Power analysis of degree-2 round functions

Gilles Van Assche[1]

[1]STMicroelectronics

SCA workshop
Šibenik, Croatia, June 2019

Based on joint work with Guido Bertoni, Joan Daemen,
Nicolas Debande, Thanh-Ha Le, Michaël Peeters, Ronny Van Keer

Outline

Outline

Typical ingredients for a side-channel attack

$$\texttt{ByteSub}(M \oplus K)$$

What do the following have in common?

- Ascon
- Gimli
- KECCAK-*p*
- XOODOO

Possible answers:

- They have 2 syllables
- They are permutations (or permutation-based schemes)
- They can be used in some duplex-based keyed mode
- Their round function has degree 2

What do the following have in common?

- Ascon
- Gimli
- KECCAK-*p*
- XOODOO

Possible answers:

- They have 2 syllables
- They are permutations (or permutation-based schemes)
- They can be used in some duplex-based keyed mode
- Their round function has degree 2

What do the following have in common?

- Ascon
- Gimli
- KECCAK-*p*
- XOODOO

Possible answers:

- They have 2 syllables
- They are permutations (or permutation-based schemes)
- They can be used in some duplex-based keyed mode
- Their round function has degree 2

What do the following have in common?

- Ascon
- Gimli
- KECCAK-*p*
- XOODOO

Possible answers:

- They have 2 syllables
- They are permutations (or permutation-based schemes)
- They can be used in some duplex-based keyed mode
- Their round function has degree 2

What do the following have in common?

- Ascon
- Gimli
- KECCAK-*p*
- XOODOO

Possible answers:

- They have 2 syllables
- They are permutations (or permutation-based schemes)
- They can be used in some duplex-based keyed mode
- Their round function has degree 2

What do the following have in common?

- Ascon
- Gimli
- KECCAK-*p*
- XOODOO

Possible answers:

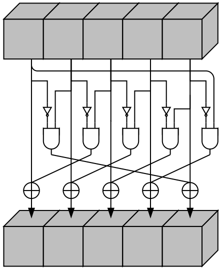
- They have 2 syllables
- They are permutations (or permutation-based schemes)
- They can be used in some duplex-based **keyed** mode
- **Their round function has degree 2**

The KECCAK-p round function

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$$

- Linear part λ followed by non-linear part χ
- $\lambda = \pi \circ \rho \circ \theta$: mixing followed by bit transposition
- χ: simple mapping operating on rows:

$$b_i \leftarrow b_i + (b_{i+1} + 1) b_{i+2}$$



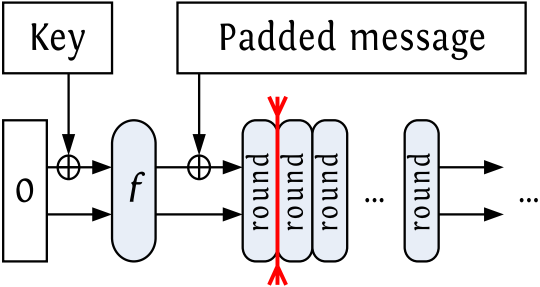
In general

Quadratic form for output bit i:

$$R_i(s) = \mathsf{s}^{\mathsf{T}}\, \mathbf{A}_i\, s + \text{constants}_i$$

Outline

Attacking keyed sponge functions / duplex objects



1. Attack the first round after absorbing known input bits
2. Compute backward by inverting the permutation

A model of the power consumption

Consumption at any time instance can be modeled as

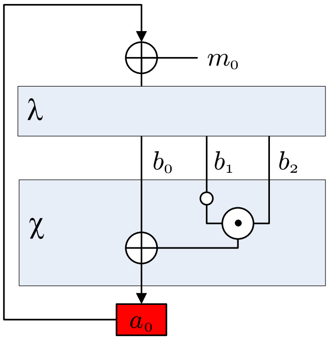
$$P = \sum_i T_i[d_i]$$

- d_i: Boolean variables that express *activity*
  - bit 1 in a given register or gate output at some stage
  - flipping of a specific register or gate output at some stage
- $T_i[0]$ and $T_i[1]$: stochastic variables

