Algebraic Side-Channel Attacks on Masked Implementations of AES

Luk Bettale   Emmanuelle Dottax   Mailody Ramphort
Idemia Crypto & Security Lab

IDEMIA

SECRYPT 2018

Physical Security

Cryptographic algorithm security

Classical security	Physical security
Algorithm: abstract mathematical object (**black box**)	Algorithm: program running on given device (**gray box**)
Only inputs and outputs are available.	Implementation-specific characteristics might leak information.

Physical Security

Cryptographic algorithm security

Classical security	Physical security
Algorithm: abstract mathematical object (**black box**)	Algorithm: program running on given device (**gray box**)
Only inputs and outputs are available.	Implementation-specific characteristics might leak information.
⇝ Classical cryptanalysis	⇝ Side-channel cryptanalysis

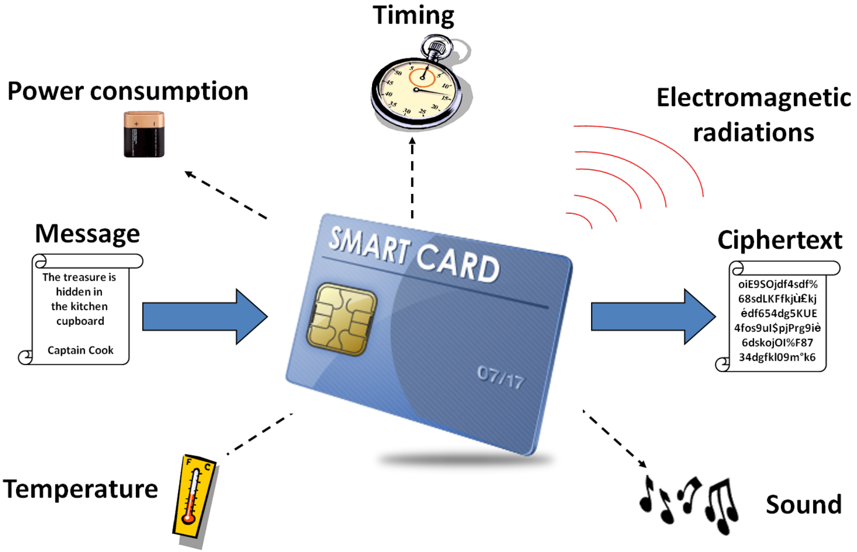
Physical Security

Cryptographic algorithm security

Classical security	Physical security
Algorithm: abstract mathematical object (**black box**)	Algorithm: program running on given device (**gray box**)
Only inputs and outputs are available.	Implementation-specific characteristics might leak information.
⇝ Classical cryptanalysis	⇝ Side-channel cryptanalysis

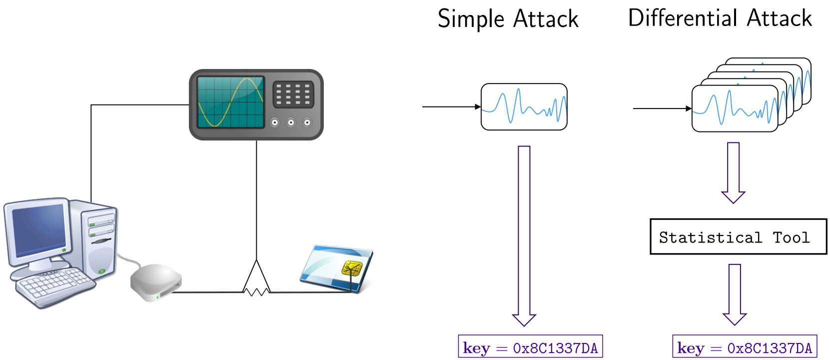
An attacker may have access to the device (e.g. smart card).

Side-Channels of a Smart-Card



Side-Channel Attacks

- Simple side-channel attack: exploit information from the leakage of **one** execution.
- Differential side-channel attack: exploit correlations between secret values and intermediate results.



Countermeasure

Principle of masking

- Randomize a variable with a random mask
- Keep the intermediate data masked all along the algorithm
- Unmask the result at the end.

