Higher-Order Side Channel Security and Mask Refreshing

J.-S. Coron,E. Prouff, M. Rivain and T. Roche
thomas.roche@ssi.gouv.fr

FSE 2013 – March 2013

Side Channel Analysis

- Side Channel Attacks (SCA) appear 15 years ago
  - 1996 : Timing Attacks
  - 1998 : Power Analysis
  - 2000 : Electromagnetic Analysis
- Numerous attacks
  - 1998 : (single-bit) DPA KocherJaffeJune1999
  - 1999 : (multi-bit) DPA Messerges99
  - 2000 : Higher-order SCA Messerges2000
  - 2002 : Template SCA ChariRaoRohatgi2002
  - 2004 : CPA BrierClavierOlivier2004
  - 2005 : Stochastic SCA SchindlerLemkePaar2006
  - 2008 : Mutual Information SCA GierlichsBatinaTuyls2008
  - etc.

Side Channel Analysis

- Side Channel Attacks (SCA) appear 15 years ago
  - 1996 : Timing Attacks
  - 1998 : Power Analysis
  - 2000 : Electromagnetic Analysis
- Numerous attacks
  - 1998 : (single-bit) DPA KocherJaffeJune1999
  - 1999 : (multi-bit) DPA Messerges99
  - 2000 : Higher-order SCA Messerges2000
  - 2002 : Template SCA ChariRaoRohatgi2002
  - 2004 : CPA BrierClavierOlivier2004
  - 2005 : Stochastic SCA SchindlerLemkePaar2006
  - 2008 : Mutual Information SCA GierlichsBatinaTuyls2008
  - etc.

SCA Countermeasures

- Masking [IBM Team at CRYPTO 1999].
  - Efficient against SCA in practice.
  - Difficult to implement for non-linear transformations.
- Shuffling [Researchers from Graz University at ACNS 2006].
  - Less efficient against SCA in practice.
  - Easy to implement for every transformation.
- Whitening [Kocher Jaffe June, CRYPTO 1999].
  - Less efficient than masking when used alone and costly in Hardware.
  - Easy to implement for every transformation.

SCA Countermeasures

- Masking [IBM Team at CRYPTO 1999].
  - Efficient against SCA in practice.
  - Difficult to implement for non-linear transformations.
- Shuffling [Researchers from Graz University at ACNS 2006].
  - Less efficient against SCA in practice.
  - Easy to implement for every transformation.
- Whitening [Kocher Jaffe June, CRYPTO 1999].
  - Less efficient than masking when used alone and costly in Hardware.
  - Easy to implement for every transformation.

SCA Countermeasures

- Masking [IBM Team at CRYPTO 1999].
  - Efficient against SCA in practice.
  - Difficult to implement for non-linear transformations.
- Shuffling [Researchers from Graz University at ACNS 2006].
  - Less efficient against SCA in practice.
  - Easy to implement for every transformation.
- Whitening [Kocher Jaffe June, CRYPTO 1999].
  - Less efficient than masking when used alone and costly in Hardware.
  - Easy to implement for every transformation.

SCA Countermeasures

- Masking [IBM Team at CRYPTO 1999].
  - Efficient against SCA in practice.
  - Difficult to implement for non-linear transformations.
- Shuffling [Researchers from Graz University at ACNS 2006].
  - Less efficient against SCA in practice.
  - Easy to implement for every transformation.
- Whitening [Kocher Jaffe June, CRYPTO 1999].
  - Less efficient than masking when used alone and costly in Hardware.
  - Easy to implement for every transformation.

Masking/Sharing Coutermeasures

Idea : consists in securing the implementation using secret sharing techniques.

- First Ideas in GoubinPatarin99 and ChariJutlaRaoRohatgi99.
- Soundness based on the following remark :

[Chari-Jutla-Rao-Rohatgi CRYPTO'99]

- Bit x masked $\mapsto x_0, x_1, \ldots, x_d$
- Leakage : $L_i \sim x_i + \mathcal{N}(\mu, \sigma^2)$
- # of leakage samples to test $((L_i)_i | x = 0) = ((L_i)_i | x = 1)$:

$$q \geq O(1)\sigma^d$$

- Until now, security proofs are not unconditional and are "limited" to so-called probing adversaries.

Masking/Sharing Coutermeasures

Idea : consists in securing the implementation using secret sharing techniques.

- First Ideas in GoubinPatarin99 and ChariJutlaRaoRohatgi99.
- Soundness based on the following remark :

[Chari-Jutla-Rao-Rohatgi CRYPTO'99]

- Bit x masked $\mapsto x_0, x_1, \ldots, x_d$
- Leakage : $L_i \sim x_i + \mathcal{N}(\mu, \sigma^2)$
- # of leakage samples to test $\left((L_i)_i | x = 0\right) = \left((L_i)_i | x = 1\right)$:

$$q \geq O(1)\sigma^d$$

- Until now, security proofs are not unconditional and are "limited" to so-called probing adversaries.

Masking/Sharing Coutermeasures

Idea : consists in securing the implementation using secret sharing techniques.

- First Ideas in GoubinPatarin99 and ChariJutlaRaoRohatgi99.
- Soundness based on the following remark :

[Chari-Jutla-Rao-Rohatgi CRYPTO'99]

- Bit x masked $\mapsto x_0, x_1, \ldots, x_d$
- Leakage : $L_i \sim x_i + \mathcal{N}(\mu, \sigma^2)$
- # of leakage samples to test $\big((L_i)_i | x = 0\big) = \big((L_i)_i | x = 1\big)$:

$$q \geq O(1)\sigma^d$$

- Until now, security proofs are not unconditional and are "limited" to so-called probing adversaries.

Masking/Sharing Coutermeasures

Idea : consists in securing the implementation using secret sharing techniques.

- First Ideas in GoubinPatarin99 and ChariJutlaRaoRohatgi99.
- Soundness based on the following remark :

[Chari-Jutla-Rao-Rohatgi CRYPTO'99]

- Bit x masked $\mapsto x_0, x_1, \ldots, x_d$
- Leakage : $L_i \sim x_i + \mathcal{N}(\mu, \sigma^2)$
- # of leakage samples to test $\big((L_i)_i | x = 0\big) = \big((L_i)_i | x = 1\big)$:

$$q \geq O(1)\sigma^d$$

- Until now, security proofs are not unconditional and are "limited" to so-called probing adversaries.

Probing Adversary

- Notion introduced in IshaiSahaiWagner, CRYPTO 2003
- A d^{th}-order probing adversary is allowed to observe **at most** d intermediate results during the overall algorithm processing.
  - Hardware interpretation : d is the maximum of wires observed in the circuit.
  - Software interpretation : d is the maximum of different timings during the processing.
- d^{th}-order probing adversary $=$ d^{th}-order SCA as introduced in Messerges99.
- Countermeasures proved to be secure against a d^{th}-order probing adv. :
  - $d = 1$: KocherJaffeJune99, BlömerGuajardoKrummel04, ProuffRivain07.
  - $d = 2$: RivainDottaxProuff08.
  - $d \geq 1$: IshaiSahaiWagner03, ProuffRoche11, GenelleProuffQuisquater11, CarletGoubinProuffQuisquaterRivain12.

