# Securing Finite Field Arithmetic in Embedded Systems

Emmanuel PROUFF

Safran Identity and Security

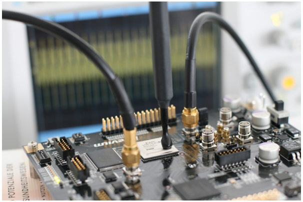
March 16, 2017

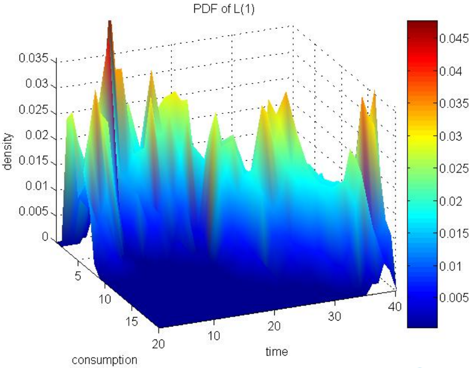
# Probability distribution function (pdf) of Electromagnetic Emanations

$$Z = S(X + k) \text{ with } X = 0 \text{ and } k = 1.$$

# Probability distribution function (pdf) of Electromagnetic Emanations

$$Z = S(X + k) \text{ with } X = 0 \text{ and } k = 1.$$



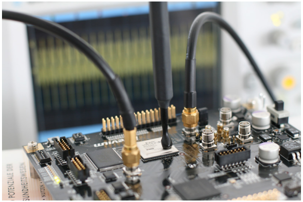


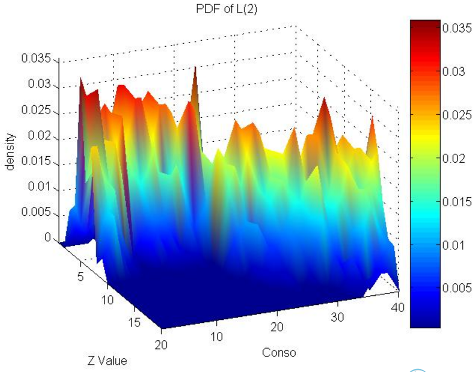


SAFRAN Morpho

## Probability distribution function (pdf) of Electromagnetic Emanations

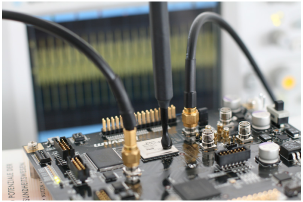
$$Z = S(X + k) \text{ with } X = 0 \text{ and } k = 2.$$

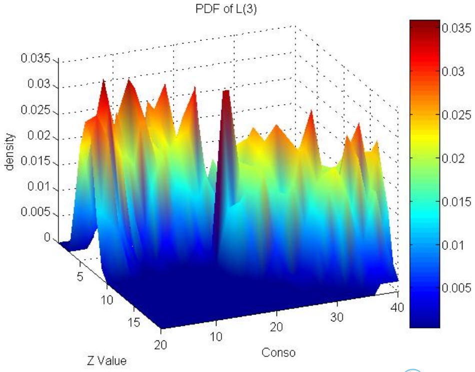




# Probability distribution function (pdf) of Electromagnetic Emanations

$$Z = S(X + k) \text{ with } X = 0 \text{ and } k = 3.$$



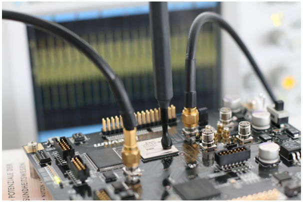


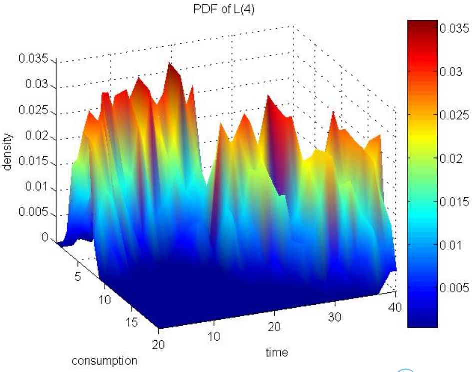


SAFRAN Morpho

# Probability distribution function (pdf) of Electromagnetic Emanations

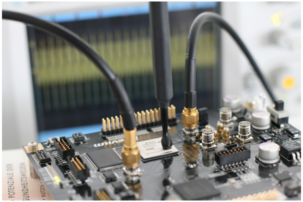
$$Z = S(X + k) \text{ with } X = 0 \text{ and } k = 4.$$

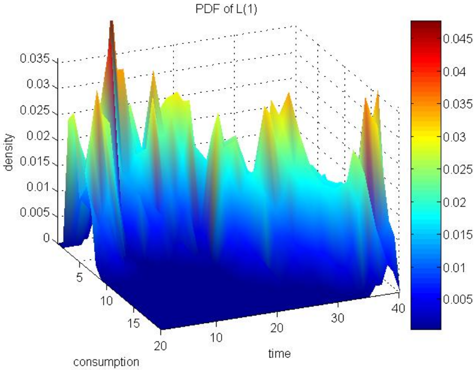




# Probability distribution function (pdf) of Electromagnetic Emanations

$$Z = S(X + k) \text{ with } X = 0 \text{ and } k \in \{1, 2, 3, 4\}.$$





## Side Channel Attacks (SCA)

- Against **each** cryptosystem and **each** implementation, find the most efficient SCA.
  - Efficiency of an SCA?
  - Which attack parameters to improve?
  - SCA common trends?
  - Attacks *versus* Characterization!

## Countermeasures

- For **each** cryptosystem, find efficient/effective countermeasures.
  - Formally define the fact that a countermeasure thwarts an SCA?
  - Which countermeasure for which SCA?
  - What makes a cryptosystem more vulnerable to SCA than another?

- Main Remark: SCA efficiency depends on the amount of noise in the observation.

- Main Remark: SCA efficiency depends on the amount of noise in the observation.

$$L = \varphi(Z) + \underbrace{\mathcal{N}}_{Noise}$$

- Main Remark: SCA efficiency depends on the amount of noise in the observation.

$$L = \varphi(Z) + \underbrace{\mathcal{N}}_{Noise}$$

- Core Idea: define mechanisms to increase the noise.

- Main Remark: SCA efficiency depends on the amount of noise in the observation.

$$L = \varphi(Z) + \underbrace{\mathcal{N}}_{Noise}$$

- Core Idea: define mechanisms to decrease the SNR.

- Main Remark: SCA efficiency depends on the amount of noise in the observation.

$$L = \varphi(Z) + \underbrace{\mathcal{N}}_{Noise}$$

- Core Idea: define mechanisms to decrease the SNR.
  - increase the noise variance.

- Main Remark: SCA efficiency depends on the amount of noise in the observation.

$$L = \varphi(Z) + \underbrace{\mathcal{N}}_{Noise}$$

- Core Idea: define mechanisms to decrease the SNR.
  - increase the noise variance.
  - force the adversary to himself decrease the SNR.

- Main Remark: SCA efficiency depends on the amount of noise in the observation.

$$L = \varphi(Z) + \underbrace{\mathcal{N}}_{Noise}$$

- Core Idea: define mechanisms to decrease the SNR.
  - increase the noise variance.
  - force the adversary to himself decrease the SNR.

