Protecting AES with Shamir's Secret Sharing Scheme

Louis Goubin and Ange Martinelli

CHES 2011, September 29, Nara Japan

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES
THALES

Outline

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

THALES

Outline

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES
THALES

Context

- Block ciphers are vulnerable to SCA.
- d-th order boolean masking is the most implemented.
- Improve security of masking schemes against SCA:

Context

- Block ciphers are vulnerable to SCA.
- d-th order boolean masking is the most implemented.
- Improve security of masking schemes against SCA:
  - Increase the order d of the masking.
    - $+$: Security of dO-masking grows exponentially with d due to intrinsic leakage noise [ChariJutlaRaoRohatgi99]
    - $-$: Efficiency of dO-masking quickly decreases with d

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

THALES

Context

- Block ciphers are vulnerable to SCA.
- d-th order boolean masking is the most implemented.
- Improve security of masking schemes against SCA:
  - Increase the order d of the masking.
    - $+$: Security of dO-masking grows exponentially with d due to intrinsic leakage noise [ChariJutlaRaoRohatgi99]
    - $-$: Efficiency of dO-masking quickly decreases with d
  - Complicate the relation between the masks and the masked variable.

$\Rightarrow$ this work

Shamir's secret sharing scheme

- a_0 secret.
- P is a polynomial s.t.
  $P(x) = a_d \cdot x^d + a_{d-1} \cdot x^{d-1} + \cdots + a_1 \cdot x + a_0$
- Each user i has $(x_i, y_i = P(x_i))_{x_i \neq 0}$
- Reconstruction:

$$a_0 = \sum_{0}^{d} y_i \cdot \beta_i$$

where $\beta_i = \displaystyle\prod_{j=0, j\neq i}^{d} \frac{-x_j}{x_i - x_j}$.

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

THALES

Outline

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

THALES

d-th order masking scheme

- Each sensitive variable b is shared as

$$(x_i, y_i)_{i=0..d}$$

- We only manipulate pairs (x_i, y_i)
- The cipher text c verifies:

$$c = \sum_{0}^{d} y_i^{final} \cdot \beta_i$$

where (x_i, y_i^{final}) is the output of the last round.

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

THALES

Masking linear layers

- AddRoundKey, ShiftRows, MixColumns computed using linear operations.
- Let $u \in \mathrm{GF}(256)$ shared as $(x_i, u_i)_{i=0..d}$, $v \in \mathrm{GF}(256)$

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

THALES

Masking linear layers

- AddRoundKey, ShiftRows, MixColumns computed using linear operations.
- Let $u \in GF(256)$ shared as $(x_i, u_i)_{i=0..d}$, $v \in GF(256)$

$$b \oplus v \quad \rightarrow (x_i', y_i') = (x_i, y_i \oplus v)$$

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES
THALES

Masking linear layers

- AddRoundKey, ShiftRows, MixColumns computed using linear operations.
- Let $u \in \mathrm{GF}(256)$ shared as $(x_i, u_i)_{i=0..d}$, $v \in \mathrm{GF}(256)$

$$\begin{aligned} b \oplus v &\quad \rightarrow (x_i', y_i') = (x_i, y_i \oplus v) \\ b \oplus u &\quad \rightarrow (x_i', y_i') = (x_i, y_i \oplus u_i) \end{aligned}$$

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES
THALES

Masking linear layers

- AddRoundKey, ShiftRows, MixColumns computed using linear operations.
- Let $u \in GF(256)$ shared as $(x_i, u_i)_{i=0..d}$, $v \in GF(256)$

$$\begin{array}{lll} b \oplus v & \rightarrow (x_i', y_i') = (x_i, y_i \oplus v) \\ b \oplus u & \rightarrow (x_i', y_i') = (x_i, y_i \oplus u_i) \\ b \cdot v & \rightarrow (x_i', y_i') = (x_i, y_i \cdot v) \end{array}$$

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

THALES

Masking AES Sbox

- SubByte can be derived from [RivainProuff10] using $x^{-1} = x^{254}$.
- **Secure square:** linear over GF(256):

$$b^2 \to (x_i', y_i') = (x_i^2, {y_i}^2)$$

- $x_i' \neq x_i \Rightarrow$ need a `RefreshMasks` operation.
- **Secure multiplication:**
  product of 2 degree d polynomials $\Rightarrow$ polynomial of degree $2d$

RefreshMasks operation

- Derived from [Ben-OrGoldwasserWigderson88]
  - Sharing each share
  - Reconstructing original value

Algorithm 1 `RefreshMasks`

INPUT: Shared representation of b, $(\alpha_i, y_i)_{i=0..d}$, chosen $(x_i)_{i=0..d}$, t such that $\alpha_i = x_i^{2^t}$
OUTPUT: Shared representation of b, $(x_i, y_i')_{i=0..d}$