Probing Adversary

- Notion introduced in IshaiSahaiWagner, CRYPTO 2003
- A d^{th}-order probing adversary is allowed to observe **at most** d intermediate results during the overall algorithm processing.
  - Hardware interpretation : d is the maximum of wires observed in the circuit.
  - Software interpretation : d is the maximum of different timings during the processing.
- d^{th}-order probing adversary $=$ d^{th}-order SCA as introduced in Messerges99.
- Countermeasures proved to be secure against a d^{th}-order probing adv. :
  - $d = 1$: KocherJaffeJune99, BlömerGuajardoKrummel04, ProuffRivain07.
  - $d = 2$: RivainDottaxProuff08.
  - $d \geq 1$: IshaiSahaiWagner03, ProuffRoche11, GenelleProuffQuisquater11, CarletGoubinProuffQuisquaterRivain12.

Higher-Order Masking Schemes

Achieving security in the probing adversary model

Definition

A *dth-order masking scheme* for an encryption algorithm $c \leftarrow \mathcal{E}(m,k)$ is an algorithm

$$(c_0, c_1, \ldots, c_d) \leftarrow \mathcal{E}'\big((m_0, m_1, \ldots, m_d), (k_0, k_1, \ldots, k_d)\big)$$

- Completeness : there exists R s.t. :

$$R(c_0, \cdots, c_d) = \mathcal{E}(m,k)$$

- Security : $\forall \{iv_1, iv_2, \ldots, iv_d\} \subseteq \{\text{intermediate var. of } \mathcal{E}'\}$:

$$\Pr\big(k \mid iv_1, iv_2, \ldots, iv_d\big) = \Pr\big(k\big)$$

For SPN (*eg.* DES, AES) the main issue is masking the S-box.

Higher-Order Masking Schemes

Achieving security in the probing adversary model

Definition

A *dth-order masking scheme* for an encryption algorithm $c \leftarrow \mathcal{E}(m, k)$ is an algorithm

$$(c_0, c_1, \ldots, c_d) \leftarrow \mathcal{E}'\big((m_0, m_1, \ldots, m_d), (k_0, k_1, \ldots, k_d)\big)$$

- Completeness : there exists R s.t. :

$$R(c_0, \cdots, c_d) = \mathcal{E}(m, k)$$

- Security : $\forall \{iv_1, iv_2, \ldots, iv_d\} \subseteq \{\text{intermediate var. of } \mathcal{E}'\}$:

$$\Pr\big(k \mid iv_1, iv_2, \ldots, iv_d\big) = \Pr\big(k\big)$$

For SPN (*eg.* DES, AES) the main issue is masking the S-box.

Masking a S-box

Original work of Ishai, Sahai and Wagner

Main idea : split the S-box computation into elementary operations and protect each of them individually.

- Original idea limited to GF(2) IshaiSahaiWagner2003
- Extended to any field in RivainProuff2010 and FaustRabinReyzinTromerVaikuntanathan2011.
- Data are split by bitwise addition : $x \longrightarrow x_0, \cdots, x_d$ s.t. $x_i \leftarrow \$$, $i > 0$, and $x_0 = \bigoplus_i x_i$.
- Masking of Linear Transformations L is easy :

$$L(x) \rightarrow \underbrace{L(x_0), L(x_1), \cdots, L(x_d)}_{L(x_0) \oplus L(x_1) \oplus \cdots \oplus L(x_d) = L(x)}$$

- Masking of non-linear transformations is an issue since the operations cannot be done on each shares separately.
  - $\rightarrow$ Problem reduces to secure multiplications !

Masking a S-box

Original work of Ishai, Sahai and Wagner

Main idea : split the S-box computation into elementary operations and protect each of them individually.

- Original idea limited to GF(2) IshaiSahaiWagner2003
- Extended to any field in RivainProuff2010 and FaustRabinReyzinTromerVaikuntanathan2011.
- Data are split by bitwise addition : $x \longrightarrow x_0, \cdots, x_d$ s.t. $x_i \leftarrow \$$, $i > 0$, and $x_0 = \bigoplus_i x_i$.
- Masking of Linear Transformations L is easy :

$$L(x) \rightarrow \underbrace{L(x_0), L(x_1), \cdots, L(x_d)}_{L(x_0) \oplus L(x_1) \oplus \cdots \oplus L(x_d) = L(x)}$$

- Masking of non-linear transformations is an issue since the operations cannot be done on each shares separately.
  - $\rightarrow$ Problem reduces to secure multiplications !

Masking a S-box

Original work of Ishai, Sahai and Wagner

Main idea : split the S-box computation into elementary operations and protect each of them individually.

- Original idea limited to GF(2) IshaiSahaiWagner2003
- Extended to any field in RivainProuff2010 and FaustRabinReyzinTromerVaikuntanathan2011.
- Data are split by bitwise addition : $x \longrightarrow x_0, \cdots, x_d$ s.t. $x_i \leftarrow \$$, $i > 0$, and $x_0 = \bigoplus_i x_i$.
- Masking of Linear Transformations L is easy :

$$L(x) \rightarrow \underbrace{L(x_0), L(x_1), \cdots, L(x_d)}_{L(x_0) \oplus L(x_1) \oplus \cdots \oplus L(x_d) = L(x)}$$

- Masking of non-linear transformations is an issue since the operations cannot be done on each shares separately.
  - $\rightarrow$ Problem reduces to secure multiplications !

Masking a S-box

Original work of Ishai, Sahai and Wagner

Main idea : split the S-box computation into elementary operations and protect each of them individually.

- Original idea limited to GF(2) IshaiSahaiWagner2003
- Extended to any field in RivainProuff2010 and FaustRabinReyzinTromerVaikuntanathan2011.
- Data are split by bitwise addition : $x \longrightarrow x_0, \cdots, x_d$ s.t. $x_i \leftarrow \$$, $i > 0$, and $x_0 = \bigoplus_i x_i$.
- Masking of Linear Transformations L is easy :

$$L(x) \rightarrow \underbrace{L(x_0), L(x_1), \cdots, L(x_d)}_{L(x_0)\oplus L(x_1)\oplus\cdots\oplus L(x_d)=L(x)}$$

- Masking of non-linear transformations is an issue since the operations cannot be done on each shares separately.
  - $\rightarrow$ Problem reduces to secure multiplications !

Masking Multiplications ×

Ishai-Sahai-Wagner Scheme (ISW)

- Outlines of the scheme :
  - Input : $(a_i)_i$, $(b_i)_i$ s.t. $\bigoplus_i a_i = a$, $\bigoplus_i b_i = b$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = a \times b$

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

- Example ($d = 2$) :

- Ishai *et al.* prove $(d/2)$th-order security
  - Extended to get dth-order security in RivainProuff10

Masking Multiplications $\times$

Ishai-Sahai-Wagner Scheme (ISW)

- Outlines of the scheme :
  - Input : $(a_i)_i$, $(b_i)_i$ s.t. $\bigoplus_i a_i = a$, $\bigoplus_i b_i = b$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = a \times b$

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

- Example ($d = 2$) :

- Ishai *et al.* prove $(d/2)$th-order security
  - Extended to get dth-order security in RivainProuff10

Masking Multiplications ×

Ishai-Sahai-Wagner Scheme (ISW)

- Outlines of the scheme :
  - Input : $(a_i)_i$, $(b_i)_i$ s.t. $\bigoplus_i a_i = a$, $\bigoplus_i b_i = b$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = a \times b$

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

- Example ($d = 2$) :

$$\begin{pmatrix} a_0 b_0 & a_0 b_1 & a_0 b_2 \\ a_1 b_0 & a_1 b_1 & a_1 b_2 \\ a_2 b_0 & a_2 b_1 & a_2 b_2 \end{pmatrix}$$

- Ishai *et al.* prove $(d/2)$th-order security
  - Extended to get dth-order security in RivainProuff10

Masking Multiplications $\times$

Ishai-Sahai-Wagner Scheme (ISW)

