A memory hierarchy protected against side-channel attacks

Ezinam Talaki

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : EEATS - Electronique, Electrotechnique, Automatique, Traitement du Signal (EEATS)
Spécialité : Nano électronique et Nano technologies
Unité de recherche : CEA/LETI

Une hiérarchie mémoire protégée contre les attaques par canaux auxiliaires

A memory hierarchy protected against side-channel attacks

Présentée par :

Ezinam TALAKI

Direction de thèse :

David HELY Maitre de Conférence, Université Grenoble Alpes	Directeur de thèse
Mathieu BOUVIER DES NOES CEA	Co-encadrant de thèse
Olivier Savry CEA	Co-encadrant de thèse

Rapporteurs :

Guy GOGNIAT
PROFESSEUR DES UNIVERSITES, UNIVERSITE BRETAGNE SUD - LORIENT VANNES
Pascal BENOIT
Maître de conférences HDR, Université de Montpellier

Thèse soutenue publiquement le **27 septembre 2022**, devant le jury composé de :

Guy GOGNIAT PROFESSEUR DES UNIVERSITES, UNIVERSITE BRETAGNE SUD - LORIENT VANNES	Rapporteur
Pascal BENOIT Maître de conférences HDR, Université de Montpellier	Rapporteur
Vincent BEROULLE PROFESSEUR DES UNIVERSITES, Grenoble INP	Examinateur
Giorgio DI NATALE DIRECTEUR DE RECHERCHE, CNRS DELEGATION ALPES	Président

Invités :

Olivier SAVRY
DOCTEUR EN SCIENCES, CEA/LETI
Mathieu BOUVIER DES NOES
DOCTEUR EN SCIENCES, CEA/LETI

Acknowledgement

C'est avec plaisir que j'utilise la première page de mon manuscrit pour remercier toutes les personnes qui ont eu un rôle dans ce long parcours qui se termine. Merci à tous ceux qui ont permis à ce travail de thèse de commencer et de s'achever, ceux qui m'ont assisté, qui m'ont guidé et encouragé.

J'aimerais d'abord remercier Pascal BENOIT, Guy GOGNIAT, Giorgio DI NATALE et Vincent BEROULLE qui ont accepté évaluer mon travail en acceptant prendre part à mon jury de thèse.

Ensuite, je tiens à remercier mes encadrants Mathieu BOUVIER DES NOES, Olivier SAVRY et David HELY pour leur soutien sans faille depuis le début de cette aventure. Merci pour les différentes corrections apportées à mes écrits, pour le temps investi afin de m'aider à bien exprimer ma problématique et à bien l'expliquer. Merci pour l'aide aussi bien technique que rédactionnelle. Merci de m'avoir motivé dans les moments de doute où les délais semblaient intenables.

Merci à Mathieu pour les différentes questions pertinentes sans lesquelles je n'aurais pas pu orienter mon travail. Merci pour l'appui technique pendant toute la première partie de la thèse.

Merci à Olivier qui m'a aidé pour la deuxième partie de cette thèse et qui m'a guidé dans les différents choix à opérer ainsi que pour les différents échanges techniques que nous avons tenus.

Merci à David qui s'est impliqué dès le début de cette thèse et qui a toujours été disponible pour corriger mes différents écrits, à des heures tardives des fois. Merci pour les différentes remarques me permettant d'améliorer ma capacité à retransmettre et à vulgariser mon travail, et aussi pour les différents échanges footballistiques que nous avons tenus.

Un grand merci à Karim qui a toujours été disponible malgré sa charge de travail, pour me permettre de pouvoir réaliser mes expériences, et aussi pour son aide concernant l'implémentation des contre-mesures proposées dans cette thèse dans un processeur d'application. Merci à tous les membres du LSOSP pour leur accueil et pour les différents échanges enrichissants. Je tiens à remercier mes collègues doctorants du service SSSEC et je leur souhaite une bonne continuation et de belles soutenances.

Je remercie aussi tous mes amis qui m'ont soutenu tout au long de cette aventure et qui m'ont permis de penser à autre chose que le travail : Alexandre, Esso, Jean-Marie, Tècle, Amine, Nikos ainsi que mon Lieutenant Firmin.