Countermeasure

Principle of masking

- Randomize a variable with a random mask
- Keep the intermediate data masked all along the algorithm
- Unmask the result at the end.

Boolean masking example

Compute $y = \mathrm{F}(x)$:

$$
\begin{aligned}
r &\leftarrow \mathrm{Random}() && \text{// mask generation}\\
\widetilde{x} &\leftarrow x \oplus r && \text{// masking}\\
\widetilde{y} &\leftarrow \mathrm{F}(\widetilde{x}) &&\\
s &\leftarrow \mathrm{F}'(r) && \text{// mask correction}\\
y &\leftarrow \widetilde{y} \oplus s && \text{// unmasking}
\end{aligned}
$$

Countermeasure

Principle of masking

- Randomize a variable with a random mask
- Keep the intermediate data masked all along the algorithm
- Unmask the result at the end.

Boolean masking example

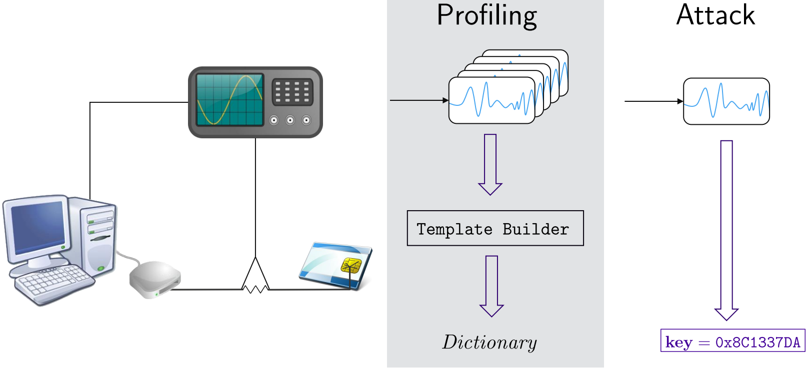
Compute $y = \mathrm{F}(x)$:

$r \leftarrow \mathrm{Random}()$	// mask generation
$\tilde{x} \leftarrow x \oplus r$	// masking
$\tilde{y} \leftarrow \mathrm{F}(\tilde{x})$	
$s \leftarrow \mathrm{F}'(r)$	// mask correction
$y \leftarrow \tilde{y} \oplus s$	// unmasking

Mask changes at each execution $\Rightarrow$ no correlations between traces.

Profiling

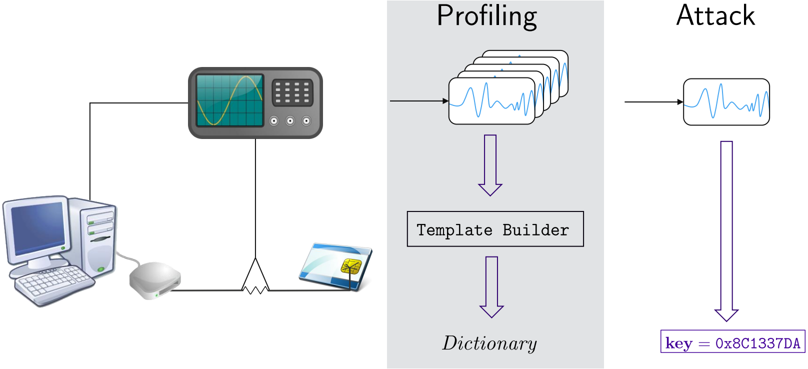
More powerful setting: the attacker can "play" with an under control version of the same device, before attacking the target.



Drawback: each "point" to attack must be profiled, requires many curves $\rightsquigarrow$ costly.

Profiling

More powerful setting: the attacker can “play” with an under control version of the same device, before attacking the target.



Drawback: each “point” to attack must be profiled, requires many curves ⇝ costly.

Question

How to use as few points as possible ?

Outline

1. Algebraic Side-Channel Attacks
2. Algebraic Modeling of Masked AES
3. Experimental Results
4. Conclusion

Algebraic Side-Channel Attacks

- Introduced by Renauld, Standaert [INSCRYPT 2009]
- Principle: model the algorithm, take leakages on intermediate values and feed an automated solver for algebraic systems.

+ semi-automatic, can achieve attack with fewer leakages.
- Leakages recovery usually requires a profiling stage or not (independent).

