

NIST Masked Circuits Feedback Compilation 2021a

NIST Masked Circuits Project¹

Updated: January 27, 2022

On June 16, 2021, the NIST [masked circuits project](#) issued a call for feedback about masking techniques for block-cipher circuits. The call included, as a reference for initiating the discussion, a proposal from K.U.Leuven researchers (received on June 1), describing three first-order masking schemes for AES. The call posed three questions about potential interest, and two questions about feasibility potential. This document compiles the call, the mentioned reference, a printout of the comments received in response to the call, and mentions two other suggested references received prior to the call.

Contents

Item 1: Call for feedback (2021-Jun-21)	2
Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)	3
Comments received in reply to the call	37
Item 3: Comments by Amir Moradi (2021-Aug-20)	37
Item 4: Comments by Francois-Xavier Standaert (2021-Aug-30)	39
Item 5: Comments by Patrick Haddad (2021-Sep-03)	40
Item 6: Comments by Elke De Mulder (2021-Sep-05)	42
Item 7: Comments by Sylvain Guilley (2021-Sep-06)	44
Item 8: Comments by Wei Cheng, Sylvain Guilley, Jean-Luc Danger (2021-Sep-06)	51
Item 9: Comments by Ventzi Nikov (2021-Sep-07)	61
Notes on other comments received before the call	64
Item 10: Comments from Fatemeh Ganji (2021-Jun-11)	64
Item 11: Comments from Siemen Dhooghe (2020-Dec-15)	64

¹Webpage: <https://csrc.nist.gov/projects/masked-circuits>; email address: masked-circuits@nist.gov.

Item 1: Call for feedback (2021-Jun-21)

Call for feedback about the Masked Circuits Project

Text originally posted on 2021-June-21 in the MC-Forum

Comments due by 2021-Sep-06

The NIST Masked Circuits project is interested in masking techniques that can improve resistance of hardware circuit implementations of block-ciphers against side-channel attacks. The newly formed forum MC-forum@list.nist.gov will be used to support open discussion; and the new project webpage <https://csrc.nist.gov/Projects/masked-circuits> will host relevant material.

It is important to gauge the stakeholders' interest in this effort and on the direction to be taken. We received a proposal "Threshold Cryptography on a Single Device by Means of Threshold Implementations" by S. Dhooghe, S. Nikova, and V. Rijmen, from KU Leuven, see attachment. It describes three first-order masking schemes for AES, requiring a set of algorithmic properties to guarantee security in the (single-probe) glitch-extended probing model. The proposal covers only the logical construction, with the result being specified as a netlist. Using this as a basis, this email serves as a call for feedback on it. We ask the community to comment on the following:

1. Potential interest:

- a) Are the glitch-extended probing model and the corresponding masking schemes addressing problems of interest for the industry and government stakeholders?
- b) What orders of protection are pertinent to consider for standardization?
- c) Will the semiconductor industry adopt this type of schemes in their product lines to supply the market with products implementing them?

2. Feasibility potential:

- a) Is there a stack of available design and production tools, widely used by the semiconductor industry, that can transfer the logical netlists into silicon gate layouts, on major hardware platforms of interest (FPGA, ASIC, etc.), while preserving the needed algorithmic properties and ensuring the needed implementation assumptions, so that minimal side-channel testing would be sufficient for the verification of the intended security guarantees?
- b) Considering practical attacks, are there identifiable deployment characteristics that justify differentiated security profiles across platforms, masking orders, and other properties?

The MC-forum@list.nist.gov forum can be used for open comments, but please send your formal comments in a PDF file, by September 6, 2021, by email to masked-circuits@nist.gov. The formal comments received by email, in PDF, will be compiled together and made available on the webpage.

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

Threshold Cryptography on a Single Device by Means of Threshold Implementations

Siemen Dhooghe, Svetla Nikova, Vincent Rijmen

imec-COSIC, ESAT, KU Leuven, Belgium
firstname.lastname@esat.kuleuven.be

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

Contents

1	Introduction	2
2	First-Order Hardware Sharings of the AES	4
2.1	Overview	4
2.2	Preliminaries	4
2.2.1	Notation	5
2.2.2	Description of the AES	5
2.2.3	The Threshold Glitch-Extended Probing Model	5
2.2.4	Boolean Sharing and Threshold Implementations	6
2.3	Changing of the Guards With Randomness	8
2.4	Overarching Structure of the AES Sharings	9
2.4.1	Multipliers	9
2.4.2	Sharing the State and Key	10
2.4.3	Sharing the Affine Transformations	10
2.4.4	Changing of the Guards Implemented	10
2.4.5	Key Schedule	11
2.5	Design I: First-Order Two-Share AES	11
2.5.1	S-Box Sharing	11
2.6	Design II: First-Order Three-Share AES	15
2.6.1	S-Box Sharing	15
2.7	Design III: First-Order Three-Share AES	18
2.7.1	S-Box Sharing	18
2.8	Security	21
2.9	Comparison Between the Designs	22
3	Algorithmic Verification Properties	25
3.1	Overview	25
3.2	List of Properties	25
3.3	Constructing the Structural Model	26
3.4	Verifying Non-Completeness	26
3.5	Verifying Uniformity	27
3.6	Verifying Probing Security	28
4	Conclusion	30

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

Chapter 1

Introduction

In this document we give a proposal to support the standardization effort for threshold cryptography on a single device. The proposal consists of two parts.

The first part involves a standardized sharing of the AES. This means the detailed provision of shared functions which provide provable algorithmic protection in the glitch-extended probing model. We propose three different first-order sharings of the AES to provide designers with possible trade-offs between the number of shares, which affects the area of the implementation, and the number of register stages, which affects the latency of the design. The designs have low randomness requirements to allow for the use of efficient and high-secure random bit generators. Second-order sharings could be added in the future.

The second part includes algorithmic properties which have to be verified on a netlist (since we focus only on ASIC related countermeasures) in order to guarantee security in the considered adversary model. These properties are detailed in a way that the readers can design a tool verifying the properties.

Certification is currently lacking from this proposal. However, we think it is important and could be a valuable part of the standard by providing a procedure on how to practically evaluate the security of a shared implementation. Our opinion is that the certification should at least be based on the procedure of ISO/IEC 17825:2016 extended with the proposal given in the paper by Whittall *et al.* [WO19]. We believe that it has to use higher-order and multivariate test vector leakage assessment [CDG⁺13] over sufficiently many power traces of an ASIC implementation.

Our goal in the first part of the proposal is to reduce a part of the industry's design cycle of secured applications. However, it does not tackle how to securely implement a generic function, instead the second part is intended to help with this. In other words, industry should still be free to design their own sharings

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

to keep/obtain market advantage and the second part of the proposal has to be used as a tool to verify the security of implementation of any design.

The verification tool from the second part can help both industry and certification labs with reducing the time for practically verifying an implementation by allowing to test netlist first. Currently, we propose three properties such that potential bugs are quickly found. We believe both the academic and industry communities can help expand these properties, resulting in a public software leading to improved efficiency and, more importantly, the efficacy of the verification.

Last but not least, over time, we believe the netlist verification can better predict certification lab evaluation results and over time further simplify the design and evaluation cycle. We provide security claims only for ASIC physical implementations profile. Although the proposed techniques are applicable also to FPGA and software implementation profiles, there will be no guarantees because certain assumptions of the adversarial model we consider might be violated.

Notation

Throughout the document we denote bits by subscript and shares by superscript. We denote the most significant bit by a bigger subscript. For example, given (a_1, \dots, a_8) then a_8 denotes the most significant bit and a_1 the least significant. We denote stochastic variables by capital letters. The symbols \oplus, \otimes denote addition and multiplication in the respective finite field, e.g. XOR and AND in \mathbb{F}_2 .

We consider sharing schemes where each secret variable x in the circuit is split into shares $\bar{x} = (x^1, x^2, \dots, x^{s_x})$ such that $x = \sum_{i=1}^{s_x} x^i$ over a binary finite field. Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$, then the function $\bar{F} : \mathbb{F}_2^{n \cdot s_x} \rightarrow \mathbb{F}_2^{m \cdot s_y}$, where we assume s_x shares per input bit and s_y shares per output bit, will be called a *sharing* of F . Each component of the shared function will be denoted by a superscript, e.g. F^i .

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

Chapter 2

First-Order Hardware Sharings of the AES

2.1 Overview

In this Chapter, we provide three variants of first-order sharings of the AES to be used in hardware ASIC implementations, each trading off the number of register stages with the number of shares. The sharings are written out in detail to aid the implementation. Software providing the input/output functionality of the shared S-boxes can be found on [GitLab](#).

Currently, no implementation costs (i.e. GE count) are included, instead rough estimates of XOR-gates, AND-gates and registers are given in Table 2.1. In addition Table 2.1 provides the number of register stages, the required randomness and the number of shares used.

The Chapter starts with introduction of AES, the probing with glitches adversarial model, and threshold implementations in Section 2.2. We then introduce a slight generalization of the changing of the guards technique in Section 2.3. Since the three introduced sharings have certain similarities, we introduce those in Section 2.4. Section 2.5 provides design I: the AES sharing with 2 shares targeting minimal area. Section 2.6 provides design II: a sharing with 3 shares targeting a trade-off between area and latency. Section 2.7 provides design III: a sharing with 3 shares targeting lower latency. A proof of the designs' security in the considered adversarial model is given in Section 2.8. Finally we provide a brief comparison of the three designs in Section 2.9.

2.2 Preliminaries

In this section we go over the used notation, introduce the AES, the probing side-channel security model, and threshold implementations.

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

2.2.1 Notation

We denote bits by subscript and shares by superscript. We denote the most significant bit by a bigger subscript. For example, given (a_1, \dots, a_8) then a_8 denotes the most significant bit and a_1 the least significant. The symbols \oplus, \otimes denote addition and multiplication in the respective finite field, e.g. XOR and AND in \mathbb{F}_2 .

2.2.2 Description of the AES

We quickly introduce the AES cipher designed by Daemen and Rijmen [DR20] and standardized by NIST [FIPS197]. There are three levels of security 128, 192, and 256. AES consists of a 128-bit state and 128, 192, or 256-bit key, respectively, divided into bytes. The cipher is composed of 10, 12, or 14 rounds, respectively, each applying an addition of a subkey, a bricklayer of S-Boxes, a `ShiftRows` operation, and a `MixColumns` operation. The AES S-Box consists of an inversion in the field \mathbb{F}_{2^8} and the application of an affine layer. This is visually represented in Figure 2.1.

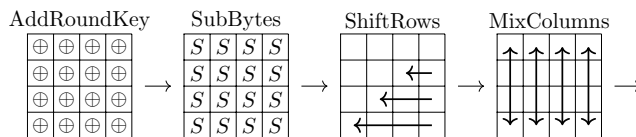


Figure 2.1: Representation of the AES.

The key schedule for AES-128, which operates on 4 columns of 32 bits each, is depicted in Figure 2.2. For each round of the AES state function, there is a parallel round of the key schedule. We provide a description for the AES-128 key schedule. Denote V_j , with $j \in \{1, \dots, 4\}$, the j^{th} word of the key state at round i and W_j the j^{th} word of the key state at round $i + 1$. Then a round of the key schedule is defined as

$$\begin{aligned}
 W_1 &= V_1 \oplus \text{RotWord}(\text{SubWord}(V_4)) + C_{i+1}, \\
 W_2 &= V_2 \oplus W_1, \\
 W_3 &= V_3 \oplus W_2, \\
 W_4 &= V_4 \oplus W_3.
 \end{aligned}
 \tag{2.1}$$

With `RotWord` the left circular shift, `SubWord` the application of four AES S-boxes, and C_{i+1} the round constants for round $i + 1$.

2.2.3 The Threshold Glitch-Extended Probing Model

This section introduces the threshold probing model.

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

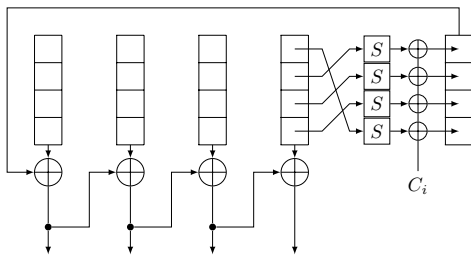


Figure 2.2: The AES-128 key schedule. The i^{th} round constants are denoted by $C_i \in \mathbb{F}_2^{32}$ and S denotes the AES S-box.

Threshold Probing. A d^{th} -order threshold probing adversary \mathcal{A} , as first proposed by Ishai *et al.* [ISW03], can view the values present on up to d gates or wires in a circuit implementing a cipher during a single execution (cipher evaluation). We note that by “probe” we do not mean a physical probe such as an EM probe. Instead, the word probe is used as an abstract concept through which an adversary can perfectly observe a part of the computation.

The adversary \mathcal{A} is computationally unbounded, and must specify the location of the probes before querying the circuit. However, the adversary can change the location of the probes over multiple cipher queries. The adversary’s interaction with the circuit is mediated through encoder and decoder algorithms, neither of which can be probed.

Glitches. The above model is extended to capture the effect of glitches on hardware. Whereas one of the adversary’s probes normally results in the value of a single wire, a glitch-extended probe allows obtaining all the registered inputs leading to the gate/wire which is probed. This extension of the probing model has been discussed in the work of Reparaz *et al.* [RBN⁺15] and formalized by Faust *et al.* [FGP⁺18]. The formulation of the latter work is as follows: “For any ϵ -input circuit gadget G , combinatorial recombinations (aka glitches) can be modeled with specifically ϵ -extended probes so that probing any output of the function allows the adversary to observe all its ϵ inputs.”

2.2.4 Boolean Sharing and Threshold Implementations

Boolean sharing was independently introduced by Goubin and Patarin [GP99] and Chari *et al.* [CJRR99]. It serves as a sound and widely-deployed countermeasure against side-channel attacks. The technique is based on splitting each secret variable $x \in \mathbb{F}_2$ in the circuit into shares $\bar{x} = (x^1, x^2, \dots, x^{s_x})$ such that $x = \sum_{i=1}^{s_x} x^i$ over \mathbb{F}_2 . A random Boolean sharing of a fixed secret is uniform if all sharings of that secret are equally likely.

There are several approaches to sharing a circuit. In this proposal, we make use of threshold implementations, proposed by Nikova *et al.* [NRR06]. In particular, we focus on “first-order threshold implementations” as those which protect against first-order side-channel attacks. The interested reader is referred to the

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

works by Bilgin *et al.* [BGN⁺14] and Beyne *et al.* [BDZ20] for more information on how to use threshold implementations to secure against higher-order attacks. In the following, the main properties of threshold implementations as introduced by Nikova *et al.* are reviewed.

A threshold implementation consists of several layers of Boolean functions, as shown in Figure 2.3. As for any shared design, a black-box encoder function generates a uniform random sharing of the input before it enters the shared circuit and the output shares are recombined by a decoder function. At the end of each layer, synchronization is ensured by means of registers which are assumed to stop glitches.

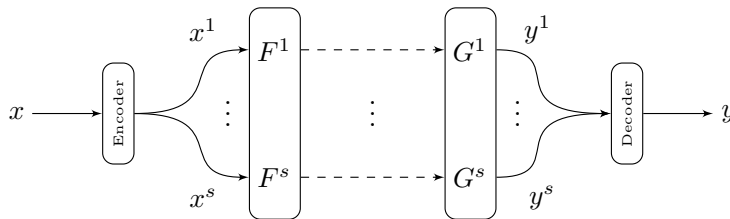


Figure 2.3: Schematic illustration of a threshold implementation assuming an equal number of input and output shares.

Let \bar{F} be a layer in the threshold implementation corresponding to a part of the circuit $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$. For example, F might be the linear layer of a block cipher. The function $\bar{F} : \mathbb{F}_2^{n s_x} \rightarrow \mathbb{F}_2^{m s_y}$, where we assume s_x shares per input bit and s_y shares per output bit, will be called a *sharing* of F . The i^{th} share of the function \bar{F} is denoted by $F^i : \mathbb{F}_2^{n s_x} \rightarrow \mathbb{F}_2^m$, for $i \in \{1, \dots, s_y\}$. Sharings can have a number of properties that are relevant in the security argument for a threshold implementation; these properties are summarized in Definition 1.

Definition 1 (Properties of first-order threshold implementations [NRR06]). Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ be a function and $\bar{F} : \mathbb{F}_2^{n s_x} \rightarrow \mathbb{F}_2^{m s_y}$ be a sharing of F . The sharing \bar{F} is said to be

1. *correct* if $\sum_{i=1}^{s_y} F^i(x^1, \dots, x^{s_x}) = F(x)$ for all $x \in \mathbb{F}_2^n$ and for all shares $x^1, \dots, x^{s_x} \in \mathbb{F}_2^n$ such that $\sum_{i=1}^{s_x} x^i = x$,
2. *non-complete* if any function F^i depends on at most $s_x - 1$ input shares measured between register stages,
3. *uniform* if \bar{F} maps a uniform random sharing of any $x \in \mathbb{F}_2^n$ to a uniform random sharing of $F(x) \in \mathbb{F}_2^m$.

In a threshold implementation, all input/outputs of the functions are stored in registers, which separate the layers. Therefore placing a glitch-extended probe in a layer of a threshold implementation returns only the inputs of the probed shared Boolean function. If all layers of a threshold implementation are non-complete and uniform, the resulting shared circuit can be proven secure in the first-order probing model with glitches [DNR19].