Enfin, je remercie l'ensemble de ma famille pour son soutien indéfectible. Mes derniers remerciements vont à ma mère Brigitte, ma fille Ehana et à ma compagne Sonia pour tout le soutien qui m'a permis de garder la motivation dans les derniers moments et d'aller au bout de cette thèse.

Dédicace spéciale à mon père Basile qui a su trouver les mots justes pour m'encourager depuis le début de mes études et jusqu'à sa disparition.

$\mathcal{E}$LABAL$\mathcal{E}$ KPEM.

Contents

Chapter 1

Introduction

Contents

1.1 Information security: embedded systems perspective

One lesson we have learned from the massive use of teleworking as well as IT tools and infrastructures since the beginning of the covid-19 crisis is the need to protect our data from potential malicious people. The French security certification agency, ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information), even reported a four times increase in the number of victims of cyber attacks since 2019 [ANS]. The electronic gadgets used in our daily lives (smartphones, smartcards, laptops) also process large amounts of data. Users may not want to leave these data in plain sight. Some attackers could aim to retrieve our sensitive data, modify it or impersonate us. These three scenarios result in compromising the confidentiality, integrity, and authenticity properties of the data respectively. These properties are supposed to be guaranteed in a secure system. However, these systems are not only gathering user data, but also system data and important information for its effective functioning. The need for security is hence clearly urgent, especially for embedded systems.

1.1.1 On the need of securing data in embedded systems

Several advances have been made in the miniaturization of embedded systems thanks to the concept of *System on Chip* (SoC), which refers to an integrated circuit (IC) that contains almost all the components needed for an embedded system. Figure 1.1 recalls the architecture of a SoC by showing its main components. The main goal of that race

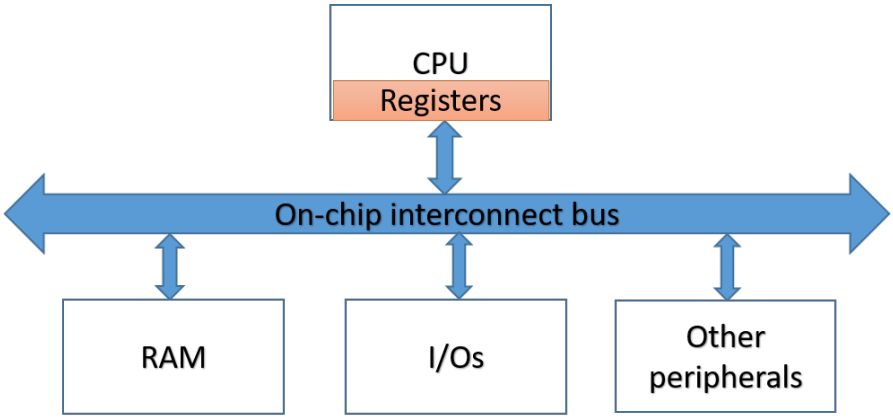


Figure 1.1: Simplified architecture of a SoC.

to miniaturization was to reduce power consumption, increase the performance of the

system and reduce the spatial footprint of previously large systems. Instead of adopting a multi-chip design, people started integrating all components needed by an embedded system on a single chip. The side effect of the increased use of such devices in all parts of our lives is that a huge amount of data manipulated by these chips makes them a real "*goldmine*" for malicious parties who would be tempted to break into them and cause serious harm to users. The attacks on System on Chips (side-channel attacks, microprobing, hardware fault injection attacks) have highlighted the vulnerability of consumer electronics. This not only applies to small micro-controllers embedded in IoT devices, but also to processors in computers and mobile phones. It raises the need to ensure the security of such data while being manipulated in the SoC. One of the most valuable actions for an attacker with access to a SoC is to be able to spy on the communication between the modules and retrieve the contents of the various memory elements in the circuit. This shows the importance of protecting the data that moves within the circuit. The oldest and best-known method for providing this security is data encryption. Through an encryption algorithm, the original data is modified according to a well-defined mathematical scheme so that the resulting data looks random to anyone who does not have the secret encryption key. However, the hardware and software implementations of such algorithms still suffer from hardware attacks that leverage the physical properties of the chip on which they are implemented.

