A memory hierarchy protected against side-channel attacks

Ezinam-Bertrand Talaki, Olivier Savry, David Hely, Mathieu Bouvier Des Noes

cryptography

MDPI

Article

A Memory Hierarchy Protected against Side-Channel Attacks

Ezinam Bertrand Talaki *, Olivier Savry, Mathieu Bouvier Des Noes and David Hely *

Univ. Grenoble Alpes, CEA, Leti, F-38000 Grenoble, France; olivier.savry@cea.fr (O.S.); mathieu.desnoes@cea.fr (M.B.D.N.)

* Correspondence: ezinam-bertrand.talaki@cea.fr (E.B.T.); david.hely@cea.fr (D.H.)

Abstract: In the vulnerability analysis of System on Chips, memory hierarchy is considered among the most valuable element to protect against information theft. Many first-order side-channel attacks have been reported on all its components from the main memory to the CPU registers. In this context, memory hierarchy encryption is widely used to ensure data confidentiality. Yet, this solution suffers from both memory and area overhead along with performance losses (timing delays), which is especially critical for cache memories that already occupy a large part of the spatial footprint of a processor. In this paper, we propose a secure and lightweight scheme to ensure the data confidentiality through the whole memory hierarchy. This is done by masking the data in cache memories with a lightweight mask generator that provides masks at each clock cycle without having to store them. Only 8-bit Initialization Vectors are stored for each mask value to enable further recomputation of the masks. The overall security of the masking scheme is assessed through a mutual information estimation that helped evaluate the minimum number of attack traces needed to succeed a profiling side-channel attack to 592 K traces in the attacking phase, which provides an acceptable security level in an analysis where an example of Signal to Noise Ratio of 0.02 is taken. The lightweight aspect of the generator has been confirmed by a hardware implementation that led to resource utilization of 400 LUTs.

Keywords: side channel attacks; masking; lightweight ciphers; NIST LWC competition

Citation: Talaki, E.B.; Savry, O.; Bouvier Des Noes, M.; Hely, D. A Memory Hierarchy Protected against Side-Channel Attacks. *Cryptography* **2022**, *6*, 19. https://doi.org/10.3390/cryptography6020019

Academic Editor: Jim Plusquellic

Received: 23 March 2022
Accepted: 13 April 2022
Published: 20 April 2022

1. Introduction

The idea of Kocher of exploiting the power consumption of a circuit that implements an encryption algorithm to retrieve the key has given rise to extensive studies on the vulnerability of System on Chips to such attacks. Originally performed on hardware cryptosystems, their targets have been widened to leverage the leakage of micro architectural components such as ALUs, register banks [1,2], interconnect bus, on-chip memories [3], hence endangering the entire memory hierarchy and targeting different types of data. To face these threats, which undermine the confidentiality of data as they move from main memory to the CPU, one of the frequently used solutions remains the encryption of the memory hierarchy to break the link between the sensitive data and the recorded power consumption related to the data transfer. Encryption schemes induce timing and area overhead which make them not suitable for application processors when applied to cipher data in cache memories. Yet, these memories are vulnerable to first-order power side-channel attacks [4] and need to be protected. It becomes more critical to find low-impact solutions since caches are the most area-consuming components on the CPU and adding mechanisms to ensure data confidentiality against power-based side-channel attacks would lead to a significant overhead on the spatial footprint of the entire processor and increase the latency of the caches.

Boolean masking has been introduced as a countermeasure to break the dependency between the sensitive data processed and the side-channel information by adding randomly generated masks to the sensitive variable. It can be applied either on a software application or a hardware module. The randomness of the mask helps ensure that any leakage gathered

by the adversary does not lead to direct recovery of the sensitive data. This solution has been widely studied and implemented in several consumer devices such as smartcards [5]. Instead of keeping encrypted data throughout the whole memory hierarchy, masking the data to be sent to the cache memory till the CPU registers could be a relevant alternative to continue ensuring data confidentiality. Applying a masking scheme in hardware requires some extra memory to store all the generated masks, which may be costly when applied to cache memories. It raises the challenge of generating mask values with low area, latency and memory overhead. In order to address this issue, we present in this paper a masking method for protecting data in cache memories against first-order side-channel attacks based on power consumption analysis. It relies on a lightweight mask generator, referred to as LightMaG, to provide a 64-bit mask value at each clock cycle using a lightweight cryptographic primitive (Subterranean 2.0 [6] in our case), ensuring a low area overhead. In this scheme, only an 8-bit Initialisation Vector (IV) is stored in the cache rather than the 64-bit mask. We also show how masking approach fits efficiently into a memory hierarchy protection mechanisms by dealing with the interface with the interconnect bus which provides encrypted data that is further decrypted and masked before being sent through the different cache memories up to the CPU registers which contain masked data and their masks with the goal of masking the execute stage of the CPU pipeline. A methodology to assess the security provided by the mask values and the required properties to achieve first-order side-channel security is also described in this paper.

The rest of the paper is organized as follows: Section 2 presents the previous work related to our proposal and the main motivation of our work. It is followed by the attacker model definition in Section 3. Background on the masking countermeasure and security evaluation of masked implementations using mutual information (MI) is provided in Section 4. Section 5 shows the results of MI estimation on simulated side-channel traces and the security requirement for the definition mask generation in order to achieve first-order security. LightMag is described in Section 6, followed by some discussion on the use of the generator and some possible enhancement in Section 7. Then comes the conclusion of the paper.

2. Motivations and Related Work

Side-channel attacks can exploit microarchitectural leakage of modern SoCs to retrieve sensitive data at any stage of the memory hierarchy. Arsath et al. proposed a framework (PLAN) to identify leaking modules in a microprocessor [3] without even needing a physical chip, as it takes a pre-synthesized netlist as input. They reported leakage on the memory hierarchy (cache, registers, RAM) and even on the ALU and FPU of an open-source RISC-V processor (Shakti-C [7]). A first-order side-channel vulnerability has been reported on SRAM cells and embedded memories while recording their power consumption [8]. Many other leakage assessments and first-order attacks have been conducted on other parts of the memory hierarchy such as interconnect bus and CPU registers in order to reveal the Hamming weight of the data, Hamming distance between two data, and even the data value directly [1,9,10]. All these work rise the necessity of mitigating such attacks on the data during the transit in the memory hierarchy. This is mostly done by encrypting these data. Several approaches have been designed with the purpose of encrypting the data in the memory hierarchy while keeping in mind the need to reduce the impact of this encryption on the performance of the processor. There are, for example, the works of Unterluggauer et al. [11] and Yin et al. [12] where the content of the main memory is encrypted before being stored and decrypted when loaded by the processor. Wong et al. also proposed a design of a memory protection unit to ensure confidentiality and integrity of data on a RISC-V SoC [13]. They specifically encrypt communication in the L2 cache-DRAM traffic, assuming all the components between the L2 cache and the CPU to be in a secure world. As opposed to the previous propositions, their solution focuses on the security of the DRAM memory content, which does not take into account the security of the data in cache memories and registers. Arsath et al. designed PARAM [3], a power

attack resistant microprocessor. They protect all the leaking modules of the processor by encrypting the data and addresses using a 4-round Feistel structure, which leads to a lightweight solution compared to previous ones. To prevent the disclosure of the secret key through side-channel attacks, the authors use the frequent re-keying strategy implemented with a remapping unit. Despite the lightweight aspect of their solution which can cope with the limited resources available in cache memories, the remapping unit will increase the cache misses since the dirty lines of the cache are written back to the memory before changing the key. In addition to performance degradation, depending on the key refreshing frequency, an adversary can mount a template attack since the authors did not provide information on that frequency.