- Secret Sharing: randomly split $Z$ into $d$ shares $Z_1$, ..., $Z_d$:

| $Z_1$ | $Z_2$ | $\cdots$ | $Z_d$ |
|---|---|---|---|

- Main Remark: SCA efficiency depends on the amount of noise in the observation.

$$L = \varphi(Z) + \underbrace{\mathcal{N}}_{Noise}$$

- Core Idea: define mechanisms to decrease the SNR.
  - increase the noise variance.
  - force the adversary to himself decrease the SNR.

- Secret Sharing: randomly split $Z$ into $d$ shares $Z_1$, ..., $Z_d$:

$$L_1 = \varphi(Z_1) + \mathcal{N}_1 \quad L_2 = \varphi(Z_2) + \mathcal{N}_2 \quad \cdots \quad L_d = \varphi(Z_d) + \mathcal{N}_d$$

  - all the $L_i$ are needed to get information on $Z$!
  - hence the adversary must combine all the $L_i$
  - lead to multiply the $\mathcal{N}_i$ altogether and to merge information and noise in a complex way.

## Adversary Game

In the implementation, find $d$ or less intermediate variables that jointly depend on a secret variable $Z$.

## Developer Game

Translate (Compile?) an implementation into a new one defeating the adversary.

Implementation = sequence of **elementary operations** which **read a memory location** and **write its result in another memory location**.

- First Issue: how to share sensitive data?

- Second Issue: how to securely process on shared data?

- **First Issue:** how to share sensitive data?
- **Related to:**
  - secret sharing *Shamir79*
  - design of error correcting codes with large dual distance *Massey93, CastagnosRennerZémor13*
  - etc.

- **Second Issue:** how to securely process on shared data?
- **Related to:**
  - secure multi-party computation *NikovaRijmenSchläffer2008 ProuffRoche2011*
  - circuit processing in presence of leakage *e.g. GoldwasserRothblum2012*
  - efficient polynomial evaluation *e.g. CarletGoubinProuffQuisquater-Rivain2012, CoronProuffRoche2012, CoronRoyVivek2014*
  - etc.

- Linear Secret Sharing with parameters $n$ and $d$:
  - $n$ elements $Z_i$ such that

$$Z = \sum_i Z_i$$

  - no sub-family of $d-1$ $Z_i$ depends on $Z$.


SAFRAN Morpho

- Linear Secret Sharing with parameters $n$ and $d$:
  - $n$ elements $Z_i$ such that

$$Z = \sum_i Z_i$$

  - no sub-family of $d-1$ $Z_i$ depends on $Z$.

- Massey (1993):

designing an $(n, d)$ linear secret sharing

$$\Longleftrightarrow$$

building a code with length $n+1$ and dual distance $d$

- Linear Secret Sharing with parameters $n$ and $d$:
  - $n$ elements $Z_i$ such that

$$Z = \sum_i Z_i$$

  - no sub-family of $d-1$ $Z_i$ depends on $Z$.

- Massey (1993):

designing an $(n, d)$ linear secret sharing

$$\Longleftrightarrow$$

building a code with length $n+1$ and dual distance $d$

- Yes, interesting, but ... who cares?

- Linear Secret Sharing with parameters $n$ and $d$:
  - $n$ elements $Z_i$ such that

$$Z = \sum_i Z_i$$

  - no sub-family of $d-1$ $Z_i$ depends on $Z$.

- Massey (1993):

designing an $(n, d)$ linear secret sharing

$$\Longleftrightarrow$$

building a code with length $n+1$ and dual distance $d$

- Yes, interesting, but ... who cares?
  - gives a general framework to describe and analyse all linear sharing schemes

- Linear Secret Sharing with parameters $n$ and $d$:
  - $n$ elements $Z_i$ such that

$$Z = \sum_i Z_i$$

  - no sub-family of $d-1$ $Z_i$ depends on $Z$.

- Massey (1993):

designing an $(n,d)$ linear secret sharing

$$\Longleftrightarrow$$

building a code with length $n+1$ and dual distance $d$

- Yes, interesting, but ... who cares?
  - gives a general framework to describe and analyse all linear sharing schemes
  - links our problems with those of a rich community

- Linear Sharing = Encoding

$$\begin{pmatrix} \color{red}{Z} & R_1 & \dots & R_{k-1} \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 & \alpha_{1,k} & \dots & \alpha_{1,n} \\ 0 & 1 & 0 & 0 & \alpha_{2,k} & \dots & \alpha_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & \alpha_{k,k} & \dots & \alpha_{k,n} \end{pmatrix}$$

$$= \begin{pmatrix} \color{red}{Z} & Z_1 & \dots & Z_{k-1} & Z_k & \dots & Z_n \end{pmatrix}$$


SAFRAN Morpho

## Linear Sharing = Encoding

$$\begin{pmatrix} Z & R_1 & \dots & R_{k-1} \end{pmatrix} \times \overbrace{\left(\mathrm{Id}_k | M\right)}^{=\mathbf{G}} = \begin{pmatrix} Z & Z_1 & \dots & Z_{k-1} & Z_k & \dots & Z_n \end{pmatrix}$$


SAFRAN Morpho

- Linear Sharing = Encoding

$$\begin{pmatrix} \color{red}{Z} & Z_1 & \dots & Z_n \end{pmatrix} \times \begin{pmatrix} \alpha_{1,k} & \dots & \dots & \alpha_{k,k} \\ \alpha_{1,k+1} & \dots & \dots & \alpha_{k,k+1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{1,n} & \dots & \dots & \alpha_{k,n} \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 0 & \dots & 0 \end{pmatrix}$$

- Linear Sharing = Encoding

$$\begin{pmatrix} \color{red}{Z} & Z_1 & \dots & Z_n \end{pmatrix} \times \overbrace{\begin{pmatrix} \vec{H}_1 & \vec{H}_2 & \dots & \vec{H}_k \end{pmatrix}}^{=\mathbf{H}} = \begin{pmatrix} 0 & 0 & \dots & 0 \end{pmatrix}$$

- Linear Sharing = Encoding

$$\begin{pmatrix} Z & Z_1 & \dots & Z_n \end{pmatrix} \times \overbrace{\begin{pmatrix} \vec{H}_1 & \vec{H}_2 & \dots & \vec{H}_k \end{pmatrix}}^{=\mathbf{H}} = \begin{pmatrix} 0 & 0 & \dots & 0 \end{pmatrix}$$

- implies for every $i \in [1..k]$:

$$Z = H_{i,0}^{-1} \sum_{j=1}^{n} Z_j \times H_{i,j} \quad .$$

where $\vec{H}_i \doteq (H_{i,0}, \cdots, H_{i,n})^\intercal$.


SAFRAN Morpho

- Linear Sharing = Encoding

$$\begin{pmatrix} Z & Z_1 & \dots & Z_n \end{pmatrix} \times \overbrace{\begin{pmatrix} \vec{H}_1 & \vec{H}_2 & \dots & \vec{H}_k \end{pmatrix}}^{=\mathbf{H}} = \begin{pmatrix} 0 & 0 & \dots & 0 \end{pmatrix}$$

- implies for every $i \in [1..k]$:

$$Z = H_{i,0}^{-1} \sum_{j=1}^{n} Z_j \times H_{i,j} \quad .$$

where $\vec{H}_i \doteq (H_{i,0}, \cdots, H_{i,n})^\intercal$.