- Outlines of the scheme :
  - Input : $(a_i)_i$, $(b_i)_i$ s.t. $\bigoplus_i a_i = a$, $\bigoplus_i b_i = b$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = a \times b$

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

- Example ($d = 2$) :

$$\begin{pmatrix} a_0 b_0 & a_0 b_1 & a_0 b_2 \\ 0 & a_1 b_1 & a_1 b_2 \\ 0 & 0 & a_2 b_2 \end{pmatrix} \oplus \begin{pmatrix} 0 & 0 & 0 \\ a_1 b_0 & 0 & 0 \\ a_2 b_0 & a_2 b_1 & 0 \end{pmatrix}$$

- Ishai *et al.* prove $(d/2)$th-order security
  - Extended to get dth-order security in RivainProuff10

Masking Multiplications $\times$

Ishai-Sahai-Wagner Scheme (ISW)

- Outlines of the scheme :
  - Input : $(a_i)_i$, $(b_i)_i$ s.t. $\bigoplus_i a_i = a$, $\bigoplus_i b_i = b$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = a \times b$

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

- Example ($d = 2$) :

$$\begin{pmatrix} a_0 b_0 & a_0 b_1 & a_0 b_2 \\ 0 & a_1 b_1 & a_1 b_2 \\ 0 & 0 & a_2 b_2 \end{pmatrix} \oplus \begin{pmatrix} 0 & a_1 b_0 & a_2 b_0 \\ 0 & 0 & a_2 b_1 \\ 0 & 0 & 0 \end{pmatrix}$$

- Ishai *et al.* prove $(d/2)$th-order security
  - Extended to get dth-order security in RivainProuff10

Masking Multiplications $\times$

Ishai-Sahai-Wagner Scheme (ISW)

- Outlines of the scheme :
  - Input : $(a_i)_i$, $(b_i)_i$ s.t. $\bigoplus_i a_i = a$, $\bigoplus_i b_i = b$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = a \times b$

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

- Example ($d = 2$) :

$$\begin{pmatrix} a_0 b_0 & a_0 b_1 \oplus a_1 b_0 & a_0 b_2 \oplus a_2 b_0 \\ 0 & a_1 b_1 & a_1 b_2 \oplus a_2 b_1 \\ 0 & 0 & a_2 b_2 \end{pmatrix}$$

- Ishai *et al.* prove $(d/2)$th-order security
  - Extended to get dth-order security in RivainProuff10

Masking Multiplications $\times$

Ishai-Sahai-Wagner Scheme (ISW)

- Outlines of the scheme :
  - Input : $(a_i)_i$, $(b_i)_i$ s.t. $\bigoplus_i a_i = a$, $\bigoplus_i b_i = b$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = a \times b$

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

- Example ($d = 2$) :

$$\begin{pmatrix} a_0 b_0 & a_0 b_1 \oplus a_1 b_0 & a_0 b_2 \oplus a_2 b_0 \\ 0 & a_1 b_1 & a_1 b_2 \oplus a_2 b_1 \\ 0 & 0 & a_2 b_2 \end{pmatrix}$$

- Ishai *et al.* prove $(d/2)$th-order security
  - Extended to get dth-order security in RivainProuff10

Masking Multiplications $\times$

Ishai-Sahai-Wagner Scheme (ISW)

- Outlines of the scheme :
  - Input : $(a_i)_i$, $(b_i)_i$ s.t. $\bigoplus_i a_i = a$, $\bigoplus_i b_i = b$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = a \times b$

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

- Example ($d = 2$) :

$$\begin{pmatrix} a_0 b_0 & a_0 b_1 \oplus a_1 b_0 & a_0 b_2 \oplus a_2 b_0 \\ 0 & a_1 b_1 & a_1 b_2 \oplus a_2 b_1 \\ 0 & 0 & a_2 b_2 \end{pmatrix} \oplus \begin{pmatrix} 0 & r_{0,1} & r_{0,2} \\ 0 & 0 & r_{1,2} \\ 0 & 0 & 0 \end{pmatrix}$$

- Ishai *et al.* prove $(d/2)$th-order security
  - Extended to get dth-order security in RivainProuff10

Masking Multiplications ×

Ishai-Sahai-Wagner Scheme (ISW)

- Outlines of the scheme :
  - Input : $(a_i)_i$, $(b_i)_i$ s.t. $\bigoplus_i a_i = a$, $\bigoplus_i b_i = b$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = a \times b$

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

- Example $(d = 2)$:

$$\begin{pmatrix} a_0 b_0 & a_0 b_1 \oplus a_1 b_0 & a_0 b_2 \oplus a_2 b_0 \\ 0 & a_1 b_1 & a_1 b_2 \oplus a_2 b_1 \\ 0 & 0 & a_2 b_2 \end{pmatrix} \oplus \begin{pmatrix} 0 & r_{0,1} & r_{0,2} \\ r_{0,1} & 0 & r_{1,2} \\ r_{0,2} & r_{1,2} & 0 \end{pmatrix}$$

- Ishai *et al.* prove $(d/2)$th-order security
  - Extended to get dth-order security in RivainProuff10

Masking Multiplications $\times$

Ishai-Sahai-Wagner Scheme (ISW)

- Outlines of the scheme :
  - Input : $(a_i)_i$, $(b_i)_i$ s.t. $\bigoplus_i a_i = a$, $\bigoplus_i b_i = b$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = a \times b$

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

- Example ($d = 2$) :

$$\begin{pmatrix} a_0 b_0 & (a_0 b_1 \oplus r_{0,1}) \oplus a_1 b_0 & (a_0 b_2 \oplus r_{0,2}) \oplus a_2 b_0 \\ r_{0,1} & a_1 b_1 & (a_1 b_2 \oplus r_{1,2}) \oplus a_2 b_1 \\ r_{0,2} & r_{1,2} & a_2 b_2 \end{pmatrix}$$

- Ishai *et al.* prove $(d/2)$th-order security
  - Extended to get dth-order security in RivainProuff10

Masking Multiplications ×

Ishai-Sahai-Wagner Scheme (ISW)

- Outlines of the scheme :
  - Input : $(a_i)_i$, $(b_i)_i$ s.t. $\bigoplus_i a_i = a$, $\bigoplus_i b_i = b$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = a \times b$

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

- Example ($d = 2$) :

$$\begin{pmatrix} a_0 b_0 & (a_0 b_1 \oplus r_{0,1}) \oplus a_1 b_0 & (a_0 b_2 \oplus r_{0,2}) \oplus a_2 b_0 \\ r_{0,1} & a_1 b_1 & (a_1 b_2 \oplus r_{1,2}) \oplus a_2 b_1 \\ r_{0,2} & r_{1,2} & a_2 b_2 \end{pmatrix}$$

- Ishai *et al.* prove $(d/2)$th-order security
  - Extended to get dth-order security in RivainProuff10

Masking Multiplications $\times$

Ishai-Sahai-Wagner Scheme (ISW)

- Outlines of the scheme :
  - Input : $(a_i)_i$, $(b_i)_i$ s.t. $\bigoplus_i a_i = a$, $\bigoplus_i b_i = b$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = a \times b$

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

- Example ($d = 2$) :

$$\begin{pmatrix} a_0 b_0 & (a_0 b_1 \oplus r_{0,1}) \oplus a_1 b_0 & (a_0 b_2 \oplus r_{0,2}) \oplus a_2 b_0 \\ r_{0,1} & a_1 b_1 & (a_1 b_2 \oplus r_{1,2}) \oplus a_2 b_1 \\ r_{0,2} & r_{1,2} & a_2 b_2 \end{pmatrix}$$

$$c_1$$

- Ishai *et al.* prove $(d/2)$th-order security
  - Extended to get dth-order security in RivainProuff10