1. **for** $i = 0$ **to** d **do**
2. $\quad \beta_i' \leftarrow \beta_i^{2^t}$
3. $\quad$ Share y_i in $(x_j, z_{i_j})_{j=0..d}$
4. **for** $i = 0$ **to** d **do**
5. $\quad (x_i, y_i') \leftarrow \left(x_i, \sum_{j=0}^{d} \beta_j' \cdot z_{j_i} \right)$
6. `return` $(x_i, y_i')_{i=0..d}$

Masking the field multiplication

- Two possibilities:
  - Adapt SMC algorithm of [Ben-OrGoldwasserWigderson88][1]
    $\Rightarrow$ huge complexity
  - Provide a new algorithm exploiting the SCA context
    $\Rightarrow$ loss of known security proof

[1] see full version at http://eprint.iacr.org/2011/516.pdf

Masking the field multiplication

- Two possibilities:
  - Adapt SMC algorithm of [Ben-OrGoldwasserWigderson88][1]
    $\Rightarrow$ huge complexity
  - Provide a new algorithm exploiting the SCA context
    $\Rightarrow$ loss of known security proof
    $\Rightarrow$ our choice.
- Idea : truncate the degree $2d$ polynomial to degree d

[1] see full version at http://eprint.iacr.org/2011/516.pdf

Masking the field multiplication

- Let $\beta_{j,k}(x)$ be defined as:
  - $\beta_j(x) = \prod_{l=0, l\neq j}^{d} \frac{x - x_l}{x_j - x_l}$.
  - $\beta_j(x) \cdot \beta_k(x) = \alpha_{2d} x^{2d} + \cdots + \alpha_d x^d + \cdots + \alpha_1 x + \alpha_0$
  - Then $\beta_{j,k}(x) = \beta_{k,j}(x) = \alpha_d x^d + \cdots + \alpha_1 x + \alpha_0$.
- $P(x) = \sum_{j=0}^{d} \sum_{k=0}^{d} y_j \cdot u_k \cdot \beta_{j,k}(x)$ verifies:
  - $degree(P) = d$
  - $P(0) = b \cdot u$
  - $\forall x \in \{x_i\}_{i=0..d},\ P(x_i) = y_i'$

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

THALES

Masking the field multiplication

Algorithm 2 Share multiplication SecMult

INPUT: Shared representation of b, $(x_i, y_i)_{i=0..d}$ and u, $(x_i, u_i)_{i=0..d}$
OUTPUT: Shares $(x_i, y'_i)_{i=0..d}$ representing the product of b and u

1. **for** $j = 0$ **to** d **do**
2. **for** $k = 0$ **to** d **do**
3. $z_{j,k} \leftarrow y_j \cdot u_k$
4. **for** $i = 0$ **to** d **do**
5. $(x_i, y'_i) \leftarrow \left(x_i, \left(\sum_{j=1}^{d} \sum_{0 \le k < j} (z_{j,k} \oplus z_{k,j}) \cdot \beta_{j,k}(x_i) \right) + \sum_{j=0}^{d} z_{j,j} \cdot \beta_{j,j}(x_i) \right)$
6. `return` $(x_i, y'_i)_{i=0..d}$

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES
THALES

Intuition of security

- Intuitively we have
  - One needs at least $d+1$ shares to define a polynomial of degree d,
  - $\beta_{j,k}(x_i)$ is independent of any secret,
  - $y_j \cdot u_k$ does not leak more information on b (resp. u) than the knowledge of y_j (resp. u_k),
- No easy security proof for `SecMult` a order d: open work.

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

THALES

Outline

Complexity of the inversion

Table: Complexity of inversion algorithms

order	XORs	multiplications	$\hat{}2^j$	Rand. bytes	RAM (bytes)
$O1$-SSS	36	54	14	6	20
$O2$-SSS	150	165	21	18	33
Od-SSS	$7d^3 + 18d^2 + 11d$	$5d^3 + 18d^2 + 22d + 9$	$7(d+1)$	$3d^2 + 3d$	$d^2 + 10d + 9$
$O1$-Bool.	20	16	6	6	7
$O2$-Bool.	56	36	9	16	12
$O3$-Bool.	108	64	12	20	18
$O4$-Bool.	176	100	15	48	25
Od-Bool.	$7d^2 + 12d$	$4d^2 + 8d + 4$	$3(d+1)$	$2d^2 + 4d$	$\frac{1}{2}d^2 + \frac{7}{2}d + 3$

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

THALES

Overall complexity

- Log/alog tables based multiplication

Table: Complexity of cipher implementations

Masking	XORs/ANDs	Table look-ups	Random bits	RAM (bits)	ROM (bits)
1O boolean	17640	16144	16896	312	6128
2O boolean	37800	32272	46080	352	6128
3O boolean	65640	54160	87552	400	6128
1O SSS	31760	37296	16240	400	6128