- masking/sharing order $< \min_{(a_1,\cdots,a_k)\in\mathbb{F}_2^k} \mathrm{HW}(\sum_i a_i \vec{H}_i) - 1$

- Linear Sharing = Encoding

$$\begin{pmatrix} Z & Z_1 & \dots & Z_n \end{pmatrix} \times \overbrace{\begin{pmatrix} \vec{H}_1 & \vec{H}_2 & \dots & \vec{H}_k \end{pmatrix}}^{=\mathbf{H}} = \begin{pmatrix} 0 & 0 & \dots & 0 \end{pmatrix}$$

- implies for every $i \in [1..k]$:

$$Z = H_{i,0}^{-1} \sum_{j=1}^{n} Z_j \times H_{i,j} \ .$$

where $\vec{H}_i \doteq (H_{i,0}, \cdots, H_{i,n})^\intercal$.

- masking/sharing order $< \min_{(a_1,\cdots,a_k)\in\mathbb{F}_2^k} \mathrm{HW}(\sum_i a_i \vec{H}_i) - 1$
- Actually masking order $= \min_{(a_1,\cdots,a_k)\in\mathbb{F}_2^k} \mathrm{HW}(\sum_i a_i \vec{H}_i) - 1$

*Massey93*

- Boolean Sharing: encoding with the matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

implies $k = n - 1$.

- **Boolean Sharing:** encoding with the matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

  implies $k = n - 1$.
- **Shamir's secret Sharing:**
  - generate a random degree-$d$ polynomial $P(X)$ such that $P(0) = Z$
  - build the $Z_i$ such that $Z_i = P(\alpha_i)$ for $n \geq 2d$ different public values $\alpha_i$.

- Boolean Sharing: encoding with the matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

implies $k = n - 1$.

- Shamir's secret Sharing:
  - generate a random degree-$d$ polynomial $P(X)$ such that $P(0) = Z$
  - build the $Z_i$ such that $Z_i = P(\alpha_i)$ for $n \geq 2d$ different public values $\alpha_i$.
- ... amounts to define a Reed-Solomon code with parameters $[n+1, d+1, \cdot]$ *McElieceSarwate81*.


SAFRAN Morpho

- Boolean Sharing: encoding with the matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

  implies $k = n - 1$.
- Shamir's secret Sharing:
  - generate a random degree-$d$ polynomial $P(X)$ such that $P(0) = Z$
  - build the $Z_i$ such that $Z_i = P(\alpha_i)$ for $n \geq 2d$ different public values $\alpha_i$.
- ... amounts to define a Reed-Solomon code with parameters $[n+1, d+1, \cdot]$ *McElieceSarwate81*.
- Main issue: minimize $n$ for a given $d$.


SAFRAN Morpho

- Securing elementary Operations:

- Securing elementary Operations:
- Original idea by *Ishai-Sahai-Wagner*: limited to GF(2)

- Securing elementary Operations:
- Original idea by *Ishai-Sahai-Wagner*: limited to GF(2)
- Extended to any field in *RivainProuff2010* and *FaustRabinReyzinTromerVaikuntanathan2011*


SAFRAN Morpho

- Securing elementary Operations:
- Original idea by *Ishai-Sahai-Wagner*: limited to GF(2)
- Extended to any field in *RivainProuff2010* and *FaustRabinReyzinTromerVaikuntanathan2011*
- Based on Boolean Sharing: $Z = Z_0 \oplus Z_1 \oplus \dots Z_d$

- Securing elementary Operations:

- Original idea by *Ishai-Sahai-Wagner*: limited to GF(2)
- Extended to any field in *RivainProuff2010* and *FaustRabinReyzinTromerVaikuntanathan2011*
- Based on Boolean Sharing: $Z = Z_0 \oplus Z_1 \oplus \ldots Z_d$
- Securing linear functions L:

$$\begin{array}{cccc} Z_0 & Z_1 & \cdots & Z_d \\ \downarrow & \downarrow & \downarrow & \downarrow \\ \mathrm{L}(Z_0) & \mathrm{L}(Z_1) & \cdots & \mathrm{L}(Z_d) \end{array}$$

- Securing elementary Operations:

- Original idea by *Ishai-Sahai-Wagner*: limited to GF(2)
- Extended to any field in *RivainProuff2010* and *FaustRabinReyzinTromerVaikuntanathan2011*
- Based on Boolean Sharing: $Z = Z_0 \oplus Z_1 \oplus \ldots Z_d$
- Securing linear functions L:

$$\begin{array}{ccccccccc} & Z_0 & & Z_1 & & \cdots & & Z_d \\ & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ \mathrm{L}(Z) = & \mathrm{L}(Z_0) & \oplus & \mathrm{L}(Z_1) & \oplus & \cdots & \oplus & \mathrm{L}(Z_d) \end{array}$$


SAFRAN Morpho

- Securing elementary Operations:
- Original idea by *Ishai-Sahai-Wagner*: limited to GF(2)
- Extended to any field in *RivainProuff2010* and *FaustRabinReyzinTromerVaikuntanathan2011*
- Based on Boolean Sharing: $Z = Z_0 \oplus Z_1 \oplus \ldots Z_d$
- Securing linear functions L:

$$\begin{array}{ccccccccc} & Z_0 & & Z_1 & & \cdots & & Z_d \\ & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ \mathrm{L}(Z) = & \mathrm{L}(Z_0) & \oplus & \mathrm{L}(Z_1) & \oplus & \cdots & \oplus & \mathrm{L}(Z_d) \end{array}$$

- Much more difficult for non-linear functions (*i.e.* multiplication)

## Securing Multiplication *IshaiSahaiWagner2003*:

- 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. $\bigoplus_i c_i = ab$

## Securing Multiplication *IshaiSahaiWagner2003*:

- 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. $\bigoplus_i c_i = ab$

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

- Securing Multiplication *IshaiSahaiWagner2003*:
  - 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. $\bigoplus_i c_i = ab$

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

- Illustration of ISW scheme for $d = 2$:

$$\begin{pmatrix} a_0b_0 & a_0b_1 & a_0b_2 \\ a_1b_0 & a_1b_1 & a_1b_2 \\ a_2b_0 & a_2b_1 & a_2b_2 \end{pmatrix}$$

- Securing Multiplication *IshaiSahaiWagner2003*:
  - 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. $\bigoplus_i c_i = ab$

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

- Illustration of ISW scheme for $d = 2$:

$$\begin{pmatrix} a_0b_0 & a_0b_1 & a_0b_2 \\ a_1b_0 & a_1b_1 & a_1b_2 \\ a_2b_0 & a_2b_1 & a_2b_2 \end{pmatrix} \oplus \begin{pmatrix} r_{0,0} & r_{0,1} & r_{0,2} \\ r_{1,0} & r_{1,1} & r_{1,2} \\ r_{2,0} & r_{2,1} & r_{2,2} \end{pmatrix}$$

where the $r_{i,j}$ are a sharing of 0.