Masking Multiplications $\times$

Ishai-Sahai-Wagner Scheme (ISW)

- Outlines of the scheme :
  - Input : $(a_i)_i$, $(b_i)_i$ s.t. $\bigoplus_i a_i = a$, $\bigoplus_i b_i = b$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = a \times b$

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

- Example $(d = 2)$:

$$\begin{array}{ccc} \left(\begin{array}{ccc} a_0 b_0 & (a_0 b_1 \oplus r_{0,1}) \oplus a_1 b_0 & (a_0 b_2 \oplus r_{0,2}) \oplus a_2 b_0 \\ r_{0,1} & a_1 b_1 & (a_1 b_2 \oplus r_{1,2}) \oplus a_2 b_1 \\ r_{0,2} & r_{1,2} & a_2 b_2 \end{array}\right) \\ \begin{array}{ccc} c_1 & \quad c_2 & \quad \end{array} \end{array}$$

- Ishai *et al.* prove $(d/2)$th-order security
  - Extended to get dth-order security in RivainProuff10

Masking Multiplications $\times$

Ishai-Sahai-Wagner Scheme (ISW)

- Outlines of the scheme :
  - Input : $(a_i)_i$, $(b_i)_i$ s.t. $\bigoplus_i a_i = a$, $\bigoplus_i b_i = b$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = a \times b$

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

- Example ($d = 2$) :

$$\begin{pmatrix} a_0 b_0 & (a_0 b_1 \oplus r_{0,1}) \oplus a_1 b_0 & (a_0 b_2 \oplus r_{0,2}) \oplus a_2 b_0 \\ r_{0,1} & a_1 b_1 & (a_1 b_2 \oplus r_{1,2}) \oplus a_2 b_1 \\ r_{0,2} & r_{1,2} & a_2 b_2 \end{pmatrix}$$
$$c_1 \qquad c_2 \qquad c_3$$

- Ishai *et al.* prove $(d/2)$th-order security
  - Extended to get dth-order security in RivainProuff10

Masking Multiplications $\times$

Ishai-Sahai-Wagner Scheme (ISW)

- Outlines of the scheme :
  - Input : $(a_i)_i$, $(b_i)_i$ s.t. $\bigoplus_i a_i = a$, $\bigoplus_i b_i = b$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = a \times b$

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

- Example ($d = 2$) :

$$\begin{pmatrix} a_0 b_0 & (a_0 b_1 \oplus r_{0,1}) \oplus a_1 b_0 & (a_0 b_2 \oplus r_{0,2}) \oplus a_2 b_0 \\ r_{0,1} & a_1 b_1 & (a_1 b_2 \oplus r_{1,2}) \oplus a_2 b_1 \\ r_{0,2} & r_{1,2} & a_2 b_2 \end{pmatrix}$$
$$\quad c_1 \qquad\qquad c_2 \qquad\qquad c_3$$

- Ishai *et al.* prove $(d/2)$th-order security
  - Extended to get dth-order security in RivainProuff10

Application to Secure Power Functions

... with a focus on the *AES power function* $x \mapsto x^{254}$

Let $\mathrm{Exp} : x \mapsto x^r$ be a power function defined over a finite field $\mathrm{GF}(2^n)$.

- Split Exp into a sequence of multiplications and squarings.
- Squaring is a GF(2)-linear operation $\longrightarrow$ easy to mask :
  - masked square : $x^2 \longrightarrow x_0^2, x_1^2, \cdots, x_d^2$
- Multiplications masked with ISW Scheme
- To reduce the overall cost of the securing, favour squaring over multiplication in the Exp evaluation method :
  - amount to look at small addition chains for r
- For AES non-linear function ($r = 254$), Rivain and Prouff proves that the evaluation can be done with 4 multiplications only (optimal).

Application to Secure Power Functions

... with a focus on the *AES power function* $x \mapsto x^{254}$

Let $\mathrm{Exp} : x \mapsto x^r$ be a power function defined over a finite field $\mathrm{GF}(2^n)$.

- Split Exp into a sequence of multiplications and squarings.
- Squaring is a GF(2)-linear operation $\longrightarrow$ easy to mask :
  - masked square : $x^2 \longrightarrow x_0^2, x_1^2, \cdots, x_d^2$
- Multiplications masked with ISW Scheme
- To reduce the overall cost of the securing, favour squaring over multiplication in the Exp evaluation method :
  - amount to look at small addition chains for r
- For AES non-linear function ($r = 254$), Rivain and Prouff proves that the evaluation can be done with 4 multiplications only (optimal).

Application to Secure Power Functions

... with a focus on the *AES power function* $x \mapsto x^{254}$

Let $\mathrm{Exp} : x \mapsto x^r$ be a power function defined over a finite field $\mathrm{GF}(2^n)$.

- Split Exp into a sequence of multiplications and squarings.
- Squaring is a GF(2)-linear operation $\longrightarrow$ easy to mask :
  - masked square : $x^2 \longrightarrow x_0^2, x_1^2, \cdots, x_d^2$
- Multiplications masked with ISW Scheme
- To reduce the overall cost of the securing, favour squaring over multiplication in the Exp evaluation method :
  - amount to look at small addition chains for r
- For AES non-linear function ($r = 254$), Rivain and Prouff proves that the evaluation can be done with 4 multiplications only (optimal).

Application to Secure Power Functions

... with a focus on the *AES power function* $x \mapsto x^{254}$

Let $\mathrm{Exp} : x \mapsto x^r$ be a power function defined over a finite field $\mathrm{GF}(2^n)$.

- Split Exp into a sequence of multiplications and squarings.
- Squaring is a GF(2)-linear operation $\longrightarrow$ easy to mask :
  - masked square : $x^2 \longrightarrow x_0^2, x_1^2, \cdots, x_d^2$
- Multiplications masked with ISW Scheme
- To reduce the overall cost of the securing, favour squaring over multiplication in the Exp evaluation method :
  - amount to look at small addition chains for r
- For AES non-linear function ($r = 254$), Rivain and Prouff proves that the evaluation can be done with 4 multiplications only (optimal).

Masking the S-box

RivainProuff10

Algorithmic description :

Input : shares $\mathbf{x}_i$ s.t. $\bigoplus_i \mathbf{x}_i = \mathbf{x}$

Output : shares y_i s.t. $\bigoplus_i y_i = \mathbf{x}^{254}$

1. $(z_i)_i \leftarrow (x_i^2)_i$ $[\bigoplus_i z_i = x^2]$

2. $\mathrm{RefreshMasks}\big((z_i)_i\big)$

3. $(y_i)_i \leftarrow \mathtt{ISW}\big((z_i)_i, (x_i)_i\big)$ $[\bigoplus_i y_i = x^3]$

4. $(w_i)_i \leftarrow (y_i^4)_i$ $[\bigoplus_i w_i = x^{12}]$

5. $\mathrm{RefreshMasks}\big((w_i)_i\big)$

6. $(y_i)_i \leftarrow \mathtt{ISW}\big((y_i)_i, (w_i)_i\big)$ $[\bigoplus_i y_i = x^{15}]$

7. $(y_i)_i \leftarrow (y_i^{16})_i$ $[\bigoplus_i y_i = x^{240}]$

8. $(y_i)_i \leftarrow \mathtt{ISW}\big((y_i)_i, (w_i)_i\big)$ $[\bigoplus_i y_i = x^{252}]$

9. $(y_i)_i \leftarrow \mathtt{ISW}\big((y_i)_i, (z_i)_i\big)$ $[\bigoplus_i y_i = x^{254}]$