Considering all the studies mentioned above, the primary motivation behind our work is how to bring data securely from the main memory to CPU registers and to prevent data eavesdropping via side-channel attacks on the execute stage of the CPU by enabling secure execution of instructions in that stage. For this purpose, a simple datapath between the DRAM and the CPU is briefly illustrated in Figure 1.

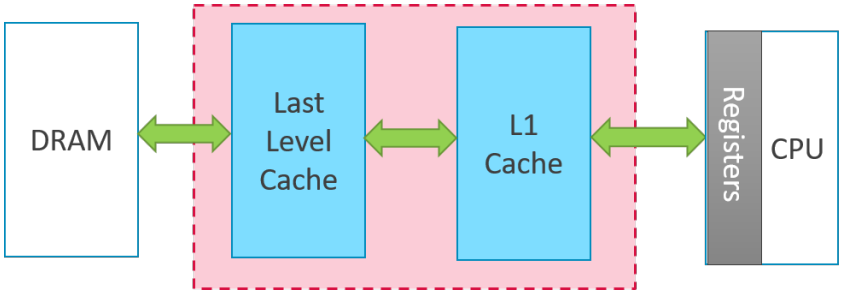

Figure 1. Simple datapath from memory to registers.

It is widely adopted in the literature that protecting such transfers can be done with encryption. Nonetheless, keeping encrypted data in cache memories (boxed in dotted lines in Figure 1) is more tricky than it seems. To do this, one needs to implement a module to decrypt the data upon a cache hit before sending it to the execute stage of the pipeline. This approach is time and area-consuming. To prevent this, boolean masking can help lighten the encryption in the cache memories. In practice, instead of sending encrypted data in the caches, we decrypt it at the output of the interconnect bus and mask it before sending it to the last level cache. At the output of the L1 cache, the masked data and the mask are sent to the registers. This provides inputs for the execute stage to process instructions on masked data. Figure 2 illustrates the described protection method which will be applied to a SoC embedding a 64 bit RISC-V CPU.

The main drawback in a masking scheme is the need to store the masks in order to be able to unmask the data. This inevitably leads to memory overhead and moreover a spatial overhead since we are dealing with caches. The manufacturing cost of a chip is directly proportional to the area size of the chip. Since caches occupy almost half of a CPU's surface, storing the masks would generate an additional cost of about 50% of the manufacturing cost of the circuit. Here comes the need for a masking scheme in which the mask storage condition is relaxed. To tackle this problem, we propose LightMaG, a lightweight mask generator that can output 64-bit masks at each clock cycle using a lightweight cipher. With this generator, only 8-bit IVs are stored in the cache (instead of 64 bits). We hence reduce the memory and delay overheads and the lightweight cipher used helps keep a low area overhead on the overall spatial footprint of the CPU. Being aware of this, it is reasonable to wonder whether our masking solution will be able to protect the data in the caches from first-order power side-channel attacks. We address this concern in Section 5 with leakage estimations on simulated traces.

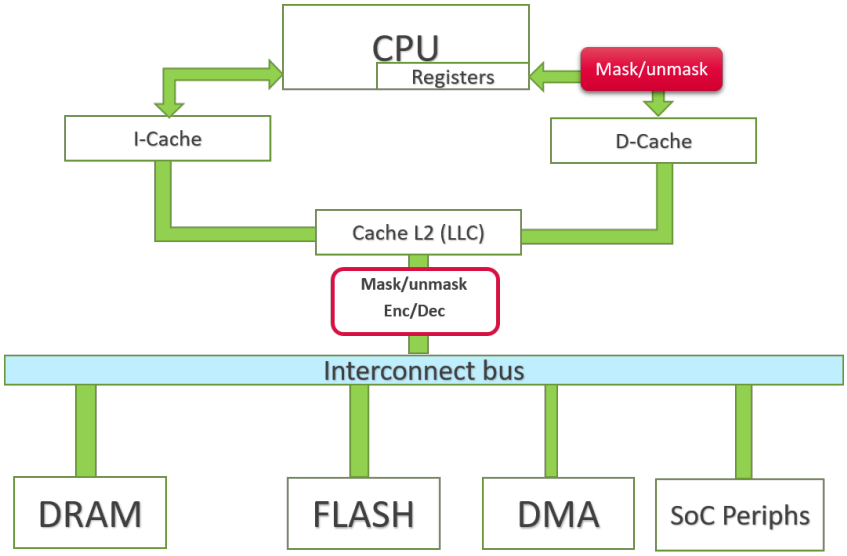

Figure 2. Overview of protected memory hierarchy.

Beyond the design of a countermeasure, this paper presents a design methodology to help designers in the various implementation options they can make to secure their circuits. Despite the accuracy that could be provided by leakage assessment experimentations on an actual prototype, using simulated traces is more valuable since experimental results from a specific circuit are only valid for a single SNR value (single noise level) whereas simulation can be run on a wide range of SNR values allowing designers to estimate the security level they can achieve according to the noise level on their chip.

3. Attacker Model

Based on the vulnerabilities of memory hierarchy to first-order side-channel attacks described earlier, we defined the resources and information an attacker could target in the memory hierarchy of a RISC-V based SoC. We build our model on top of the Dolev-Yao attacker model [14], considering the attacker to be able to intercept or forge any message in the "non-protected world" of the SoC like depicted in Figure 3.

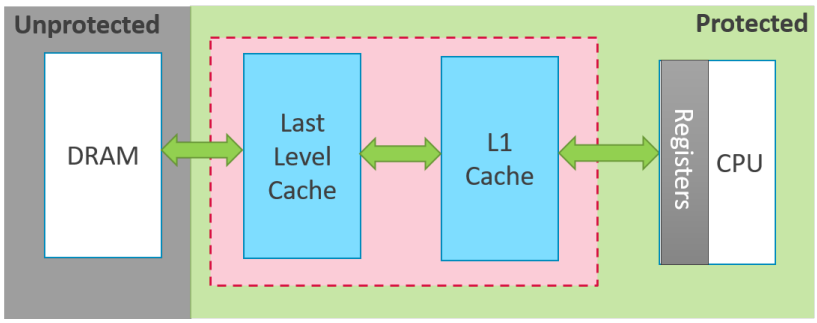

Figure 3. Protected vs unprotected parts against invasive attacks on the SoC.

Invasive hardware attacks such as microprobing and Focused Ion Beams (FIB) on the CPU and caches are not considered in this attacker model. The possible attacks of the adversary are hence limited to first-order such as template attacks or CPA. Higher-order attacks are out of the scope of our analysis. The underlying Dolev-Yao model also allows us to assert that the attacker cannot perform cryptanalysis based on the mathematical construction of the cryptographic primitives used for the communication between the CPU and the main memory. We define the following attributes of the attacker:

- Knowledge of the architecture:the RISC-V ISA documentation is publicly available hence a good knowledge of the circuit's (micro)architecture is expected since the implementation characteristics will also be available.
- Access to the chip: the attacker has:
  - *physical access*: power consumption and EM radiation of the circuit
  - *logical access*: registers, programming the CPU, I/Os, GPIOs.
- Main goal: retrieve any data in the memory hierarchy or at least their Hamming weight (HW) or Hamming distance (HD).