## Securing Multiplication *IshaiSahaiWagner2003*:

- 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. $\bigoplus_i c_i = ab$

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

## Illustration of ISW scheme for $d = 2$:

$$\begin{pmatrix} a_0b_0 \oplus r_{0,0} & a_0b_1 \oplus r_{0,1} & a_0b_2 \oplus r_{0,2} \\ a_1b_0 \oplus r_{1,0} & a_1b_1 \oplus r_{1,1} & a_1b_2 \oplus r_{1,2} \\ a_2b_0 \oplus r_{2,0} & a_2b_1 \oplus r_{2,1} & a_2b_2 \oplus r_{2,2} \end{pmatrix}$$

## Securing Multiplication *IshaiSahaiWagner2003*:

- 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. $\bigoplus_i c_i = ab$

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

## Illustration of ISW scheme for $d = 2$:

$$\begin{pmatrix} a_0b_0 \oplus r_{0,0} & a_0b_1 \oplus r_{0,1} & a_0b_2 \oplus r_{0,2} \\ a_1b_0 \oplus r_{1,0} & a_1b_1 \oplus r_{1,1} & a_1b_2 \oplus r_{1,2} \\ a_2b_0 \oplus r_{2,0} & a_2b_1 \oplus r_{2,1} & a_2b_2 \oplus r_{2,2} \end{pmatrix}$$

$c_1$


SAFRAN Morpho

## Securing Multiplication *IshaiSahaiWagner2003*:

- 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. $\bigoplus_i c_i = ab$

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

## Illustration of ISW scheme for $d = 2$:

$$\begin{pmatrix} a_0b_0 \oplus r_{0,0} & a_0b_1 \oplus r_{0,1} & a_0b_2 \oplus r_{0,2} \\ a_1b_0 \oplus r_{1,0} & a_1b_1 \oplus r_{1,1} & a_1b_2 \oplus r_{1,2} \\ a_2b_0 \oplus r_{2,0} & a_2b_1 \oplus r_{2,1} & a_2b_2 \oplus r_{2,2} \end{pmatrix}$$

$$c_1 \qquad\qquad c_2$$

## Securing Multiplication *IshaiSahaiWagner2003*:

- 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. $\bigoplus_i c_i = ab$

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

## Illustration of ISW scheme for $d = 2$:

$$\begin{pmatrix} a_0b_0 \oplus r_{0,0} & a_0b_1 \oplus r_{0,1} & a_0b_2 \oplus r_{0,2} \\ a_1b_0 \oplus r_{1,0} & a_1b_1 \oplus r_{1,1} & a_1b_2 \oplus r_{1,2} \\ a_2b_0 \oplus r_{2,0} & a_2b_1 \oplus r_{2,1} & a_2b_2 \oplus r_{2,2} \end{pmatrix}$$
$$\quad c_1 \qquad\qquad c_2 \qquad\qquad c_3$$


SAFRAN Morpho

- **Securing Multiplication** *IshaiSahaiWagner2003*:
  - 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. $\bigoplus_i c_i = ab$

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

- **Illustration of ISW scheme for $d = 2$:**

$$\begin{pmatrix} a_0b_0 \oplus r_{0,0} & a_0b_1 \oplus r_{0,1} & a_0b_2 \oplus r_{0,2} \\ a_1b_0 \oplus r_{1,0} & a_1b_1 \oplus r_{1,1} & a_1b_2 \oplus r_{1,2} \\ a_2b_0 \oplus r_{2,0} & a_2b_1 \oplus r_{2,1} & a_2b_2 \oplus r_{2,2} \end{pmatrix}$$
$$\quad c_1 \qquad\qquad\qquad c_2 \qquad\qquad\qquad c_3$$

- Actually, we can do it with $(d+1)^2/2$ random values instead of $(d+1)^2$ *(Ishai, Sahai, Wagner, CRYPTO 2003)*, and even in $d + d^2/4$ *(Belaid et al, Eurocrypt 2016)*.
- **Problematic:** Random Complexity of a $d$-secure multiplication?

# Securing any Polynomial evaluation

- Write the s-box $\mathrm{S} : \{0,1\}^n \rightarrow \{0,1\}^m$ as a polynomial function over $\mathrm{GF}(2^n)$:

$$\mathrm{S}(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{2^n-1} x^{2^n-1}$$


SAFRAN Morpho

# Securing any Polynomial evaluation

- Write the s-box $\mathrm{S} : \{0,1\}^n \to \{0,1\}^m$ as a polynomial function over $\mathrm{GF}(2^n)$:

$$\mathrm{S}(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{2^n-1} x^{2^n-1}$$

- Four kinds of operations over $\mathrm{GF}(2^n)$:
  1. additions
  2. scalar multiplications (*i.e.* by constants)
  3. squares
  4. regular multiplications

# Securing any Polynomial evaluation

- Write the s-box $\mathrm{S} : \{0,1\}^n \to \{0,1\}^m$ as a polynomial function over $\mathrm{GF}(2^n)$:

$$\mathrm{S}(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{2^n-1} x^{2^n-1}$$

- Four kinds of operations over $\mathrm{GF}(2^n)$:
  1. additions
  2. scalar multiplications (*i.e.* by constants)
  3. squares
  4. regular multiplications

- Schemes with complexity $O(d)$ for the 3 first kinds
  - $(x+y) \longrightarrow (x_0+y_0), (x_1+y_1), \cdots, (x_d+y_d)$
  - $x^2 \longrightarrow x_0^2, x_1^2, \cdots + x_d^2$
  - $a \cdot x \longrightarrow a \cdot x_0, a \cdot x_1, \cdots, a \cdot x_d$

# Securing any Polynomial evaluation

- Write the s-box $S : \{0,1\}^n \to \{0,1\}^m$ as a polynomial function over $GF(2^n)$:

$$S(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{2^n-1} x^{2^n-1}$$

- Four kinds of operations over $GF(2^n)$:
  1. additions
  2. scalar multiplications (*i.e.* by constants)
  3. squares
  4. regular multiplications $\Rightarrow$ *nonlinear multiplications*

- Schemes with complexity $O(d)$ for the 3 first kinds
  - $(x+y) \longrightarrow (x_0+y_0), (x_1+y_1), \cdots, (x_d+y_d)$
  - $x^2 \longrightarrow x_0^2, x_1^2, \cdots + x_d^2$
  - $a \cdot x \longrightarrow a \cdot x_0, a \cdot x_1, \cdots, a \cdot x_d$

- Schemes with complexity $O(d^2)$ for the non-linear multiplication

*IshaiSahaiWagner2004*

**Definition** (*CarletGoubinProuffQuisquaterRivain2012*)

The masking complexity of $S$ is the minimal number of non-linear multiplications needed for its evaluation.

**Definition** (*CarletGoubinProuffQuisquaterRivain2012*)

The masking complexity of $S$ is the minimal number of non-linear multiplications needed for its evaluation.

Problematic 1: compute the masking complexity of any $S$ (at least bounds).

**Definition** (*CarletGoubinProuffQuisquaterRivain2012*)

The masking complexity of $S$ is the minimal number of non-linear multiplications needed for its evaluation.

Problematic 1: compute the masking complexity of any $S$ (at least bounds).

Problematic 2: find evaluations methods efficient for the masking complexity criterion.

**Definition** (*CarletGoubinProuffQuisquaterRivain2012*)

The masking complexity of $S$ is the minimal number of non-linear multiplications needed for its evaluation.

Problematic 1: compute the masking complexity of any $S$ (at least bounds).

Problematic 2: find evaluations methods efficient for the masking complexity criterion.