Simplified model

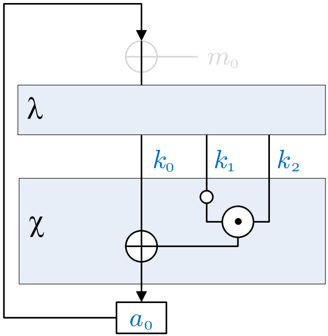
$$P = \alpha + \sum_i (-1)^{d_i}$$

DPA applied to an unprotected implementation



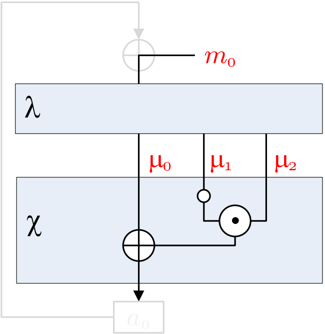
- Leakage exploited: switching consumption of register bit 0
- Value switches from a_0 to $b_0 + (b_1 + 1)b_2$
- Activity equation: $d = a_0 + b_0 + (b_1 + 1)b_2$

DPA applied to an unprotected implementation



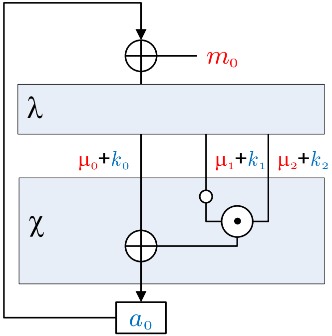
- Take the case $M = 0$
- We call K the input of χ-block if $M = 0$
- K will be our target

DPA applied to an unprotected implementation



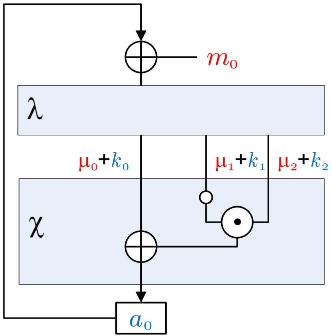
- We call the effect of M at input of χ: μ
- $\mu = \lambda(M)$
- Linearity of λ: $B = K + \lambda(M)$

DPA applied to an unprotected implementation



- $d = a_0 + k_0 + (k_1 + 1)(k_2) + \mu_0 + (\mu_1 + 1)\mu_2 + k_1\mu_2 + k_2\mu_1$
- Fact: value of $q = a_0 + k_0 + (k_1 + 1)k_2$ is same for all traces
- Let M_0: traces with $d = q$ and M_1: $d = q + 1$

DPA applied to an unprotected implementation



- Selection: $s(M, K^*) = \mu_0 + (\mu_1 + 1)\mu_2 + k_1^*\mu_2 + k_2^*\mu_1$
- Values of μ_1 and μ_2 computed from M
- Hypothesis has two bits only: k_1^* and k_2^*

DPA applied to an unprotected implementation

- Correct hypothesis K
  - traces in M_0: $d = q$
  - traces in M_1: $d = q + 1$
- Incorrect hypothesis $K^* = K + \Delta$
  - trace in M_0: $d = q + \mu_1\delta_2 + \mu_2\delta_1$
  - trace in M_1: $d = q + \mu_1\delta_2 + \mu_2\delta_1 + 1$
- Remember: $\mu = \lambda(M)$
  - random inputs M lead to random μ_1 and μ_2
  - Incorrect hypothesis: d uncorrelated with $\{M_0, M_1\}$

In general

Quadratic form for output bit i:

$$R_i(s) = \mathbf{s}^{\mathsf{T}} \mathbf{A}_i \, s + \text{constants}_i$$

After the first round after absorbing the message:

$$d_i(M, K) = \alpha_i(M) + \beta_i(K) + K^{\mathsf{T}}\Gamma_i M$$

Selection function:

$$s_i(M, K^*) = \alpha_i(M) + K^{\mathsf{T}}\Gamma_i M$$

(In)correct guess:

$$s_i(M, K + \epsilon) = s_i(M, K) + \epsilon^{\mathsf{T}}\Gamma_i M$$

[Bertoni, Daemen, Debande, Le, Peeters, Van Assche, HASP 2012]