1.1.2 Vulnerability of cryptographic modules to hardware attacks

The various efforts to standardize encryption algorithms by standardization bodies such as NIST, FIPS, etc. demonstrate the scientific community's long experience in designing reliable encryption algorithms with a suitable level of security against so-called cryptanalysis attacks. However, the range of possible attacks is not limited to cryptanalysis. Once the encryption algorithm is implemented on a SoC, the attacker has a wide range of choices in terms of attacks that can be carried out (either on software or hardware) depending on his knowledge of the target and his skills. For example, he can exploit the circuit that implements the encryption algorithm to find the encryption key and then be able to decrypt the data. This has given rise to the class of attacks related to the exploitation of hardware implementations. Hardware has long been viewed as a trusted party supporting the whole computer system and whose only purpose is to execute instructions passed from the software. Hardware security has come a long way since the emergence of hardware Trojans. Nowadays, there are more advanced

attacks such as fault injection, hardware reverse engineering, physical and software side-channel attacks, etc.

1.1.3 Side-channel attacks on System on Chips

Within the class of hardware attacks is the category comprising side-channel attacks. While running a program, the CPU generates some interactions between all the components of the SoC in order to operate on data related to the current program. This can be by loading data from the main memory, transferring it on the interconnect bus, storing it in cache memories and registers, and performing operations on it in the execute stage of the pipeline. In the case of cryptographic algorithms, these data are the plaintext, the ciphertext, the key, and all the intermediate variables generated through the computation. All these components mobilized during the execution of the program are made up of gates whose power consumption depends on the manipulated data [MOP08]. That power consumption leads to the spread of other physical quantities such as electromagnetic emanations. There are also other channels through which it is possible to infer the computation going on in the CPU and the internal data that is processed. Those quantities, also referred to as side channels, are illustrated in Figure 1.2.

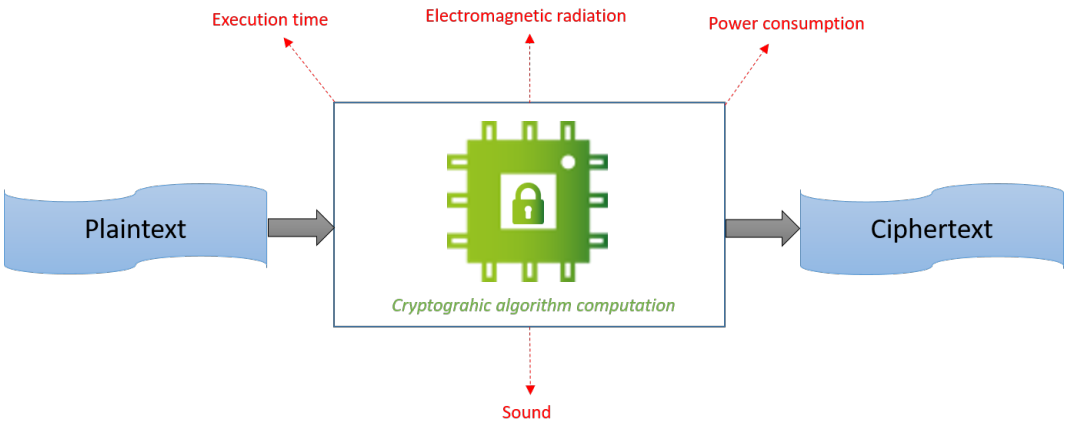


Figure 1.2: Side channels of a cryptographic module.

Exploiting these channels is considered particularly harmful because it is based on the observation of the circuit, without any perturbation of the chip's behavior. This is not the case for perturbation-based attacks like fault injection attacks that uses laser or electromagnetic glitches. Reported for the first time by Kocher *et al.* [KJJ99], side-

channel attacks have been since then widely applied by attackers in order to leverage power consumption or electromagnetic emanations of target devices to reveal sensitive information.

Mainly used to reveal secret keys used in symmetric and asymmetric cryptographic modules on SoCs (DES, AES, PRESENT, RSA, ECC...), these attacks are increasingly used to attack any type of micro-architectural component on a SoC. These attacks will be more described in chapter 2 with a look at the various information that was available and the corresponding countermeasures developed in the literature.