Masking the S-box

RivainProuff10

Algorithmic description :

Input : shares $\mathbf{x}_i$ s.t. $\bigoplus_i \mathbf{x}_i = \mathbf{x}$
Output : shares y_i s.t. $\bigoplus_i y_i = \mathbf{x}^{254}$

1. $(z_i)_i \leftarrow (x_i^2)_i$ $[\bigoplus_i z_i = x^2]$
2. $\mathrm{RefreshMasks}\big((z_i)_i\big)$
3. $(y_i)_i \leftarrow \mathtt{ISW}\big((z_i)_i, (x_i)_i\big)$ $[\bigoplus_i y_i = x^3]$
4. $(w_i)_i \leftarrow (y_i^4)_i$ $[\bigoplus_i w_i = x^{12}]$
5. $\mathrm{RefreshMasks}\big((w_i)_i\big)$
6. $(y_i)_i \leftarrow \mathtt{ISW}\big((y_i)_i, (w_i)_i\big)$ $[\bigoplus_i y_i = x^{15}]$
7. $(y_i)_i \leftarrow (y_i^{16})_i$ $[\bigoplus_i y_i = x^{240}]$
8. $(y_i)_i \leftarrow \mathtt{ISW}\big((y_i)_i, (w_i)_i\big)$ $[\bigoplus_i y_i = x^{252}]$
9. $(y_i)_i \leftarrow \mathtt{ISW}\big((y_i)_i, (z_i)_i\big)$ $[\bigoplus_i y_i = x^{254}]$

Masking the S-box

RivainProuff10

Algorithmic description :

Input : shares $\mathbf{x}_i$ s.t. $\bigoplus_i \mathbf{x}_i = \mathbf{x}$
Output : shares y_i s.t. $\bigoplus_i y_i = \mathbf{x}^{254}$
1. $(z_i)_i \leftarrow (x_i^2)_i$ $[\bigoplus_i z_i = x^2]$
2. $\text{RefreshMasks}\big((z_i)_i\big)$
3. $(y_i)_i \leftarrow \texttt{ISW}\big((z_i)_i, (x_i)_i\big)$ $[\bigoplus_i y_i = x^3]$
4. $(w_i)_i \leftarrow (y_i^4)_i$ $[\bigoplus_i w_i = x^{12}]$
5. $\text{RefreshMasks}\big((w_i)_i\big)$
6. $(y_i)_i \leftarrow \texttt{ISW}\big((y_i)_i, (w_i)_i\big)$ $[\bigoplus_i y_i = x^{15}]$
7. $(y_i)_i \leftarrow (y_i^{16})_i$ $[\bigoplus_i y_i = x^{240}]$
8. $(y_i)_i \leftarrow \texttt{ISW}\big((y_i)_i, (w_i)_i\big)$ $[\bigoplus_i y_i = x^{252}]$
9. $(y_i)_i \leftarrow \texttt{ISW}\big((y_i)_i, (z_i)_i\big)$ $[\bigoplus_i y_i = x^{254}]$

Masking the S-box

RivainProuff10

Algorithmic description :

Input : shares $\mathbf{x}_i$ s.t. $\bigoplus_i \mathbf{x}_i = \mathbf{x}$
Output : shares y_i s.t. $\bigoplus_i y_i = \mathbf{x}^{254}$

1. $(z_i)_i \leftarrow (x_i^2)_i$ $\quad [\bigoplus_i z_i = x^2]$
2. RefreshMasks$\big((z_i)_i\big)$
3. $(y_i)_i \leftarrow \texttt{ISW}\big((z_i)_i, (x_i)_i\big)$ $\quad [\bigoplus_i y_i = x^3]$
4. $(w_i)_i \leftarrow (y_i^4)_i$ $\quad [\bigoplus_i w_i = x^{12}]$
5. RefreshMasks$\big((w_i)_i\big)$
6. $(y_i)_i \leftarrow \texttt{ISW}\big((y_i)_i, (w_i)_i\big)$ $\quad [\bigoplus_i y_i = x^{15}]$
7. $(y_i)_i \leftarrow (y_i^{16})_i$ $\quad [\bigoplus_i y_i = x^{240}]$
8. $(y_i)_i \leftarrow \texttt{ISW}\big((y_i)_i, (w_i)_i\big)$ $\quad [\bigoplus_i y_i = x^{252}]$
9. $(y_i)_i \leftarrow \texttt{ISW}\big((y_i)_i, (z_i)_i\big)$ $\quad [\bigoplus_i y_i = x^{254}]$

Security

- Security proved against a d^{th}-order probing adversary
- RefreshMasks assumed to be out of the scope of the proof.
- A simple (and assumed to be secure) algorithm is proposed to refresh the masks :

Input : shares z_i s.t. $\bigoplus_i z_i = z$
Output : new shares z'_i s.t. $\bigoplus_i z'_i = z$
1. **for** $i = 1$ **to** d **do**
2. $tmp \leftarrow \mathtt{rand}(n)$
3. $z_0 \leftarrow z_0 \oplus tmp$
4. $z'_i \leftarrow z_i \oplus tmp$

The Flaw

Let us focus on the three first steps of Rivain-Prouff's scheme.

1. $(z_i)_i \leftarrow (x_i^2)_i$
2. $(z'_i)_i \leftarrow \text{RefreshMasks}\big((z_i)_i\big)$
3. $(y_i)_i \leftarrow \texttt{ISW}\big((z'_i)_i, (x_i)_i\big)$

- By construction, at the $d/2^{\text{th}}$iteration of RefreshMasks :

- By definition, ISW involves the following processings (cross-products) :

$$z'_i \times x_{i+d/2}$$

for all $\in [1; d/2]$

The Flaw

Let us focus on the three first steps of Rivain-Prouff's scheme.

1. $(z_i)_i \leftarrow (x_i^2)_i$
2. $(z'_i)_i \leftarrow \text{RefreshMasks}\big((z_i)_i\big)$
3. $(y_i)_i \leftarrow \texttt{ISW}\big((z'_i)_i, (x_i)_i\big)$

- By construction, at the $d/2^{\text{th}}$iteration of RefreshMasks :

$$z_0 = z \oplus \bigoplus_{1 \leq i \leq d/2} z'_i \oplus \bigoplus_{d/2+1 \leq i \leq d} z_i$$

- By definition, ISW involves the following processings (cross-products) :

$$z'_i \times x_{i+d/2}$$

for all $\in [1; d/2]$

The Flaw

Let us focus on the three first steps of Rivain-Prouff's scheme.

1. $(z_i)_i \leftarrow (x_i^2)_i$
2. $(z'_i)_i \leftarrow \text{RefreshMasks}\big((z_i)_i\big)$
3. $(y_i)_i \leftarrow \texttt{ISW}\big((z'_i)_i, (x_i)_i\big)$

- By construction, at the $d/2^{\text{th}}$iteration of RefreshMasks :

$$z_0 = z \oplus \bigoplus_{1 \leq i \leq d/2} z'_i \oplus \bigoplus_{d/2+1 \leq i \leq d} {x_i}^2$$

- By definition, ISW involves the following processings (cross-products) :

$$z'_i \times x_{i+d/2}$$

for all $\in [1; d/2]$

The Flaw

Let us focus on the three first steps of Rivain-Prouff's scheme.