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

2.3 Changing of the Guards With Randomness

The changing of the guards method proposed by Daemen [Dae17] is a technique that transforms a non-complete sharing into a uniform and non-complete sharing. The technique works by embedding the sharing into a Feistel-like structure. In this paper, we slightly generalize the method by considering a first-order probing secure sharing. Such a sharing potentially requires multiple register stages and extra randomness to guarantee its security. The adapted changing of the guards method still ensures uniformity and first-order probing security while allowing the re-use of the randomness. An example of the method with two shares is shown in Figure 2.4.

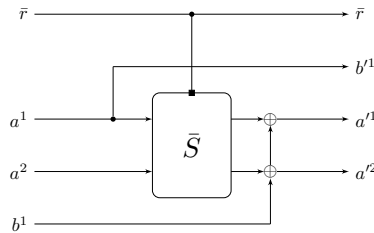


Figure 2.4: Changing of the guards method with two shares where the shared S-box \bar{S} uses the randomness \bar{r} .

We give the changing of the guards method formally.

Definition 2. The changing of the guards method applied to a shared map \bar{S} given inputs (a^1, \dots, a^s) , (b^1, \dots, b^{s-1}) , and randomness \bar{r} is calculated as follows

$$\bar{r}' = \bar{r} \tag{2.2}$$

$$a'^1 = S^1(a^1, \dots, a^s, \bar{r}) \oplus b^1, \quad \dots, \quad a'^{s-1} = S^{s-1}(a^1, \dots, a^s, \bar{r}) \oplus b^{s-1}, \tag{2.3}$$

$$a'^s = S^s(a^1, \dots, a^s, \bar{r}) \oplus b^1 \oplus \dots \oplus b^{s-1} \tag{2.4}$$

$$b'^1 = a^1, \quad \dots, \quad b'^{s-1} = a^{s-1}. \tag{2.5}$$

In general we refer to the (b^1, \dots, b^{s-1}) as the *guards* of the shared S-box \bar{S} .

We show that the changing of the guards construction with randomness retains the correctness and probing security properties from \bar{S} , but makes the sharing uniform.

Theorem 2.3.1. *The method from Definition 2 is correct, first-order probing secure, and uniform.*

Proof. Correctness of the construction follows from the correctness of \bar{S} and the fact that each share b^i is added to two different output shares in Equations 2.3-2.4.

First-order probing security of the construction, assuming a joint uniform input, follows from the first-order probing security of \bar{S} and the facts that the

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

share b^s is not used in the construction and that each share b^i is calculated using only one share a^i using Equations 2.3-2.4.

For the proof of uniformity, we first take an arbitrary input secret a . We show that the above construction is invertible. In other words, given the secret a and the outputs (a^1, \dots, a^s) , (b^1, \dots, b^{s-1}) , \bar{r}' , we show it is possible to construct the inputs (a^1, \dots, a^s) , (b^1, \dots, b^{s-1}) , \bar{r} .

Since the input secret a is given, we can construct the inputs (a^1, \dots, a^s) from (b^1, \dots, b^{s-1}) using Equation 2.5. From \bar{r}' we can evidently construct \bar{r} since the two are equal (see Equation 2.2). By running (a^1, \dots, a^s) and \bar{r} through \bar{S} and XORing the output (a^1, \dots, a^s) , we can also construct (b^1, \dots, b^s) (see Equations 2.3-2.4) which concludes the proof. \square

Thus, the changing of the guards method allows for the transformation of any first-order probing secure sharing into a uniform one which allows the re-use of the randomness used in the S-box.

2.4 Overarching Structure of the AES Sharings

In this section we provide the common parts of all first-order designs. Only the sharing of the S-box differs for each design. These are detailed in Sections 2.5, 2.6, and 2.7.

2.4.1 Multipliers

Since we share the S-box using the tower field approach by Canright [Can05], in the computation of the sharing of the S-box, we make use of multipliers over \mathbb{F}_{2^2} and \mathbb{F}_{2^4} . We note that these equations only hold for the multipliers in the shared S-box and not for any other operation in the AES like the `MixColumns`. We recall the equations for the multiplication over \mathbb{F}_{2^2} . These map the two 2-bit inputs (a_1, a_2) , (b_1, b_2) to the 2-bit output (c_1, c_2) as follows:

$$\begin{aligned} c_1 &= (a_2 \oplus a_1) \otimes (b_2 \oplus b_1) \oplus (a_1 \otimes b_1) \\ c_2 &= (a_2 \oplus a_1) \otimes (b_2 \oplus b_1) \oplus (a_2 \otimes b_2) \end{aligned} \tag{2.6}$$

We then recall the equations for the multiplication over \mathbb{F}_{2^4} . This maps the two 4-bit inputs (a_1, a_2, a_3, a_4) , (b_1, b_2, b_3, b_4) to the 4-bit output (c_1, c_2, c_3, c_4) as follows:

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

$$\begin{aligned}
c_1 &= (a_4 \otimes b_4) \oplus (a_2 \otimes b_4) \oplus (a_3 \otimes b_3) \oplus (a_1 \otimes b_3) \oplus (a_4 \otimes b_2) \oplus (a_1 \otimes b_2) \\
&\quad \oplus (a_3 \otimes b_1) \oplus (a_2 \otimes b_1) \oplus (a_1 \otimes b_1) \\
c_2 &= (a_4 \otimes b_4) \oplus (a_3 \otimes b_4) \oplus (a_2 \otimes b_4) \oplus (a_1 \otimes b_4) \oplus (a_4 \otimes b_3) \oplus (a_2 \otimes b_3) \\
&\quad \oplus (a_4 \otimes b_2) \oplus (a_3 \otimes b_2) \oplus (a_2 \otimes b_2) \oplus (a_4 \otimes b_1) \oplus (a_1 \otimes b_1) \quad (2.7) \\
c_3 &= (a_3 \otimes b_4) \oplus (a_2 \otimes b_4) \oplus (a_4 \otimes b_3) \oplus (a_3 \otimes b_3) \oplus (a_1 \otimes b_3) \oplus (a_4 \otimes b_2) \\
&\quad \oplus (a_2 \otimes b_2) \oplus (a_3 \otimes b_1) \oplus (a_1 \otimes b_1) \\
c_4 &= (a_4 \otimes b_4) \oplus (a_2 \otimes b_4) \oplus (a_1 \otimes b_4) \oplus (a_3 \otimes b_3) \oplus (a_2 \otimes b_3) \oplus (a_4 \otimes b_2) \\
&\quad \oplus (a_3 \otimes b_2) \oplus (a_2 \otimes b_2) \oplus (a_1 \otimes b_2) \oplus (a_4 \otimes b_1) \oplus (a_2 \otimes b_1)
\end{aligned}$$

2.4.2 Sharing the State and Key

For the sharing of the AES state and key, we use classical Boolean sharing. The key and state are each extended by $8 \cdot (s - 1)$ random bits, with s the number of shares, for the changing of the guards technique as introduced in Section 2.3.

For two-share designs, each byte of the state or key x is shared using a random byte r as follows:

$$x^1 = x \oplus r \quad x^2 = r \quad (2.8)$$

For three-share designs, each byte of the state or key x is shared using random bytes (r_1, r_2) as follows:

$$x^1 = x \oplus r_1 \oplus r_2 \quad x^2 = r_1 \quad x^3 = r_2 \quad (2.9)$$

2.4.3 Sharing the Affine Transformations

The sharing of the linear transformations (like `MixColumns`, `ShiftRows`, or the affine layer in the S-box) is simply done share-wise. Constants are added to the first share of the relevant variable.

2.4.4 Changing of the Guards Implemented

In this section we explain how to implement the changing of the guards method. We instantiate an extra random sharing of zero at the start of execution. The output of each shared S-box is reshared using the current guards. The guards are then replaced by the input of that shared S-box.

For two-share designs, given a guard g , a shared S-box with inputs (a^1, a^2) and the changing of the guards is calculated as follows:

$$a'^1 = S^1(a^1, a^2) \oplus g \quad a'^2 = S^2(a^1, a^2) \oplus g, \quad (2.10)$$

the guard is then replaced as follows

$$g' = a^1. \quad (2.11)$$

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

For three-share designs, given guards g^1, g^2 , a shared S-box with inputs (a^1, a^2, a^3) and the changing of the guards is calculated as follows:

$$\begin{aligned} a'^1 &= S^1(a^1, a^2, a^3) \oplus g^1 & a'^2 &= S^2(a^1, a^2, a^3) \oplus g^2 \\ a'^3 &= S^3(a^1, a^2, a^3) \oplus g^1 \oplus g^2, \end{aligned} \quad (2.12)$$

the guards are then replaced as follows

$$g'^1 = a^1 \quad g'^2 = a^2. \quad (2.13)$$

The designer is free to choose which guards are used to refresh an S-box. For example, the designer can choose to calculate the S-boxes row by row or column by column, it does not affect the first-order probing security of the design.

2.4.5 Key Schedule

The AES S-boxes in the key schedule are shared in the same way as the one from the state function, this is detailed in Sections 2.5, 2.6, and 2.7. The shared linear layers work share-wise. Finally, a changing of the guards structure is applied over the S-boxes in `SubWord`, this changing of the guards method follows the explanation above. The guards used in the state function can also be used in the key schedule. Meaning that the changing of the guards technique can be used across both the state function as the key schedule.

In case the shared key schedule is run to refresh the shared secret key, one first refreshes the shared master key after which, the shared key schedule is run to produce the refreshed round keys.

Thus, for two-share designs the key (k^1, k^2) is refreshed using 128 fresh random bits r , as follows:

$$k'^1 = k^1 \oplus r \quad k'^2 = k^2 \oplus r \quad (2.14)$$

For three-share designs the key (k^1, k^2, k^3) is refreshed using 256 fresh random bits (r_1, r_2) , as follows:

$$k'^1 = k^1 \oplus r_1 \oplus r_2 \quad k'^2 = k^2 \oplus r_1 \quad k'^3 = k^3 \oplus r_2 \quad (2.15)$$

2.5 Design I: First-Order Two-Share AES

This section describes the first S-box design of a two-share first-order AES. Compared to the three-share AES in Sections 2.6 and 2.7, this sharing is lower in the number of logic gates and higher in the number of register stages.

2.5.1 S-Box Sharing

The method is shown in Figure 2.5 and is divided into six stages and uses 54 random bits in total. These random bits can be re-used over all shared S-boxes in both the AES state function as the key schedule.

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

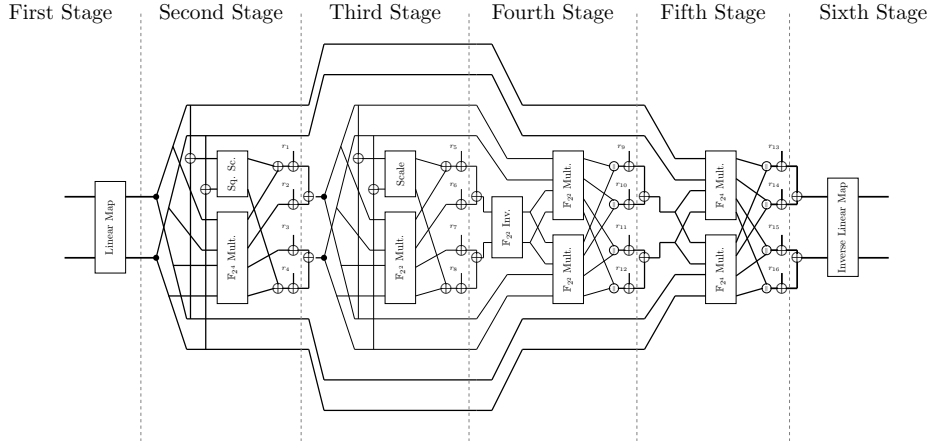


Figure 2.5: Representation of the S-box of design I. Register stages are denoted by dashed vertical lines.

First Stage.

The first operation occurring in the decomposed S-box performs a change of basis through a linear map. Its sharing requires instantiating this linear map once for each share. This mapping is implemented in combinational logic and it maps the 8-bit input (a_1^i, \dots, a_8^i) to the 8-bit output (y_1^i, \dots, y_8^i) for each share $i \in \{1, 2\}$ as follows:

$$\begin{aligned}
 y_1^i &= a_8^i \oplus a_7^i \oplus a_6^i \oplus a_3^i \oplus a_2^i \oplus a_1^i & y_5^i &= a_8^i \oplus a_5^i \oplus a_4^i \oplus a_2^i \oplus a_1^i \\
 y_2^i &= a_7^i \oplus a_6^i \oplus a_5^i \oplus a_1^i & y_6^i &= a_1^i \\
 y_3^i &= a_7^i \oplus a_6^i \oplus a_2^i \oplus a_1^i & y_7^i &= a_7^i \oplus a_6^i \oplus a_1^i \\
 y_4^i &= a_8^i \oplus a_7^i \oplus a_6^i \oplus a_1^i & y_8^i &= a_7^i \oplus a_4^i \oplus a_3^i \oplus a_2^i \oplus a_1^i
 \end{aligned} \tag{2.16}$$

Second Stage.

We consider the parallel application of nonlinear multiplication and affine Square Scaling (Sq. Sc.) as one single function $d = b \otimes c \oplus SqSc(b \oplus c)$. The affine square scaling $SqSc$ maps the 4-bit input (x_1, \dots, x_4) to the 4-bit output (y_1, \dots, y_4) as follows:

$$\begin{aligned}
 y_1 &= x_1 & y_3 &= x_2 \oplus x_4 \\
 y_2 &= x_1 \oplus x_2 & y_4 &= x_1 \oplus x_3
 \end{aligned} \tag{2.17}$$

For the parallel multiplier and square scaling with random nibbles (r_1, \dots, r_4) , the resulting equations are given by:

$$\begin{aligned}
 d^1 &= b^1 \otimes c^1 \oplus SqSc(b^1 \oplus c^1) \oplus r_1 & d^3 &= b^2 \otimes c^1 \oplus r_3 \\
 d^2 &= b^1 \otimes c^2 \oplus r_2 & d^4 &= b^2 \otimes c^2 \oplus SqSc(b^2 \oplus c^2) \oplus r_4
 \end{aligned} \tag{2.18}$$

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

where $r_4 = r_1 \oplus r_2 \oplus r_3$.

Third Stage.

This stage is similar to the second stage. First, the four output shares (d^1, \dots, d^4) from the previous stage are compressed to reduce the number of output shares back to two. More specifically, the shares (d^1, \dots, d^4) are mapped to (e^1, e^2) as follows:

$$e^1 = d^1 \oplus d^2 \qquad e^2 = d^3 \oplus d^4 \qquad (2.19)$$

These nibbles are then split in 2-bit couples for further operation. The Scaling operation (Sc) replaces the similar affine Square Scaling and is executed alongside the multiplication in \mathbb{F}_{2^2} as one function $h = f \otimes g \oplus Sc(f \oplus g)$. The scaling operation maps a 2-bit input (x_1, x_2) to the 2-bit output (y_1, y_2) as follows:

$$y_1 = x_1 \oplus x_2 \qquad y_2 = x_2 \qquad (2.20)$$

For the parallel multiplier and scaling with random two-bits (r_5, \dots, r_8), the resulting equations are given by:

$$\begin{aligned} h^1 &= f^1 \otimes g^1 \oplus Sc(f^1 \oplus g^1) \oplus r_5 & h^3 &= f^2 \otimes g^1 \oplus r_7 \\ h^2 &= f^1 \otimes g^2 \oplus r_6 & h^4 &= f^2 \otimes g^2 \oplus Sc(f^2 \oplus g^2) \oplus r_8 \end{aligned} \qquad (2.21)$$

where $r_8 = r_5 \oplus r_6 \oplus r_7$.

Fourth Stage.

First, the four output shares (h^1, \dots, h^4) from the previous stage are compressed to reduce the number of output shares back to two. More specifically, the shares (h^1, \dots, h^4) are mapped to (k^1, k^2) as follows:

$$k^1 = h^1 \oplus h^2 \qquad k^2 = h^3 \oplus h^4 \qquad (2.22)$$

The rest of the fourth stage is composed of an inversion and two parallel multiplications in \mathbb{F}_{2^2} . The inversion Inv in \mathbb{F}_{2^2} is linear and is implemented by swapping the bits using wires. The inversion operation maps a 2-bit input (x_1, x_2) to the 2-bit output (y_1, y_2) as follows:

$$y_1 = x_2 \qquad y_2 = x_1 \qquad (2.23)$$

The outputs of the multiplications are concatenated, denoted by \oplus in Figure 2.5, to form 4-bit values in \mathbb{F}_{2^4} . For the inversion and the parallel multipliers $l = f \otimes Inv(k)$ and $m = Inv(k) \otimes g$ the resulting equations are given by:

$$\begin{aligned} l^1 &= f^1 \otimes Inv(k^1) & l^3 &= f^2 \otimes Inv(k^1) \\ l^2 &= f^1 \otimes Inv(k^2) & l^4 &= f^2 \otimes Inv(k^2) \\ m^1 &= Inv(k^1) \otimes g^1 & m^3 &= Inv(k^2) \otimes g^1 \\ m^2 &= Inv(k^1) \otimes g^2 & m^4 &= Inv(k^2) \otimes g^2 \end{aligned} \qquad (2.24)$$

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

The two outputs of the parallel multipliers are then concatenated l being the most significant bits and m the least significant. The concatenated bits are refreshed with the random nibbles (r_9, \dots, r_{12}) , as follows:

$$\begin{aligned} n^1 &= l^1 \oplus m^1 \oplus r_9 & n^3 &= l^3 \oplus m^3 \oplus r_{11} \\ n^2 &= l^2 \oplus m^2 \oplus r_{10} & n^4 &= l^4 \oplus m^4 \oplus r_{12} \end{aligned} \quad (2.25)$$

where $r_{12} = r_9 \oplus r_{10} \oplus r_{11}$.