1.2 Leakage of micro-architectural components

1.2.1 On the side-channel vulnerability of the micro-architecture

Once the link between the consumption of logic gates and the manipulated data is established, a relevant application is the retrieval of encryption keys on hardware devices. At this point, it is relevant to target physical implementations of cryptographic algorithms. People were only trying to attack these specific modules in the SoC. Mindful of the power of such attacks, the community's interest gradually shifted to other parts of the micro-architecture that was left out of the security analysis till then. It allows an evaluator to identify and understand the origin of observed leakage regardless of the program running on a device. Hence, components like registers, ALU, cache memories, and interconnect bus started being targeted by exploiting their side-channel leakage.

1.2.2 The specific case of interconnect buses and registers

Data transfers in a SoC are done through the channels known as buses. They are used to connect different modules in order to ensure the proper functioning of the entire system. In all available SoC architectures, there is a central bus that connects the majority of the modules, commonly referred to as the interconnect bus. Due to its central position in the system, all the data of the circuit potentially goes through it. This creates an attractive attack surface for an adversary since as soon as the data transiting through it is known to the attacker, the whole system can be considered compromised. In addition, a lot of secret information such as encryption keys or user-sensitive data can be found there.

Registers are the smallest memory elements that are usually located close to the CPU, where intermediate computational variables are stored. Such internal data can often be sensitive and therefore need to be protected from any malicious person.

In view of the above, the registers and the interconnect bus, considered the nerve center of the SoC, must be protected against side-channel attacks in order to guarantee the confidentiality and integrity of the data handled inside.

1.2.3 Attacker model definition

The various side-channel vulnerabilities reported on microarchitectural components led to the definition of an attacker model that will be considered throughout this thesis, as it will serve as a starting point for finding a suitable countermeasure to these vulnerabilities. To ensure security of data in the memory hierarchy against side-channel attacks, the resources and information an attacker could target in the memory hierarchy of a RISC-V based SoC have been defined. The model is built on top of the Dolev-Yao attacker model [DY83], considering the attacker to be able to intercept or forge any message in the "non-protected world" of the SoC like depicted in Figure 1.3. Invasive

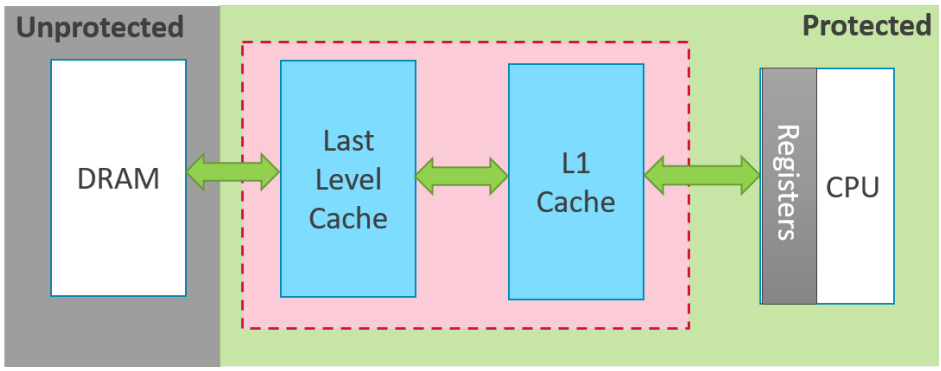


Figure 1.3: Protected vs unprotected parts against invasive attacks on the SoC.

hardware attacks such as microprobing [SK08] and Focused Ion Beams (FIB) on the CPU and caches are not considered in this attacker model. Higher-order attacks are out of the scope of our analysis. The possible attacks of the adversary are hence limited to first-order such as template attacks or Correlation Power Analysis (CPA). 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 attacker has access to the functional specifications of the SoC (datasheet, user manual...) usually available for users or software developers. Implementation details such as source code and hardware design are considered critical and not known to the attacker.
- **Access to the chip:** the attacker has:
  - *physical access*: he can measure power consumption and EM radiation of the circuit
  - *logical access*: possibility to run a software program on the CPU in order to access registers, I/Os, GPIOs, and data in available memories.
- **Main goal:** the attacker can retrieve any data in the memory hierarchy or at least their Hamming weight (HW) or Hamming distance (HD) via first-order side-channel attacks.
- **Expertise:** in order to run such attacks, he is considered to have some knowledge about algorithms, protocols, hardware structures, principles, and concepts of security and side-channel attacks.