1. $(z_i)_i \leftarrow (x_i^2)_i$
2. $(z'_i)_i \leftarrow \text{RefreshMasks}\big((z_i)_i\big)$
3. $(y_i)_i \leftarrow \texttt{ISW}\big((z'_i)_i, (x_i)_i\big)$

- By construction, at the $d/2^{\text{th}}$iteration of RefreshMasks :

$$z_0 = z \oplus \bigoplus_{1 \le i \le d/2} z'_i \oplus \bigoplus_{d/2+1 \le i \le d} {x_i}^2$$

- By definition, ISW involves the following processings (cross-products) :

$$z'_i \times x_{i+d/2}$$

for all $\in [1; d/2]$

The Flaw

$$z_0 = z \oplus \bigoplus_{1 \le i \le d/2} z'_i \oplus \bigoplus_{d/2+1 \le i \le d} {x_i}^2 \quad \rightarrow \ell_0$$

$$z'_i \times x_{i+d/2} \quad \forall i \in [1; d/2] \quad \rightarrow \ell_i$$

- The $d/2$ leakage values ℓ_i bring information on all the shares z'_i and $x_{i+d/2}$ for $i \le d/2$.
- This information is combined with ℓ_0 to retrieve information on (a.k.a. unmask) z.
  - Indeed $\Pr[z \mid (\ell_i)_i, \ell_o] \neq \Pr[z]$.

First (natural) Countermeasure

- Replace the RefreshMasks call by a call to ISW s.t. :
  - the first input is the sharing (of x) to refresh and
  - the second input is a sharing of 1.
- By definition, ISW will indeed outputs a new sharing of $x \times 1$.
- We get :

  1. $(z_i)_i \leftarrow (x_i^2)_i$
  2. $(z_i)_i \leftarrow \texttt{ISW}\big((z_i)_i, (1_i)_i\big)$ $(1_i)_i$ sharing of 1
  3. $(y_i)_i \leftarrow \texttt{ISW}\big((z_i)_i, (x_i)_i\big)$
  4. $(w_i)_i \leftarrow (y_i^4)_i$
  5. $(w_i)_i \leftarrow \texttt{ISW}\big((w_i)_i, ((1'_i)_i\big)$ $(1'_i)_i$ sharing of 1
  6. $(y_i)_i \leftarrow \texttt{ISW}\big((y_i)_i, (w_i)_i\big)$
  7. $(y_i)_i \leftarrow (y_i^{16})_i$
  8. $(y_i)_i \leftarrow \texttt{ISW}\big((y_i)_i, (w_i)_i\big)$
  9. $(y_i)_i \leftarrow \texttt{ISW}\big((y_i)_i, (z_i)_i\big)$

- Problem : security difficult to prove !

Second Countermeasure Proposal

- Principle : Replace every processing of $h(x) = x \cdot x^{2^j}$ s.t.
  1. $(z_i)_i \leftarrow (x_i^{2^j})_i$ — $(z_i)_i$ sharing of x^{2^j}
  2. $\text{Refreshmasks}((z_i)_i)$
  3. $(y_i)_i \leftarrow \texttt{ISW}((z_i)_i, (x_i)_i)$ — $(y_i)_i$ sharing of $x \cdot x^{2^j}$

  by a single processing of a new algorithm ISW'
- Core idea :

$$
\begin{aligned}
y &= \bigoplus_i a_i \cdot \bigoplus_i a_i^{2^j} \\
&= \bigoplus_i a_i^{2^j+1} \oplus \bigoplus_{i<k} \left(a_i \cdot a_k^{2^j} \oplus a_k \cdot a_i^{2^j} \right) \\
&= \bigoplus_i h(a_i) \oplus \bigoplus_{i<k} f(a_i, a_k)
\end{aligned}
$$

  involve the new function $f(x, y) = x \cdot y^{2^j} \oplus x^{2^j} \cdot y$
  - f is bilinear, thus we have

$$(\textit{Property} *) \qquad f(x, y) = h(x \oplus y) \oplus h(x) \oplus h(y)$$

Second Countermeasure Proposal

- Principle : Replace every processing of $h(x) = x \cdot x^{2^j}$ s.t.
  1. $(z_i)_i \leftarrow (x_i^{2^j})_i$ — $(z_i)_i$ sharing of x^{2^j}
  2. Refreshmasks$((z_i)_i)$
  3. $(y_i)_i \leftarrow \texttt{ISW}((z_i)_i, (x_i)_i)$ — $(y_i)_i$ sharing of $x \cdot x^{2^j}$

  by a single processing of a new algorithm ISW'
- Core idea :

$$\begin{aligned} y &= \bigoplus_i a_i \cdot \bigoplus_i a_i^{2^j} \\ &= \bigoplus_i a_i^{2^j+1} \oplus \bigoplus_{i<k} \left(a_i \cdot a_k^{2^j} \oplus a_k \cdot a_i^{2^j}\right) \\ &= \bigoplus_i h(a_i) \oplus \bigoplus_{i<k} f(a_i, a_k) \end{aligned}$$

  involve the new function $f(x, y) = x \cdot y^{2^j} \oplus x^{2^j} \cdot y$
  - f is bilinear, thus we have

$$(Property\ *) \qquad f(x, y) = h(x \oplus y) \oplus h(x) \oplus h(y)$$

Second Countermeasure Proposal

- Principle : Replace every processing of $h(x) = x \cdot x^{2^j}$ s.t.
  1. $(z_i)_i \leftarrow (x_i^{2^j})_i$ — $(z_i)_i$ sharing of x^{2^j}
  2. Refreshmasks$((z_i)_i)$
  3. $(y_i)_i \leftarrow \texttt{ISW}((z_i)_i, (x_i)_i)$ — $(y_i)_i$ sharing of $x \cdot x^{2^j}$

  by a single processing of a new algorithm ISW'
- Core idea :

$$
\begin{aligned}
y &= \bigoplus_i a_i \cdot \bigoplus_i a_i^{2^j} \\
&= \bigoplus_i a_i^{2^j+1} \oplus \bigoplus_{i<k} \left(a_i \cdot a_k^{2^j} \oplus a_k \cdot a_i^{2^j} \right) \\
&= \bigoplus_i h(a_i) \oplus \bigoplus_{i<k} f(a_i, a_k)
\end{aligned}
$$

  involve the new function $f(x, y) = x \cdot y^{2^j} \oplus x^{2^j} \cdot y$
  - f is bilinear, thus we have

(*Property* $*$) $\quad f(x, y) = h(x \oplus y) \oplus h(x) \oplus h(y)$

Masking of Power functions $x \mapsto x^{2^j+1}$

Outlines of the new scheme ISW'

- I/O :
  - Input : $(a_i)_i$ s.t. $\bigoplus_i a_i = a$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = h(a) = a \times a^{2^j}$
- Example ($d = 2$) :
- Security against d^{th}order probing adversary is given in the paper.

Masking of Power functions $x \mapsto x^{2^j+1}$

Outlines of the new scheme ISW'

- I/O :
  - Input : $(a_i)_i$ s.t. $\bigoplus_i a_i = a$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = h(a) = a \times a^{2^j}$
- Example ($d = 2$) :

$$\begin{pmatrix} a_0^{2^j+1} & a_0 \cdot a_1^{2^j} & a_0 \cdot a_2^{2^j} \\ a_1 \cdot a_0^{2^j} & a_1^{2^j+1} & a_1 \cdot a_2^{2^j} \\ a_2 \cdot a_0^{2^j} & a_2 \cdot a_1^{2^j} & a_2^{2^j+1} \end{pmatrix}$$

- Security against d^{th}order probing adversary is given in the paper.