Fifth Stage.

Stage 5 is similar to stage 4. The difference of the two stages lies in the absence of the inversion operation and the multiplications being performed in \mathbb{F}_{2^4} instead of \mathbb{F}_{2^2} . First, the four output shares (n^1, \dots, n^4) from the previous stage are compressed to reduce the number of output shares back to two. More specifically, the shares (n^1, \dots, n^4) are mapped to (o^1, o^2) as follows:

$$\begin{aligned} o^1 &= n^1 \oplus n^2 & o^2 &= n^3 \oplus n^4 \end{aligned} \quad (2.26)$$

For the parallel multipliers $p = b \otimes o$ and $q = o \otimes c$ the resulting equations are given by:

$$\begin{aligned} p^1 &= b^1 \otimes o^1 & p^3 &= b^2 \otimes o^1 \\ p^2 &= b^1 \otimes o^2 & p^4 &= b^2 \otimes o^2 \\ q^1 &= o^1 \otimes c^1 & q^3 &= o^2 \otimes c^1 \\ q^2 &= o^1 \otimes c^2 & q^4 &= o^2 \otimes c^2 \end{aligned} \quad (2.27)$$

The two outputs of the parallel multipliers are then concatenated p being the most significant bits and q the least significant. The concatenated bits are refreshed with the random nibbles (r_{13}, \dots, r_{16}) , as follows:

$$\begin{aligned} s^1 &= p^1 \oplus q^1 \oplus r_{13} & s^3 &= p^3 \oplus q^3 \oplus r_{15} \\ s^2 &= p^2 \oplus q^2 \oplus r_{14} & s^4 &= p^4 \oplus q^4 \oplus r_{16} \end{aligned} \quad (2.28)$$

where $r_{16} = r_{13} \oplus r_{14} \oplus r_{15}$.

The additional XORs from the changing of the guards method, described in Section 2.4.4, can already be added at this stage. Denoting this values by t^1 , then this addition is performed as follows:

$$\begin{aligned} s'^1 &= s^1 \oplus t^1 & s'^3 &= s^3 \\ s'^2 &= s^2 & s'^4 &= s^4 \oplus t^1 \end{aligned} \quad (2.29)$$

Sixth Stage.

First, the four output shares (s^1, \dots, s^4) from the previous stage are compressed to reduce the number of output shares back to two. More specifically, the shares (s^1, \dots, s^4) are mapped to (t^1, t^2) as follows:

$$\begin{aligned} t^1 &= s^1 \oplus s^2 & t^2 &= s^3 \oplus s^4 \end{aligned} \quad (2.30)$$

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

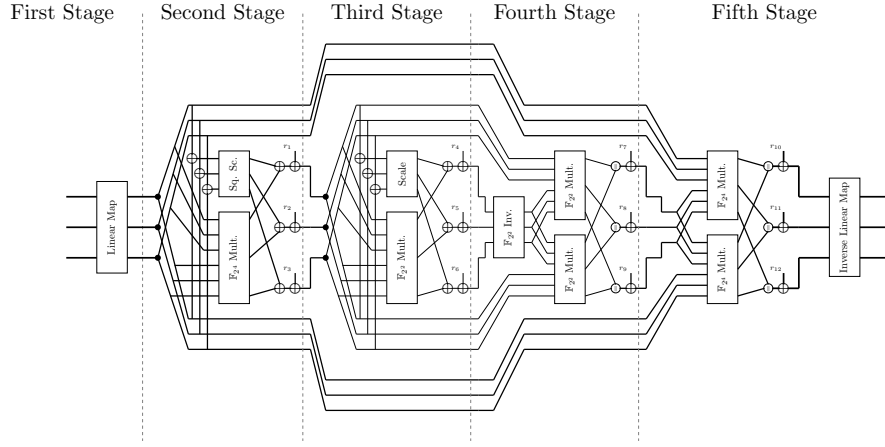


Figure 2.6: Representation of the S-box of design II. Register stages are denoted by dashed vertical lines.

Then, the inverse linear map (including the AES affine transformation) is performed. This linear function maps the 8-bit input (t_1^i, \dots, t_8^i) to the 8-bit output (y_1^i, \dots, y_8^i) for each share i as follows:

$$\begin{aligned}
 y_1^i &= t_6^i \oplus t_4^i & y_5^i &= t_8^i \oplus t_7^i \oplus t_6^i \oplus t_5^i \oplus t_4^i \\
 y_2^i &= t_8^i \oplus t_4^i & y_6^i &= t_7^i \oplus t_6^i \oplus t_4^i \oplus t_3^i \oplus t_1^i \\
 y_3^i &= t_7^i \oplus t_1^i & y_7^i &= t_6^i \oplus t_5^i \oplus t_2^i \\
 y_4^i &= t_8^i \oplus t_6^i \oplus t_4^i & y_8^i &= t_7^i \oplus t_5^i \oplus t_2^i
 \end{aligned} \tag{2.31}$$

The constant 0x63 is then added to the first share.

2.6 Design II: First-Order Three-Share AES

This section describes the second S-box design of a three-share first-order AES. Compared to the two-share AES in Section 2.5, this sharing is higher in the number of logic gates and lower in the number of register stages.

2.6.1 S-Box Sharing

The method is shown in Figure 2.6 and is divided into five stages and uses 36 random bits in total. These random bits can be re-used over all shared S-boxes in both the AES state function as the key schedule.

First Stage.

The first operation occurring in the decomposed S-box performs a change of basis through a linear map. Its sharing requires instantiating this linear map

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

once for each share. This mapping is implemented in combinational logic and it maps the 8-bit input (a_1^i, \dots, a_8^i) to the 8-bit output (y_1^i, \dots, y_8^i) for each share $i \in \{1, 2, 3\}$ as follows:

$$\begin{aligned}
 y_1^i &= a_8^i \oplus a_7^i \oplus a_6^i \oplus a_3^i \oplus a_2^i \oplus a_1^i & y_5^i &= a_8^i \oplus a_5^i \oplus a_4^i \oplus a_2^i \oplus a_1^i \\
 y_2^i &= a_7^i \oplus a_6^i \oplus a_5^i \oplus a_1^i & y_6^i &= a_1^i \\
 y_3^i &= a_7^i \oplus a_6^i \oplus a_2^i \oplus a_1^i & y_7^i &= a_7^i \oplus a_6^i \oplus a_1^i \\
 y_4^i &= a_8^i \oplus a_7^i \oplus a_6^i \oplus a_1^i & y_8^i &= a_7^i \oplus a_4^i \oplus a_3^i \oplus a_2^i \oplus a_1^i
 \end{aligned} \tag{2.32}$$

Second Stage.

We consider the parallel application of nonlinear multiplication and affine Square Scaling (Sq. Sc.) as one single function $d = b \otimes c \oplus SqSc(b \oplus c)$. The affine square scaling $SqSc$ maps the 4-bit input (x_1, \dots, x_4) to the 4-bit output (y_1, \dots, y_4) as follows:

$$\begin{aligned}
 y_1 &= x_1 & y_3 &= x_2 \oplus x_4 \\
 y_2 &= x_1 \oplus x_2 & y_4 &= x_1 \oplus x_3
 \end{aligned} \tag{2.33}$$

For the parallel multiplier and square scaling with random nibbles (r_1, r_2, r_3) , the resulting equations are given by:

$$\begin{aligned}
 d^1 &= b^1 \otimes c^1 \oplus b^1 \otimes c^2 \oplus b^2 \otimes c^1 \oplus SqSc(b^1 \oplus c^1) \oplus r_1 \\
 d^2 &= b^2 \otimes c^2 \oplus b^2 \otimes c^3 \oplus b^3 \otimes c^2 \oplus SqSc(b^2 \oplus c^2) \oplus r_2 \\
 d^3 &= b^3 \otimes c^3 \oplus b^3 \otimes c^1 \oplus b^1 \otimes c^3 \oplus SqSc(b^3 \oplus c^3) \oplus r_3
 \end{aligned} \tag{2.34}$$

where $r_3 = r_1 \oplus r_2$.

Third Stage.

This stage is similar to the second stage. The nibbles are split in 2-bit couples for further operation. The Scaling operation (Sc) replaces the similar affine Square Scaling and is executed alongside the multiplication in \mathbb{F}_{2^2} as one function $h = f \otimes g \oplus Sc(f \oplus g)$. The scaling operation maps a 2-bit input (x_1, x_2) to the 2-bit output (y_1, y_2) as follows:

$$\begin{aligned}
 y_1 &= x_1 \oplus x_2 & y_2 &= x_2
 \end{aligned} \tag{2.35}$$

For the parallel multiplier and scaling with random two-bits (r_4, r_5, r_6) , the resulting equations are given by:

$$\begin{aligned}
 h^1 &= f^1 \otimes g^1 \oplus f^1 \otimes g^2 \oplus f^2 \otimes g^1 \oplus Sc(f^1 \oplus g^1) \oplus r_4 \\
 h^2 &= f^2 \otimes g^2 \oplus f^2 \otimes g^3 \oplus f^3 \otimes g^2 \oplus Sc(f^2 \oplus g^2) \oplus r_5 \\
 h^3 &= f^3 \otimes g^3 \oplus f^3 \otimes g^1 \oplus f^1 \otimes g^3 \oplus Sc(f^3 \oplus g^3) \oplus r_6
 \end{aligned} \tag{2.36}$$

where $r_6 = r_4 \oplus r_5$.

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

Fourth Stage.

The fourth stage is composed of an inversion and two parallel multiplications in \mathbb{F}_{2^2} . The inversion Inv in \mathbb{F}_{2^2} is linear and is implemented by swapping the bits. The inversion operation maps a 2-bit input (x_1, x_2) to the 2-bit output (y_1, y_2) as follows:

$$y_1 = x_2 \qquad y_2 = x_1 \qquad (2.37)$$

The outputs of the multiplications are concatenated, denoted by \oplus in Figure 2.6, to form 4-bit values in \mathbb{F}_{2^4} . For the inversion and the parallel multipliers $l = f \otimes Inv(h)$ and $m = Inv(h) \otimes g$ the resulting equations are given by:

$$\begin{aligned} l^1 &= f^1 \otimes Inv(h^1) \oplus f^1 \otimes Inv(h^2) \oplus f^2 \otimes Inv(h^1) \\ l^2 &= f^2 \otimes Inv(h^2) \oplus f^2 \otimes Inv(h^3) \oplus f^3 \otimes Inv(h^2) \\ l^3 &= f^3 \otimes Inv(h^3) \oplus f^3 \otimes Inv(h^1) \oplus f^1 \otimes Inv(h^3) \\ m^1 &= Inv(h^1) \otimes g^1 \oplus Inv(h^1) \otimes g^2 \oplus Inv(h^2) \otimes g^1 \\ m^2 &= Inv(h^2) \otimes g^2 \oplus Inv(h^2) \otimes g^3 \oplus Inv(h^3) \otimes g^2 \\ m^3 &= Inv(h^3) \otimes g^3 \oplus Inv(h^3) \otimes g^1 \oplus Inv(h^1) \otimes g^3 \end{aligned} \qquad (2.38)$$

The two outputs of the parallel multipliers are then concatenated l being the most significant bits and m the least significant. The concatenated bits are refreshed with the random nibbles (r_7, r_8, r_9) , as follows:

$$\begin{aligned} n^1 &= l^1 \oplus m^1 \oplus r_7 \\ n^2 &= l^2 \oplus m^2 \oplus r_8 \\ n^3 &= l^3 \oplus m^3 \oplus r_9 \end{aligned} \qquad (2.39)$$

where $r_9 = r_7 \oplus r_8$.

Fifth Stage.

Stage 5 is similar to stage 4. The difference of the two stages lies in the absence of the inversion operation and the multiplications being performed in \mathbb{F}_{2^4} instead of \mathbb{F}_{2^2} . For the parallel multipliers $p = b \otimes n$ and $q = n \otimes c$ the resulting equations are given by:

$$\begin{aligned} p^1 &= b^1 \otimes n^1 \oplus b^1 \otimes n^2 \oplus b^2 \otimes n^1 \\ p^2 &= b^2 \otimes n^2 \oplus b^2 \otimes n^3 \oplus b^3 \otimes n^2 \\ p^3 &= b^3 \otimes n^3 \oplus b^3 \otimes n^1 \oplus b^1 \otimes n^3 \\ q^1 &= n^1 \otimes c^1 \oplus n^1 \otimes c^2 \oplus n^2 \otimes c^1 \\ q^2 &= n^2 \otimes c^2 \oplus n^2 \otimes c^3 \oplus n^3 \otimes c^2 \\ q^3 &= n^3 \otimes c^3 \oplus n^3 \otimes c^1 \oplus n^1 \otimes c^3 \end{aligned} \qquad (2.40)$$

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

The two outputs of the parallel multipliers are then concatenated p being the most significant bits and q the least significant. The concatenated bits are refreshed with the random nibbles (r_{10}, r_{11}, r_{12}) , as follows:

$$\begin{aligned} s^1 &= p^1 \oplus q^1 \oplus r_{10} \\ s^2 &= p^2 \oplus q^2 \oplus r_{11} \\ s^3 &= p^3 \oplus q^3 \oplus r_{12} \end{aligned} \tag{2.41}$$

where $r_{10} = r_{11} \oplus r_{12}$.

Finally, the inverse linear map (including the AES affine transformation) is performed. This linear function maps the 8-bit input (s_1^i, \dots, s_8^i) to the 8-bit output (y_1^i, \dots, y_8^i) for each share i as follows:

$$\begin{aligned} y_1^i &= s_6^i \oplus s_4^i & y_5^i &= s_8^i \oplus s_7^i \oplus s_6^i \oplus s_5^i \oplus s_4^i \\ y_2^i &= s_8^i \oplus s_4^i & y_6^i &= s_7^i \oplus s_6^i \oplus s_4^i \oplus s_3^i \oplus s_1^i \\ y_3^i &= s_7^i \oplus s_1^i & y_7^i &= s_6^i \oplus s_5^i \oplus s_2^i \\ y_4^i &= s_8^i \oplus s_6^i \oplus s_4^i & y_8^i &= s_7^i \oplus s_5^i \oplus s_2^i \end{aligned} \tag{2.42}$$

The constant 0x63 is then added to the first share.

2.7 Design III: First-Order Three-Share AES

This section describes the third S-box design of a three-share first-order AES. Compared to the three-share AES in Section 2.6, this sharing is even higher in the number of logic gates and lower in the number of register stages.

2.7.1 S-Box Sharing

The method is shown in Figure 2.7 and is divided into four stages and uses 40 random bits in total. These random bits can be re-used over all shared S-boxes in both the AES state function as the key schedule.

First Stage.

The first operation occurring in the decomposed S-box performs a change of basis through a linear map. Its sharing requires instantiating this linear map once for each share. This mapping is implemented in combinational logic and it maps the 8-bit input (a_1^i, \dots, a_8^i) to the 8-bit output (y_1^i, \dots, y_8^i) for each share $i \in \{1, 2, 3\}$ as follows:

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

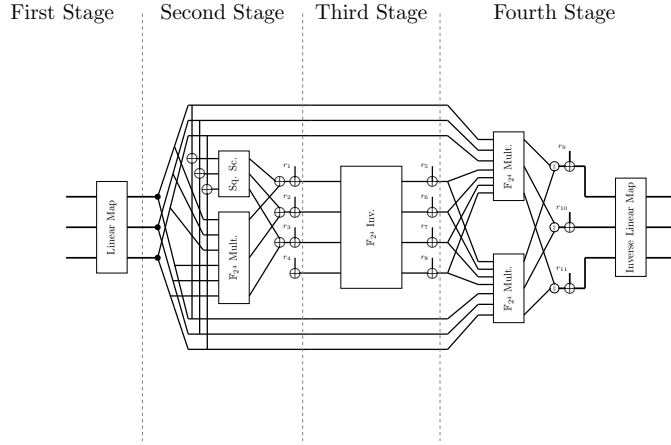


Figure 2.7: Representation of the S-box of design III. Register stages are denoted by dashed vertical lines.

$$\begin{aligned}
 y_1^i &= a_8^i \oplus a_7^i \oplus a_6^i \oplus a_3^i \oplus a_2^i \oplus a_1^i & y_5^i &= a_8^i \oplus a_5^i \oplus a_4^i \oplus a_2^i \oplus a_1^i \\
 y_2^i &= a_7^i \oplus a_6^i \oplus a_5^i \oplus a_1^i & y_6^i &= a_1^i \\
 y_3^i &= a_7^i \oplus a_6^i \oplus a_2^i \oplus a_1^i & y_7^i &= a_7^i \oplus a_6^i \oplus a_1^i \\
 y_4^i &= a_8^i \oplus a_7^i \oplus a_6^i \oplus a_1^i & y_8^i &= a_7^i \oplus a_4^i \oplus a_3^i \oplus a_2^i \oplus a_1^i
 \end{aligned} \tag{2.43}$$

Second Stage.