1.3 Motivations of the thesis

It makes no doubt that the leakage exploited in side-channel attacks on cryptographic algorithms stems from the microarchitecture. Due to their sensitivity, encryption algorithms attract many attackers. For this reason, several studies focus on how to exploit side channels to find the secret keys used inside. However, these modules do not constitute the only critical parts of a circuit. All the microarchitectural components of the chip can suffer from such attacks as they manipulate different kinds of data that can be as critical as encryption keys. Typically, the vulnerability of interconnect bus and registers to side-channel attacks raises the concern about the effect such attacks would have on these key components in a SoC. This yields also the challenge of how to guarantee the confidentiality and integrity of communications between a CPU and the main memory, as well as the other modules on the SoC (e.g. FLASH, DMA, peripherals). They are among the most critical elements in terms of security, because all the

internal data of the circuit can be found there. An attacker with access to the contents of the CPU registers or to the data passing on the bus through a side-channel attack can breach the confidentiality of all the data (instruction disassembly, proprietary code recovery, user's sensitive data disclosure). This thesis focuses on reaching the following goals:

- Demonstrate and quantify the capabilities of an adversary to successfully perform side-channel attacks to retrieve information on the data either during their transfer on the interconnect bus or when they are stored in the registers.
- Design and implement countermeasures to these attacks taking into account the strong constraints of size, performance, and power consumption required by the need to move towards increasingly compact and efficient SoCs. This is done in light of the information on the success rate of previously held attacks.

This work is meant to be implemented on a SoC embedding a 64-bit RISC-V application processor.

1.4 Contributions of this thesis

This thesis focuses on the study, quantification, and protection of micro-architectural elements against side-channel without restricting its scope to cryptographic modules only. In order to achieve the goals highlighted in section 1.3, several milestones have been set up and led to the contributions hereby summarised.

1. **Side-channel leakage characterization of interconnect bus and CPU registers**

   One of the first steps toward the protection of such components consists in conducting a leakage assessment. Hence, the focus is on assessing side-channel vulnerabilities of the interconnect bus and registers. The goal is to run some characterization tests to precisely define the side-channel leakage model of those components. Among the available tests, correlation test and SNR test have been used to confirm the Hamming weight (HW) and/or Hamming distance (HD) leakage models already observed on flip-flops by Moradi *et al.* [Mor+08]. HW leakage has been observed on the interconnect bus with a peak correlation coefficient of 0.8 which is quite high. On the other side, HW and HD leakages have been observed on CPU registers with lower peak values but are clearly distinguishable.

2. **Template attacks on interconnect bus and registers: threat quantification**

   After confirming the leakage behavior of the target modules, an effort has been made to evaluate the probability of success when exploiting their power leakage through a template attack [CRR02]. It helps quantify the threats on such modules if a malicious adversary attempts to exploit their side-channel leakage. As a result, the HW of data in registers is retrieved with a success rate of 90% using 50 attack traces. The HD between two data consecutively stored in a register was retrieved using 200 attack traces to reach the same success rate. For the interconnect bus, the HW of data is recovered with 15 attack traces at a success rate of 99%. On the other hand, an investigation is done to check whether targeting directly the value of the data on the interconnect bus is realistic or not. The template attack led to a success probability of 0.96 for 200 attack traces and a residual complexity of $2^{13.2}$ for enumerating the remaining values to test after the ranking from the template attack.