The relevance of this attacker model lies in the fact that various observation attacks have been demonstrated on all the parts of the memory hierarchy. A template attack on the interconnection bus has made it possible to recover the HW of the data passing through it ([10]) and also the HW and HD of the data in a CPU register. Moreover, the possibility of directly retrieving the value of the data was demonstrated in that same work, with a probability of 0.96 for 200 attack traces and a residual enumeration complexity of $2^{13.2}$. Cache memories vulnerability to first-order power analysis attack has been exhibited by Giterman et al. [8].

4. Background

4.1. Masking against Side-Channel Attacks

Side-channel attacks exploit the link between processed data and the corresponding observable physical quantity of a given device. Masking is part of countermeasures based on the adjustment of individual modules or the design datapath to improve security. It aims at removing that link by adding a random mask value to the actual data. Hence, all cryptographic operations are modified in order to process masked input data and return masked output data. In this context, an attacker has to analyze multiple time instants in the side-channel trace since the original data has been split into several parts. It can be more efficient than solutions such as secure logic styles in terms of overheads but is more case-specific and hardly applicable to other modules.

Two masking methods have been proposed by Messerges [15]: arithmetic masking and boolean masking. A mask m is added to a given data x of size k to produce a masked data.

- Arithmetic masking: $x_m = (x - m)\ mod\ 2^k$
- Boolean masking: $x_m = x \oplus m$

The cryptographic algorithm is no longer computed on the secret variable but on the two independent shares x_m and m. Many masking schemes have been introduced such as Threshold implementation [16], Domain Oriented Masking [17], Unified Masking [18]... Instead of using only one random value (first-order masking), one can increase the number of masks to d for example ($d + 1$ shares). The goal here is to protect against side-channel attacks that target the computation on several variables at a time. In this scheme, which is analogous to the d-probing secure circuit of Ishai et al. [19], an attacker that can record the instantaneous leakages of up to d shares of the secret variable, cannot retrieve the sensitive information. Initially seen as an empirical solution, many efforts have been made to provide formal security proofs of masking [20,21].

There have been significant improvements since the original idea, both in higher-order masking [22,23], in attacking masked circuits [24] and in formal proofs of different masking schemes, whether at software level or hardware level. In the rest of this paper, when referring to masking, we imply first order boolean masking for hardware modules.

4.2. Evaluating the Security of Masked Implementation

When investigating the robustness of circuits to first-order side-channel attacks, leakage analysis methods are used to check whether any leaks could lead to an attack. These methods only shows the existence of leakage without proving whether that leakage is exploitable or not and how this can be done. Various tools have been developed for this purpose (TVLA [25], t-test [26], correlation test [27], mutual information analysis [28], etc.).

In the case of masked implementations, these tools are still applicable, but mutual information, which makes it possible to capture all kinds of dependencies between side-channel traces and sensitive data, is still the most relevant tool, unlike, for example, the correlation test, which only points out linear dependencies. The mutual information (MI) between two random variables is the amount of the information learned on one of them when observing the other. It has been introduced as a generic distinguisher to exploit all side-channel information available in a trace with respect to a given variable [28]. Besides using MI only for leakage assessment, it can be used to give theoretical requirement on the masks generation in order to get an overall secure masking scheme. This is shown by Grosso et al. when trying to find the impact of randomness on the performances of masking schemes [29]. They compute the MI on a masked AES S-box using different masks distribution and for different noise variances. They confirmed the need for uniform masks for an optimally secured masking scheme since it is well-known that uniform distribution lead to maximum entropy, which is desired in masks generation.

The topic of evaluation of secure implementations has always been of interest in the research community. The number of traces required for a successful attack has been widely adopted as a sound metric for evaluating such implementations. Given that mutual information is the most appropriate way to analyse the leakage of a masked implementation, the challenge now is to determine how many traces it will take to carry out a successful attack based on an MI analysis. This question has been answered by Cherisey et al. by linking the number of traces required to succeed an attack and the MI through an inequality [30] recalled in Equation (1). A recent work [31] recall the link provided in [30] and argue that it can be used for both low and high number of traces, which was not the case in the initial work.

$$\frac{f(SR)}{MI} \leq N_{traces} \tag{1}$$

In this inequality, f is a known, invertible, strictly increasing function defined in [30] and that depends on the success rate SR. The only parameter to estimate here is the mutual information. MI estimation techniques have been investigated (histogram method and kernel density estimation [32]) since the perspective of its use in the context of side channels has been stated. For our study, we used MINE [33], a neural network based MI estimation tool that has been proven by the authors to perform better than the classical methods, especially with higher dimension traces. MINE is fed with the simulated traces and the set of labels corresponding to the HW of the unmasked data. Like any neural network, a learning phase is necessary to allow the tool to increase its efficiency in the classification phase. As opposed to classical deep learning context where the network is trained for further predictions, the MI is obtained by evaluating the loss function which was proven to be up-bounded by the MI. In order to prevent overfitting, the input data is split into training data and validation data. The estimated MI is then the maximum value of the loss function on the validation data.

5. Security Requirement for Mask Generation

In [29], the authors focused only on the masks distribution and noise level to analyse the security of a masking scheme. To complement their analysis, we suggest that the security of a masked implementation depends on the number of traces needed to successfully attack it. To this end, we define the following security objective for a masked implementation: "*The number of attack traces required to retrieve the unmasked data with a profiling attack has to be at least 10,000 traces*". Profiling attacks have been chosen because they are assumed to be the worst-case attack scenario for the attacked device. The threshold of 10,000 attack traces is due to the fact that most attacks published in the literature use less than 2000 traces for the attack phase, considering that it is difficult to obtain more exploitable traces on a circuit that is not under control by the adversary. Hence, setting a threshold of 10k traces allows for a reasonable security margin. We also set this threshold assuming that the attacker

can gather infinite number of traces on the profiling device, which is consistent with the principle of the attack.

Reaching this security goal is challenging if there is no substantial noise on the circuit, even if uniform masks are used. This shows the key role of noise as complementary approach to masking countermeasure. The goal of the experiments presented in this section is therefore to verify whether it is still possible to reach that objective with a biased mask distributions instead of uniform masks, and with what noise level this could be done. To do this, MI is estimated using MINE with simulated side-channel traces of masked implementation in which we vary the distribution of the masks and the noise level.

5.1. Generating Simulated Traces

Side-channel leakage based on power consumption is commonly modeled by the sum of a deterministic part related to the ongoing operation at the time of the measurement and a non-deterministic part considered as noise. This noise may be related to the measuring instrument or either to other ongoing operations that are not being targeted or to the operation of other components in the circuit. It is usually considered to be a realisation of a Gaussian variable with mean zero and standard deviation σ. According to this model, we simulate leakage traces for a sensitive variable v with Equation (2).

$$L(v) = \psi(v) + Noise \tag{2}$$

The function that links the value of v to the observed physical leakage, namely the leakage model, is represented in this equation by ψ. The traces are generated using 8-bit variables. Some random data are generated and "XORed" with the masks to obtain the masked data. We then construct traces containing two samples with the Hamming weight (HW) as leakage model ($\psi = HW$). The first sample represents the leakage of the mask and the second represents the leakage of the masked data. The standard deviation of the noise has been kept variable so that it allows us to study the influence of noise on the efficiency of masked implementations and also to evaluate the randomness properties of the masks.