We consider the parallel application of nonlinear multiplication and affine Square Scaling (Sq. Sc.) as one single function $d = b \otimes c \oplus SqSc(b \oplus c)$. The affine square scaling $SqSc$ maps the 4-bit input (x_1, \dots, x_4) to the 4-bit output (y_1, \dots, y_4) as follows:

$$\begin{aligned}
 y_1 &= x_1 & y_3 &= x_2 \oplus x_4 \\
 y_2 &= x_1 \oplus x_2 & y_4 &= x_1 \oplus x_3
 \end{aligned} \tag{2.44}$$

For the parallel multiplier and square scaling with random nibbles (r_1, r_2, r_3, r_4) , the resulting equations are given by:

$$\begin{aligned}
 d^1 &= b^1 \otimes c^1 \oplus b^1 \otimes c^2 \oplus b^2 \otimes c^1 \oplus SqSc(b^1 \oplus c^1) \oplus r_1 \\
 d^2 &= b^2 \otimes c^2 \oplus b^2 \otimes c^3 \oplus b^3 \otimes c^2 \oplus SqSc(b^2 \oplus c^2) \oplus r_2 \\
 d^3 &= b^3 \otimes c^3 \oplus b^3 \otimes c^1 \oplus b^1 \otimes c^3 \oplus SqSc(b^3 \oplus c^3) \oplus r_3 \\
 d^4 &= r_4
 \end{aligned} \tag{2.45}$$

where $r_4 = r_1 \oplus r_2 \oplus r_3$. Extra randomness is used to temporarily increase the number of shares in order to improve latency.

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

Third Stage.

The third stage is composed of an inversion in \mathbb{F}_{2^4} . For the inversion Inv in \mathbb{F}_{2^4} , the resulting equations are given by:

$$\begin{aligned}
 y_1 &= x_1 \otimes x_3 \oplus x_1 \otimes x_4 \oplus x_2 \otimes x_3 \otimes x_4 \oplus x_2 \otimes x_4 \oplus x_3 \\
 y_2 &= x_1 \otimes x_3 \otimes x_4 \oplus x_1 \otimes x_4 \oplus x_2 \otimes x_4 \oplus x_3 \oplus x_4 \\
 y_3 &= x_1 \otimes x_2 \otimes x_4 \oplus x_1 \otimes x_3 \oplus x_1 \oplus x_2 \otimes x_3 \oplus x_2 \otimes x_4 \\
 y_4 &= x_1 \otimes x_2 \otimes x_3 \oplus x_1 \oplus x_2 \otimes x_3 \oplus x_2 \otimes x_4 \oplus x_2
 \end{aligned} \tag{2.46}$$

As the equations for the sharing are too large, we simply explain how to share a cubic and quadratic term. Consider the cubic term xyz , the i^{th} share is calculated as follows, where the convention is used that the superscripts wrap around at four.

$$\begin{aligned}
 xyz^i &= x^i y^i z^i \oplus x^i y^{i+1} z^i \oplus x^i y^i z^{i+1} \oplus x^i y^{i+1} z^{i+1} \oplus x^i y^{i+1} z^{i+2} \oplus x^i y^{i+2} z^{i+1} \\
 &\oplus x^i y^i z^{i+2} \oplus x^i y^{i+2} z^i \oplus x^i y^{i+2} z^{i+2} \oplus x^{i+2} y^{i+1} z^i \oplus x^{i+1} y^{i+2} z^i \\
 &\oplus x^{i+2} y^i z^{i+1} \oplus x^{i+1} y^i z^{i+2} \oplus x^{i+2} y^{i+1} z^{i+1} \oplus x^{i+2} y^{i+2} z^{i+1} \\
 &\oplus x^{i+2} y^{i+1} z^{i+2}
 \end{aligned} \tag{2.47}$$

The i^{th} share of a quadratic term xy is calculated as follows:

$$xy^i = x^i y^i \oplus x^i y^{i+1} \oplus x^{i+1} y^i \oplus x^i y^{i+2} \oplus x^{i+2} y^i \tag{2.48}$$

The linear terms are added share-wise.

The output of the inversion (e^1, \dots, e^4) is then refreshed with the random nibbles (r_5, r_6, r_7, r_8) , as follows:

$$\begin{aligned}
 f^1 &= e^1 \oplus r_5 \\
 f^2 &= e^2 \oplus r_6 \\
 f^3 &= e^3 \oplus r_7 \\
 f^4 &= e^4 \oplus r_8
 \end{aligned} \tag{2.49}$$

where $r_8 = r_5 \oplus r_6 \oplus r_7$.

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

Fourth Stage.

In the fourth stage we have two parallel multipliers over \mathbb{F}_{2^4} . For the parallel multipliers $g = b \otimes f$ and $h = f \otimes c$ the resulting equations are given by:

$$\begin{aligned}
g^1 &= b^1 \otimes f^1 \oplus b^1 \otimes f^2 \oplus b^1 \otimes f^3 \oplus b^2 \otimes f^1 \\
g^2 &= b^2 \otimes f^2 \oplus b^2 \otimes f^3 \oplus b^2 \otimes f^4 \oplus b^3 \otimes f^2 \\
g^3 &= b^3 \otimes f^3 \oplus b^3 \otimes f^4 \oplus b^3 \otimes f^1 \oplus b^1 \otimes f^4 \\
h^1 &= f^1 \otimes c^1 \oplus f^1 \otimes c^2 \oplus f^1 \otimes c^3 \oplus f^2 \otimes c^1 \\
h^2 &= f^2 \otimes c^2 \oplus f^2 \otimes c^3 \oplus f^2 \otimes c^4 \oplus f^3 \otimes c^2 \\
h^3 &= f^3 \otimes c^3 \oplus f^3 \otimes c^4 \oplus f^3 \otimes c^1 \oplus f^1 \otimes c^4
\end{aligned} \tag{2.50}$$

The two outputs of the parallel multipliers are then concatenated g being the most significant bits and h the least significant. The concatenated bits are refreshed with the random nibbles (r_9, r_{10}, r_{11}) , as follows:

$$\begin{aligned}
k^1 &= g^1 \oplus h^1 \oplus r_9 \\
k^2 &= g^2 \oplus h^2 \oplus r_{10} \\
k^3 &= g^3 \oplus h^3 \oplus r_{11}
\end{aligned} \tag{2.51}$$

where $r_{11} = r_{10} \oplus r_9$.

Finally, the inverse linear map (including the AES affine transformation) is performed. This linear function maps the 8-bit input (k_1^i, \dots, k_8^i) to the 8-bit output (y_1^i, \dots, y_8^i) for each share i as follows:

$$\begin{aligned}
y_1^i &= k_6^i \oplus k_4^i & y_5^i &= k_8^i \oplus k_7^i \oplus k_6^i \oplus k_5^i \oplus k_4^i \\
y_2^i &= k_8^i \oplus k_4^i & y_6^i &= k_7^i \oplus k_6^i \oplus k_4^i \oplus k_3^i \oplus k_1^i \\
y_3^i &= k_7^i \oplus k_1^i & y_7^i &= k_6^i \oplus k_5^i \oplus k_2^i \\
y_4^i &= k_8^i \oplus k_6^i \oplus k_4^i & y_8^i &= k_7^i \oplus k_5^i \oplus k_2^i
\end{aligned} \tag{2.52}$$

The constant 0x63 is then added to the first share.

2.8 Security

In this section we argue the first-order probing security of the three designs. We show this by arguing that the designs are threshold implementations, see Definition 1. We refer to Dhooghe *et al.* [DNR19] for the proof that a threshold implementation is first-order robust probing secure.

The shared S-boxes are first-order probing secure due to the extra randomness added to each register stage. This refreshing works as follows. Given an input (a^1, \dots, a^s) , an arbitrary shared map \bar{F} , and randomness r^1, \dots, r^{s-1} , refreshing is done as follows, for $i \in \{1, \dots, s-1\}$

$$a^i = F^i(a^1, \dots, a^s) \oplus r^i, \quad a^s = F^s(a^1, \dots, a^s) \oplus r^1 \oplus \dots \oplus r^{s-1}. \tag{2.53}$$

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

Lemma 2.8.1. *The refreshing following Equation (2.53) gives a uniform output.*

Proof. We show that the function, taking (a^1, \dots, a^s) and (r^1, \dots, r^{s-1}) as input and (a^1, \dots, a^s) , $F^i(a^1, \dots, a^s) \oplus r^i$ for $i \in \{1, \dots, s-1\}$, and $F^s(a^1, \dots, a^s) \oplus r^1 \oplus \dots \oplus r^{s-1}$ as output, is invertible. Removing the (a^1, \dots, a^s) then gives a balanced (or uniform) output of the refreshing detailed in Equation 2.53.

The derivation is straightforward. Since (a^1, \dots, a^s) is given in the output, one can calculate $F^i(a^1, \dots, a^s)$ for $i \in \{1, \dots, s\}$. Subtracting this from the output (a^1, \dots, a^{s-1}) then gives (r^1, \dots, r^{s-1}) showing the map is invertible. \square

Theorem 2.8.2. *Designs I, II, and III from Sections 2.5, 2.6, and 2.7 are threshold implementations as given by Definition 1.*

Proof. First, each design uses a changing of the guards method. We refer to Theorem 2.3.1 for the proof that the shared input and output of each shared S-box is uniform.

Since the linear layers of the construction are evidently non-complete, since they work share-wise, and uniform, since the unshared linear functions are permutations, these comply to the properties from Definition 1.

We then show that the shared S-box from designs I, II, and III are first-order probing secure. Since a probe in the designs can only view one shared S-box, it suffices to show that each stage in the shared S-box complies to the threshold implementation properties.

Each stage in the shared S-box either maps a part of the input to the output (such as in the outer wires of Figure 2.5) or the output is shared using the randomness \bar{r} . Due to the changing of the guards structure this randomness \bar{r} is joint uniform with the input of the shared S-box. From Lemma 2.8.1, we find that each stage of the shared S-box of design I, II, or III is uniform.

The non-completeness is verified on sight from the equations given in Sections 2.5, 2.6, 2.7. More specifically, it can be seen that each multiplication or inversion in the designs is shared in a non-complete way.

As a result, the shared S-boxes from designs I, II, and III are itself threshold implementations and thus first-order robust probing secure. \square

2.9 Comparison Between the Designs

We compare the three designs in terms of the number of shares, stages, logic gates, and randomness cost. The comparison can be viewed in Table 2.1. We give a more detailed analysis for each design separately. While we only provide rough unoptimized numbers for the number of logic gates, they still allow for a comparison between the designs.

Design I. We give the cost of each stage separately.

- The first stage requires 48 XOR gates and 16 registers.

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

- The second stage requires 38 XOR gates, 4 \mathbb{F}_{2^4} multipliers, and 32 registers.
- The third stage requires 26 XOR gates, 4 \mathbb{F}_{2^2} multipliers, and 32 registers.
- The fourth stage requires 20 XOR gates, 16 \mathbb{F}_{2^2} multipliers, and 32 registers.
- The fifth stage requires 40 XOR gates, 16 \mathbb{F}_{2^4} multipliers, and 32 registers.
- The sixth stage requires 54 XOR gates.

Concerning the randomness cost, design I requires 128 bits to share the plaintext, 128 bits to share the key, 8 bits to instantiate the changing of the guards, and 54 bits for the shared S-box (which are re-used every S-box). Thus, the design requires a total of 318 random bits per encryption.

Design II. We give the cost of each stage separately.

- The first stage requires 72 XOR gates and 24 registers.
- The second stage requires 69 XOR gates, 9 \mathbb{F}_{2^4} multipliers, and 36 registers.
- The third stage requires 33 XOR gates, 9 \mathbb{F}_{2^2} multipliers, and 42 registers.
- The fourth stage requires 60 XOR gates, 36 \mathbb{F}_{2^2} multipliers, and 36 registers.
- The fifth stage requires 175 XOR gates and 36 \mathbb{F}_{2^4} multipliers.

Concerning the randomness cost, design II requires 256 bits to share the plaintext, 256 bits to share the key, 16 bits to instantiate the changing of the guards, and 36 bits for the shared S-box (which are re-used every S-box). Thus, the design requires a total of 564 random bits per encryption.

Design III. We give the cost of each stage separately.

- The first stage requires 72 XOR gates and 24 registers.
- The second stage requires 69 XOR gates, 9 \mathbb{F}_{2^4} multipliers, and 40 registers.
- The third stage requires 480 XOR gates, 712 AND gates, and 40 registers.
- The fourth stage requires 115 XOR gates and 48 \mathbb{F}_{2^4} multipliers.

Concerning the randomness cost, design III requires 256 bits to share the plaintext, 256 bits to share the key, 16 bits to instantiate the changing of the guards, and 40 bits for the shared S-box (which are re-used every S-box). Thus, the design requires a total of 568 random bits per encryption.

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

	shares	stages	XOR	AND	reg.	total randomness
Design I (2.5)	2	6	1546	880	144	318
Design II (2.6)	3	5	2299	1980	138	564
Design III (2.7)	3 [†]	4	2788	2992	104	568

Table 2.1: The number of shares, register stages, XORs, AND gates, and bit registers the shared S-box requires and how many random bits the entire AES design needs, including the masking of the plaintext and key.

[†] One stage in the design works on four shares.

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

Chapter 3

Algorithmic Verification Properties

3.1 Overview

The aim of this chapter is to specify algorithmic verification properties which can be used to verify whether the sharing is correctly implemented. The goal is twofold to shorten and ease the design cycle and to reduce the need for lab evaluation experiments.

We explain these properties such that it should be clear how to verify them given a netlist. The properties also return where the algorithmic errors occur such that the designer can correct them.

Whereas in the previous chapter only first-order security was discussed, with higher-order security to be added in the future, this chapter handles general higher-order security.

3.2 List of Properties

We consider the verification for probing security order d . We consider the following three properties to verify a sharing of the AES was implemented safely.

- Register to register non-completeness of order d .
- Input and output uniformity.
- Glitch-extended probing security of order d .

To reduce complexity, it suffices to verify the properties of the shared AES S-box instead of the entire design. The above properties capture higher-order univariate and single cipher-round multivariate attacks, but do not cover multi-round attacks as such a verification would be too computationally intensive. We also assume the sharing of the linear layers is done share-wise and as such need

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

no explicit verification. However, we note to take care when placing registers before or after the linear layer in order to avoid non-completeness issues.

3.3 Constructing the Structural Model

The verification properties in this chapter are described to operate on netlists. These can be extracted from HDL written in VHDL or Verilog. This netlist should be parsed, and a structural model of the circuit in the shape of a Binary Decision Diagram (BDD) should be built using wires, pins, and gates as objects connected to one another.

Additionally, a header should be made in which important data for the tool is specified. The data should include the names of the sensitive inputs, the outputs of every layer of registers, random numbers, and the outputs to be analyzed.

3.4 Verifying Non-Completeness

We describe the verification of non-completeness using the work by Arribas *et al.* [ANR18]. The first-order non-completeness property was already discussed in Definition 1 in the previous chapter. We recall the mathematical definition of the more general d^{th} -order non-completeness property.

Definition 3 (d^{th} -Order Non-Completeness). The sharing \bar{F} is said to be d^{th} -order non-complete if any function in d or fewer components of the shared function F^i depends on at most $s_x - 1$ input shares.

The above property needs to hold between every two register stages. The verification of “first-order non-completeness” is done by collecting the input dependencies of every primary output (at a register stage) with a recursive search through the binary decision tree. For each output, one should check whether all shares for the same variable are included. This is to be repeated for every register stage. If this is the case, the design fails non-completeness.

We provide the properties in a matrix notation. Consider the function in the netlist has m outputs and n inputs. Construct a binary matrix

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{bmatrix}. \quad (3.1)$$

In case the i^{th} output depends on the j^{th} input, then $a_{i,j} = 1$. The designer specifies which variables are secret and which are the shares. The tool then checks for each row in the matrix whether all shares of a value occur in the matrix. In case this is true, then the sharing fails the verification and the tool returns the output variable for which non-completeness failed.

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

This is generalized to d^{th} -order non-completeness by considering all possible combinations of d outputs. The difference now is that the dependencies of the d outputs are combined. If any combination of d output variables fail non-completeness, the design fails the property. For the matrix method. The tool simply builds the same matrix but then considers all combinations of d rows. Rows are combined by taking the OR between the values on the same column. Afterwards, the regular non-completeness check is performed as specified above.

3.5 Verifying Uniformity

We describe the verification of uniformity as described in the work by Arribas *et al.* [ANR18]. The definition of uniformity was given in Definition 1 from the previous chapter.

The uniformity property is verified between the shared input and output of the shared AES S-box. To verify the property, one should calculate the output distribution of the shared S-box by simulating all possible shared inputs. The frequencies of occurrence of each output vector should be calculated for the different input sharings. If all frequencies are the same for every unshared input, then the module is said to preserve uniformity.

The property is evaluated by checking whether for certain unshared inputs, every shared output has the same frequency of occurrence when all possible sharings of this input are evaluated. This is repeated for every possible unshared input together with, if added, all random bits specified. This is specified in Algorithm 1. This algorithm uses the function `Frequency` which takes an array and an integer and returns how many times the integer occurs in the array.

Algorithm 1: Uniformity Verification.

Input: m to n -bit S-box \bar{S} taking r -bit randomness.

Output: Boolean denoting the uniformity of \bar{S} .

```

for  $\bar{x} \in \mathbb{F}_2^m$  do
  for  $\bar{r} \in \mathbb{F}_2^r$  do
    output.append( $\bar{S}_{\bar{r}}(\bar{x})$ );
  end
end
for  $i = 1$  to  $2^n$  do
  if  $\text{Frequency}(\text{output}, i) \neq 2^{m-n}$  then
    return False;
  end
end
return True;

```

In case the verification fails, the designer should be notified that the implemented shared S-box fails uniformity.