3. **Countermeasure: memory hierarchy protection against side-channel attacks**

   A formal and accurate assessment and quantification of the adversary's abilities when having a power consumption record of the interconnect bus and registers helps define a suitable countermeasure to protect such modules. In the countermeasure definition, the goal is not only protecting these two components but all the memory hierarchy. This is because protecting them individually may lead to overall security that is not as complete as implementing a security scheme that is meant for the whole memory hierarchy. The proposed solution is based on authenticated encryption using a lightweight cipher. Yet, encrypting the content of cache memories will incur large overheads both for spatial footprint, memory size, and latency. To avoid performance losses, a lightweight mask generation scheme is proposed for masking data in caches instead of keeping encrypted data inside them. The proposed mask generator can output 64-bit masks at each clock cycle using a lightweight cipher (Subterranean 2.0 [Dae+20]). Instead of storing all the 64 bits of the generated masks, only an 8-bit Initialisation Vector (IV) is stored. The masks can be recomputed if needed from the IVs. The security level of the overall scheme is then assessed through the residual Mutual Information of the leakage of the masked data.

1.5 Organisation of the manuscript

In addition to this introductory chapter, the rest of this manuscript consists of the following chapters organized as follows:

- **Chapter 2** reviews the different side-channel attacks published in the literature while targeting cryptographic algorithms. A state-of-the-art of the existing methodologies for side-channel leakage assessment and the countermeasures to such attacks is also provided. The Chapter ends with a focus on microarchitectural leakage by summarizing the various studies on the exploitation and mitigation of such leakages.

- **Chapter 3** presents the first part of this work which focuses on side-channel leakage assessment of interconnect bus and general purpose registers of the CPU. Further on, the identified leakages have been exploited through template attacks in order to gain information on the data handled by such modules. This leads to the quantification of the capabilities of the identified attacker model.

- **Chapter 4** presents the investigations on the protection scheme to implement to ensure security of data in the memory hierarchy. A security analysis of the designed solution is also provided, as well as a description of the overall functioning of the system once the countermeasure is integrated.

- **Chapter 5** summarizes the contribution of the thesis and discusses the limitations and perspectives.

1.6 Context of the thesis

This thesis is part of the Nanotrust project which aims at developing a System on Chip that is intrinsically secure against side-channel and fault injection attacks. Starting from the CVA6 processor [ZB19] which is a 64-bit RISC-V based System on Chip, the goal is to add countermeasures against side channels, fault injection, cache attacks, etc. to obtain an application processor that relieves the software developer of the necessity to think about security as it is intrinsically handled by the hardware and the operating system. Hence, the countermeasures that have been proposed to protect memory hier-

archy from side-channel attacks are meant to be implemented on the final application processor of the Nanotrust project.

1.7 List of publications

The work carried out in this thesis has yielded the following publications and patent:

1. E. B. Talaki, M. Bouvier Des Noes, O. Savry, and D. Hely, "Side-channel Leakage Assessment On RISC-V Architecture", Trudevice Workshop, Mar. 2020.
2. E. B. Talaki, M. Bouvier Des Noes, O. Savry, D. Hely, S. Bacles-Min, and R. Lemaire, "Exposing Data Value On a Risc-V Based SoC", in 2021 IEEE Physical Assurance and Inspection of Electronics (PAINE), Nov. 2021.
3. E. B. Talaki, O. Savry, M. Bouvier Des Noes, and D. Hely, "A Memory Hierarchy Protected against Side-Channel Attacks", MDPI Cryptography, June 2022.
4. E. B. Talaki, O. Savry "Procédé de protection contre les attaques par canaux auxiliaires", Patent FR 2201933, March 2022.

Résumé du chapitre

L'utilisation massive du télétravail ainsi que des outils et infrastructures informatiques depuis le début de la crise du covid-19 est la nécessité de protéger nos données contre d'éventuelles personnes malveillantes. L'Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI), a même signalé une multiplication par quatre du nombre de victimes de cyberattaques depuis 2019. Les gadgets électroniques utilisés dans notre vie quotidienne (smartphones, cartes à puce, ordinateurs portables) traitent également de grandes quantités de données et certains attaquants pourraient chercher à récupérer nos données sensibles, à les modifier ou à se faire passer pour nous. Ces trois scénarios ont pour conséquence de compromettre respectivement les propriétés de confidentialité, d'intégrité et d'authenticité des données. Le besoin de sécurité est donc clairement urgent, en particulier pour les systèmes embarqués. Parmi les différentes attaques existantes sur ces systèmes, cette thèse se propose d'étudier celles visant un certain nombre de composants des *système sur puces* (System on Chip ou SoC en Anglais) existant dans les systèmes embarqués.