5.2. Mutual Information Estimation

Depending on the type of distribution from which the masks come, a higher or lower MI can be obtained. In order to better understand this effect, we aim to characterize different masks distributions for 8-bit variables. MI is estimated between a side-channel record of a masked implementation and the target sensitive variable using simulated traces. Simulation enables the control and change of the masks generation algorithm to observe the resulting MI as a function of the Signal to Noise Ratio (SNR) which is linked to the noise level in the circuit. For this purpose, we defined two types of mask distributions.

- *Uniform masks*: They correspond to mask values generated uniformly over the 2^n possible values, where n is the size of the mask.
- *biased mask-x*: A bias is introduced in the mask values distribution. That is, among all the possible values, we randomly pick x values that will be used as masks. It means that the set of possible values has been reduced to those x values, which will lead to more repetition of the masks.

Figure 4 explains the experimental process we have been through for the MI estimation. For each noise variance, we generate the masks according to a given distribution (uniform or biased) and we simulate the corresponding side-channel leakage as explained in Section 5.1. Then we compute the MI between the trace and the unmasked data using the HW leakage model. 1000 MI estimations per noise variance were ran to remove measurement fluctuations due to noise.

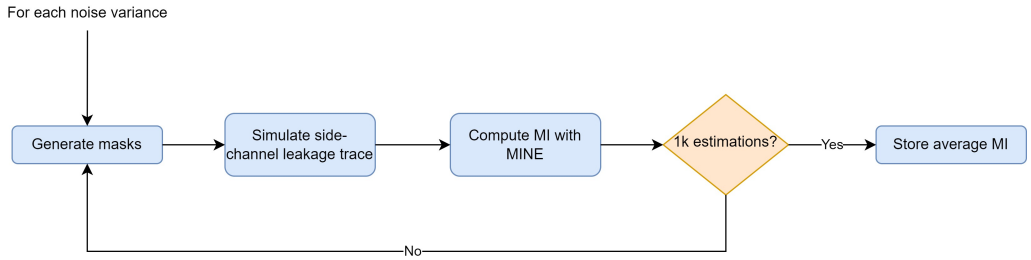

Figure 4. MI estimation process.

The simulations were done for different noise levels reflected in the SNR values. For a n-bit data and a noise variance σ^2, we compute the SNR with Equation (3).

$$SNR = \frac{n}{4\sigma^2} \tag{3}$$

For the biased masks, we used $x = 2, 4, 8, 16$. The result is depicted in Figure 5.

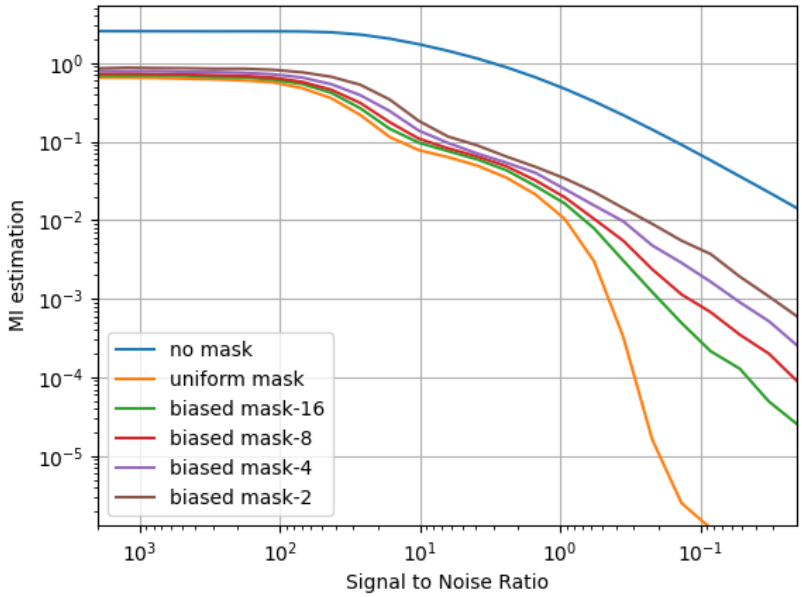

Figure 5. MI estimation for different noise and bias levels.

With a closer look at this figure, we can notice that for low noise levels, the MI is almost the same and quite high (around 1) whether the masks are biased or not. We can also see that in the presence of a high noise level (SNR between 0.01 and 0.1), the MI is significantly reduced for both uniform and biased masks. This highlights the soundness of masking when combined with noise [34]. Using the inequality (1), we can quickly realize that a potential attack would need at least 10k attack traces to succeed if biased masks are used (biased mask-16 and biased mask-8) on a device where the SNR is 10^{-2}. From these results, we can now define the architecture of a mask generator while keeping in mind that even if the output masks are biased, the security goal is still reachable.

6. The Lightweight Mask Generator (LightMaG)

In order to provide a secure masking scheme that meets the performance and area requirements associated with caches, the security and architectural constraints that any mask generator should satisfy are defined hereafter.

- **Security constraint:** the generated masks have to enable the protection of the masking scheme against an attacker with up to 10,000 attack traces.

- **Architectural constraints:**
  - The mask generator must have a low area footprint since it is intended to be implemented near the cache memories.
  - A new mask has to be available at each clock cycle
  - It must be possible to recalculate the masks as needed

In order to respect the security constraint, a TRNG would do the job as it will produce uniform masks that will require less noise than biased masks to respect our constraint. However, this solution does not respect the architectural constraint concerning the on-demand reproducibility of masks unless they are stored, which would double the size of the cache memory and thus be unacceptable to us due to area constraint. The experiments conducted in Section 5 have proven that depending on the bias, a low MI can be obtained if there is enough noise on the SoC where the generator is implemented and hence, satisfy the security requirement. Our goal then is to define a mask generator that complies with the architectural constraint while having the right bias level to also comply with the security requirement. To do this, we build our generator on top of the round function of a lightweight cryptographic primitive, Subterranean 2.0. Subterranean ranked first in hardware benchmarking either on FPGA [35] or ASIC [36] in terms of spatial footprint and speed compared to other lightweight ciphers submitted to NIST LWC competition [37]. Those results motivate the architectural choice we made for the underlying primitive of the generator. In addition to the lightweight aspect, its cryptographic properties helps ensure the ability to reproduce the masks when needed without storing the whole masks but only some input parameters. Depending on the inputs of the mask generator, the bias level in the output masks will be more or less high. In the following, we give a detailed description of the generator, starting with Subterranean round function which is the starting block of the generator. This is followed by a security evaluation of the produced masks in a worst case scenario.

6.1. Subterranean Round Function as Starting Block

The round function R of Subterranean applies a four-step transformation on a state of size 257 bits: $R = \pi \circ \theta \circ \iota \circ \chi$. Let S denote the state and S_i the ith bit of S ($0 \leq i \leq 256$). The four steps are defined as follows.

- $\chi : S_i \leftarrow S_i + (S_{i+1} + 1).S_{i+2}$
- $\iota : S_i \leftarrow S_i + \delta_i$; $\delta_i = 1 \; for \; i = 0, \; 0 \; otherwise$
- $\theta : S_i \leftarrow S_i + S_{i+3} + S_{i+3}$
- $\pi : S_i \leftarrow S_{12i}$

The authenticated encryption scheme of Subterranean invokes this round function multiple times after absorbing the encryption key, the associated data, the nonce, and the message. Although the limited spatial footprint of the whole algorithm demonstrated by Mohajerani et al. [35] (891 LUTs), we chose to apply only a couple of rounds on a predefined state in order to keep a very low area and latency overhead. A trade-off has been made between the number of rounds to use to meet the architectural requirements and the achievable security and we came up to two rounds. This choice is argued later in this section with the security analysis of the produced masks.

