} , since he could get all $r_{ij}^{(k)}$ with a single probe.
- \Rightarrow robustness of PRG is not needed.

Our technique

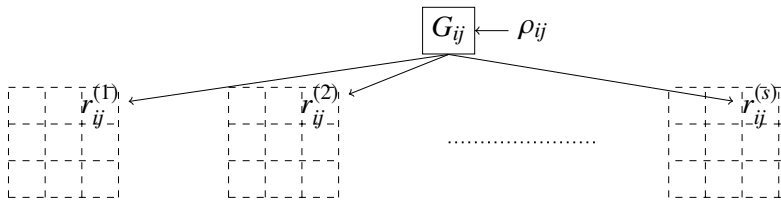
- The r_{ij} 's in all ISW multiplication gadgets are generated by a dedicated PRG G_{ij} .



Security of ISW

- If the attacker probes a given r_{ij} , we can give to the attacker all other $r_{ij}^{(k)}$ for free.
- The attacker has no advantage in probing the PRG G_{ij} , since he could get all $r_{ij}^{(k)}$ with a single probe.
- \Rightarrow robustness of PRG is not needed.

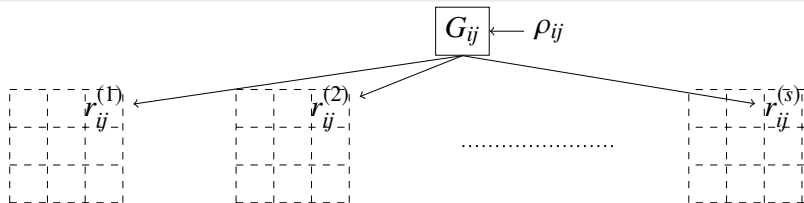
Our technique (2)



r -wise independence of PRG

- With some mask refreshing, every intermediate variable depends on at most a *single* random $r_{ij}^{(k)}$.
- With t probes, the adversary gets information about at most t randoms $r_{ij}^{(k)}$ generated by the PRG G_{ij} .
- \Rightarrow we can use a PRG with r -wise independence $r = t$.

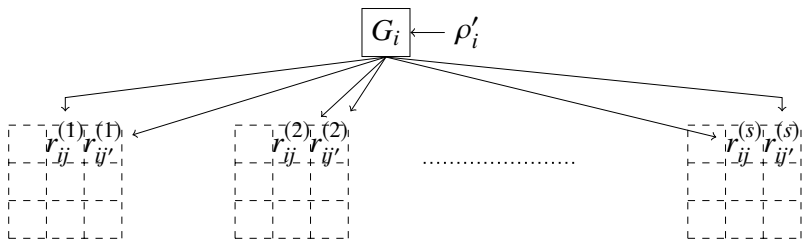
Our technique (3)



Randomness complexity

- There are $O(t^2)$ randoms r_{ij} , so we need $O(t^2)$ independent PRGs G_{ij}
- Each PRG requires $r = t$ true randoms in \mathbb{F}
 - with $|\mathbb{F}| = O(|C|)$
- Randomness complexity $O(t^3 \log |C|)$
 - instead of $O(t^{3+\varepsilon} \log^k |C|)$ in [IKL13]
 - without expander graphs

Improved randomness complexity



Randomness complexity

- Each PRG G_i now generates the full row of randoms $r_{ij}^{(k)}$
 - We need $O(t)$ independent PRGs instead of $O(t^2)$.
- Each PRG requires $r = t$ true randoms in \mathbb{F}
 - with $|\mathbb{F}| = O(t|C|)$
- Randomness complexity $O(t^2(\log t + \log |C|))$

Application to AES

- Concrete AES implementation of our construction
 - 44 MHz ARM-Cortex M3 processor
 - Slow TRNG: 1500 cycles per TRNG byte.
 - Source code publicly available
- Minimization of TRNG calls

	Rivain-Prouff		Our construction		
	TRNG	MCycles	TRNG	MCycles	ratio
$t = 2$	2880	20.6	48	14.1	0.68
$t = 3$	5760	40.2	108	34.7	0.86

- Minimization of running time

	Rivain-Prouff		Our construction		
	TRNG	MCycles	TRNG	MCycles	ratio
$t = 2$	2880	20.6	642	9.8	0.48
$t = 3$	5760	40.2	1056	15.5	0.39

Application to AES

- Concrete AES implementation of our construction
 - 44 MHz ARM-Cortex M3 processor
 - Slow TRNG: 1500 cycles per TRNG byte.
 - Source code publicly available
- Minimization of TRNG calls

	Rivain-Prouff		Our construction		
	TRNG	MCycles	TRNG	MCycles	ratio
$t = 2$	2880	20.6	48	14.1	0.68
$t = 3$	5760	40.2	108	34.7	0.86

- Minimization of running time

	Rivain-Prouff		Our construction		
	TRNG	MCycles	TRNG	MCycles	ratio
$t = 2$	2880	20.6	642	9.8	0.48
$t = 3$	5760	40.2	1056	15.5	0.39

Application to AES

- Concrete AES implementation of our construction
 - 44 MHz ARM-Cortex M3 processor
 - Slow TRNG: 1500 cycles per TRNG byte.
 - Source code publicly available
- Minimization of TRNG calls

	Rivain-Prouff		Our construction		
	TRNG	MCycles	TRNG	MCycles	ratio
$t = 2$	2880	20.6	48	14.1	0.68
$t = 3$	5760	40.2	108	34.7	0.86

- Minimization of running time

	Rivain-Prouff		Our construction		
	TRNG	MCycles	TRNG	MCycles	ratio
$t = 2$	2880	20.6	642	9.8	0.48
$t = 3$	5760	40.2	1056	15.5	0.39

Conclusion

Reducing randomness complexity of high-order masking

- $O(t^2 |C|)$ in original ISW
- From $O(t^{3+\varepsilon} \log^k |C|)$ in [IKL+13] with robust PRG
- to $O(t^2(\log t + \log |C|))$ with r -wise independent PRG
- Simple construction without expander graphs.

Application to AES

- Concrete implementation of AES with PRG.
- Only 48 bytes of TRNG instead of 2880 in [RP10] for $t = 2$.
- Implementation on a real-life processor : 50 % speedup.

Reducing randomness complexity of high-order masking

- $O(t^2 |C|)$ in original ISW
- From $O(t^{3+\varepsilon} \log^k |C|)$ in [IKL+13] with robust PRG
- to $O(t^2(\log t + \log |C|))$ with r -wise independent PRG
- Simple construction without expander graphs.

Application to AES

- Concrete implementation of AES with PRG.
- Only 48 bytes of TRNG instead of 2880 in [RP10] for $t = 2$.
- Implementation on a real-life processor : 50 % speedup.