Algebraic Side-Channel Attacks

key = ??????????

m_0 = 0x01234567

c_0 = 0x8B3C01DE

⋮

m_k = 0xA1B2C3D4

c_k = 0x7D09A226

Algebraic Side-Channel Attacks

key = ??????????

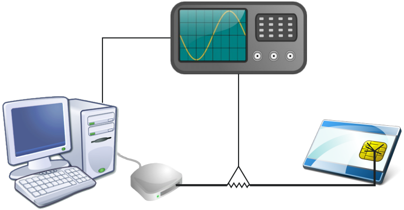
m_0 = 0x01234567

c_0 = 0x8B3C01DE

$\vdots$

m_k = 0xA1B2C3D4

c_k = 0x7D09A226

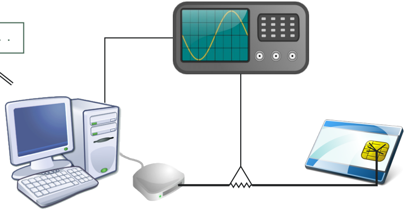


Algebraic Side-Channel Attacks

key = ??????????

m_0 = 0x01234567
c_0 = 0x8B3C01DE
⋮
m_k = 0xA1B2C3D4
c_k = 0x7D09A226

$\mathrm{HW}(A_0) = 4\ldots$

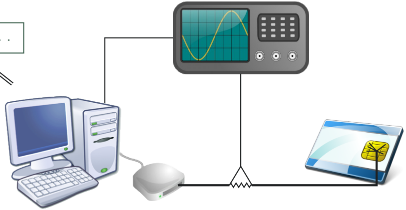


Algebraic Side-Channel Attacks

$$
\begin{aligned}
x_0 + x_1 x_2 + x_1 x_4 &= 0 \\
x_0 x_1 x_4 + x_3 x_4 + x_2 &= 0 \\
&\vdots \\
x_0 x_2 x_3 + x_0 x_2 + x_1 x_2 &= 0
\end{aligned}
$$

key = ??????????

$\mathrm{HW}(A_0) = 4 \ldots$



m_0 = 0x01234567
c_0 = 0x8B3C01DE
$\vdots$
m_k = 0xA1B2C3D4
c_k = 0x7D09A226

Algebraic Side-Channel Attacks

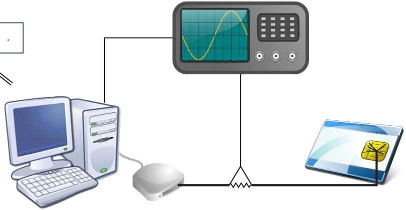
key = ??????????

$$x_0 + x_1 x_2 + x_1 x_4 = 0$$
$$x_0 x_1 x_4 + x_3 x_4 + x_2 = 0$$
$$\vdots$$
$$x_0 x_2 x_3 + x_0 x_2 + x_1 x_2 = 0$$