In general

Quadratic form for output bit i:

$$R_i(s) = \mathbf{s}^{\mathsf{T}}\, \mathbf{A}_i\, s + \text{constants}_i$$

After the first round after absorbing the message:

$$d_i(M, K) = \alpha_i(M) + \beta_i(K) + K^{\mathsf{T}}\Gamma_i M$$

Selection function:

$$s_i(M, K^*) = \alpha_i(M) + K^{\mathsf{T}}\Gamma_i M$$

(In)correct guess:

$$s_i(M, K + \epsilon) = s_i(M, K) + \epsilon^{\mathsf{T}}\Gamma_i M$$

[Bertoni, Daemen, Debande, Le, Peeters, Van Assche, HASP 2012]

In general

Quadratic form for output bit i:

$$R_i(s) = \mathbf{s}^\mathsf{T}\, \mathbf{A}_i\, s + \text{constants}_i$$

After the first round after absorbing the message:

$$d_i(M, K) = \alpha_i(M) + \beta_i(K) + K^\mathsf{T}\Gamma_i M$$

Selection function:

$$s_i(M, K^*) = \alpha_i(M) + K^\mathsf{T}\Gamma_i M$$

(In)correct guess:

$$s_i(M, K + \epsilon) = s_i(M, K) + \epsilon^\mathsf{T}\Gamma_i M$$

[Bertoni, Daemen, Debande, Le, Peeters, Van Assche, HASP 2012]

In general

Quadratic form for output bit i:

$$R_i(s) = \mathbf{s}^\mathsf{T}\,\mathbf{A}_i\,s + \text{constants}_i$$

After the first round after absorbing the message:

$$d_i(M,K) = \alpha_i(M) + \beta_i(K) + K^\mathsf{T}\Gamma_i M$$

Selection function:

$$s_i(M,K^*) = \alpha_i(M) + K^\mathsf{T}\Gamma_i M$$

(In)correct guess:

$$s_i(M,K+\epsilon) = s_i(M,K) + \epsilon^\mathsf{T}\Gamma_i M$$

[Bertoni, Daemen, Debande, Le, Peeters, Van Assche, HASP 2012]

Outline

Secret sharing

- Countermeasure at algorithmic level:
  - Split variables in *random* shares: $x = a \oplus b \oplus \dots$
  - Keep computed variables *independent* from *native* variables
  - Protection against n-th order DPA: at least $n + 1$ shares

Software: two-share masking

- $\chi : x_i \leftarrow x_i + (x_{i+1} + 1)x_{i+2}$ becomes:

$$\begin{aligned} a_i &\leftarrow a_i + (a_{i+1} + 1)a_{i+2} + a_{i+1}b_{i+2} \\ b_i &\leftarrow b_i + (b_{i+1} + 1)b_{i+2} + b_{i+1}a_{i+2} \end{aligned}$$

- Independence from native variables, if:
  - we compute left-to-right
  - we avoid leakage in register or bus transitions
- λ becomes:

$$\begin{aligned} a &\leftarrow \lambda(a) \\ b &\leftarrow \lambda(b) \end{aligned}$$

Software: two-share masking (faster)

- Making it faster!
- χ becomes:

$$\begin{aligned} a_i &\leftarrow a_i + (a_{i+1} + 1)a_{i+2} + a_{i+1}b_{i+2} + (b_{i+1} + 1)b_{i+2} + b_{i+1}a_{i+2} \\ b_i &\leftarrow b_i \end{aligned}$$

- Precompute $R = b + \lambda(b)$
- λ becomes:

$$\begin{aligned} a &\leftarrow \lambda(a) + R \\ b &\leftarrow b \end{aligned}$$

Software: two-share masking (faster)