For monomials: amounts to look for short 2-addition-chain exponentiations.

**Definition** (*CarletGoubinProuffQuisquaterRivain2012*)

The masking complexity of $S$ is the minimal number of non-linear multiplications needed for its evaluation.

Problematic 1: compute the masking complexity of any $S$ (at least bounds).

Problematic 2: find evaluations methods efficient for the masking complexity criterion.

For monomials: amounts to look for short 2-addition-chain exponentiations.

For polynomials: amounts to find efficient decompositions;

- Knuth-Eve algorithm *VonZurGathenNoker2003*
- or the Cyclotomic Method *CarletGoubinProuffQuisquaterRivain2012*
- or Coron-Roy-Vivek's method *CoronRoyVivek2014*

# Cyclotomic Method

- Cyclotomic class of $\alpha$ : $C_\alpha = \{\alpha \cdot 2^j \bmod (2^n - 1);\ j \leq n\}$

# Cyclotomic Method

- Cyclotomic class of $\alpha$ : $C_\alpha = \{\alpha \cdot 2^j \bmod (2^n - 1);\ j \leq n\}$
- $\beta \in C_\alpha \Leftrightarrow C_\beta = C_\alpha$:

# Cyclotomic Method

- Cyclotomic class of $\alpha$ : $C_\alpha = \{\alpha \cdot 2^j \bmod (2^n - 1);\ j \leq n\}$
- $\beta \in C_\alpha \Leftrightarrow C_\beta = C_\alpha$:
  - $x^\alpha$ deduced from $x^\beta$ with 0 nonlinear multiplication

# Cyclotomic Method

- Cyclotomic class of $\alpha$ : $C_\alpha = \{\alpha \cdot 2^j \bmod (2^n - 1);\ j \leq n\}$
- $\beta \in C_\alpha \Leftrightarrow C_\beta = C_\alpha$:
  - $x^\alpha$ deduced from $x^\beta$ with 0 nonlinear multiplication
  - $x^\alpha$ and $x^\beta$ have the same masking complexity

# Cyclotomic Method

- Cyclotomic class of $\alpha$ : $C_\alpha = \{\alpha \cdot 2^j \bmod (2^n - 1);\ j \leq n\}$
- $\beta \in C_\alpha \Leftrightarrow C_\beta = C_\alpha$:
  - $x^\alpha$ deduced from $x^\beta$ with 0 nonlinear multiplication
  - $x^\alpha$ and $x^\beta$ have the same masking complexity

$$\begin{aligned} \mathrm{S}(x) &= a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6 + a_7x^7 \\ &\quad + a_8x^8 + a_9x^9 + a_{10}x^{10} + a_{11}x^{11} + a_{12}x^{12} + \ldots \end{aligned}$$

# Cyclotomic Method

- Cyclotomic class of $\alpha$ : $C_\alpha = \{\alpha \cdot 2^j \bmod (2^n - 1);\ j \leq n\}$
- $\beta \in C_\alpha \Leftrightarrow C_\beta = C_\alpha$:
  - $x^\alpha$ deduced from $x^\beta$ with 0 nonlinear multiplication
  - $x^\alpha$ and $x^\beta$ have the same masking complexity

$$\begin{aligned} \mathrm{S}(x) &= a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^7 \\ &\quad + a_8 x^8 + a_9 x^9 + a_{10} x^{10} + a_{11} x^{11} + a_{12} x^{12} + \ldots \end{aligned}$$


SAFRAN Morpho

# Cyclotomic Method

- Cyclotomic class of $\alpha$ : $C_\alpha = \{\alpha \cdot 2^j \bmod (2^n - 1);\ j \leq n\}$
- $\beta \in C_\alpha \Leftrightarrow C_\beta = C_\alpha$:
  - $x^\alpha$ deduced from $x^\beta$ with 0 nonlinear multiplication
  - $x^\alpha$ and $x^\beta$ have the same masking complexity

$$\begin{aligned} \mathrm{S}(x) &= a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^7 \\ &\quad + a_8 x^8 + a_9 x^9 + a_{10} x^{10} + a_{11} x^{11} + a_{12} x^{12} + \dots \end{aligned}$$


SAFRAN Morpho

# Cyclotomic Method

- Cyclotomic class of $\alpha$ : $C_\alpha = \{\alpha \cdot 2^j \bmod (2^n - 1);\ j \leq n\}$
- $\beta \in C_\alpha \Leftrightarrow C_\beta = C_\alpha$:
  - $x^\alpha$ deduced from $x^\beta$ with 0 nonlinear multiplication
  - $x^\alpha$ and $x^\beta$ have the same masking complexity

$$
\begin{aligned}
\mathrm{S}(x) \quad = \quad & a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + \ a_5 x^5 + a_6 x^6 + a_7 x^7 \\
& + a_8 x^8 + a_9 x^9 + \ a_{10} x^{10} + a_{11} x^{11} + a_{12} x^{12} + \dots
\end{aligned}
$$

# Cyclotomic Method

- Cyclotomic class of $\alpha$ : $C_\alpha = \{\alpha \cdot 2^j \bmod (2^n - 1);\ j \leq n\}$
- $\beta \in C_\alpha \Leftrightarrow C_\beta = C_\alpha$:
  - $x^\alpha$ deduced from $x^\beta$ with 0 nonlinear multiplication
  - $x^\alpha$ and $x^\beta$ have the same masking complexity

$$
\begin{aligned}
\mathrm{S}(x) \quad = \quad & a_0 + a_1 x + a_2 x^2 + a_4 x^4 + a_8 x^8 + \ldots \\
& + a_3 x^3 + a_6 x^6 + a_{12} x^{12} + a_{24} x^{24} + \ldots \\
& + a_5 x^5 + a_{10} x^{10} + a_{20} x^{20} + a_{40} x^{40} + \ldots \\
& + \ldots
\end{aligned}
$$


SAFRAN Morpho

# Cyclotomic Method

- Cyclotomic class of $\alpha$ : $C_\alpha = \{\alpha \cdot 2^j \bmod (2^n - 1);\ j \leq n\}$
- $\beta \in C_\alpha \Leftrightarrow C_\beta = C_\alpha$:
  - $x^\alpha$ deduced from $x^\beta$ with 0 nonlinear multiplication
  - $x^\alpha$ and $x^\beta$ have the same masking complexity

$$
\begin{aligned}
S(x) \quad = \quad & a_0 + a_1 x + a_2 x^2 + a_4 x^4 + a_8 x^8 + \ldots \\
& + a_3 x^3 + a_6 (x^3)^2 + a_{12} (x^3)^4 + a_{24} (x^3)^8 + \ldots \\
& + a_5 x^5 + a_{10} (x^5)^2 + a_{20} (x^5)^4 + a_{40} (x^5)^8 + \ldots \\
& + \ldots
\end{aligned}
$$

# Cyclotomic Method

- Cyclotomic class of $\alpha$ : $C_\alpha = \{\alpha \cdot 2^j \bmod (2^n - 1);\ j \leq n\}$
- $\beta \in C_\alpha \Leftrightarrow C_\beta = C_\alpha$:
  - $x^\alpha$ deduced from $x^\beta$ with 0 nonlinear multiplication
  - $x^\alpha$ and $x^\beta$ have the same masking complexity

$$\mathrm{S}(x) = a_0 + L_1(x) + L_3(x^3) + L_5(x^5) + \dots$$

where

- $L_1(X) = a_1 X + a_2 X^2 + a_4 X^4 + a_8 X^8 + \dots$
- $L_3(X) = a_3 X + a_6 X^2 + a_{12} X^4 + a_{24} X^8 + \dots$
- $L_5(X) = a_5 X + a_{10} X^2 + a_{20} X^4 + a_{40} X^8 + \dots$
- ...