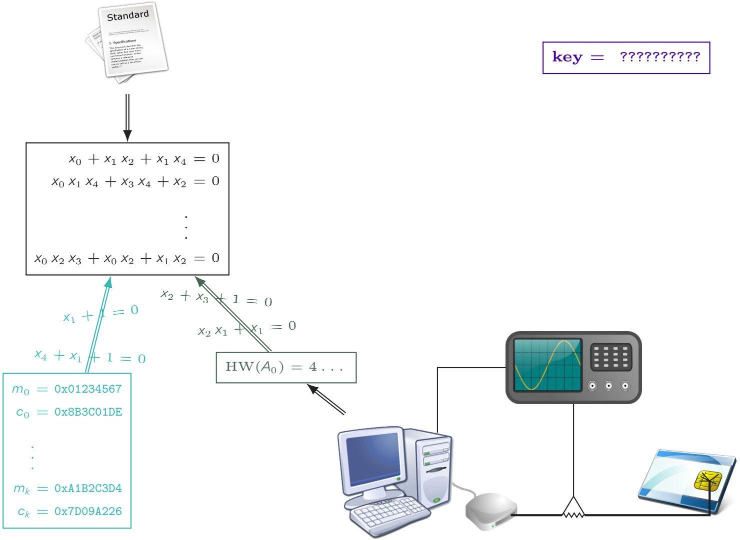
$$x_1 + 1 = 0$$
$$x_4 + x_1 + 1 = 0$$

m_0 = 0x01234567
c_0 = 0x8B3C01DE
⋮
m_k = 0xA1B2C3D4
c_k = 0x7D09A226

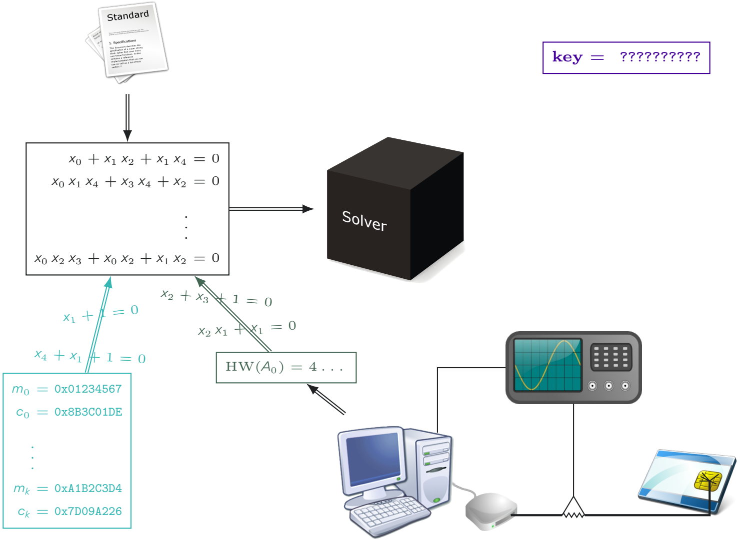
$\mathrm{HW}(A_0) = 4 \ldots$



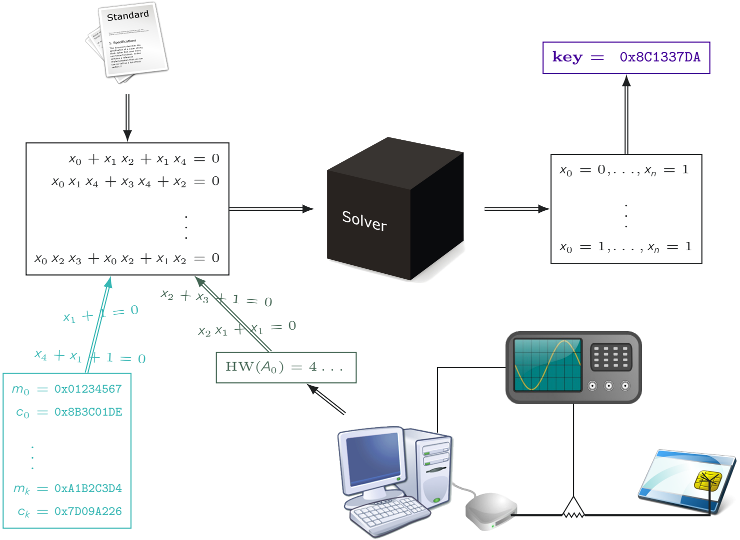
Algebraic Side-Channel Attacks



Algebraic Side-Channel Attacks



Algebraic Side-Channel Attacks



State of the Art

Algebraic Side-Channel Attacks

- Foundations
  - Renauld, Standaert [INSCRYPT 2009]
  - Renauld, Standaert, Veyrat-Charvillon [CHES 2009]
- Improvements
  - Oren, Kirschbaum, Popp, Wool [CHES 2010]
- Error handling
  - Zhao, Wang, Guo, Zhang, Shi, Liu, Wu [CASC 2011]
  - Oren, Renauld, Standaert, Wool [CHES 2012]

State of the Art

Algebraic Side-Channel Attacks

- Foundations
  - Renauld, Standaert [INSCRYPT 2009]
  - Renauld, Standaert, Veyrat-Charvillon [CHES 2009]
- Improvements
  - Oren, Kirschbaum, Popp, Wool [CHES 2010]
- Error handling
  - Zhao, Wang, Guo, Zhang, Shi, Liu, Wu [CASC 2011]
  - Oren, Renauld, Standaert, Wool [CHES 2012]