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

3.6 Verifying Probing Security

The verification of the d^{th} -order probing security property is explained from the work by Knichel *et al.* [KSM20]. The properties explored in this section have the purpose to provide some necessary conditions the sharing requires to resist some basic forms of higher-order univariate or multivariate attacks.

We recall from Section 2.2.3, that in the d -probing security model, an adversary can probe (observe) up to d intermediate values during the processing of sensitive information. Probing security is defined as the probes being statistically independent of any sensitive (random) input. More precisely, the joint distribution of the considered set of probes has to be independent of the joint distribution of all sensitive inputs. This can be formally defined as:

Definition 4 (d -Probing Security). A shared S-box \bar{S} with secret input set $X \in \mathbb{F}_2^n$ is d -probing secure, if and only if for any observation set Q containing d wires, X is statistically independent of the observation set, i.e., the following condition holds:

$$\Pr[Q|X] = \Pr[Q]. \quad (3.2)$$

Since the design is made for a hardware implementations, physical default in terms of glitches should be taken into consideration as discussed in Section 2.2.3. Therefore, a probe not only accesses the field element of the driving gate but also can access all stable field elements of the last synchronization points which drive the probed gate (having a path to the driving gate in the circuit graph). The verification considers this extension called “glitch-extended probing”.

The verification takes all possible d combinations of wires across the entire shared AES S-box and checks whether these have statistical dependence on the unshared input of the S-box. This can be achieved by simulating the shared S-box for all possible shared inputs and verifying the probabilistic property of Definition 4 for each d -set of wires. For this verification, we go over the definition of a probability mass function and joint mass functions. The probability mass function provides the probability of all possible values for a set of discrete random variables based on their probability distribution

Definition 5 (Probability Mass Function). Let \mathbf{X} be a set of discrete random variables. The probability mass function $p_{\mathbf{X}}(x)$ is defined as:

$$p_{\mathbf{X}}(x) = \Pr[\mathbf{X} = x]. \quad (3.3)$$

Based on this, given two arbitrary sets of discrete random variables, the joint probability mass function between these two variable sets then is defined as follows.

Definition 6 (Joint Probability Mass Function). Let \mathbf{X}, \mathbf{Y} be two sets of discrete random variables. The joint probability mass function $p_{\mathbf{X}, \mathbf{Y}}(x, y)$ is defined as:

$$p_{\mathbf{X}, \mathbf{Y}}(x, y) = \Pr[\mathbf{X} = x \text{ and } \mathbf{Y} = y]. \quad (3.4)$$

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

The above mass functions should be calculated using histograms going over all \mathbf{X} , \mathbf{Y} and pairs (\mathbf{X}, \mathbf{Y}) for the joint probability mass function.

Verifying statistical dependence is then done using the following theorem as outlined by Knichel *et al.* [KSM20].

Theorem 3.6.1. *Let \mathbf{X} , \mathbf{Y} be two sets of binary random variables. Then \mathbf{X} and \mathbf{Y} are statistically independent, if and only if $p_{\mathbf{X}', \mathbf{Y}'}(a, b) = p_{\mathbf{X}'}(a) \cdot p_{\mathbf{Y}'}(b)$ for any fixed values a and b and every possible combination of $\mathbf{X}' \subset \mathbf{X}$ and $\mathbf{Y}' \subset \mathbf{Y}$.*

The above theorem can then be used for a verification algorithm for probing security. This is outlined in Algorithm 2. Since the shared circuit consists of binary variables, verifying the mass distributions $p_{\mathbf{X}}(a)$ for $a = 1$ is sufficient.

Algorithm 2: *d*-Probing Verification.

Input: \mathbf{X} – Set of n secret variables.

Output: Either True or a set of probes failing *d*-probing security.

```

foreach probing set  $\mathbf{Q}$  with  $|\mathbf{Q}| = d$  do
  for  $t = 1$  to  $n$  do
    foreach  $\mathbf{X}' \subset \mathbf{X}$  with  $|\mathbf{X}'| = t$  do
      if  $p_{\mathbf{Q}, \mathbf{X}'}(1, 1) \neq p_{\mathbf{Q}}(1) \cdot p_{\mathbf{X}'}(1)$  then
        return  $\mathbf{Q}$ ;
      end
    end
  end
end
return True;

```

In case the verification finds a flaw, it should return the set of probes which provoked it. That way the designer can debug the implementation. The tool can choose to either stop after one flaw is found or return all possible flaws.

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

Chapter 4

Conclusion

We explained the key points which should be included in the standardization effort for threshold cryptography on a single device. Additionally, we proposed some solutions for the first two parts.

The first part involves the proposition of several maskings of the AES. We believe a standard way to mask the AES is beneficial for both industry as academia to either ease the design cycle or to have a standard research object. The second part involves algorithmic properties to verify whether a netlist has some cryptographic qualities. These properties can be transformed into a (potentially open source) tool which can be improved by both industry as academia. Over time, we believe such a tool can grow out to significantly reduce costs in industry as well as push forward research.

Currently, we did not discuss practical verification such as test vector leakage assessment. However, we believe standard lab practices and testing procedures should be included in the standard.

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

Bibliography

- [ANR18] V. Arribas, S. Nikova, and V. Rijmen. Vermi: Verification tool for masked implementations. In *25th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2018, Bordeaux, France, December 9-12, 2018*, pages 381–384. IEEE, 2018. doi:10.1109/ICECS.2018.8617841.
- [BDZ20] T. Beyne, S. Dhooghe, and Z. Zhang. Cryptanalysis of masked ciphers: A not so random idea. In *ASIACRYPT 2020, Part I*, LNCS, pages 817–850. Springer, Heidelberg, Dec. 2020. doi:10.1007/978-3-030-64837-4_27.
- [BGN⁺14] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen. Higher-order threshold implementations. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of LNCS, pages 326–343. Springer, Heidelberg, Dec. 2014. doi:10.1007/978-3-662-45608-8_18.
- [Can05] D. Canright. A very compact S-box for AES. In J. R. Rao and B. Sunar, editors, *CHES 2005*, volume 3659 of LNCS, pages 441–455. Springer, Heidelberg, Aug. / Sept. 2005. doi:10.1007/11545262_32.
- [CDG⁺13] J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, and P. Rohatgi. Test vector leakage assessment (tvla) methodology in practice, 2013.
- [CJRR99] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of LNCS, pages 398–412. Springer, Heidelberg, Aug. 1999. doi:10.1007/3-540-48405-1_26.
- [Dae17] J. Daemen. Changing of the guards: A simple and efficient method for achieving uniformity in threshold sharing. In W. Fischer and N. Homma, editors, *CHES 2017*, volume 10529 of LNCS, pages 137–153. Springer, Heidelberg, Sept. 2017. doi:10.1007/978-3-319-66787-4_7.
- [DNR19] S. Dhooghe, S. Nikova, and V. Rijmen. Threshold implementations in the robust probing model. In B. Bilgin, S. Petkova-

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

- Nikova, and V. Rijmen, editors, *Proceedings of ACM Workshop on Theory of Implementation Security Workshop, TIS@CCS 2019, London, UK, November 11, 2019*, pages 30–37. ACM, 2019. doi:10.1145/3338467.3358949.
- [DR20] J. Daemen and V. Rijmen. *The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition*. Information Security and Cryptography. Springer, 2020.
- [FGP⁺18] S. Faust, V. Grosso, S. M. D. Pozo, C. Paglialonga, and F.-X. Standaert. Composable masking schemes in the presence of physical defaults & the robust probing model. *IACR TCHES*, 2018(3):89–120, 2018. doi:10.13154/tches.v2018.i3.89-120. <https://tches.iacr.org/index.php/TCHES/article/view/7270>.
- [FIPS197] National Institute of Standards and Technology. Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce, Nov. 2001.
- [GP99] L. Goubin and J. Patarin. DES and differential power analysis (the “duplication” method). In Çetin Kaya. Koç and C. Paar, editors, *CHES’99*, volume 1717 of *LNCS*, pages 158–172. Springer, Heidelberg, Aug. 1999. doi:10.1007/3-540-48059-5*15.
- [ISW03] Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, Heidelberg, Aug. 2003. doi:10.1007/978-3-540-45146-4*27.
- [KSM20] D. Knichel, P. Sasdrich, and A. Moradi. SILVER - statistical independence and leakage verification. In S. Moriai and H. Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I*, volume 12491 of *Lecture Notes in Computer Science*, pages 787–816. Springer, 2020. doi:10.1007/978-3-030-64837-4.26.
- [NRR06] S. Nikova, C. Rechberger, and V. Rijmen. Threshold implementations against side-channel attacks and glitches. In P. Ning, S. Qing, and N. Li, editors, *ICICS 06*, volume 4307 of *LNCS*, pages 529–545. Springer, Heidelberg, Dec. 2006.
- [RBN⁺15] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede. Consolidating masking schemes. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 764–783. Springer, Heidelberg, Aug. 2015. doi:10.1007/978-3-662-47989-6*37.

Item 2: Proposal by S. Dhooghe, S. Nikova, V. Rijmnen (2021-Jun-01)

- [WO19] C. Whitnall and E. Oswald. A critical analysis of ISO 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules'). In S. D. Galbraith and S. Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 256–284. Springer, Heidelberg, Dec. 2019. [doi:10.1007/978-3-030-34618-8_9](https://doi.org/10.1007/978-3-030-34618-8_9).

Item 3: Comments by Amir Moradi (2021-Aug-20)

From: Amir Moradi
Sent: Friday, August 20, 2021 06:35

Dear Masked Circuits team,

Many thanks for your effort and dedication to this topic, which I fully support.

I have some comments which I would like to share with you:

I personally published several papers presenting hardware masked AES (and other lightweight ciphers) which are based on manual optimizations WRT area, latency, randomness, etc. The designs proposed by the KU Leuven team (sent around) are also from the same nature, i.e., manually-crafted and optimized designs. These are for sure useful, but I have some concerns:

- These designs are usually for a certain security order, and cannot be easily extended to higher orders.
- They are for particular design architectures, e.g., round-based, serial, etc. If the engineer (company) wants to adjust the implementation to their needs (e.g., they need a column-serial design), any modification on the design architecture may lead to a security vulnerability, when the engineer does not have enough expertise.
- The (probing) security of such designs usually cannot be proven. There are some open-source tools (e.g., SILVER <https://eprint.iacr.org/2020/634>) to examine the security of hardware designs, but they cannot handle large circuits. For example, for AES, first-order designs (with 2 shares) can still be evaluated by SILVER, but not larger ones, even a first-order design with 3 shares.

All these issues convince me to think about another direction when the standardization (or even recommendation for secure implementations) is the main topic. I think, the designs which are constructed by secure composable gadgets are a potential solution. Precisely, I refer to notions like PINI (<https://eprint.iacr.org/2018/438>) and gadgets like HPC2 (<https://eprint.iacr.org/2020/185>). If the circuit is built by such gadgets (which are designed for any arbitrary security order), the security of the construction can be proven (at any security order). Of course, the research is still ongoing in this direction with optimizing the gadgets (WRT area, latency, demand for fresh randomness, etc.) and developing

Item 3: Comments by Amir Moradi (2021-Aug-20)

tools which can translate unprotected designs to their masked variant using those gadgets in an automated way. As an example, AGEMA has been developed for this purpose (<https://eprint.iacr.org/2021/569>). Hence, any engineer (who just knows VHDL/Verilog) can construct provably-secure masked implementations.

If we standardize gadgets and how the circuits should be made by those gadgets (this can include suggestions to use open-source tools like AGEMA), the above-mentioned concerns would not be valid anymore. The drawback of gadget-constructed circuits is their overhead (mainly latency and amount of required fresh masks). I am sure, this will be improved in the near future. In short, manually-crafted designs always win considering the efficiency parameters, but their security is always questionable, particularly if the design is even slightly modified. This forces everybody to focus on experimental analyses (leakage assessment, t-test, etc.) to have a feeling about the security/vulnerability of such manually-crafted designs.

Sorry for the long email. Of course, I am available for any discussions or even meetings for this matter.

Regards,
Amir

Item 4: Comments by Francois-Xavier Standaert (2021-Aug-30)

From: Francois-Xavier Standaert
Date: Monday, August 30, 2021 at 16:02

[...]

I had a look at the proposal. My high-level concern is that the note describes a first-order secure scheme, with limited possibilities of extension towards higher orders and formal analyses, which I think would have made sense 5 years ago but falls a bit behind the current state-of-the-art.

Based on the discussions during the workshop, I would favor solutions that scale at arbitrary security orders, of which the security can be proven/analyzed for full implementations (i.e., including composability reasoning and tested thanks to formal methods). There are various options for this purpose in the literature (in terms of composability notions, tools, ...), the choice of which depending on the type of implementation (e.g., specialized to hardware/ specialized to software/ applicable to both). I also think the concrete evaluation procedure must go significantly beyond leakage detection.

These are of course only high-level comments. Depending on whether you find them relevant or not, I am of course happy to detail them more or to discuss them during meetings. I understand you also need more technical feedback but since it's been a while the workshop took place, I think it makes sense to clarify the general ideas/goals again.

Best regards,

François-Xavier Standaert

Item 5: Comments by Patrick Haddad (2021-Sep-03)



Potential interest:

As clearly explained in [DNR21] and formalized in [FGP+18], the glitch-extended probing model allows obtaining all the inputs leading to either a target gate or wire in a circuit.

Such model relies on very strong assumptions. Indeed, the knowledge of the attacked implementation must be very precise, the targeted internal variables must leak sufficiently and the side channel experimental setup must be able to acquire this leakage with a very good accuracy.

Assuming this adversary model, robustness against a chosen order of side channel attacks remains achievable thanks to the implementation of masking schemes composed of several layers of Boolean functions whose outputs are synchronized by means of registers which are assumed to stop glitches. Those masking schemes are called "threshold implementations". However, such masking schemes have strong negative impact on performances and complexity of block-ciphers hardwired in integrated circuits. Even the fastest design proposed in [DNR21] remains too slow vs. performance needs for many applications where a robust implementation is required.

As explained in [DNR21], threshold implementation is one of numerous approaches to randomly sharing a circuit: There are other ways to achieve side channel robustness under less stringent yet realistic adversary model assumptions. Assessing and defining a realistic adversary model for a given chip, technology and application is a key challenge, including for threshold implementations for which it translates into selecting the proper order of protection to target.

Robustness against side channel attacks isn't the only important consideration. Performances and hardware-complexity are key considerations that can't be ignored. To minimize impact on performances and complexity, the adversary model should be driven by realistic and practical threats faced by the final application of the circuit. The countermeasure often needs to be tailored to the actual threats in order to minimize its impact on performances, cost or power-consumption of the circuit.

In conclusion, even if the glitch-extended probing model is very attractive and well-suited to applications that can bear its resulting performance and cost impacts, it can be an overly-aggressive (vs. practical) threat model resulting in masking schemes that are unfortunately often too slow or complex for adoption in real-life applications.

Standardizing a specific adversary model and the corresponding protection without considering the characteristics of each application is likely to hinder technical innovation and rule-out alternatives better suited for many applicative use cases and constraints.

Feasibility potential:

As explained in [DNR21], A n-order threshold implementations only provides a theoretical guarantee that the implementation is robust against side channel attacks whose order is lower than $n+1$.

Tools used by semiconductor companies allows to check the design compliance with the intended masking scheme. They will never guarantee the practical unfeasibility of attacks that are out of

Item 5: Comments by Patrick Haddad (2021-Sep-03)



the adversary model. As an example, the feasibility of a 2nd order attack will depend on the information quality provided by the side channel and it is impacted by the technologic node, the chip layout and the experimental setup used for the side channel measurements. Many parameters making attacks practical in the non-simulated world can't be verified using the design flow tools widely used by the semiconductor industry.

As a conclusion, practical robustness of a masked circuits is what matters. For sure theoretical analysis are extremely useful but they are based on adversary models that must be experimentally validated. Moreover, the practical unfeasibility of the attacks remaining theoretical feasible must also be practically validated.

The results of the practical validation depend on deployment characteristics and they can justify differentiated approaches across platforms.

Bibliography

[DNR21] S. Dhooghe, S. Nikova, V. Rijmen
Threshold Cryptography on a Single Device by Means of Threshold Implementations.

[FGP+18] S. Faust, V. Grosso, S. M. D. Pozo, C. Paglialonga, and F.-X. Standaert.
Composable masking schemes in the presence of physical defaults & the robust probing model.
IACR TCHES, 2018(3):89{120,2018. doi:10.13154/tches.v2018.i3.89-120.
<https://tches.iacr.org/index.php/TCHES/article/view/7270>.

Item 6: Comments by Elke De Mulder (2021-Sep-05)

From: De Mulder, Elke
Sent: Sunday, September 5, 2021 20:04

[...]

On behalf of Rambus, we would like to submit our comments to call 2021a. Please find them below.

Best regards,
Elke De Mulder

1. Potential Interest

a. Are the glitch-extended probing model and the corresponding masking schemes addressing problems of interest for the industry and government stakeholders?

We could see the glitch-extended probing model as an example, but it may be too restrictive for practical use, i.e., could get too expensive depending on the use case. Also, while we could see it as an example, we would be concerned that if it was presented as a guidance, it may become the de-facto solution to adhere to the standard.

b. What orders of protection are pertinent to consider for standardization?