In the original submission, the authors defined an absorption function that injects 33 bits into the state and an extraction function that extracts 32 bits of the state. To meet our need to have 64-bit masks, we changed the structure of these functions to absorb (respectively extract) 128 bits instead of 33 bits (respectively 32 bits). Extracting 128 bits will help mask the 64-bit data and an optional 64-bit integrity tag to ensure both confidentiality and integrity of the data in the caches.

6.2. Architecture of LightMaG

The architecture of the proposed mask generator is shown in Figure 6.

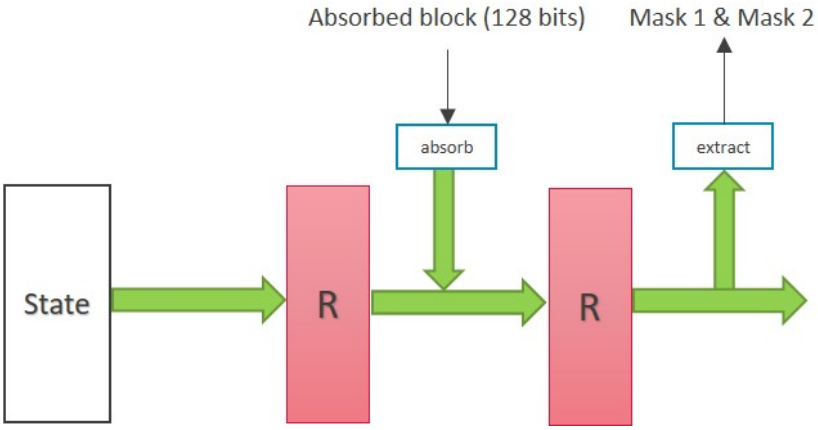

Figure 6. Architecture of LightMaG.

This generator is intended to be used for masking the content of cache memories. Therefore, we use some attributes of the data as inputs of the generator. The state is initialized with:

- A 128-bit key K,
- An 8-bit IV,
- The address of the data to mask/unmask,
- An Address Space Identifier (ASID): assigned by the Operating System (OS) to each process to distinguish its virtual address space from that of other processes (to avoid flushing the TLBs upon a context switch). This identifier is used in both ARM and RISC-V architectures.
- A Pointer identifier (Ptr_id) that allows making countermeasures against temporal and spatial memory vulnerabilities [38]

$$State \leftarrow K||ASID||IV||Ptr\_id||address||0^{16}||1^{16} \quad (4)$$

The 256 LSB bits of the initial state are permuted by the mean of DES permutation according to the scheme in Figure 7 before applying the first round.

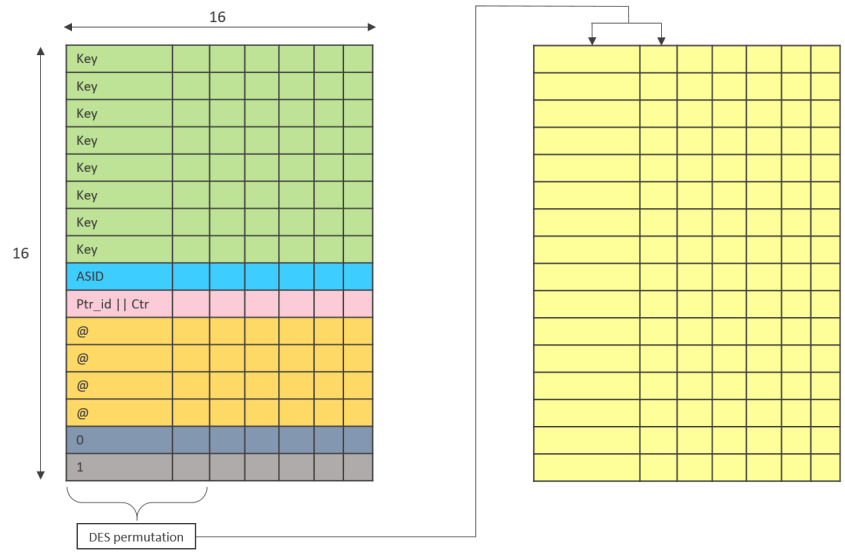

Figure 7. Permutation of initial state.

At the output of the first round, a 128-bit block is absorbed using the modified absorption function previously defined. The composition of the absorbed block is given by Equation (5) where @(48) denotes the 48 LSB bits of the address.

$$Absorbed\ block \leftarrow Ptr\_id||IV||@(48)||Perm\_DES(\overline{Ptr\_id||IV||@(48)}) \quad (5)$$

The resulting state is then sent into the second round of Subterranean. Two masks (2×64 bits) are extracted after the second round. These will be used to mask the data in the cache memories.

The structure of the initial state and the permutation performed on it, as well as the structure of the absorbed block, ensure that the masks produced are consequently variable, as can be seen in Figure 8 which shows the cumulative distribution functions of each byte of the masks (Mask 1). Here we can observe distribution functions that approach the one of uniform distribution. The same behaviour is observed in the distribution of the other 64 bits (Mask 2).

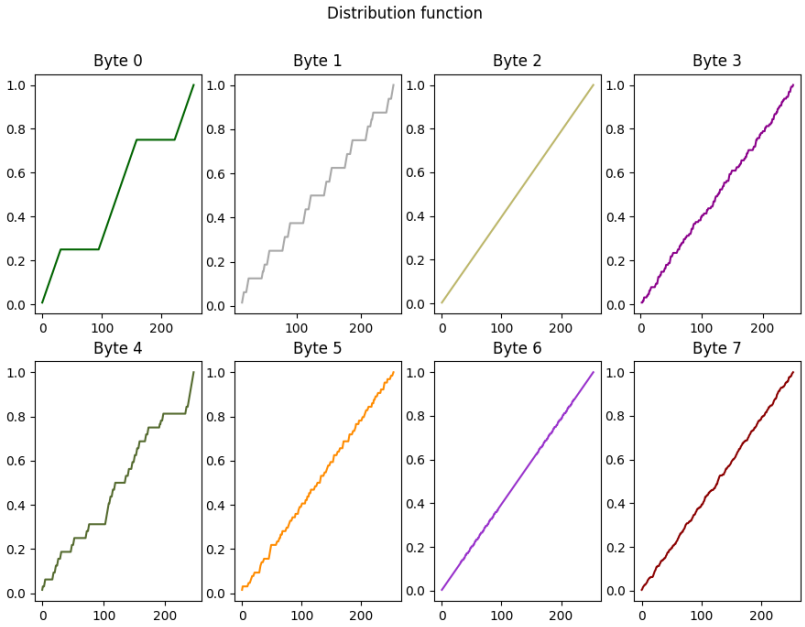

Figure 8. Cumulative distribution function of the mask values.

6.3. Security Evaluation of LightMaG Masks

After defining the architecture of the mask generator, an evaluation is imperative to check the compliance of LightMaG masks to the security requirement previously defined. Therefore, we compute MI values obtained with these masks and uniform masks for different noise levels (represented here by the SNR), again using simulated traces with HW leakage model. In order to generate the masks for the simulation, we set a fixed value (randomly chosen) for all the parameters of the initial state (address, key, Ptr_id, ASID) except the IV which is incremented between each computation. In this configuration, we will have only 256 different masks out of the 2^{64} possible masks meaning that we have highly biased masks. Figure 9 shows the MI as a function of the SNR.