# Cyclotomic Method

To sum up:

# Cyclotomic Method

To sum up:

1. Compute one power per cyclotomic class $x$, $x^3$, $x^5$, $x^7$, ...

# Cyclotomic Method

To sum up:

1. Compute one power per cyclotomic class $x$, $x^3$, $x^5$, $x^7$, ...
2. Evaluate the corresponding linearized polynomials $L_i(x^i)$

## Cyclotomic Method

To sum up:

1. Compute one power per cyclotomic class $x$, $x^3$, $x^5$, $x^7$, ...
2. Evaluate the corresponding linearized polynomials $L_i(x^i)$
3. Then compute the sum

$$\mathrm{S}(x) = a_0 + L_1(x) + L_3(x^3) + L_5(x^5) + L_7(x^7) + \dots$$

# Cyclotomic Method

To sum up:

1. Compute one power per cyclotomic class $x$, $x^3$, $x^5$, $x^7$, ...
2. Evaluate the corresponding linearized polynomials $L_i(x^i)$
3. Then compute the sum

$$\mathrm{S}(x) = a_0 + L_1(x) + L_3(x^3) + L_5(x^5) + L_7(x^7) + \dots$$

*CarletGoubinProuffQuisquaterRivain2012*

Number of nonlinear multiplications

$=$

$\#\{$**cyclotomic classes involved in** $\mathrm{S}\}\backslash(C_0 \cup C_1)$

# Knuth-Eve's Method for SBox Evaluation

$$\begin{aligned} \mathrm{S}(x) &= a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^7 \\ &\quad + a_8 x^8 + a_9 x^9 + a_{10} x^{10} + a_{11} x^{11} + a_{12} x^{12} + \ldots \end{aligned}$$

# Knuth-Eve's Method for SBox Evaluation

$$
\begin{aligned}
S(x) \quad = \quad & a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^7 \\
& + a_8 x^8 + a_9 x^9 + a_{10} x^{10} + a_{11} x^{11} + a_{12} x^{12} + \dots
\end{aligned}
$$

# Knuth-Eve's Method for SBox Evaluation

$$\mathrm{S}(x) = a_0 + a_2x^2 + a_4x^4 + a_6x^6 + a_8x^8 + \ldots$$
$$a_1x + a_3x^3 + a_5x^5 + a_7x^7 + a_9x^9 + \ldots$$

# Knuth-Eve's Method for SBox Evaluation

$$\begin{aligned} S(x) &= a_0 + a_2x^2 + a_4x^4 + a_6x^6 + a_8x^8 + \dots \\ &\quad (a_1 + a_3x^2 + a_5x^4 + a_7x^6 + a_9x^8 + \dots) \cdot x \end{aligned}$$

- Nonlinear mult. : 1


SAFRAN Morpho

# Knuth-Eve's Method for SBox Evaluation

$$\begin{aligned} \mathrm{S}(x) &= a_0 + a_2x^2 + a_4x^4 + a_6x^6 + a_8x^8 + \dots \\ &\quad (a_1 + a_3x^2 + a_5x^4 + a_7x^6 + a_9x^8 + \dots) \cdot x \end{aligned}$$

- Nonlinear mult. : 1

# Knuth-Eve's Method for SBox Evaluation

$$\begin{aligned} S(x) &= a_0 + a_2X + a_4X^2 + a_6X^3 + a_8X^4 + \ldots \\ &\quad (a_1 + a_3X + a_5X^2 + a_7X^3 + a_9X^4 + \ldots) \cdot x \end{aligned}$$

where $X = x^2$

- Nonlinear mult. : 1

# Knuth-Eve's Method for SBox Evaluation

$$\begin{aligned} \mathrm{S}(x) &= a_0 + a_2 X + a_4 X^2 + a_6 X^3 + a_8 X^4 + \ldots \\ &\quad (a_1 + a_3 X + a_5 X^2 + a_7 X^3 + a_9 X^4 + \ldots) \cdot x \end{aligned}$$

where $X = x^2$

- Nonlinear mult. : 1

# Knuth-Eve's Method for SBox Evaluation

$$\begin{aligned} S(x) &= a_0 + a_4X^2 + a_8X^4 + \ldots + a_2X + a_6X^3 + \ldots \\ &\quad (a_1 + a_5X^2 + a_9X^4 + \ldots + a_3x^2 + a_7X^3 + \ldots) \cdot x \end{aligned}$$

where $X = x^2$

- Nonlinear mult. : 1

# Knuth-Eve's Method for SBox Evaluation

$$\begin{aligned} \mathrm{S}(x) &= a_0 + a_4X^2 + a_8X^4 + \ldots + (a_2 + a_6X^2 + \ldots) \cdot X + \\ &\quad (a_1 + a_5X^2 + a_9X^4 + \ldots + (a_3 + a_7X^2 + \ldots) \cdot X) \cdot x \end{aligned}$$

where $X = x^2$

- Nonlinear mult. : 1+2

# Knuth-Eve's Method for SBox Evaluation

$$\begin{aligned} \mathrm{S}(x) &= a_0 + a_4x^4 + a_8x^8 + \ldots + (a_2 + a_6x^4 + \ldots) \cdot x^2 + \\ & (a_1 + a_5x^4 + a_9x^8 + \ldots + (a_3 + a_7x^4 + \ldots) \cdot x^2) \cdot x \end{aligned}$$

- Nonlinear mult. : 1+2

# Knuth-Eve's Method for SBox Evaluation

$$
\begin{aligned}
\mathrm{S}(x) \quad = \quad & a_0 + a_4X + a_8X^2 + \ldots + (a_2 + a_6X + \ldots) \cdot x^2 + \\
& (a_1 + a_5X + a_9X^2 + \ldots + (a_3 + a_7X + \ldots) \cdot x^2) \cdot x
\end{aligned}
$$

where $X = x^4$

- Nonlinear mult. : 1+2

# Knuth-Eve's Method for SBox Evaluation

$$\begin{aligned} \mathrm{S}(x) &= a_0 + a_4 X + a_8 X^2 + \ldots + (a_2 + a_6 X + \ldots) \cdot x^2 + \\ &\quad (a_1 + a_5 X + a_9 X^2 + \ldots + (a_3 + a_7 X + \ldots) \cdot x^2) \cdot x \end{aligned}$$

where $X = x^4$

- Nonlinear mult. : $1 + 2 + \cdots + 2^{r-1} = 2^r - 1$


SAFRAN Morpho

# Knuth-Eve's Method for SBox Evaluation

$$\begin{aligned} \mathrm{S}(x) &= a_0 + a_4 X + a_8 X^2 + \ldots + (a_2 + a_6 X + \ldots) \cdot x^2 + \\ &\quad (a_1 + a_5 X + a_9 X^2 + \ldots + (a_3 + a_7 X + \ldots) \cdot x^2) \cdot x \end{aligned}$$

where $X = x^4$

- Nonlinear mult. : $1+2+\cdots+2^{r-1} = 2^r - 1$
- and the evaluation of $2^{r+1}$ polynomials in $X = x^{2^r}$