Observations

- Few details/study on masked implementations
- Advance in Machine Learning (ML): more accurate leakages

State of the Art

Algebraic Side-Channel Attacks

- Foundations
  - Renauld, Standaert [INSCRYPT 2009]
  - Renauld, Standaert, Veyrat-Charvillon [CHES 2009]
- Improvements
  - Oren, Kirschbaum, Popp, Wool [CHES 2010]
- Error handling
  - Zhao, Wang, Guo, Zhang, Shi, Liu, Wu [CASC 2011]
  - Oren, Renauld, Standaert, Wool [CHES 2012]

Observations

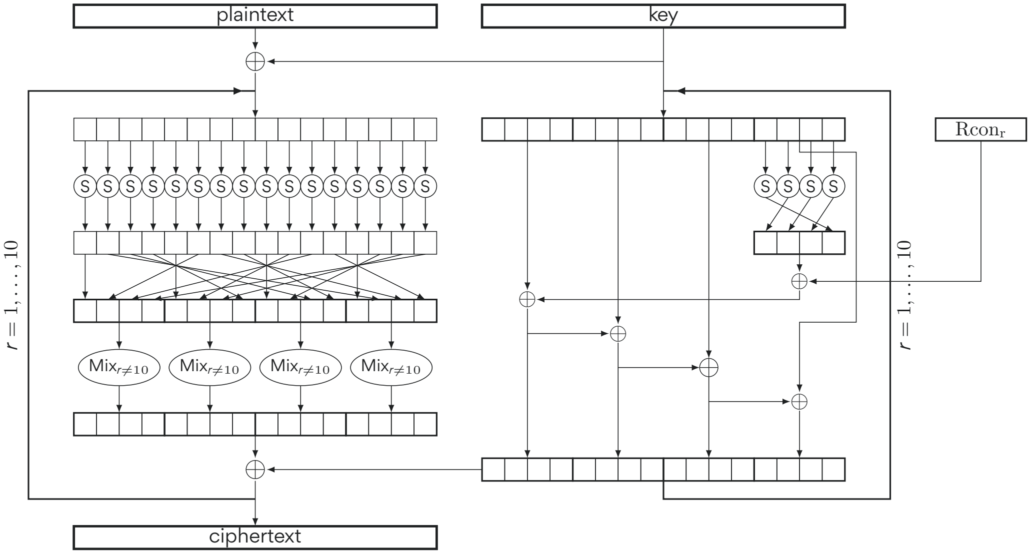
- Few details/study on masked implementations
- Advance in Machine Learning (ML): more accurate leakages

$\rightsquigarrow$ no error handling

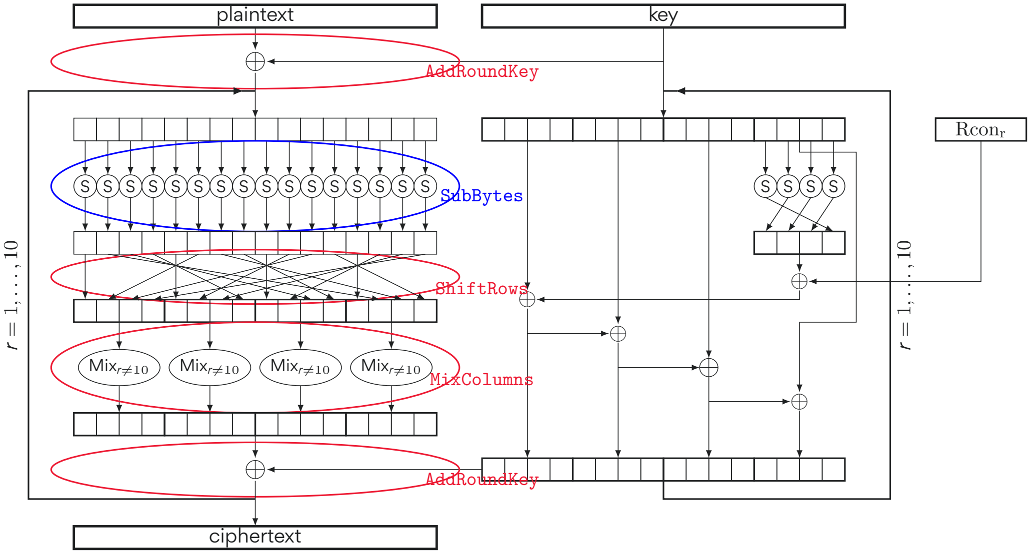
Contributions

- Work on state-of-the-art embedded code: masked!
- Consider exact leakages given by profiling step
- Try to minimize number of leakages to minimize profiling
- Apply ASCA to different masking schemes.