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES
THALES

Overall complexity

- Log/alog tables based multiplication

Table: Complexity of cipher implementations

Masking	XORs/ANDs	Table look-ups	Random bits	RAM (bits)	ROM (bits)
1O boolean	17640	16144	16896	312	6128
2O boolean	37800	32272	46080	352	6128
3O boolean	65640	54160	87552	400	6128
1O SSS	31760	37296	16240	400	6128

Outline

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

THALES

Leakage model

- Each sensitive variable Z manipulated as

$$U_i = (x_i, P(x_i))_{i=0..d}$$

  where $P(0) = Z$
- Hamming weight model with additional Gaussian noise

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES
THALES

Leakage model

- Each sensitive variable Z manipulated as

$$U_i = (x_i, P(x_i))_{i=0..d}$$

  where $P(0) = Z$
- Hamming weight model with additional Gaussian noise
- No d-th order leakage thanks to Shamir's sharing scheme

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES
THALES

Leakage model

- Each sensitive variable Z manipulated as

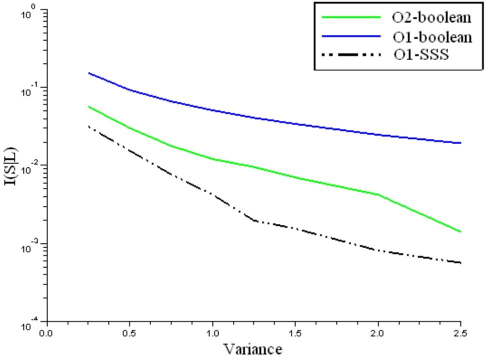
$$U_i = (x_i, P(x_i))_{i=0..d}$$

  where $P(0) = Z$
- Hamming weight model with additional Gaussian noise
- No d-th order leakage thanks to Shamir's sharing scheme
- What about $(d+1)$-th order leakage ?

Information Theoretic Analysis

- Follows the approach of [StandaertMalkingYung09]
  - Mutual information evaluation

Figure: Mutual Information values with respect to σ^2 (logarithmic scale).



UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES
THALES

Higher-Order DPA Evaluation

- Optimal correlation [ProuffRivainBévan09]:

$$\rho = \sqrt{\frac{\mathrm{Var}\left[\mathrm{E}\left[\prod_i \overline{L}_i | Z = z\right]\right]}{\mathrm{Var}\left[\prod_i \overline{L}_i\right]}}$$

- Boolean masking [RivainProuffDoget09]:

$$\rho_{\text{bool}} = (-1)^d \frac{\sqrt{n}}{(n + 4\sigma^2)^{\frac{d+1}{2}}}$$

- 1O-SSS masking:

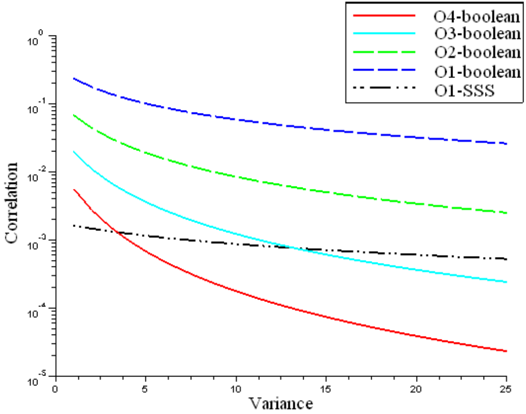
$$\rho_{\text{SSS}} = \sqrt{\frac{n^3 \cdot (2^{n+1} - 4^n - 1)}{\alpha_2 \cdot \sigma^4 + \alpha_1 \cdot \sigma^2 + \alpha_0}}$$

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

THALES

Higher-Order DPA Evaluation

Figure: Correlation values with respect to σ^2 (logarithmic scale).



UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

THALES

Attack simulations

Table: Number of leakage measurements for a 90% success rate.

Attack \ SNR	$+\infty$	1	1/2	1/5	1/10
Attacks against Boolean Masking					
2O-DPA on 1O Boolean Masking	150	500	1500	6000	20 000
2O-MIA on 1O Boolean Masking	100	5000	15 000	50 000	160 000
3O-DPA on 2O Boolean Masking	1500	9000	35 000	280 000	$> 10^6$
3O-MIA on 2O Boolean Masking	160	160 000	650 000	$> 10^6$	$> 10^6$
Attacks against SSS Masking					
2O-DPA on 1O SSS Masking	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$
2O-MIA on 1O SSS Masking	500 000	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$
3O-DPA on 2O SSS Masking	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$
3O-MIA on 2O SSS Masking	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$	$> 10^6$

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

THALES

Outline

UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

THALES

Conclusion

- New alternative to higher order boolean masking
- Good complexity-security trade-off for high level security:
  - $1O$-SSS complexity $\approx 2O$ boolean
  - $1O$-SSS security $\approx 3O$ boolean
- Open work:
  - Security proof for `SecMult`
  - Try other secret sharing as masking scheme

End of the talk

Thank you for your attention

Questions / comments ?