# Knuth-Eve's Method for SBox Evaluation

$$\begin{aligned} \mathrm{S}(x) &= a_0 + a_4 X + a_8 X^2 + \ldots + (a_2 + a_6 X + \ldots) \cdot x^2 + \\ & (a_1 + a_5 X + a_9 X^2 + \ldots + (a_3 + a_7 X + \ldots) \cdot x^2) \cdot x \end{aligned}$$

where $X = x^4$

- Nonlinear mult. : $1+2+\cdots+2^{r-1} = 2^r - 1$
- and the evaluation of $2^{r+1}$ polynomials in $X = x^{2^r}$
  - we derive $X^j$ for $j < 2^{n-r}$


SAFRAN Morpho

# Knuth-Eve's Method for SBox Evaluation

$$\begin{aligned} \mathrm{S}(x) &= a_0 + a_4 X + a_8 X^2 + \ldots + (a_2 + a_6 X + \ldots) \cdot x^2 + \\ & (a_1 + a_5 X + a_9 X^2 + \ldots + (a_3 + a_7 X + \ldots) \cdot x^2) \cdot x \end{aligned}$$

where $X = x^4$

- Nonlinear mult. : $1+2+\cdots+2^{r-1} = 2^r - 1$
- and the evaluation of $2^{r+1}$ polynomials in $X = x^{2^r}$
  - we derive $X^j$ for $j < 2^{n-r}$
  - $2^{n-r-1} - 1$ nonlinear mult.

# Knuth-Eve's Method for SBox Evaluation

$$\begin{aligned} \mathrm{S}(x) &= a_0 + a_4 X + a_8 X^2 + \ldots + (a_2 + a_6 X + \ldots) \cdot x^2 + \\ & (a_1 + a_5 X + a_9 X^2 + \ldots + (a_3 + a_7 X + \ldots) \cdot x^2) \cdot x \end{aligned}$$

where $X = x^4$

- Nonlinear mult. : $1+2+\cdots+2^{r-1} = 2^r - 1$
- and the evaluation of $2^{r+1}$ polynomials in $X = x^{2^r}$
  - we derive $X^j$ for $j < 2^{n-r}$
  - $2^{n-r-1} - 1$ nonlinear mult.

$$\Rightarrow 2^{n-r-1} + 2^r - 2 \ \text{ nonlinear mult.}$$

# Coron-Roy-Viveks (CRV) Method

# Coron-Roy-Viveks (CRV) Method

- Build $s$ cyclotomic classes $C_i$ s.t. $\{X^i; a_i \neq 0\} \subseteq C + C$ with $C = \bigcup_i C_i$.
  - define $\mathcal{P}$ as the set of polynomials with monomials in $C$ only.


SAFRAN Morpho

# Coron-Roy-Viveks (CRV) Method

- Build $s$ cyclotomic classes $C_i$ s.t. $\{X^i; a_i \neq 0\} \subseteq C + C$ with $C = \bigcup_i C_i$.
  - define $\mathcal{P}$ as the set of polynomials with monomials in $C$ only.
- Fix $t$ polynomials $q_i(x) \in \mathcal{P}$ and find $t+1$ polynomials $p_i(x) \in \mathcal{P}$ s.t.

$$S(x) = \sum_{i=1}^{t} p_i(x) \times q_i(x) + p_t(x)$$

## Coron-Roy-Viveks (CRV) Method

- Build $s$ cyclotomic classes $C_i$ s.t. $\{X^i; a_i \neq 0\} \subseteq C + C$ with $C = \bigcup_i C_i$.
  - define $\mathcal{P}$ as the set of polynomials with monomials in $C$ only.
- Fix $t$ polynomials $q_i(x) \in \mathcal{P}$ and find $t+1$ polynomials $p_i(x) \in \mathcal{P}$ s.t.

$$S(x) = \sum_{i=1}^{t} p_i(x) \times q_i(x) + p_t(x)$$

- Number of non-linear multiplications $N = s + t - 2$ s.t.:

$$N \geqslant 2\sqrt{\frac{2^n}{n}} \ .$$


SAFRAN Morpho

## Coron-Roy-Viveks (CRV) Method

- Build $s$ cyclotomic classes $C_i$ s.t. $\{X^i; a_i \neq 0\} \subseteq C + C$ with $C = \bigcup_i C_i$.
  - define $\mathcal{P}$ as the set of polynomials with monomials in $C$ only.
- Fix $t$ polynomials $q_i(x) \in \mathcal{P}$ and find $t+1$ polynomials $p_i(x) \in \mathcal{P}$ s.t.

$$S(x) = \sum_{i=1}^{t} p_i(x) \times q_i(x) + p_t(x)$$

- Number of non-linear multiplications $N = s + t - 2$ s.t.:

$$N \geqslant 2\sqrt{\frac{2^n}{n}} \ .$$

- Note: there always exists a polynomial whose evaluation requires at least $\sqrt{\frac{2^n}{n}} - 2$ non-linear multiplications CoronRoyVivek14.

- CRV's method amounts to solve the linear system:

$$\begin{cases} \sum_{i=1}^{t} p_i(e_1) \times q_i(e_1) & + \quad p_{t+1}(e_1) & = \quad S(e_1) \\ \sum_{i=1}^{t} p_i(e_2) \times q_i(e_2) & + \quad p_{t+1}(e_2) & = \quad S(e_2) \\ \vdots & & \\ \sum_{i=1}^{t} p_i(e_{2^n}) \times q_i(e_{2^n}) & + \quad p_{t+1}(e_{2^n}) & = \quad S(e_{2^n}) \end{cases}$$

with (around) $(t+1) \times n \times s$ unknowns and $2^n$ equations.

- CRV's method amounts to solve the linear system:

$$\begin{cases} \sum_{i=1}^{t} p_i(e_1) \times q_i(e_1) & + & p_{t+1}(e_1) & = & S(e_1) \\ \sum_{i=1}^{t} p_i(e_2) \times q_i(e_2) & + & p_{t+1}(e_2) & = & S(e_2) \\ \vdots \\ \sum_{i=1}^{t} p_i(e_{2^n}) \times q_i(e_{2^n}) & + & p_{t+1}(e_{2^n}) & = & S(e_{2^n}) \end{cases}$$

with (around) $(t+1) \times n \times s$ unknowns and $2^n$ equations.

- Necessary condition:

$$(t+1) \times n \times s \geqslant 2^n \quad .$$

- In practice, the condition was sufficient.


SAFRAN Morpho

## Asymptotic Complexities

- Cyclotomic Method: $O(\frac{2^n-1}{n}d^2)$.
- Knuth-Eve's Method: $O(2^{n/2}d^2)$.
- Coron-Roy-Vivek's Method (heuristic): $O(\sqrt{\frac{2^n}{n}}d^2)$

## Practical (worst case) Complexities

| $n$ | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|
| Knuth-Eve | 3 | 5 | 11 | 17 | 33 | 52 | 105 |
| Cyclotomic | 4 | 6 | 10 | 14 | 22 | 30 | 46 |
| CRV | 2 | 4 | 5 | 7 | 10 | 14 | 19 |

algebraic degree of a polynomial: greatest Hamming weight of the power of its monomials (with non-zero coefficients).

algebraic degree of a polynomial: greatest Hamming weight of the power of its monomials (with non-zero coefficients).