The AES block cipher



The AES block cipher



Algebraic Modeling of AES

Number of variables

One bit $\Leftrightarrow$ one variable

- 128 variables for key bits, 128 variables for input bits
- (10×128) variables for intermediate states (128 per inner round)
- (10×128) variables for subkeys (128 per inner rounds)
- 128 variables for output bits

Equations

- SBox: each output bit as function of 8 input bits
- Linear parts: Lin. combination of SBoxes' output bit
- Only one equation for each new state/subkey bit

Algebraic Modeling of AES

Number of variables

One bit $\Leftrightarrow$ one variable

- 128 variables for key bits, 128 variables for input bits
- (10×128) variables for intermediate states (128 per inner round)
- (10×128) variables for subkeys (128 per inner rounds)
- 128 variables for output bits

Equations

- SBox: each output bit as function of 8 input bits $\rightsquigarrow$ total degree $= 8$
- Linear parts: Lin. combination of SBoxes' output bit
- Only one equation for each new state/subkey bit

Algebraic Modeling of AES

Number of variables

One bit $\Leftrightarrow$ one variable

- 128 variables for key bits, 128 variables for input bits
- (10×128) variables for intermediate states (128 per inner round)
- (10×128) variables for subkeys (128 per inner rounds)
- 128 variables for output bits

Equations

- SBox: each output bit as function of 8 input bits $\rightsquigarrow$ total degree $= 8$
- Linear parts: Lin. combination of SBoxes' output bit $\rightsquigarrow$ no increase in degree
- Only one equation for each new state/subkey bit

Algebraic Modeling of AES

Number of variables

One bit $\Leftrightarrow$ one variable

- 128 variables for key bits, 128 variables for input bits
- (10×128) variables for intermediate states (128 per inner round)
- (10×128) variables for subkeys (128 per inner rounds)
- 128 variables for output bits

Equations

- SBox: each output bit as function of 8 input bits $\rightsquigarrow$ total degree $= 8$
- Linear parts: Lin. combination of SBoxes' output bit $\rightsquigarrow$ no increase in degree
- Only one equation for each new state/subkey bit

Total: 2688 equations of degree at most 8, in 2944 variables.

Attack context

Target of evaluation

- component: 8-bit micro-controller
- leakage: Hamming weight (HW) of manipulated values
- noise: not considered (perfect leakage)

Algebraic Modeling of Hamming Weight leakages

General Principle

$HW\left(\sum_{i=0}^{n-1} b_i\, 2^i\right) = w \Leftrightarrow$ exactly w bits among the b_i's are 1.

Algebraic Modeling of Hamming Weight leakages

General Principle

$HW\left(\sum_{i=0}^{n-1} b_i 2^i\right) = w \Leftrightarrow$ exactly w bits among the b_i's are 1.

Equations

- At most w bits are $1 \Leftrightarrow$ all products of $w+1$ bits are 0:

$$\prod_{i \in \mathcal{S}_1} x_i = 0\,, \quad \ldots \quad , \quad \prod_{i \in \mathcal{S}_k} x_i = 0\,,$$

  for each $w+1$ elements subset $\mathcal{S}_1, \ldots, \mathcal{S}_k$ of $\{0, \ldots, n-1\}$.

- At least w bits are $1 \Leftrightarrow$ sum of all products of w bits is positive:

$$\sum_{j=1}^{\ell} \prod_{i \in \mathcal{S}_j} x_i \geqslant 1\,,$$

  for all w elements subsets $\mathcal{S}_j$ of $\{0, \ldots, n-1\}$.