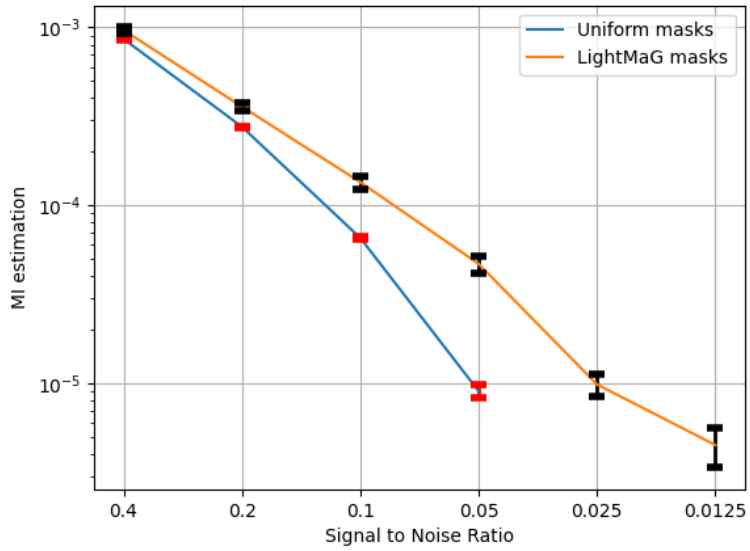

Figure 9. Comparing MIs: Uniform vs. LightMaG masks.

Masks provided by LightMaG led to higher MI than those coming from a uniform distribution, even for higher noise levels (lower SNR). This is understandable since our generator does not claim to be a TRNG. The security requirement specifies the minimum number of attack traces from which we can argue that the produced masks lead to first-order secure masking scheme. Using inequality (1), we compute this number for different success rates and a fixed MI. The results are presented in Table 1.

Table 1. Number of traces needed to for different success rates. MI set to 10^{-5} (SNR = 0.02). Data are masked with LightMaG masks.

SR	0.80	0.90	0.95	0.99
$f(SR)$	4.48	5.25	5.62	5.92
$Min(N_{traces})$	448,000	525,000	562,000	592,000

We choose the MI corresponding to a SNR of 0.02 as example, which is consistent with the SNR values observed on consumer application processors [39]. Figure 10 illustrates the SNR computed on an ARMv7 platform embedding a Cortex-A7 dual-core which is an application processor able to run an embedded linux.

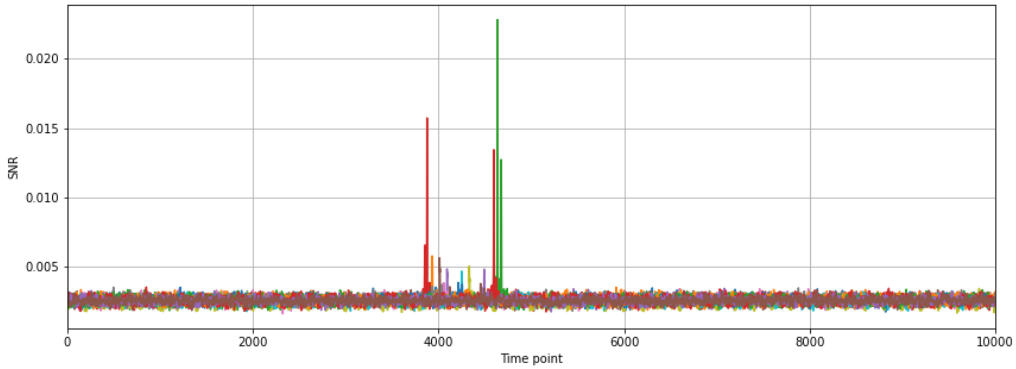

Figure 10. SNR on ARMv7 platform with Cortex-A7 dual core.

The results in Table 1 show that an attacker needs at least 592 K attack traces to get a 99% success rate to retrieve the unmasked data using a profiled side-channel attack such

as a template attack. This assumes that the attacker has an infinite number of traces in the profiling phase to build perfect profiles. These results show that our mask generator satisfies the security requirement if the SNR on the chip is 0.02, meaning that it provides better security than unprotected systems where less than 10 K attack traces are needed to retrieve the secret information. We recall that the value of 0.02 is not set here as a threshold but is the typical SNR observed on SoCs intended for IoT systems for examples. Table 1 can as well be computed for other values of SNR using inequality (1). Since our goal is to provide designers with a range of SNR in which our proposal lead to first-order secure cache memories, we first estimate the maximum MI an attacker can obtain with 10K traces (with (1)) and then derive the upper-bound on the SNR from the results depicted in Figure 9. This led to a maximum MI of 5.92×10^{-4} and a maximum SNR of 0.05 for a success rate of 99%. Hence, our solution can ensure security of circuits with SNR lower than 0.05 against first-order side-channel attack on the caches. That beying said, the typical SNR on application processors has been shown to be mostly of 0.02 which is less than 0.05.

6.4. Hardware Implementation and Performance Analysis

The masking generator has been implemented on Xilinx Kintex-7 FPGA of the Digilent Genesys 2 board in order to confirm the lightweight aspect that is claimed. The design has been synthesized using default Vivado settings. The implementation part has been done using the "*remap_LUT*" option which enables a combination of LUTs in order to reduce the area. The resulting resource utilization report shows 400 LUTs (6-input LUTs) used for a max frequency of 150 MHz. The original design of the CVA6, a 64-bit RISC-V SoC [40] used 66,510 LUTs [41]. Accordingly, the ratio between the resource utilization of LightMaG and that of the CVA6 leads to an overhead of 0.6%. The running frequency also implies that the generator will not be on the critical path if integrated on CVA6 processor that runs at 50 MHz. The performance of the SoC being quite dependent on the latency of the caches, we can ensure that our generator will not degrade the performance of the cache since it can produce the masks in one clock cycle. In addition, the generator can run in parallel which makes it transparent to the cache in terms of latency. On the other side, encrypting/decrypting 64-bit data in cache memory using the same cryptographic primitive (the whole Subterranean algorithm) requires 10 clock cycles. This means that a 10-cycle delay is added upon each load operation from the CPU whereas only 2 cycles are needed to get masked data. It also shows performance gain of masking solution compared to classical encryption of data in cache memories. These results allow us to confirm the lightweight and fast mask generation aspects of LightMaG.

7. Discussion and Perspectives

The mask generator proposed in this paper is meant to be used in a constrained environment where the available area and memory are limited and there is a need for fast masks generation. Many application processors designers are quite hesitant to implement masking solutions because of the area and memory overhead introduced by the need for fresh randomness. This is less the case for smartcards where the masking solution is more implemented [5]. The ability of LightMaG to avoid storing whole masks would be of great help in reducing the costs of memory expansion on the circuit.

The MI estimations realized in Section 6.3 gave an idea of the expected number of traces to reach a given success rate when trying to retrieve the unmasked data from a power consumption record containing the leakage of the masked data and the mask itself. More significantly, it allowed to establish an SNR range in which our solution leads to better security than unprotected devices. Let's assume that before trying to retrieve the unmasked data, an attacker targets first the mask generator itself in order to find the secret key used to generate the masks. The inputs of the generator are the key, ASID, address of the data to mask, the pointer Id, and the IV. The address, ASID, and pointer Id are publicly available and known to the attacker. The only unknown information is the IV and the key. A potential attack scenario would be to repetitively send the same data at the same

physical address and collect the power consumption traces of the circuit to first detect when the masks are being computed (which is not so obvious) and then perform a Correlation Power Analysis (CPA) or Mutual Information Analysis (MIA). A template attack would not be feasible because it assumes that the attacker has at disposal a fully controllable circuit, which cannot hold. After all, the key used by the generator is not provided by the software so it cannot be modified by a user. To succeed in his attack, he will need to find the IV and the key. This adds more complexity in the computation since the security complexity is not only 2^{128} but 2^{128+8} due to the 8-bit IV that is unknown. Yet, a *divide and conquer* approach can help lower the latter complexity and make the CPA attack feasible. In such circumstances, a re-keying strategy [42] can make the attack more difficult. The key replacement can be done either after flushing all the data of the process or as soon as there is a collision of ASIDs since the OS flushes the data of all processes before assigning a new ASID when a collision occurs.