Masking of Power functions $x \mapsto x^{2^j+1}$

Outlines of the new scheme `ISW'`

- I/O :
  - Input : $(a_i)_i$ s.t. $\bigoplus_i a_i = a$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = h(a) = a \times a^{2^j}$
- Example ($d = 2$) :

$$\begin{pmatrix} a_0^{2^j+1} & a_0 \cdot a_1^{2^j} & a_0 \cdot a_2^{2^j} \\ 0 & a_1^{2^j+1} & a_1 \cdot a_2^{2^j} \\ 0 & 0 & a_2^{2^j+1} \end{pmatrix} \oplus \begin{pmatrix} 0 & 0 & 0 \\ a_1 \cdot a_0^{2^j} & 0 & 0 \\ a_2 \cdot a_0^{2^j} & a_2 \cdot a_1^{2^j} & 0 \end{pmatrix}$$

- Security against d^{th}order probing adversary is given in the paper.

Masking of Power functions $x \mapsto x^{2^j+1}$

Outlines of the new scheme ISW'

- I/O :
  - Input : $(a_i)_i$ s.t. $\bigoplus_i a_i = a$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = h(a) = a \times a^{2^j}$
- Example ($d = 2$) :

$$\begin{pmatrix} a_0^{2^j+1} & a_0 \cdot a_1^{2^j} & a_0 \cdot a_2^{2^j} \\ 0 & a_1^{2^j+1} & a_1 \cdot a_2^{2^j} \\ 0 & 0 & a_2^{2^j+1} \end{pmatrix} \oplus \begin{pmatrix} 0 & a_1 \cdot a_0^{2^j} & a_2 \cdot a_0^{2^j} \\ 0 & 0 & a_2 \cdot a_1^{2^j} \\ 0 & 0 & 0 \end{pmatrix}$$

- Security against d^{th}order probing adversary is given in the paper.

Masking of Power functions $x \mapsto x^{2^j+1}$

Outlines of the new scheme ISW'

- I/O :
  - Input : $(a_i)_i$ s.t. $\bigoplus_i a_i = a$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = h(a) = a \times a^{2^j}$
- Example ($d = 2$) :

$$\begin{pmatrix} a_0^{2^j+1} & a_0 \cdot a_1^{2^j} \oplus a_1 \cdot a_0^{2^j} & a_0 \cdot a_2^{2^j} + a_2 \cdot a_0^{2^j} \\ 0 & a_1^{2^j+1} & a_1 \cdot a_2^{2^j} \oplus a_2 \cdot a_1^{2^j} \\ 0 & 0 & a_2^{2^j+1} \end{pmatrix}$$

- Security against d^{th}order probing adversary is given in the paper.

Masking of Power functions $x \mapsto x^{2^j+1}$

Outlines of the new scheme ISW'

- I/O :
  - Input : $(a_i)_i$ s.t. $\bigoplus_i a_i = a$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = h(a) = a \times a^{2^j}$
- Example ($d = 2$) :

$$\begin{pmatrix} h(a_0) & f(a_0, a_1) & f(a_0, a_2) \\ 0 & h(a_1) & f(a_1, a_2) \\ 0 & 0 & h(a_2) \end{pmatrix}$$

- Security against d^{th}order probing adversary is given in the paper.

Masking of Power functions $x \mapsto x^{2^j+1}$

Outlines of the new scheme ISW'

- I/O :
  - Input : $(a_i)_i$ s.t. $\bigoplus_i a_i = a$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = h(a) = a \times a^{2^j}$
- Example ($d = 2$) :

$$\begin{pmatrix} h(a_0) & f(a_0, a_1) & f(a_0, a_2) \\ 0 & h(a_1) & f(a_1, a_2) \\ 0 & 0 & h(a_2) \end{pmatrix}$$

- Security against d^{th}order probing adversary is given in the paper.

Masking of Power functions $x \mapsto x^{2^j+1}$

Outlines of the new scheme ISW'

- I/O :
  - Input : $(a_i)_i$ s.t. $\bigoplus_i a_i = a$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = h(a) = a \times a^{2^j}$
- Example ($d = 2$) :

$$\begin{pmatrix} h(a_0) & f(a_0, a_1) & f(a_0, a_2) \\ 0 & h(a_1) & f(a_1, a_2) \\ 0 & 0 & h(a_2) \end{pmatrix} \oplus \begin{pmatrix} 0 & r_{0,1} & r_{0,2} \\ 0 & 0 & r_{1,2} \\ 0 & 0 & 0 \end{pmatrix}$$

- Security against d^{th}order probing adversary is given in the paper.

Masking of Power functions $x \mapsto x^{2^j+1}$

Outlines of the new scheme ISW'

- I/O :
  - Input : $(a_i)_i$ s.t. $\bigoplus_i a_i = a$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = h(a) = a \times a^{2^j}$
- Example $(d = 2)$:

$$\begin{pmatrix} h(a_0) & f(a_0, a_1) & f(a_0, a_2) \\ 0 & h(a_1) & f(a_1, a_2) \\ 0 & 0 & h(a_2) \end{pmatrix} \oplus \begin{pmatrix} 0 & r_{0,1} & r_{0,2} \\ r_{0,1} & 0 & r_{1,2} \\ r_{0,2} & r_{1,2} & 0 \end{pmatrix}$$

- Security against d^{th}order probing adversary is given in the paper.

Masking of Power functions $x \mapsto x^{2^j+1}$

Outlines of the new scheme ISW'

- I/O :
  - Input : $(a_i)_i$ s.t. $\bigoplus_i a_i = a$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = h(a) = a \times a^{2^j}$
- Example ($d = 2$) :

$$\begin{pmatrix} h(a_0) & f(a_0, a_1) + r_{0,1} & f(a_0, a_2) + r_{0,2} \\ r_{0,1} & h(a_1) & f(a_1, a_2) + r_{1,2} \\ r_{0,2} & r_{1,2} & h(a_2) \end{pmatrix}$$

- Security against d^{th}order probing adversary is given in the paper.

Masking of Power functions $x \mapsto x^{2^j+1}$

Outlines of the new scheme ISW'

- I/O :
  - Input : $(a_i)_i$ s.t. $\bigoplus_i a_i = a$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = h(a) = a \times a^{2^j}$
- Example ($d = 2$) :

$$\begin{pmatrix} h(a_0) & f(a_0, a_1) + r_{0,1} & f(a_0, a_2) + r_{0,2} \\ r_{0,1} & h(a_1) & f(a_1, a_2) + r_{1,2} \\ r_{0,2} & r_{1,2} & h(a_2) \end{pmatrix}$$

- Security against d^{th}order probing adversary is given in the paper.

Masking of Power functions $x \mapsto x^{2^j+1}$

Outlines of the new scheme ISW'

- I/O :
  - Input : $(a_i)_i$ s.t. $\bigoplus_i a_i = a$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = h(a) = a \times a^{2^j}$
- Example $(d = 2)$: by Property * on f

$$
\begin{aligned}
f(a_i, a_j) \oplus r_{i,j} &= h(a_i \oplus a_j) \oplus h(a_i) \oplus h(a_j) \oplus r_{i,j} \\
&= \Big(h((a_i \oplus r'_{i,j}) \oplus a_j) \oplus h(r'_{i,j}) \Big) \oplus \\
&\qquad \Big(h(a_i \oplus r'_{i,j}) \oplus r_{i,j} \oplus h(a_j \oplus r'_{i,j}) \Big)
\end{aligned}
$$

- Security against d^{th}order probing adversary is given in the paper.