Algebraic Modeling of Hamming Weight leakages

General Principle

$HW\left(\sum_{i=0}^{n-1} b_i\, 2^i\right) = w \Leftrightarrow$ exactly w bits among the b_i's are 1.

Equations

- At most w bits are $1 \Leftrightarrow$ all products of $w+1$ bits are 0:

$$\prod_{i \in \mathcal{S}_1} x_i = 0\,, \quad \ldots \quad , \quad \prod_{i \in \mathcal{S}_k} x_i = 0\,,$$

  for each $w+1$ elements subset $\mathcal{S}_1, \ldots, \mathcal{S}_k$ of $\{0, \ldots, n-1\}$.

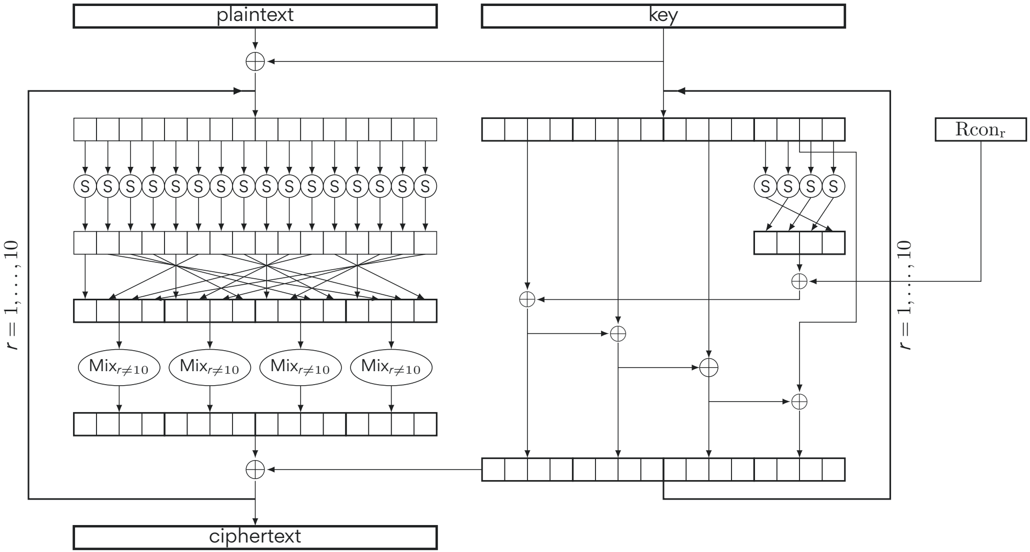
- At least w bits are $1 \Leftrightarrow$ sum of all products of w bits is positive:

$$\sum_{j=1}^{\ell} \prod_{i \in \mathcal{S}_j} x_i \geqslant 1\,,$$

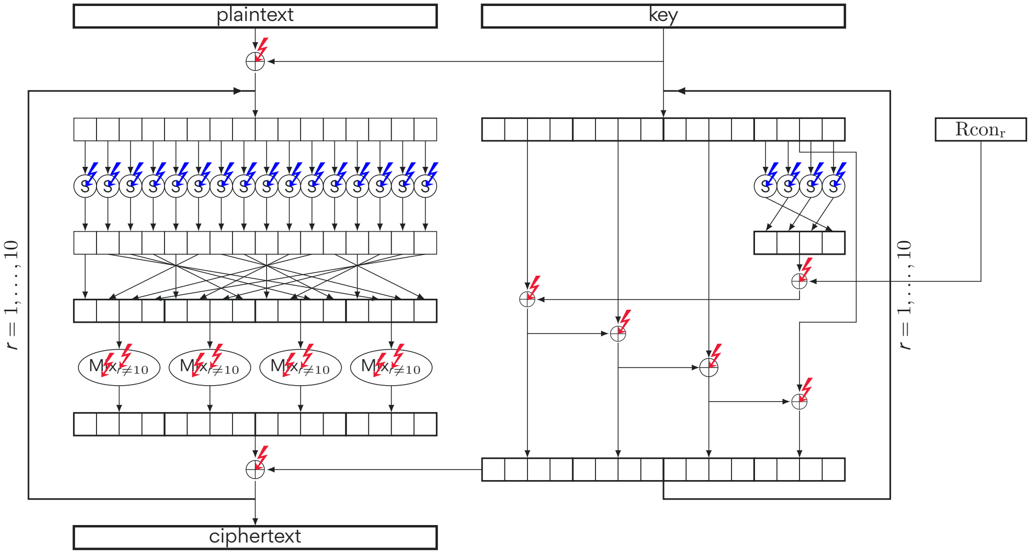
  for all w elements subsets $\mathcal{S}_j$ of $\{0, \ldots, n-1\}$.