## Secure Evaluation of a Polynomial $h(x)$ with algebraic degree $s$

$h(x)$ a polynomial with algebraic degree $s$

$$h\left(\sum_{i=1}^{d} a_i\right) = \sum_{j\leq s} c_j \sum_{\substack{I\subseteq[1;d]\\|I|=j}} h\left(\sum_{i\in I} a_i\right) \quad ,$$

where $c_j$ are constant binary coefficients.


SAFRAN Morpho

algebraic degree of a polynomial: greatest Hamming weight of the power of its monomials (with non-zero coefficients).

## Secure Evaluation of a Polynomial $h(x)$ with algebraic degree $s$

$h(x)$ a polynomial with algebraic degree $s$

$$h\left(\sum_{i=1}^{d} a_i\right) = \sum_{j \leq s} c_j \sum_{\substack{I \subseteq [1;d] \\ |I| = j}} h\left(\sum_{i \in I} a_i\right) ,$$

where $c_j$ are constant binary coefficients.

Hence: securing at order $d$ reduces to securing at order $s$.

algebraic degree of a polynomial: greatest Hamming weight of the power of its monomials (with non-zero coefficients).

## Secure Evaluation of a Polynomial $h(x)$ with algebraic degree $s$

$h(x)$ a polynomial with algebraic degree $s$

$$h\Big(\sum_{i=1}^{d} a_i\Big) = \sum_{j\leq s} c_j \sum_{\substack{I\subseteq[1;d]\\|I|=j}} h\Big(\sum_{i\in I} a_i\Big) \ ,$$

where $c_j$ are constant binary coefficients.

Hence: securing at order $d$ reduces to securing at order $s$.
Leads to the secure evaluation methods with complexity $O(d^s)$.

algebraic degree of a polynomial: greatest Hamming weight of the power of its monomials (with non-zero coefficients).

## Secure Evaluation of a Polynomial $h(x)$ with algebraic degree $s$

$h(x)$ a polynomial with algebraic degree $s$

$$h\left(\sum_{i=1}^{d} a_i\right) = \sum_{j \leq s} c_j \sum_{\substack{I \subseteq [1;d] \\ |I|=j}} h\left(\sum_{i \in I} a_i\right) ,$$

where $c_j$ are constant binary coefficients.

Hence: securing at order $d$ reduces to securing at order $s$.
Leads to the secure evaluation methods with complexity $O(d^s)$.
Example: securing degree-2 functions is as complex as securing a multiplication (with ISW scheme).

algebraic degree of a polynomial: greatest Hamming weight of the power of its monomials (with non-zero coefficients).

## Secure Evaluation of a Polynomial $h(x)$ with algebraic degree $s$

$h(x)$ a polynomial with algebraic degree $s$

$$h\left(\sum_{i=1}^{d} a_i\right) = \sum_{j \leq s} c_j \sum_{\substack{I \subseteq [1;d] \\ |I|=j}} h\left(\sum_{i \in I} a_i\right) ,$$

where $c_j$ are constant binary coefficients.

Hence: securing at order $d$ reduces to securing at order $s$.
Leads to the secure evaluation methods with complexity $O(d^s)$.
Example: securing degree-2 functions is as complex as securing a multiplication (with ISW scheme).
Efficient (compared to SoA) for small $s$ or $n \ll d^s$.

# Conclusions

## Conclusions

- We need algorithmic countermeasures with formal proof of resistance.

## Conclusions

- We need algorithmic countermeasures with formal proof of resistance.
- We need formal models fitting the physical reality of devices AND enabling relatively simple proofs.

## Conclusions

- We need algorithmic countermeasures with formal proof of resistance.
- We need formal models fitting the physical reality of devices AND enabling relatively simple proofs.
- Countermeasures must be efficient AND resistant against powerful adversaries.

## Conclusions

- We need algorithmic countermeasures with formal proof of resistance.
- We need formal models fitting the physical reality of devices AND enabling relatively simple proofs.
- Countermeasures must be efficient AND resistant against powerful adversaries.
- Links with many other rich fields: ECC, MPC, efficient processing in short characteristic, etc.

## Conclusions

- We need algorithmic countermeasures with formal proof of resistance.
- We need formal models fitting the physical reality of devices AND enabling relatively simple proofs.
- Countermeasures must be efficient AND resistant against powerful adversaries.
- Links with many other rich fields: ECC, MPC, efficient processing in short characteristic, etc.
- Many open issues...

## Conclusions

- We need algorithmic countermeasures with formal proof of resistance.
- We need formal models fitting the physical reality of devices AND enabling relatively simple proofs.
- Countermeasures must be efficient AND resistant against powerful adversaries.
- Links with many other rich fields: ECC, MPC, efficient processing in short characteristic, etc.
- Many open issues...
    - Improve proof techniques (automatize them?)

# Conclusions

- We need algorithmic countermeasures with formal proof of resistance.
- We need formal models fitting the physical reality of devices AND enabling relatively simple proofs.
- Countermeasures must be efficient AND resistant against powerful adversaries.
- Links with many other rich fields: ECC, MPC, efficient processing in short characteristic, etc.
- Many open issues...
  - Improve proof techniques (automatize them?)
  - Improve existing techniques / adapt them to the SCA context (*e.g.* decrease the complexity to securely process the multiplication)


SAFRAN Morpho

## Conclusions

- We need algorithmic countermeasures with formal proof of resistance.
- We need formal models fitting the physical reality of devices AND enabling relatively simple proofs.
- Countermeasures must be efficient AND resistant against powerful adversaries.
- Links with many other rich fields: ECC, MPC, efficient processing in short characteristic, etc.
- Many open issues...
  - Improve proof techniques (automatize them?)
  - Improve existing techniques / adapt them to the SCA context (*e.g.* decrease the complexity to securely process the multiplication)
  - Reduce the randomness consumption of existing techniques

## Conclusions

- We need algorithmic countermeasures with formal proof of resistance.
- We need formal models fitting the physical reality of devices AND enabling relatively simple proofs.
- Countermeasures must be efficient AND resistant against powerful adversaries.
- Links with many other rich fields: ECC, MPC, efficient processing in short characteristic, etc.
- Many open issues...
  - Improve proof techniques (automatize them?)
  - Improve existing techniques / adapt them to the SCA context (*e.g.* decrease the complexity to securely process the multiplication)
  - Reduce the randomness consumption of existing techniques
  - Find Efficient Evaluation methods

# Conclusions

- We need algorithmic countermeasures with formal proof of resistance.
- We need formal models fitting the physical reality of devices AND enabling relatively simple proofs.
- Countermeasures must be efficient AND resistant against powerful adversaries.
- Links with many other rich fields: ECC, MPC, efficient processing in short characteristic, etc.
- Many open issues...
  - Improve proof techniques (automatize them?)
  - Improve existing techniques / adapt them to the SCA context (*e.g.* decrease the complexity to securely process the multiplication)
  - Reduce the randomness consumption of existing techniques
  - Find Efficient Evaluation methods
  - For TI, find generic constructions secure at order $d$

Thank you for your attention!
Questions/Remarks?