Masking of Power functions $x \mapsto x^{2^j+1}$

Outlines of the new scheme ISW'

- I/O :
  - Input : $(a_i)_i$ s.t. $\bigoplus_i a_i = a$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = h(a) = a \times a^{2^j}$
- Example ($d = 2$) :

$$\begin{aligned} f(a_i, a_j) \oplus r_{i,j} &= h(a_i \oplus a_j) \oplus h(a_i) \oplus h(a_j) \oplus r_{i,j} \\ &= \Big(h((a_i \oplus r'_{i,j}) \oplus a_j) \oplus h(r'_{i,j}) \Big) \oplus \\ &\qquad \Big(h(a_i \oplus r'_{i,j}) \oplus r_{i,j} \oplus h(a_j \oplus r'_{i,j}) \Big) \end{aligned}$$

- Security against d^{th}order probing adversary is given in the paper.

Masking of Power functions $x \mapsto x^{2^j+1}$

Outlines of the new scheme ISW'

- I/O :
  - Input : $(a_i)_i$ s.t. $\bigoplus_i a_i = a$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = h(a) = a \times a^{2^j}$
- Example ($d = 2$) :

$$
\begin{aligned}
f(a_i, a_j) \oplus r_{i,j} &= h(a_i \oplus a_j) \oplus h(a_i) \oplus h(a_j) \oplus r_{i,j} \\
&= \Big(h((a_i \oplus r'_{i,j}) \oplus a_j) \oplus h(r'_{i,j}) \Big) \oplus \\
&\qquad \Big(h(a_i \oplus r'_{i,j}) \oplus r_{i,j} \oplus h(a_j \oplus r'_{i,j}) \Big)
\end{aligned}
$$

- Security against d^{th}order probing adversary is given in the paper.

Masking of Power functions $x \mapsto x^{2^j+1}$

Outlines of the new scheme ISW'

- I/O :
  - Input : $(a_i)_i$ s.t. $\bigoplus_i a_i = a$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = h(a) = a \times a^{2^j}$
- Example ($d = 2$) :

$$\begin{pmatrix} h(a_0) & f(a_0, a_1) + r_{0,1} & f(a_0, a_2) + r_{0,2} \\ r_{0,1} & h(a_1) & f(a_1, a_2) + r_{1,2} \\ r_{0,2} & r_{1,2} & h(a_2) \end{pmatrix}$$

$$\begin{matrix} \downarrow & \quad\quad & \downarrow & \quad\quad & \downarrow \\ c_0 & & c_1 & & c_2 \end{matrix}$$

- Security against d^{th}order probing adversary is given in the paper.

Masking of Power functions $x \mapsto x^{2^j+1}$

Outlines of the new scheme ISW'

- I/O :
  - Input : $(a_i)_i$ s.t. $\bigoplus_i a_i = a$
  - Output : $(c_i)_i$ s.t. $\bigoplus_i c_i = h(a) = a \times a^{2^j}$
- Example ($d = 2$) :

$$\begin{pmatrix} h(a_0) & f(a_0, a_1) + r_{0,1} & f(a_0, a_2) + r_{0,2} \\ r_{0,1} & h(a_1) & f(a_1, a_2) + r_{1,2} \\ r_{0,2} & r_{1,2} & h(a_2) \end{pmatrix}$$

$$\downarrow \qquad\qquad \downarrow \qquad\qquad \downarrow$$

$$c_0 \qquad\qquad c_1 \qquad\qquad c_2$$

- Security against d^{th}order probing adversary is given in the paper.

Second Countermeasure Final Proposal

We eventually get :

1. $(z_i)_i \leftarrow (x_i^2)_i$
2. $(y_i)_i \leftarrow \mathtt{ISW}'\big((x_i)_i, j = 1\big)$
3. $(w_i)_i \leftarrow (y_i^4)_i$
4. $(y_i)_i \leftarrow \mathtt{ISW}'\big((y_i)_i, j = 2\big)$
5. $(y_i)_i \leftarrow (y_i^{16})_i$
6. $(y_i)_i \leftarrow \mathtt{ISW}\big((y_i)_i, (w_i)_i\big)$
7. $(y_i)_i \leftarrow \mathtt{ISW}\big((y_i)_i, (z_i)_i\big)$

It is not only more secure than the first Rivain-Prouff proposal, but also more efficient $\longrightarrow$ see timings in the paper.

Second Countermeasure Final Proposal

We eventually get :

1. $(z_i)_i \leftarrow (x_i^2)_i$
2. $(y_i)_i \leftarrow \mathtt{ISW}'\big((x_i)_i, j = 1\big)$
3. $(w_i)_i \leftarrow (y_i^4)_i$
4. $(y_i)_i \leftarrow \mathtt{ISW}'\big((y_i)_i, j = 2\big)$
5. $(y_i)_i \leftarrow (y_i^{16})_i$
6. $(y_i)_i \leftarrow \mathtt{ISW}\big((y_i)_i, (w_i)_i\big)$
7. $(y_i)_i \leftarrow \mathtt{ISW}\big((y_i)_i, (z_i)_i\big)$

It is not only more secure than the first Rivain-Prouff proposal, but also more efficient $\longrightarrow$ see timings in the paper.

Second Countermeasure Final Proposal

We eventually get :

1. $(z_i)_i \leftarrow (x_i^2)_i$
2. $(y_i)_i \leftarrow \mathtt{ISW}'\big((x_i)_i, j = 1\big)$
3. $(w_i)_i \leftarrow (y_i^4)_i$
4. $(y_i)_i \leftarrow \mathtt{ISW}'\big((y_i)_i, j = 2\big)$
5. $(y_i)_i \leftarrow (y_i^{16})_i$
6. $(y_i)_i \leftarrow \mathtt{ISW}\big((y_i)_i, (w_i)_i\big)$
7. $(y_i)_i \leftarrow \mathtt{ISW}\big((y_i)_i, (z_i)_i\big)$

It is not only more secure than the first Rivain-Prouff proposal, but also more efficient $\rightarrow$ see timings in the paper.

Summary 1/2

Security enhancement

- No need of the Refresh Mask Procedure
- Global security of the Masking Scheme yet to prove :
  e.g. $y = x^{14}$

  1. $(z_i)_i \leftarrow (x_i^2)_i$
  2. $(y_i)_i \leftarrow \mathtt{ISW}'\big((x_i)_i, j = 1\big)$
  3. $(w_i)_i \leftarrow (y_i^4)_i$
  4. $(y_i)_i \leftarrow \mathtt{ISW}\big((z_i)_i, (w_i)_i\big)$

i.e. composable security of dth-order secure sub-routines.

Summary 1/2

Security enhancement

- No need of the Refresh Mask Procedure
- Global security of the Masking Scheme yet to prove :
  e.g. $y = x^{14}$
  1. $(z_i)_i \leftarrow (x_i^2)_i$
  2. $(y_i)_i \leftarrow \mathtt{ISW}'\big((x_i)_i, j = 1\big)$
  3. $(w_i)_i \leftarrow (y_i^4)_i$
  4. $(y_i)_i \leftarrow \mathtt{ISW}\big((z_i)_i, (w_i)_i\big)$

i.e. composable security of dth-order secure sub-routines.

Summary 2/2

Efficiency enhancement

- $h(x) = x \cdot x^{2^j}$ are processed efficiently (lookup tables).
  $\hookrightarrow$ only 2 expensive secure multiplication in the AES s-box processing.
- can we do better ?
  $\hookrightarrow$ find the optimal expression of $x \mapsto x^{2^{254}}$ w.r.t. the number of multiplications, squarings and $h(\cdot)$.

Summary 2/2

Efficiency enhancement

- $h(x) = x \cdot x^{2^j}$ are processed efficiently (lookup tables).
  $\hookrightarrow$ only 2 expensive secure multiplication in the AES s-box processing.
- can we do better ?
  $\hookrightarrow$ find the optimal expression of $x \mapsto x^{2^{254}}$ w.r.t. the number of multiplications, squarings and $h(\cdot)$.