This should be left to the end user and the use case at hand.

c. Will the semiconductor industry adopt this type of schemes in their product lines to supply the market with products implementing them?

As an IP provider, we do not directly make the final decision on what goes into a product, at most we give some implicit guidance by providing a set of solutions. This means that we will adopt the standard if the customer push is there. If not, only in case the standard makes sense security/overhead wise, we would consider adopting it. Anything going into a product needs justification. In that sense it is also important that the standard remains flexible since not all end products have the same security requirements.

2. Feasibility potential:

a. Is there a stack of available design and production tools, widely used by the semiconductor industry, that can transfer the logical netlists into silicon gate layouts, on major hardware platforms of interest (FPGA, ASIC, etc.), while preserving the needed algorithmic properties and ensuring the needed implementation assumptions, so that minimal side-channel testing would be sufficient for the verification of the intended security guarantees?

In our experience the commercial tools do not lend themselves to minimizing side-channel testing in an obvious manner. One reason for that could be that there are no clear criteria set, so standardization may be helpful in that respect. We believe theoretical testing/property setting could be feasible for small components but remain doubtful that practical tests could be

Item 6: Comments by Elke De Mulder (2021-Sep-05)

completely rendered unnecessary at a higher level. Current academic tools may be too restrictive, and extensive alterations would be necessary to be useful in an industry setting. We would also be concerned about the added cost of those features in commercial tools, particularly since the market is limited. While full integration may be too expensive, there may be some security properties that could be turned into click-button options.

b. Considering practical attacks, are there identifiable deployment characteristics that justify differentiated security profiles across platforms, masking orders, and other properties?

A secret's lifetime and the value being protected are important stakeholders here. Using the number of traces to attack as the sole criteria is only justifiable in low noise settings. Thirdly, ease/cost of exploitation should be considered in differentiation of security profiles, e.g. if (the typical easier exploitable) remote attacks are a realistic possibility for the product under consideration, side-channel attacks may fall into a lower security profile than in a situation where those remote attacks are not a possibility.

Additionally, we would like to put the focus on certain criteria we think are important. We prefer a solution to be as flexible as possible, which means, as much as possible, platform independent (ASIC, FPGA, microcontroller). We prefer high useability and low documentation/guidance requirements. Also, security testability must be provided. Will there be a testing standard? We think that is of utmost importance. Is the end user going to be asked to supply seeding or will there be an integrated source for random data? Will non-crypto PRNGs (pseudo RNG) be allowed to derive random shares from the seed at runtime or would there only be room for standardized (crypto) PRNGs to derive shares for the side-channel countermeasures?

Lastly, we want to reiterate that suggestions may stick: we would rather not see examples turn into de-facto standards or that the standard is so strict that market adoption of future innovations is discouraged. To give an example, after AES was standardized, academia continued research into block ciphers and improved designs significantly, but the improved ciphers have had very little commercial impact because they are not NIST-approved ciphers. If NIST had standardized one specific masking scheme 5 or 10 years ago, chances are that randomness reduction techniques invented since (which improve masking efficiency without sacrificing security) would never have found a commercial application. The efficiency of masking schemes is constantly evolving and good choices are very implementation / use-case specific. Any NIST standard should encourage corporate R&D departments to be free to choose the most efficient schemes for a product if the security requirements for the product are fulfilled.

**** This message and any attachments are for the sole use of the intended recipient(s). It may contain information that is confidential and privileged. If you are not the intended recipient of this message, you are prohibited from printing, copying, forwarding or saving it. Please delete the message and attachments and notify the sender immediately. ****

Rambus Inc.

Item 7: Comments by Sylvain Guilley (2021-Sep-06)

Security Level Assessment of Masking Techniques

Sylvain Guilley, CTO of Secure-IC.
E-Mail: sylvain.guilley@secure-ic.com

September 6, 2021

Abstract

This document is a public comment to the “*Call for feedback about the Masked Circuits (MC) Project*”, originally posted on 2021-June-21 in the MC-Forum.

Introduction

Assessing the security level of a cryptographic implementation is an important topic for the industry. Indeed, final users of technologies embedding cryptography deserve to get objective information about the actual resistance of their product, when facing side-channel analyses.

Standards are already in place in this respect. Typically ISO/IEC 19790:2012 prescribes to leverage ISO/IEC 17825:2016. And ISO/IEC 15408:2009 also comes with Security Functional Requirements and a very pragmatic Vulnerability Assurance quotation system.

In this document, I aim at:

1. Clarifying the current industrial understanding of side-channel protections, and rating of implementations;
2. Providing answers to the public consultation about “NIST Masked Circuits project”;
3. Offering a clear vision for a validation of the cryptographic implementations against side-channel analyses.

Those three points are addressed in the three following sections.

Item 7: Comments by Sylvain Guilley (2021-Sep-06)

1 Industry expectations w.r.t. side-channel analyses

1.1 Industry organization

The threat of side-channel analyses (SCA) is tangible. The industry shall therefore provide concrete answers regarding the protection of final users against this class of attacks.

The **security objective** is therefore to provide technology to resist to some extent to SCA. State-of-the-art technical solutions are numerous (see some reference books, such as [8, 10]), but indeed, they mostly boil down to:

- **reduce leakage**: Action is to balance leakage or hinder its measurement;
- **increase noise**: Action is to disallow controlability by the user, enforce parallelism, implement stealthy start/stop, or randomize the computation.

We understand that NIST effort focuses on one specific method, i.e., that of “randomization” of sensitive values involved in block cipher inner computations. One must be aware that there are multiple realizations (often patented), such as: random interrupts, fake/decoy operations, shuffling, masking, etc.

Again, NIST appears to further focus on one techniques, namely “masking”. Obviously, in order to maximize the potential of this very specific technique, some questions arise. For instance: how to randomize data at rest, in computations, in transit (moving in and out), how much entropy, etc.

The industry and the civil society aim at defining the level of protection. As already mentioned, ISO/IEC 19790 and ISO/IEC 15408 are validated and operational testing/evaluation methods, deployed in several schemes (CMVP, Common Criteria, etc.), which provide fully satisfactory answers to this question.

1.2 Academic perspective

Still, we notice that the scientific community addresses the question from a different angle, with “design metrics”. We notice a fundamental **disconnect** with the pragmatic security evaluation needs.

Let us nevertheless analyse this perspective. We notice:

- a trend for security-by-design, based on:
 - design rationale, and
 - models: bit/word level, security order, probing vs moment, etc.
- a care for implementation issues, in that:
 - models are abstract, and therefore, practically,
 - the security-by-design approach must be guided.

Item 7: Comments by Sylvain Guilley (2021-Sep-06)

Let us for instance discuss two models, namely “probing” [5] vs “moment” [9]:

- Probing:
 - probing security is very strong. But, if used as a requirement, it is **not** specific to block ciphers. It concerns, in general, probing with probing stations / LVP / FIB / etc. of key generation/transfer on bus/computation/storage. In this respect, it is not specific to "Masked Circuits Project". For example, probing one bit (say MSB or LSB) of an ECDSA nonce allows to recover the long term key (with LLL attack).
 - in practice, for industry applications, this model is **overkill**.
- Moments:
 - Recall as well issues such as “linear probing”, which wreaks havoc even probing secure schemes (please refer to [4]). Considering evaluation schemes: they target on result. Recall ISO/IEC 17825 (FIPS 140 Appendix F): 10,000 or 100,000 traces. Idem for Common Criteria scheme, which quotes attacks in terms of time to perpetrate the attack.
 - The moment model is much more relevant in this respect. As a matter of fact, it is known how to relate number of traces to countermeasures (see Appendix B of ISO/IEC 20085-2 [6] and recent publications such as [1, 2]).

The approach of the NIST “Masked Cryptography” appears very local to gates within the cryptographic implementations. But it is very well known as well, from industry experience, that SCA correction is the result of multiple requirements:

- key loading, key scheduling, key zeroization,
- datapath consideration, with data is standby at the input of some “dead” or “unselected” MUXes, etc.
- leakage in distance, etc.
- how randomness is generated, and how randomness is reused
- reorganization (structure modification, engineering change order, optimization by gates removal) of the RTL by tools

And regarding glitches, they are one spurious source of leakage. But there are others, as analysed for instance in [3] and depicted in Fig. 1. Also, hard to model leakages (such as leakage triggered by one static value at a gate while another input is changing [14]) have not be brought to design constraints as of now (as far as we know).

Item 7: Comments by Sylvain Guilley (2021-Sep-06)

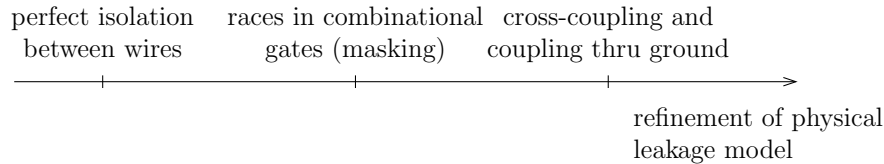


Figure 1: Hierarchy of SCA bias attributable to gate-level leakage

2 Answers to the public consultation

The questions from NIST are in blue, whereas Secure-IC’s answers are printed in black color.

2.1 1. Potential interest

- a) Are the glitch-extended probing model and the corresponding masking schemes addressing problems of interest for the industry and government stakeholders?
 - The “probing model” in general is irrelevant for the industry, as state-of-the-art attacks consider sensors which aggregate the side-channel over a macro- or meso-sopic area of the Device Under Test. Please refer to [7, §5.4.4].
- b) What orders of protection are pertinent to consider for standardization?
 - The “protection order” is a design metric, and therefore does not relate to security assurance. Considerations such as implementation corrections, signal-to-noise ratio, soundness of the scheme, capability of the attacker, etc. (please refer to [6, Annex B]) are also part of the security assessment. Thus “protection order” can be used as one possible option for being a vendor evidence, but shall not be considered a requirement.
- c) Will the semiconductor industry adopt this type of schemes in their product lines to supply the market with products implementing them?
 - We notice that the two web portals <https://csrc.nist.gov/projects/cryptographic-module-validation-program/validated-modules> and <https://www.commoncriteriaportal.org/products/> already list cryptographic modules implemented SCA protections.

2.2 2. Feasibility potential:

- a) Is there a stack of available design and production tools, widely used by the semiconductor industry, that can transfer the logical netlists into silicon gate layouts, on major hardware platforms of interest (FPGA, ASIC,

Item 7: Comments by Sylvain Guilley (2021-Sep-06)

etc.), while preserving the needed algorithmic properties and ensuring the needed implementation assumptions, so that minimal side-channel testing would be sufficient for the verification of the intended security guarantees?

- Yes, the described approach is sound. Refer for instance to PESC concept (Protect Evaluate Service & Certify): <https://www.secure-ic.com/solutions/>.
- b) Considering practical attacks, are there identifiable deployment characteristics that justify differentiated security profiles across platforms, masking orders, and other properties?
 - Security expectations are specific to use-cases. From a validation stand-point, this is captured by the “levels” (1 to 4) in ISO/IEC 19790 and the “quotation” in ISO/IEC 15408.

3 Secure-IC’s vision of cryptographic implementations

3.1 Security-by-design

The rationale of the side-channel protections shall be explained. This is a common requirement already enforced by ISO/IEC 17825 (see e.g., FIPS 140-3 IG) and ISO/IEC 15408.

3.2 DevSecOps

Given the fact that SCA threats are proteiform, generic detection methods and remediation flows shall be put in place.

The ability to assess whether protections are functional very early is a key factor for achieving these results. Therefore, presilicon evaluation methodology and tools are instrumental. We bring attention of NIST that such practices are readily available, for instance:

- Methods are described scholarly in [13];
- Tools are introduced as [12, 11].

Such tools make it possible to check for SCA issues at design stage, and therefore this maximizes the chance for the final implementation to be immune to SCA.

References

- [1] Wei Cheng, Sylvain Guilley, Claude Carlet, Jean-Luc Danger, and Sihem Mesnager. Information leakages in code-based masking: A unified quantification approach. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):465–495, 2021.

Item 7: Comments by Sylvain Guilley (2021-Sep-06)

- [2] Wei Cheng, Yi Liu, Sylvain Guilley, and Olivier Rioul. Attacking masked cryptographic implementations: Information-theoretic bounds. *CoRR*, abs/2105.07436, 2021.
- [3] Thomas De Cnudde, Begül Bilgin, Benedikt Gierlichs, Ventzislav Nikov, Svetla Nikova, and Vincent Rijmen. Does Coupling Affect the Security of Masked Implementations? In Sylvain Guilley, editor, *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*, volume 10348 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2017.
- [4] Jean-Luc Danger, Sylvain Guilley, Annelie Heuser, Axel Legay, and Ming Tang. Physical Security Versus Masking Schemes. In Çetin Kaya Koç, editor, *Cyber-Physical Systems Security.*, pages 269–284. Springer, 2018.
- [5] Yuval Ishai, Amit Sahai, and David Wagner. Private Circuits: Securing Hardware against Probing Attacks. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, August 17–21 2003. Santa Barbara, California, USA.
- [6] ISO/IEC JTC 1/SC 27/WG 3. ISO/IEC CD 20085-1:2017 (E). Information technology - Security techniques — Test tool requirements and test tool calibration methods for use in testing non-invasive attack mitigation techniques in cryptographic modules — Part 1: Calibration method and apparatus, January 25 2017.
- [7] ISO/IEC JTC 1/SC 27/WG 3. ISO/IEC 20085-1:2019 (en). Information technology Security techniques — Test tool requirements and test tool calibration methods for use in testing non-invasive attack mitigation techniques in cryptographic modules — Part 1: Test tools and techniques, 2019.
- [8] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, December 2006. ISBN 0-387-30857-1, <http://www.dpabook.org/>.
- [9] Amir Moradi and François-Xavier Standaert. Moments-Correlating DPA. In Begül Bilgin, Svetla Nikova, and Vincent Rijmen, editors, *Proceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016 Vienna, Austria, October, 2016*, pages 5–15. ACM, 2016.
- [10] Maamar Ouladj and Sylvain Guilley. *Side-Channel Analysis of Embedded Systems - An Efficient Algorithmic Approach*. Springer, 2021. DOI: 10.1007/978-3-030-77222-2.
- [11] Secure-IC. Virtualyzer tool (VTZ). <https://www.secure-ic.com/solutions/virtualyzer/> and <https://cadforassurance.org/tools/design-for-trust/virtualyzer/>, Accessed online on March 4th, 2021.

Item 7: Comments by Sylvain Guilley (2021-Sep-06)

- [12] Secure-IC. Catalyze tool, 2021. <https://cadforassurance.org/tools/software-assurance/catalyze/>, <https://www.secure-ic.com/solutions/catalyze/>, accessed on July 2nd 2021.
- [13] Youssef Souissi, Adrien Facon, and Sylvain Guilley. Virtual Security Evaluation - An Operational Methodology for Side-Channel Leakage Detection at Source-Code Level. In Claude Carlet, Sylvain Guilley, Abderrahmane Nitaj, and El Mamoun Souidi, editors, *Codes, Cryptology and Information Security - Third International Conference, C2SI 2019, Rabat, Morocco, April 22-24, 2019, Proceedings - In Honor of Said El Hajji*, volume 11445 of *Lecture Notes in Computer Science*, pages 3–12. Springer, 2019.
- [14] Takeshi Sugawara, Daisuke Suzuki, Minoru Saeki, Mitsuru Shiozaki, and Takeshi Fujino. On Measurable Side-Channel Leaks Inside ASIC Design Primitives. In Guido Bertoni and Jean-Sébastien Coron, editors, *CHES*, volume 8086 of *Lecture Notes in Computer Science*, pages 159–178. Springer, 2013.

Item 8: Comments by Wei Cheng, Sylvain Guilley, Jean-Luc Danger (2021-Sep-06)

Comments on Proposals of NIST Masked Circuits Project and Introduction of Code-based Masking

Wei Cheng¹, Sylvain Guilley^{2,1} and Jean-Luc Danger^{1,2}

¹ LTCI, Télécom Paris, Institut Polytechnique de Paris, Palaiseau, France
{wei.cheng, jean-luc.danger}@telecom-paris.fr

² Secure-IC S.A.S., Paris, France
sylvain.guilley@secure-ic.com

Abstract. The present document is a formal comment to the NIST Masked Circuits project. The co-authors are experts aiming at developing innovative technology to provably secure embedded systems. We have been studying several masking techniques aiming at improving resistance of hardware circuit implementations of block-ciphers against side-channel attacks.

In this document, we first present “Generalized Code-based Masking” (later on referred to GCM) scheme, as:

- a generalization of traditional Boolean masking scheme (see [CJRR99]);
- a consolidation of several existing schemes.

Second, we let the NIST Masked Circuits project know about relevant metrics to assess the security level of GCM, that are complementary to proof-based evaluation.

Keywords: Side-channel attacks · Leakage quantification · Signal-to-Noise Ratio (SNR) · Mutual Information (MI) · Inner Product Masking (IPM) · Shamir’s Secret Sharing (SSS) · Generalized Code-based Masking (GCM) · Coding theory.

1 Introduction

Protecting cryptographic implementations (be they hardware or software) against side-channel attacks is an important and long-lasting topic. First of all, attackers get more and more powerful, always leveraging stronger exploitation techniques (such as multivariate attacks, supervised machine learning or deep learning analysis, etc.). Second, the full potential of countermeasures is still to emerge. In this respect, several considerations must be taken into account:

- proper protection vis-à-vis a threat,
- optimal protection design given a restricted implementation cost,
- complete assessment and verification of concrete security of an implementation.