- Making it faster!
- χ becomes:

$$a_i \leftarrow a_i + (a_{i+1} + 1)a_{i+2} + a_{i+1}b_{i+2} + (b_{i+1} + 1)b_{i+2} + b_{i+1}a_{i+2}$$

- Precompute $R = b + \lambda(b)$
- λ becomes:

$$a \leftarrow \lambda(a) + R$$

Hardware: two shares are not enough

- Unknown order in combinatorial logic!

$$a_i \leftarrow a_i + (a_{i+1} + 1)a_{i+2} + a_{i+1}b_{i+2}$$

Using a threshold secret-sharing scheme

- Idea: incomplete computations only
  - Each circuit does not leak anything
    [Nikova, Rijmen, Schläffer 2008]
- Number of shares: at least 1 + algebraic degree

  3 shares are needed for χ
- Glitches as second-order effect
  - A glitch can leak about two shares, say, $a + b$
  - Another part can leak c
  - $\Rightarrow$ as if two shares only!

Using a threshold secret-sharing scheme

- Idea: incomplete computations only
  - Each circuit does not leak anything
    [Nikova, Rijmen, Schläffer 2008]
- Number of shares: at least 1 + algebraic degree

3 shares are needed for χ

- Glitches as second-order effect
  - A glitch can leak about two shares, say, $a + b$
  - Another part can leak c
  - $\Rightarrow$ as if two shares only!

Using a threshold secret-sharing scheme

- Idea: incomplete computations only
  - Each circuit does not leak anything
    [Nikova, Rijmen, Schläffer 2008]
- Number of shares: at least 1 + algebraic degree

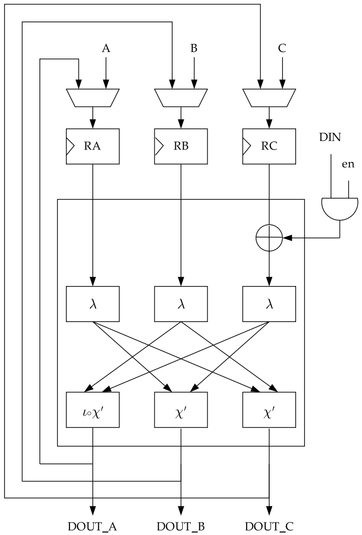
  3 shares are needed for χ
- Glitches as second-order effect
  - A glitch can leak about two shares, say, $a + b$
  - Another part can leak c
  - $\Rightarrow$ as if two shares only!

Three-share masking for χ

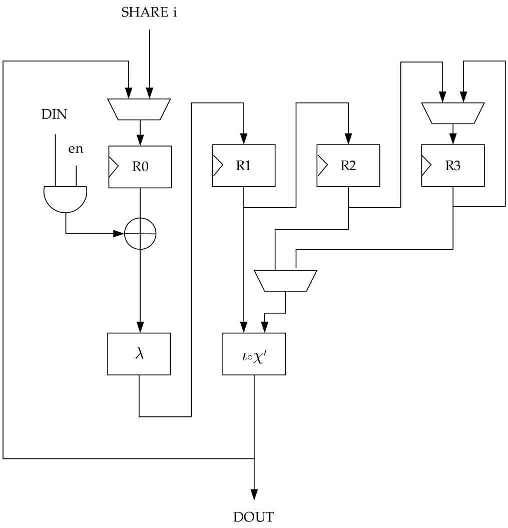
- Implementing χ in three shares:

$$
\begin{aligned}
a_i &\leftarrow b_i + (b_{i+1} + 1)b_{i+2} + b_{i+1}c_{i+2} + c_{i+1}b_{i+2} \\
b_i &\leftarrow c_i + (c_{i+1} + 1)c_{i+2} + c_{i+1}a_{i+2} + a_{i+1}c_{i+2} \\
c_i &\leftarrow a_i + (a_{i+1} + 1)a_{i+2} + a_{i+1}b_{i+2} + b_{i+1}a_{i+2}
\end{aligned}
$$

One-cycle round architecture



Three-cycle round architecture



Any questions?

Thanks for your attention!

https://keccak.team/