The choice of using two rounds of subterranean allowed us to generate the masks in one cycle while having a low spatial occupation. We could try to increase the number of rounds (say to 3 or 4) to have a lower MI but in this case, more than one cycle will be necessary to generate the masks plus a larger size of the generator, which does not respect the defined architectural constraints.

Regarding the security provided by the masks, it is worth recalling that for the simulation ran in Section 6.3, all parameters forming the initial state have been fixed except the IV that increments from one trace to another. Our simulation thus illustrates the worst case because expecting a CPU to make reads and writes at the same address is not realistic. We believe that a real-world scenario with changing parameters will result in MI values that are closer to the ones from uniform masks. We also recall that according to the memory protection architecture illustrated in Figure 2, there will be at least two implementations of LightMaG, one near the CPU and the second near the last level cache. The two modules will certainly handle different data at the same time (and different input parameters also), increasing the overall noise in the circuit. This has the benefit of hardening potential side-channel attacks on the mask generator. Ongoing work focuses on conducting such experimentation with a hardware implementation of the module and side-channel traces from the actual device.

In case one wants to get rid of the IV, replacing it with some attribute of the data to mask (e.g., integrity bits from ECC memories) could be an alternative. Yet, while using this method, we need to have in mind that the same data will always be masked with the same mask, enabling a possible template attack on the generator, unless the key is changed frequently. This will also reduce the security complexity to that of the key only. Nonetheless, there is no memory overhead with this solution because there is no IV to be stored anymore. The well-known tradeoff between security and performance shows up here and the designers will have to choose to keep the IV and have better security and some memory overhead or replace it with integrity bits that will not be stored and lose some security for the sake of performance. Another tradeoff has to be made on the desired bias level in the masks. In our simulation scenario, the bias level is directly induced by the IV. One could think of increasing the size of the IV to reduce the bias but this would necessitate to store more than 8-bit variables for every 64-bit data in the caches. In the end, this choice also has to be done by the designers between performances and security.

8. Conclusions

Masking countermeasure is widely adopted as a solution against side-channel attacks. In this paper, our main concern is to ensure data confidentiality in the memory hierarchy by providing a fast mask generation mechanism to avoid carrying encrypted data in cache memories while keeping their content protected against first-order side-channel attacks. The data are encrypted on the interconnect bus, decrypted and masked before entering the caches, and stay masked up to the CPU where computations could be done on masked data.

After finding that biased masks distribution could also contribute to a secure masking scheme if there is enough noise on the chip, we presented our lightweight mask generator and a security analysis of the masks produced by LightMaG compared to uniform masks. The results show a gap between the MIs obtained from the two methods but the work of Cherysey et al. [30] allow us to assess that a potential attacker will need at least 592 K attack traces to get 99% chances to retrieve the unmasked data on a chip with a SNR of 0.02, which is a large number of traces. Beyond this, although the relevance of the security analysis given for an SNR of 0.02, we primarily intended to provide designers with a range of SNR ($SNR \leq 0.05$) in which our solution fits better their needs both in terms of security and performances.

The advantage of the solution lies in the reduced area overhead (only two rounds of Subterranean resulting in 400 LUTs after design implementation on FPGA) and the fact that no mask will be stored for further unmasking except the 8-bit IV. We also gave some insights on the possible use of LightMaG to lighten a memory hierarchy encryption mechanism. This generator is intended to motivate hardware designers to implement masking countermeasure in their SoCs. Integrating the mask generation module in a complete SoC design and run experimentations on a real ASIC circuit is the focus of our next work.

Author Contributions: Conceptualization, E.B.T., O.S., M.B.D.N. and D.H.; Supervision, O.S., M.B.D.N. and D.H.; Writing—original draft, E.B.T.; Writing—review & editing, O.S. and M.B.D.N. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded thanks to the French national program "Programme Investissement d'Avenir IRT Nanoelec" ANR-10-AIRT-05.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Barenghi, A.; Pelosi, G. Side-channel security of superscalar CPUs: Evaluating the Impact of Micro-architectural Features. In Proceedings of the 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 24–28 June 2018; pp. 1–6.
2. You, S.C.; Kuhn, M.G. A Template Attack to Reconstruct the Input of SHA-3 on an 8-bit Device. In Proceedings of the International Workshop on Constructive Side-Channel Analysis and Secure Design, Lugano, Switzerland, 1–3 April 2020; pp. 25–42.
3. Arsath K F, M.; Ganesan, V.; Bodduna, R.; Rebeiro, C. PARAM: A Microprocessor Hardened for Power Side-Channel Attack Resistance. In Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST) 2020, Virtual Event, 7–11 December 2020; pp. 23–34.
4. Rožić, V.; Dehaene, W.; Verbauwhede, I. Design solutions for securing SRAM cell against power analysis. In Proceedings of the 2012 IEEE International Symposium on Hardware-Oriented Security and Trust, San Francisco, CA, USA, 3–4 June 2012; pp. 122–127.
5. Kocher, P.C.; Jaffe, J.M.; Jun, B.C. Cryptographic Computation Using Masking to Prevent Differential Power Analysis and Other Attacks. US Patent US7668310B2, 23 February 2010.
6. Daemen, J.; Massolino, P.M.C.; Mehrdad, A.; Rotella, Y. The Subterranean 2.0 cipher suite. *IACR Trans. Symmetric Cryptol.* **2020**, 262–294. Available online: https://tosc.iacr.org/index.php/ToSC/article/view/8622 (accessed on 6 November 2021). [CrossRef]
7. Gala, N.; Menon, A.; Bodduna, R.; Madhusudan, G.S.; Kamakoti, V. SHAKTI processors: An open-source hardware initiative. In Proceedings of the 2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID), Kolkata, India, 4–8 January 2016; pp. 7–8.
8. Giterman, R.; Keren, O.; Fish, A. A 7T security oriented SRAM bitcell. *IEEE Trans. Circuits Syst. II Express Briefs* **2018**, *66*, 1396–1400. [CrossRef]
9. Marshall, B.; Page, D.; Webb, J. MIRACLE: MIcRo-ArChitectural Leakage Evaluation. *IACR Cryptol. ePrint Arch.* **2021**, *2021*, 261. [CrossRef]
10. Talaki, E.B.; Noes, M.B.D.; Savry, O.; Hely, D.; Bacles-Min, S.; Lemaire, R. Exposing Data Value On a Risc-V Based SoC. In Proceedings of the 2021 IEEE Physical Assurance and Inspection of Electronics (PAINE), Washington, DC, USA, 30 November–2 December 2021; pp. 1–8.
11. Unterluggauer, T.; Werner, M.; Mangard, S. MEAS: Memory encryption and authentication secure against side-channel attacks. *J. Cryptogr. Eng.* **2019**, *9*, 137–158. [CrossRef] [PubMed]