Total: at most 71 equations of degree at most 8.

Leakages Location



Leakages Location



Algebraic Modeling of Masked AES

Masking against DSCA

- Linear parts: easily propagates with boolean masking
- Non-linear parts: specific algorithmic required
  - Masked SBox recomputation (in RAM)
  - Operations on smaller field ($\mathrm{GF}(16)$)

Impact on modeling

- **1 byte** mask: same for each state byte + temporary mask
- **16 bytes** mask: full state mask

Algebraic Modeling of Masked AES

Masking against DSCA

- Linear parts: easily propagates with boolean masking
- Non-linear parts: specific algorithmic required
  - Masked SBox recomputation (in RAM)
  - Operations on smaller field $(\mathrm{GF}(16))$

Impact on modeling

- **1 byte** mask: same for each state byte + temporary mask $\rightsquigarrow$ 16 extra variables.
- **16 bytes** mask: full state mask $\rightsquigarrow$ 128 extra variables.

Algebraic Modeling of Masked AES

Masking against DSCA

- Linear parts: easily propagates with boolean masking
- Non-linear parts: specific algorithmic required
  - Masked SBox recomputation (in RAM)
  - Operations on smaller field ($\mathrm{GF}(16)$)

Impact on modeling

- **1 byte** mask: same for each state byte + temporary mask $\rightsquigarrow$ 16 extra variables.
- **16 bytes** mask: full state mask $\rightsquigarrow$ 128 extra variables.
- Possible extra equations for non-linear parts

Framework/Setting

- Algebraic equations (ANF) generated using Magma computer algebra system
- Equations converted into **SAT**isfiability problem instance (CNF)
- CNF solved using CryptoMiniSAT SAT-solver
- Leakages are simulated within the framework
- Timeout on solving (4 hours)

Experimental Results Summary

Table: Best results for each setting (**one** known plaintext/ciphertext pair)

	Rnds	nb. Leakages	Success Rate
`KeySchedule`	1-5	64	100%
`Plain`	1	48	100%
`Partial`	4-5	96	100%
`1Mask`	1	48	12.5%
	1	84	87.5%
`16Mask`	-	-	0%
`1MaskGF16`	1	64	100%
`16MaskGF16`	1	128	12.5%
	1-2	320	100%

Experimental Results Summary

Table: Best results for each setting (**one** known plaintext/ciphertext pair)

	Rnds	nb. Leakages	Success Rate
`KeySchedule`	1-5	64	100%
`Plain`	1	48	100%
`Partial`	4-5	96	100%
`1Mask`	1	48	12.5%
	1	84	87.5%
`16Mask`	-	-	0%
`1MaskGF16`	1	64	100%
`16MaskGF16`	1	128	12.5%
	1-2	320	100%

Success rate can be increased with many plaintext/ciphertext pairs.

Conclusion

Results Analysis

- Key scheduling should be protected
- Partial masking: vulnerable
- 1 byte only mask: vulnerable
- $\mathrm{GF}(16)$: SBox computation leaks a lot of information
- 16 bytes mask: depend on implementation
- Security against classical DSCA $\not\Rightarrow$ security against ASCA

Conclusion

Results Analysis

- Key scheduling should be protected
- Partial masking: vulnerable
- 1 byte only mask: vulnerable
- $\mathrm{GF}(16)$: SBox computation leaks a lot of information
- 16 bytes mask: depend on implementation
- Security against classical DSCA $\not\Rightarrow$ security against ASCA

- Results conditioned by quality of leakages.
- Masked implementations in inaccurate leakages context ?

Questions?

IDEMIA

Join us on

www.idemia.com