We have been analyzing thoroughly those questions, and we recommend GCM, which will be presented in Sec. 2. The evaluation tools and metrics to evaluate GCM are presented in Sec. 3. Eventually, our contribution to the NIST Masked Circuits project is given in Sec. 4.

Item 8: Comments by Wei Cheng, Sylvain Guilley, Jean-Luc Danger (2021-Sep-06)

2 General Code-based Masking Scheme

Generalization is a promising approach to unify different masking schemes. In this trend, the code-based masking generalizes many existing schemes, including Boolean masking, Inner Product masking (IPM), Leakage Squeezing (LS) [CDG⁺14, CG18] and Direct Sum masking (DSM) [BCC⁺14, PGS⁺17]. To the best of our knowledge, the generalized code-based masking (GCM) [WMCS20, CGC⁺21a] is the most generic scheme in this respect. In particular, polynomial masking [GM11, PR11] is also a special case of GCM, which is built upon Shamir’s secret sharing (SSS) scheme [Sha79].

Let $X \in \mathbb{F}_{2^t}^k$ and $Y \in \mathbb{F}_{2^t}^t$ be respectively the sensitive variable and t random masks. Then the encoded variable in GCM writes:

$$Z = X\mathbf{G} + Y\mathbf{H} \in \mathbb{F}_{2^t}^n, \tag{1}$$

given that $k + t \leq n$, where \mathbf{G} and \mathbf{H} are generator matrices of two codes \mathcal{C} and \mathcal{D} , respectively. For the sake of simplicity, we take $k = 1$, but essentially, the GCM can use packed secret sharing techniques [GSF13, WMCS20] to improve the performance by parallelism. However, the side-channel security evaluation of encoding is similar to any k , since each of the k sensitive variables is encoded similarly.

The overview of connections between these masking schemes is shown in Fig. 1, where the four intersecting areas are:

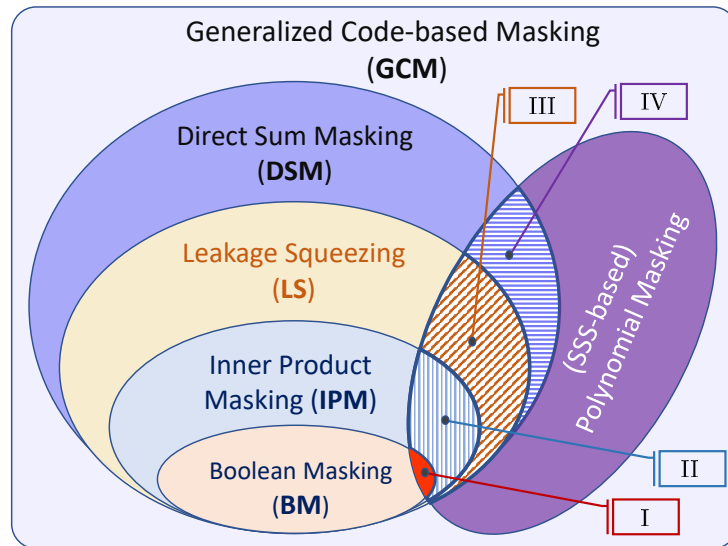


Figure 1: Overview of code-based masking schemes [CGC⁺21a]. In particular, all intersections I, II, III, and IV mean that $n = t + 1$ in SSS-based masking, where the two codes \mathcal{C} and \mathcal{D} are complementary.

- Intersection I: as pointed out in [CS21], Boolean masking can be considered as a special case of polynomial masking for small enough parameters ($n \leq 6$ or equivalently $t \leq 5$).
- Intersection II: in [BFG15], the authors claimed that the polynomial masking is a special case of IPM. However, this generalization does not indicate the exact connections between SSS-scheme and the Reed-Solomon codes. Indeed, if we take

Item 8: Comments by Wei Cheng, Sylvain Guilley, Jean-Luc Danger (2021-Sep-06)

W. Cheng, S. Guilley, J.-L. Danger

3

the polynomial evaluations in encoding into consideration, the generalization from SSS-based masking to IPM is valid only when $n = 2$ and $t = 1$.

- Intersections III and IV: in SSS-based masking, if $n = t + 1$, the codes \mathcal{C} and \mathcal{D} are complementary, therefore they can be viewed as DSM (or LS) scheme. Otherwise, if $n > t + 1$, the corresponding masking schemes are out of DSM’s scope. On the other side, the linear codes for DSM may not be converted into SSS-based schemes since the codes in SSS are endowed with a specific algebraic structure.

From a coding-theoretic view, we provide a technical overview of above masking schemes in Tab. 1, including the linear codes \mathcal{C} and \mathcal{D} , and the corresponding generator matrices. We refer the interested reader to [CGC⁺21a] for more details.

Table 1: Encodings in IPM, LS, DSM, SSS-based masking and GCM, revisited [CGC⁺21a].

	IPM [BFG15, BFG ⁺ 17]	LS [CDG ⁺ 14]	DSM [BCC ⁺ 14, PGS ⁺ 17]	SSS-based masking [GM11, PR11]	GCM [WMCS20]
Conditions on \mathcal{C} and \mathcal{D}	$\mathcal{C} \cap \mathcal{D} = \{0\}$, $\mathcal{C} + \mathcal{D} = \mathbb{K}^n$	$\mathcal{C} \cap \mathcal{D} = \{0\}$, $\mathcal{C} + \mathcal{D} = \mathbb{K}^n$	$\mathcal{C} \cap \mathcal{D} = \{0\}$, $\mathcal{C} + \mathcal{D} = \mathbb{K}^n$	$\mathcal{C} \cap \mathcal{D} = \{0\}$	$\mathcal{C} \cap \mathcal{D} = \{0\}$
$\mathbf{G} \in \mathbb{K}^{k \times n}$	$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \end{pmatrix}$	$\mathbf{G} \in \mathbb{K}^{k \times n}$	$\mathbf{G} \in \mathbb{K}^{k \times n}$	$\begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}$	$\mathbf{G} \in \mathbb{K}^{k \times n}$
$\mathbf{H} \in \mathbb{K}^{t \times n}$	$\begin{pmatrix} \alpha_1 & 1 & 0 & \cdots & 0 \\ \alpha_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_t & 0 & 0 & \cdots & 1 \end{pmatrix}$	$\mathbf{H} \in \mathbb{K}^{t \times n}$	$\mathbf{H} \in \mathbb{K}^{t \times n}$	$\begin{pmatrix} \alpha_1^1 & \alpha_1^2 & \cdots & \alpha_1^n \\ \alpha_2^1 & \alpha_2^2 & \cdots & \alpha_2^n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_t^1 & \alpha_t^2 & \cdots & \alpha_t^n \end{pmatrix}$	$\mathbf{H} \in \mathbb{K}^{t \times n}$
Security parameters:	$k = 1, n = t + 1$	$n = k + t$. \mathbf{G}, \mathbf{H} can be any matrices	$n = k + t$. \mathbf{G}, \mathbf{H} can be any matrices	$n \geq k + t$ and $f_X(X)$. In glitch-free case, $n \geq 2t + 1$ [PR11]	$n \geq k + t$. \mathbf{G}, \mathbf{H} can be any matrices

The most significant benefit of utilizing code-based masking is the higher security order than the simple Boolean masking given the same number of shares. Taking 2-share IPM over \mathbb{F}_{2^8} [BFG⁺17, CGC⁺21b] as an instance, when appropriate public parameters are chosen, the side-channel security order can be maximized to 3 under the bit-probing model [PGS⁺17], which is higher than 1 in Boolean masking. Moreover, the security orders are enlarged to 7 vs. 2 (IPM vs. Boolean one) in 3-share scenarios [CGC⁺21b, Tab. 2].

Currently, the side-channel security order of GCM has been connected to the dual distance of \mathcal{D} [PGS⁺17, CG18], which is denoted as $d_{\mathcal{D}}^{\perp}$. As a special case, the security order t in IPM and DSM is equal to $d_{\mathcal{D}}^{\perp} - 1$ since the two codes \mathcal{C} and \mathcal{D} are complementary. However, as pointed out in [CGC⁺21b], the dual distance of \mathcal{D} is not sufficient to characterize the concrete side-channel resistance of IPM, hence a new framework with a new parameter (more precisely $B_{d_{\mathcal{D}}^{\perp}}$, which counts the number of codewords of Hamming weight equal to $d_{\mathcal{D}}^{\perp}$ in \mathcal{D}^{\perp}) is proposed to model IPM’s concrete security level more accurately. Finally, the framework is extended for leakage quantification of GCM in [CGC⁺21a] when \mathcal{C} and \mathcal{D} are not necessary to be complementary.

3 Tools for the Evaluation of Masked Circuits

According to different leakage models and the abstraction level of cryptographic implementations, evaluation tools are roughly classified into three categories as follows.

- Firstly, the conformance-based leakage detection aims at answering the following question at a high abstraction level: *does the device under test leak side-channel information?* [CGJ⁺13, BDGN14, MRSS18]. Those statistical tools include Welch’s t -test, NICV, χ^2 -test, etc. A similar approach is detailed in ISO/IEC 17825 [ISO].

Item 8: Comments by Wei Cheng, Sylvain Guilley, Jean-Luc Danger (2021-Sep-06)

- Secondly, the information-theoretic evaluation aims at measuring side-channel leakages by utilizing information-theoretic measures [SMY09, WO11]. In essential, it usually provides information-theoretic bounds on the probability of success for any side-channel distinguishers given a set side-channel measurements [dCGRP19, CLGR21]. The frequently used measures include Shannon mutual information (MI), Kullback-Leibler divergence, conditional entropy, etc.
- Finally, the attack-based evaluation is at the core of side-channel security evaluation, which aims at assessing the probability of success of a specific side-channel distinguisher. Relying on large variety of side-channel distinguishers like correlation power analysis [BCO04], template attacks [CRR02], stochastic attacks [SLP05], higher-order optimal distinguisher [BGHR14], etc, the attack-based evaluation provides more accurate assessment of leakage, which captures device-specific features of side-channel leakage. In this respect, some metrics include the signal-to-noise ratio (SNR) for leakage detection and success rate as an ultimate attacking metric.

Summing up, the conformance-based leakage detection only provides qualitative assertion on whether the masked circuits leak or not, while other two evaluations give quantitative assessment of concrete side-channel security.

In the following, we concentrate on MI and SNR, and recall their definitions and related state-of-the-arts.

3.1 Quantifying Leakage by MI

Let $Z = (Z_1, Z_2, \dots, Z_n)$ denote the encoded intermediate with n shares and X be the secret. Considering a simplified example by ignoring the noise, we assume the leakage of each share is $\mathcal{L}_i = Z_i$ under the identity leakage model and $\mathcal{L} = \sum_i \mathcal{L}_i$ is the total leakage. To launch a successful attack, an adversary needs to find the (smallest) key-depend statics, namely raising d such that $\mathbb{E}[\mathcal{L}^d|X] \neq \mathbb{E}[\mathcal{L}^d]$, but $\mathbb{E}[\mathcal{L}^i|X] = \mathbb{E}[\mathcal{L}^i]$ for all $i < d$. Equivalently, an adversary needs the smallest d such that $\mathbb{V}[\mathbb{E}[\mathcal{L}^d|X]] \neq 0$, which measures the informative part in \mathcal{L} .

More formally, let $P(Z)$ be a leakage function and let N denote the independent Gaussian noise with zero mean and variance $\mathbb{V}[N] = \sigma_{total}^2 \propto \sigma^{2d}$ (\propto means proportional to σ^{2d}) [CGC⁺21b]. Then, the leakage is:

$$\mathcal{L} = P(Z) + N. \quad (2)$$

Therefore, the MI between leakage \mathcal{L} and the sensitive variable X is defined as

$$I[\mathcal{L}; X] = H[\mathcal{L}] - H[\mathcal{L}|X], \quad (3)$$

where:

- the total entropy is:

$$H[\mathcal{L}] = - \int_l \mathbb{P}[l] \log_2 \mathbb{P}[l] dl \quad (4)$$

- the conditional entropy $H[\mathcal{L}|X]$ is:

$$H[\mathcal{L}|X] = - \sum_{x \in \mathbb{F}_2^k} \mathbb{P}[x] \int_l \mathbb{P}[l|x] \log_2 \mathbb{P}[l|x] dl. \quad (5)$$

In multivariate cases, two entropies are computed on $\mathcal{L} = (\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_d)$ for a d -variate MI by a d -D integral on continuous variables. While in monivariate cases, two entropies are computed by an one dimensional integral.

Therefore, the next theorem is for quantifying the information leakages by MI in GCM.

Item 8: Comments by Wei Cheng, Sylvain Guilley, Jean-Luc Danger (2021-Sep-06)

Theorem 1 (Leakage Quantification by MI [CGC⁺21a]). *Let a device be protected by the GCM scheme as $Z = X\mathbf{G} + Y\mathbf{H}$. Assume the leakages of the device can be represented in the form: $\mathcal{L} = P(Z) + N$. Then the MI between \mathcal{L} and X is estimated as:*

$$I[\mathcal{L}; X] = \begin{cases} 0, & \text{if } \deg(P) < d_{\mathcal{D}}^{\perp} \\ \frac{d_{\mathcal{D}}^{\perp} B'_{d_{\mathcal{D}}^{\perp}}}{2 \ln 2 \cdot 2^{2d_{\mathcal{D}}^{\perp}}} \times \frac{1}{\sigma^{2d_{\mathcal{D}}^{\perp}}} + \mathcal{O}\left(\frac{1}{\sigma^{2(d_{\mathcal{D}}^{\perp}+1)}}\right), & \text{if } \deg(P) = d_{\mathcal{D}}^{\perp}, \text{ when } \sigma \rightarrow +\infty \end{cases} \quad (6)$$

where σ is the standard deviation of noise in the leakage of each share.

An illustration of MI on several optimal instances of GCM with $n = 2$ different parameters are shown in Fig. 2, in comparison with the unprotected case and the Boolean masking with $n \in \{2, 3\}$. The first observation is that from an information-theoretic perspective, the optimal instances of GCM with $n = 2$ are always better than the Boolean one with $n = 2$ at all ranges of noise levels. This confirms the superiority of optimal code-based masking upon the Boolean one.

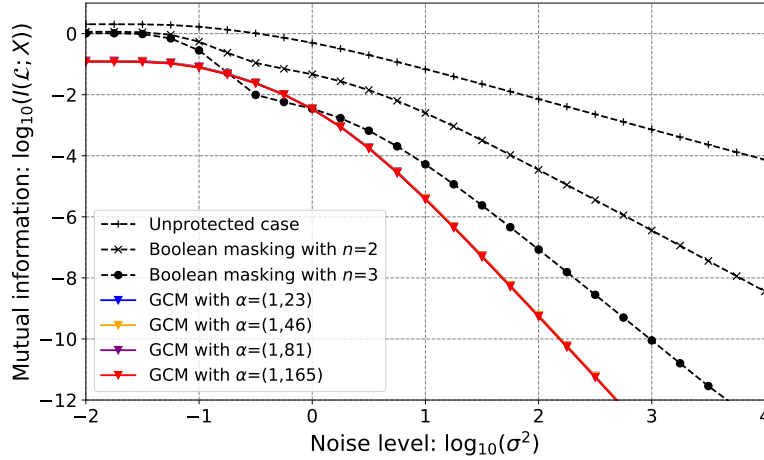


Figure 2: Information-theoretic evaluation on four instances of GCM with optimal parameters. Note that there are 12 groups of parameters for GCM with $n = 2$ and $t = 1$ [CGC⁺21a].

Secondly, **the optimal GCM instances (with $n = 2$ shares) are much better than the Boolean masking even with $n = 3$** except a small interval of noise levels. In particular, when $\sigma^2 > 1.0$, the optimal GCM instances always perform better. In fact, since the dual distance of optimal GCM instances equals 4, they would be comparable to the Boolean masking with $n = 4$ (the fourth-order Boolean masking).

At last, the four GCM instances have almost the same side-channel security from an information-theoretic perspective. Indeed, these four instances have the same $d_{\mathcal{D}}^{\perp}$ and $B'_{d_{\mathcal{D}}^{\perp}}$. As a result, MI curves for the four instances are superimposed together as presented in Fig. 2. It is worth mentioning that the linear codes in these four instances are not mutually equivalent in a view of coding theory.

3.2 Quantifying Leakage by SNR

Taking the same notations as above, we have $\mathbb{V}[\mathbb{E}[P(Z) + N|X]] = \mathbb{V}[\mathbb{E}[P(Z)|X]]$, where $Z = X\mathbf{G} + Y\mathbf{H} \in \mathbb{K}^n = \mathbb{F}_2^n$ is the encoding in GCM (Equ. 1). The SNR of leakages is defined as:

$$\text{SNR} = \frac{\mathbb{V}[\mathbb{E}[\mathcal{L}|X]]}{\mathbb{V}[N]} = \frac{\mathbb{V}[\mathbb{E}[P(Z)|X]]}{\sigma_{total}^2}. \quad (7)$$

Item 8: Comments by Wei Cheng, Sylvain Guilley, Jean-Luc Danger (2021-Sep-06)

Therefore, the following theorem quantifies the leakages in the GCM scheme by SNR.