12. Yin, T.; Xin, G.; Han, J. HPME: A High-Performance Hardware Memory Encryption Engine Based on RISC-V TEE. In Proceedings of the 2020 IEEE 15th International Conference on Solid-State Integrated Circuit Technology (ICSICT), Kunming, China, 3–6 November 2020; pp. 1–3.
13. Wong, M.M.; Haj-Yahya, J.; Chattopadhyay, A. SMARTS: Secure memory assurance of RISC-V trusted SoC. In Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy, Los Angeles, CA, USA, 2 June 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 1–8.
14. Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
15. Messerges, T.S. Securing the AES Finalists against Power Analysis Attacks. In Proceedings of the Fast Software Encryption, New York, NY, USA, 10–12 April 2000; Lecture Notes in Computer Science; Goos, G., Hartmanis, J., van Leeuwen, J., Schneier, B., Eds.; Springer: Berlin/Heidelberg, Germany, 2001; pp. 150–164. [CrossRef]
16. Nikova, S.; Rechberger, C.; Rijmen, V. Threshold implementations against side-channel attacks and glitches. In Proceedings of the International Conference on Information and Communications Security, Raleigh, NC, USA, 4–7 December 2006; pp. 529–545.
17. Groß, H.; Mangard, S.; Korak, T. Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order. In Proceedings of the TIS@ CCS, Vienna, Austria, 24–28 October 2016; p. 3.
18. Gross, H.; Mangard, S. A unified masking approach. *J. Cryptogr. Eng.* **2018**, *8*, 109–124. [CrossRef]
19. Ishai, Y.; Sahai, A.; Wagner, D. Private Circuits: Securing Hardware against Probing Attacks. In Proceedings of the Advances in Cryptology—CRYPTO 2003, Santa Barbara, CA, USA, 17–21 August 2003; Lecture Notes in Computer Science; Boneh, D., Ed.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 463–481.
20. Prouff, E.; Rivain, M. Masking against side-channel attacks: A formal security proof. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, 26–30 May 2013; pp. 142–159.
21. Rivain, M.; Prouff, E. Provably Secure Higher-Order Masking of AES. In Proceedings of the Cryptographic Hardware and Embedded Systems, Santa Barbara, CA, USA, 17–20 August 2010; Lecture Notes in Computer Science; Mangard, S., Standaert, F.X., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 413–427.
22. Schramm, K.; Paar, C. Higher Order Masking of the AES. In Proceedings of the Topics in Cryptology—CT-RSA 2006, San Jose, CA, USA, 13–17 February 2005; Pointcheval, D., Ed.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 208–225.
23. Journault, A.; Standaert, F.X. Very High Order Masking: Efficient Implementation and Security Evaluation. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2017, Taipei, Taiwan, 25–28 September 2017; Fischer, W., Homma, N., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 623–643.
24. Moos, T.; Moradi, A.; Schneider, T.; Standaert, F.X. Glitch-Resistant Masking Revisited—Or Why Proofs in the Robust Probing Model are Needed. Cryptology ePrint Archive. Report 2018/490. 2018. Available online: https://ia.cr/2018/490 (accessed on 18 March 2022) .
25. Becker, G.; Cooper, J.; DeMulder, E.; Goodwill, G.; Jaffe, J.; Kenworthy, G.; Kouzminov, T.; Leiserson, A.; Marson, M.; Rohatgi, P. Test vector leakage assessment (TVLA) methodology in practice. In Proceedings of the International Cryptographic Module Conference, Gaithersburg, MD, USA, 24–26 September 2003; Volume 1001, p. 13.
26. Standaert, F.X. How (not) to use welch's t-test in side-channel security evaluations. In Proceedings of the International Conference on Smart Card Research and Advanced Applications, Montpellier, France, 12–14 November 2018; pp. 65–79.
27. Brier, E.; Clavier, C.; Olivier, F. Correlation power analysis with a leakage model. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Cambridge, MA, USA, 11–13 August 2004; pp. 16–29.
28. Batina, L.; Gierlichs, B.; Prouff, E.; Rivain, M.; Standaert, F.X.; Veyrat-Charvillon, N. Mutual information analysis: A comprehensive study. *J. Cryptol.* **2011**, *24*, 269–291. [CrossRef]
29. Grosso, V.; Standaert, F.X.; Faust, S. Masking vs. multiparty computation: How large is the gap for AES? *J. Cryptogr. Eng.* **2014**, *4*, 47–57. [CrossRef]
30. de Chérisey, E.; Guilley, S.; Rioul, O.; Piantanida, P. Best information is most successful. *Cryptol. Eprint Arch.* **2019**. [CrossRef]
31. Masure, L.; Dumas, C.; Prouff, E. A comprehensive study of deep learning for side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2020**, 348–375. [CrossRef]
32. Prouff, E.; Rivain, M. Theoretical and practical aspects of mutual information-based side channel analysis. *Int. J. Appl. Cryptogr.* **2010**, *2*, 121–138. [CrossRef]
33. Cristiani, V.; Lecomte, M.; Maurine, P. Leakage Assessment through Neural Estimation of the Mutual Information. In Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS), Rome, Italy, 22–25 June 2020; Volume 12418, pp. 144–162. [CrossRef]
34. Chari, S.; Jutla, C.S.; Rao, J.R.; Rohatgi, P. Towards Sound Approaches to Counteract Power-Analysis Attacks. In Proceedings of the Advances in Cryptology–0CRYPTO' 99, Santa Barbara, CA, USA, 15–19 August 1999; Lecture Notes in Computer Science; Wiener, M., Ed.; Springer: Berlin/Heidelberg, Germany, 1999; pp. 398–412. [CrossRef]
35. Mohajerani, K.; Haeussler, R.; Nagpal, R.; Farahmand, F.; Abdulgadir, A.; Kaps, J.P.; Gaj, K. FPGA Benchmarking of Round 2 Candidates in the NIST Lightweight Cryptography Standardization Process: Methodology, Metrics, Tools, and Results. *IACR Cryptol. ePrint Arch.* **2020**, *2020*, 1207.
36. Aagaard, M.D.; Zidaric, N. ASIC Benchmarking of Round 2 Candidates in the NIST Lightweight Cryptography Standardization Process. *IACR Cryptol. ePrint Arch.* **2021**, *49*, 1–49.
37. Computer Security Division Information Technology Laboratory. *Lightweight Cryptography*; CSRC: Gaithersburg, MD, USA, 2017.

38. Nasahl, P.; Schilling, R.; Werner, M.; Hoogerbrugge, J.; Medwed, M.; Mangard, S. CrypTag: Thwarting physical and logical memory vulnerabilities using cryptographically colored memory. In Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security, Virtual Event, 7–11 June 2021; pp. 200–212.
39. Yan, Y.; Yu, J.; Guo, P.; Guo, J. Utility analysis and evaluation method study of side channel information. *J. Electron.* **2013**, *30*, 500–508. [CrossRef]
40. Zaruba, F.; Benini, L. The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2019**, *27*, 2629–2640. [CrossRef]
41. Koene, D. Implementation and Evaluation of Packed-SIMD Instructions for a RISC-V Processor. Master's Thesis, Delft University of Technology, Delft, The Netherlands, 2021.
42. Medwed, M.; Standaert, F.X.; Großschädl, J.; Regazzoni, F. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In Proceedings of the International Conference on Cryptology in Africa, Stellenbosch, South Africa, 3–6 May 2010; pp. 279–296.