Theorem 2 (Leakage Quantification by SNR [CGC⁺21a]). *Let a device be protected by the GCM scheme as $Z = X\mathbf{G} + Y\mathbf{H}$. Assume the device is leaking in Hamming weight model in the form: $\mathcal{L} = P(Z) + N$. Then the SNR of the exploitable leakages is:*

$$SNR = \frac{\mathbb{V}[\mathbb{E}[P(Z)|X]]}{\sigma_{total}^2} = \frac{B'_{d_{\mathcal{D}}}}{\sigma_{total}^2} \left(\frac{d_{\mathcal{D}}!}{2^{d_{\mathcal{D}}}} \right)^2,$$

where σ_{total}^2 is the total noise such that $\sigma_{total}^2 \propto \sigma^{2d}$. Moreover, $B'_{d_{\mathcal{D}}}$ is the adjusted kissing number defined in [CGC⁺21a]. \mathcal{C}^{\perp} and \mathcal{D}^{\perp} are the dual codes of \mathcal{C} and \mathcal{D} , respectively.

3.3 Attack-based Evaluation by Success Rate

Considering a key-recovery attack in SCA, the ultimate metric is the success rate indicating the probability that an adversary can succeed in recovering the secret key. In the following, we launch the *optimal attack* [BGHR14] which adopts higher-order optimal distinguishers based on Maximum-Likelihood (ML) rule.

In order to have a thorough comparison, we evaluate the success rate P_s of the optimal attack under different noise levels. In particular, we deploy the commonly used Hamming weight leakage model with Gaussian noises as above. We depict in Fig. 3 the number of traces to achieve $P_s \geq 95\%$ versus different noise levels.

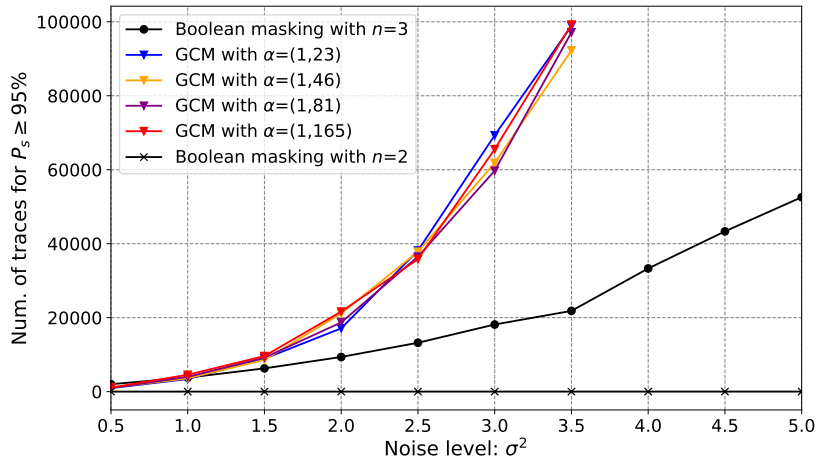


Figure 3: Attack-based evaluation on four instances of GCM with optimal parameters. Note that the success rate P_s is computed by 200 repetitions of optimal attacks.

As shown in Fig. 3, the attack-based evaluation (by carrying out the optimal multivariate attack) validates that the optimal GCM instances performs better than two instances of Boolean masking, and even more when the noise level increases. Thus, this confirms that **optimal GCM instances with $n = 2$ shares are more secure than Boolean masking even with $n = 3$ shares**. In fact, given a sufficient amount of noise, the number of traces needed for a successful attack is exponential to the masking order [CJRR99, PR13]. As a consequence, the actual security order of the optimal GCM instances should be greater than the third-order Boolean masking as verified by the information-theoretic evaluation in Fig. 2.

Item 8: Comments by Wei Cheng, Sylvain Guilley, Jean-Luc Danger (2021-Sep-06)

W. Cheng, S. Guilley, J.-L. Danger

7

4 Contribution to NIST Masked Circuits Project

In view of the state-of-the-art on constructions and evaluations of masked circuits of block ciphers, our contributions are the following:

1. The side-channel security of masked circuits must be evaluated as the effort for an attacker to extract the secret key.
2. State-of-the-art metrics, such as masking order, are qualitatively linked to this objective: assuming the masking scheme is implemented correctly, more security is expected by deploying more advanced masking schemes, e.g., code-based masking with optimal parameters. In particular, we provide the best achievable parameters for the most general code-based masking.
3. However, we stress that the security achieved by the Boolean masking scheme of order n_1 can be much lower than that of any masking scheme of order n_2 despite $n_1 > n_2$. Indeed, as we explained in previous sections, the same masking order (which is a design metric) can lead to implementations of different strength, highly depending on the masking scheme parameters.
4. In this respect, we support the fact that MI and SNR are more operational metrics to characterize the security level of an implementation, in the context of hardware circuit implementations of block-ciphers. The reason for this claim is that MI and SNR are quantitatively related to the key extraction attack success rate.
5. We insist, as shown in the note (as well in scholar papers, such as [CGC⁺21a, CGC⁺21b, RCG21]) that it is possible to bring a formal model, in that the MI and the SNR can be computed based on the countermeasure specifications. In return, our approach would provide a best-practice guideline for the application of code-based masking in protected cryptographic circuits.
6. At last, we highlight that the code-based masking can also be extended to detect faults, as a countermeasure against fault injection attacks [CCG⁺21, WMCS20]. However, it is still an open problem for constructing general elementary masked operations (e.g., addition and multiplication) in the presence of faults. We leave it for future investigation.

References

- [BCC⁺14] Julien Bringer, Claude Carlet, Hervé Chabanne, Sylvain Guilley, and Houssem Maghrebi. Orthogonal Direct Sum Masking - A Smartcard Friendly Computation Paradigm in a Code, with Builtin Protection against Side-Channel and Fault Attacks. In David Naccache and Damien Sauveron, editors, *Information Security Theory and Practice. Securing the Internet of Things - 8th IFIP WG 11.2 International Workshop, WISTP 2014, Heraklion, Crete, Greece, June 30 - July 2, 2014. Proceedings*, volume 8501 of *Lecture Notes in Computer Science*, pages 40–56. Springer, 2014.
- [BCO04] Éric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

Item 8: Comments by Wei Cheng, Sylvain Guilley, Jean-Luc Danger (2021-Sep-06)

- [BDGN14] Shivam Bhasin, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. NICV: Normalized Inter-Class Variance for Detection of Side-Channel Leakage. In *International Symposium on Electromagnetic Compatibility (EMC '14 / Tokyo)*. IEEE, May 12-16 2014. Session OS09: EM Information Leakage. Hitotsubashi Hall (National Center of Sciences), Chiyoda, Tokyo, Japan.
- [BFG15] Josep Balasch, Sebastian Faust, and Benedikt Gierlichs. Inner Product Masking Revisited. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 486–510. Springer, 2015.
- [BFG⁺17] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, Clara Paglialonga, and François-Xavier Standaert. Consolidating Inner Product Masking. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 724–754. Springer, 2017.
- [BGHR14] Nicolas Bruneau, Sylvain Guilley, Annelie Heuser, and Olivier Rioul. Masks Will Fall Off – Higher-Order Optimal Distinguishers. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 344–365. Springer, 2014.
- [CCG⁺21] Wei Cheng, Claude Carlet, Kouassi Goli, Jean-Luc Danger, and Sylvain Guilley. Detecting faults in inner product masking scheme. *J. Cryptogr. Eng.*, 11(2):119–133, 2021.
- [CDG⁺14] Claude Carlet, Jean-Luc Danger, Sylvain Guilley, Housseem Maghrebi, and Emmanuel Prouff. Achieving side-channel high-order correlation immunity with leakage squeezing. *J. Cryptographic Engineering*, 4(2):107–121, 2014.
- [CG18] Claude Carlet and Sylvain Guilley. Statistical properties of side-channel and fault injection attacks using coding theory. *Cryptography and Communications*, 10(5):909–933, 2018.
- [CGC⁺21a] Wei Cheng, Sylvain Guilley, Claude Carlet, Jean-Luc Danger, and Sihem Mesnager. Information leakages in code-based masking: A unified quantification approach. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):465–495, 2021.
- [CGC⁺21b] Wei Cheng, Sylvain Guilley, Claude Carlet, Sihem Mesnager, and Jean-Luc Danger. Optimizing Inner Product Masking Scheme by a Coding Theory Approach. *IEEE Trans. Inf. Forensics Secur.*, 16:220–235, 2021.
- [CGJ⁺13] Jeremy Cooper, Gilbert Goodwill, Joshua Jaffe, Gary Kenworthy, and Pankaj Rohatgi. Test Vector Leakage Assessment (TVLA) Methodology in Practice, Sept 24–26 2013. International Cryptographic Module Conference (ICMC), Holiday Inn Gaithersburg, MD, USA.

Item 8: Comments by Wei Cheng, Sylvain Guilley, Jean-Luc Danger (2021-Sep-06)

W. Cheng, S. Guilley, J.-L. Danger

9

- [CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.
- [CLGR21] Wei Cheng, Yi Liu, Sylvain Guilley, and Olivier Rioul. Attacking masked cryptographic implementations: Information-theoretic bounds. *CoRR*, abs/2105.07436, 2021.
- [CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski, Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.
- [CS21] Nicolas Costes and Martijn Stam. Redundant code-based masking revisited. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(1):426–450, 2021.
- [dCGRP19] Éloi de Chérisey, Sylvain Guilley, Olivier Rioul, and Pablo Piantanida. Best Information is Most Successful — Mutual Information and Success Rate in Side-Channel Analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):49–79, 2019.
- [GM11] Louis Goubin and Ange Martinelli. Protecting AES with Shamir’s Secret Sharing Scheme. In Preneel and Takagi [PT11], pages 79–94.
- [GSF13] Vincent Grosso, François-Xavier Standaert, and Sebastian Faust. Masking vs. Multiparty Computation: How Large Is the Gap for AES? In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 400–416. Springer, 2013.
- [ISO] ISO/IEC JTC 1/SC 27/WG 3. ISO/IEC 17825:2016: Information technology – Security techniques – Testing methods for the mitigation of non-invasive attack classes against cryptographic modules. <https://www.iso.org/standard/60612.html>.
- [MRSS18] Amir Moradi, Bastian Richter, Tobias Schneider, and François-Xavier Standaert. Leakage Detection with the χ^2 -Test. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):209–237, 2018.
- [PGS⁺17] Romain Poussier, Qian Guo, François-Xavier Standaert, Claude Carlet, and Sylvain Guilley. Connecting and Improving Direct Sum Masking and Inner Product Masking. In Thomas Eisenbarth and Yannick Teglia, editors, *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers*, volume 10728 of *Lecture Notes in Computer Science*, pages 123–141. Springer, 2017.
- [PR11] Emmanuel Prouff and Thomas Roche. Higher-Order Glitches Free Implementation of the AES Using Secure Multi-party Computation Protocols. In Preneel and Takagi [PT11], pages 63–78.
- [PR13] Emmanuel Prouff and Matthieu Rivain. Masking against Side-Channel Attacks: A Formal Security Proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International*

Item 8: Comments by Wei Cheng, Sylvain Guilley, Jean-Luc Danger (2021-Sep-06)

- Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.
- [PT11] Bart Preneel and Tsuyoshi Takagi, editors. *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 – October 1, 2011. Proceedings*, volume 6917 of *LNCS*. Springer, 2011.
- [RCG21] Olivier Rioul, Wei Cheng, and Sylvain Guilley. Cumulant expansion of mutual information for quantifying leakage of a protected secret. *CoRR*, abs/2102.02468, 2021.
- [Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.
- [SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A Stochastic Model for Differential Side Channel Cryptanalysis. In *LNCS*, editor, *CHES*, volume 3659 of *LNCS*, pages 30–46. Springer, Sept 2005. Edinburgh, Scotland, UK.
- [SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In *EUROCRYPT*, volume 5479 of *LNCS*, pages 443–461. Springer, April 26-30 2009. Cologne, Germany.
- [WMCS20] Weijia Wang, Pierrick Méaux, Gaëtan Cassiers, and François-Xavier Standaert. Efficient and Private Computations with Code-Based Masking. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2):128–171, 2020.
- [WO11] Carolyn Whitnall and Elisabeth Oswald. A Comprehensive Evaluation of Mutual Information Analysis Using a Fair Evaluation Framework. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 316–334. Springer, 2011.

Item 9: Comments by Ventzi Nikov (2021-Sep-07)



Dear Masked Circuit Forum,

Please, find below NXP feedback to the NIST Masked Circuits project and to the proposal "Threshold Cryptography on a Single Device by Means of Threshold Implementations".

1. Potential interest:

- a) Are the glitch-extended probing model and the corresponding masking schemes addressing problems of interest for the industry and government stakeholders?

To a certain level this is useful model which allows one to provide security proofs but it provides too much (unrealistic) power to the attacker, it does not cover all attack scenarios, it does not provide a link between the provable security of order d and the minimum number of traces/measurements required to see a leakage (or to reduce the rank of the master key below 2^{32} using a key enumeration algorithm) for a given signal to noise ratio. Therefore it is a good model to start with but it requires further improvements especially on the link with the number of traces without leakage.

- b) What orders of protection are pertinent to consider for standardization?

We consider only ASIC implementations. We think that order of protection beyond 4th is becoming too expensive and other methods (e.g. re-keying) might be better to consider. But for us the main question is how many traces are required by the attacker in order to see a leakage, since the same AES IP with given order of protection will be secure in a noisy IC and may become insecure in a low-noise (low-power) product.

- c) Will the semiconductor industry adopt this type of schemes in their product lines to supply the market with products implementing them?

We have already been using such a technology in our products.

2. Feasibility potential:

- a) Is there a stack of available design and production tools, widely used by the semiconductor industry, that can transfer the logical netlists into silicon gate layouts, on major hardware platforms of interest (FPGA, ASIC, etc.), while preserving the needed algorithmic properties and ensuring the needed

Item 9: Comments by Ventzi Nikov (2021-Sep-07)



implementation assumptions, so that minimal side-channel testing would be sufficient for the verification of the intended security guarantees?

There are RTL coding rules and tools which preserve the algorithmic properties, however they cannot guarantee that all assumptions hold, that is why certain side-channel testing is always performed.

- b) Considering practical attacks, are there identifiable deployment characteristics that justify differentiated security profiles across platforms, masking orders, and other properties?
- *Countermeasures against physical attacks are implementation related and hence are not an interoperability issue – i.e. it should be fine to adopt several different solutions.*
 - *Product security requirements (for different market segments) may differ a lot, the semiconductor industry needs to have flexible tools to achieve different levels of security and at the same time be cost-efficient. Here efficiency may relate to area, power, latency, energy, etc. Flexibility implies to have a variety of techniques and gadgets which can be used to achieve the design goals.*
 - *Products will differ in their signal-to-noise ratio and hence the used masking order in order to achieve the desired protection level. If the standard goes to one (assumably high) level of protection (profile) then many products may not use this standard (e.g. by cost reasons) and hence might go to dubious methods of protection. Therefore we would like to have standard supporting different security levels (profiles) especially to be relevant for low-power / low-energy microcontrollers and products like IoT.*
3. *We find the proposal to be a good first step which guarantees security in the single-probe glitch-extended probing model. Naturally we would like to see it extended to address the above listed concerns.*

Specific comments on the document:

Page 2:

“In other words, industry should still be free to design their own sharings ...”

Does this refer to other “generic functions” or also AES?

Item 9: Comments by Ventzi Nikov (2021-Sep-07)



Page 6:

"The adversary's interaction with the circuit is mediated through encoder and decoder algorithms, neither of which can be probed."

Maybe mention that this is only an assumption for the proof. In practice, you can still attack it, but it does not deliver useful information as it works on plain/ciphertexts?

Page 14: "[s]tage 4"

Page 24:

Table 2.1: Mixes metrics for Sbox and total AES, maybe introduce another line at the top with "Sbox" and "AES"

Page 25:

"To reduce complexity, it suffices to verify the properties of the shared AES S-box instead of the entire design"

Not sure about this one. Could there not be some imprecise control logic setup, which leaks a function of all shares? Usually that should not happen, but it might, e.g., with some transition leakages. It might be out of scope for this document, but could be useful to the reader to mention this here.

"The above properties capture higher-order univariate and single cipher-round multivariate attacks,"

Not sure about this one either, if we just look at one Sbox instead of the whole layer. Since we reuse randomness to a degree, there might be something to gain attacking multiple Sboxes of the same round, e.g., some horizontal attack?

Page 26:

"However, we note to take care when placing registers before or after the linear layer in order to avoid non-completeness issues."

This should be detailed further. Maybe give an example.

Page 27:

"For the matrix method[.] The tool"

Best Regards,
NXP

Notes on other comments received before the call

Item 10: Comments from Fatemeh Ganji (2021-Jun-11)

On 2021-Jun-11, Fatemeh Ganji called attention to a draft of their work, jointly with Ana Covi and Domenic Forte, “Circuit Masking From Theory to Standardization: A Comprehensive Survey for Hardware Security Researchers and Practitioner”, whose latest online version appears to be the one from 2021-Jun-24 available at <https://arxiv.org/abs/2106.12714>.

Item 11: Comments from Siemen Dhooghe (2020-Dec-15)

On 2020-Dec-15, Siemen Dhooghe called attention to their work, jointly with Time Beyne and Zhenda Zhang, “Cryptanalysis of Masked Ciphers: A Not So Random Idea” (doi:[10.1007/978-3-030-64837-4_27](https://doi.org/10.1007/978-3-030-64837-4_27)), whose latest online version appears to be the one from 2021-Jul-16 available at <https://eprint.iacr.org/2020/993>.