Les transferts de données dans un SoC se font par l'intermédiaire de canaux appelés bus. Ils sont utilisés pour connecter différents modules afin d'assurer le bon fonctionnement de l'ensemble du système. Dans toutes les architectures de SoC disponibles, il existe un bus central qui relie la majorité des modules, communément appelé bus d'interconnexion. En raison de sa position centrale dans le système, toutes les données du circuit passent potentiellement par lui. Cela crée une surface d'attaque attrayante pour un adversaire, car dès que les données qui y transitent sont connues de l'attaquant, l'ensemble du système peut être considéré comme compromis. En outre, de nombreuses informations secrètes, telles que des clés de chiffrement ou des données sensibles pour l'utilisateur, peuvent s'y trouver.

Les registres sont les plus petits éléments de mémoire, généralement situés à proximité de l'unité centrale de calcul, où sont stockées les variables de calcul intermédiaires. Ces données internes peuvent souvent être sensibles et doivent donc être protégées contre toute personne malveillante.

Compte tenu de ce qui précède, les registres et le bus d'interconnexion, considérés comme le centre nerveux du SoC, doivent être protégés contre les attaques à canal latéral afin de garantir la confidentialité et l'intégrité des données qui y sont traitées.

Cette thèse s'inscrit dans le cadre du projet Nanotrust qui vise à développer un sys-

tème sur puce intrinsèquement sécurisé contre les attaques par canaux auxiliaires et par injection de fautes. Ele a permis de proposer une caractérisation fine des fuites du bus d'interconnexion et des registres afin de pouvoir réaliser une attaque par profilage sur les données y passant. Ensuite, une solution de protection de tout le système a été proposée en tenant compte des contraintes de performance requises pour un système embarqué.

Chapter 2

Side-channel attacks: From cryptographic keys to all embedded information

Contents

2.1 Overcoming encryption algorithms with side-channel attacks

Upon the rise of cryptography, assessing the security of proposed encryption algorithms has mainly been done through classical cryptanalysis which exploits mathematical weaknesses of cryptographic algorithms. After implementing these algorithms on SoCs, hardware attacks started proliferating and especially side-channel attacks. Attackers leverage the power consumption of hardware devices to retrieve sensitive information.

A link has been established between the data manipulated by a CMOS gate and its energy consumption [MOP08]. That is, the amount of power consumed for a transition from '0' to '1' differs from that of '1' to '0'. From that point, it is possible to find a correlation between the actual data handled in a hardware component and the power consumption of the latter. During execution of a cryptographic operations, depending on the amount of energy consumed by the module, it is possible to guess the value of the data that is being computed. This is particularly interesting when secret information is being processed, namely the encryption key or intermediate variables of the computation. Here, the power consumption is considered as a side channel, a backdoor through which an attacker can reveal the secret information of a mathematically secure encryption algorithm. There exist also other kinds of side channel, such as execution time, electromagnetic radiations, sound... Side-channel attacks are hence hardware attacks that rely on the observation of side channels to reveal secret information inside an implementation. The typical equipment needed to perform this attack consists mainly of an oscilloscope to observe and record the side channel information from the target circuit and a computer to process it.

From a practical point of view, the attacker has several obstacles to overcome to successfully retrieve the researched data.

- **Leakage acquisition:** The first step of the attack is to collect the power consumption of the attacked circuit. Since consumer circuits are not manufactured with the primary intention of providing the user with the ability to observe power consumption, it becomes difficult for the attacker to find the right measurement point. It might be the right pin for measuring the current consumption of the target module or the right coordinates for measuring the electromagnetic radiation

of the module. To achieve this, he will have to perform several tests and compare the accuracy of the attack results.

- **Noise in measurements:** Once the barrier of measurement points finding has been removed, the next issue to be addressed is that of noise in the measurements. Like any communication channel, the side channels are considered noisy. As a result, the signal observed on the scope includes a useful signal and some noise. The useful signal is the deterministic part of the signal, corresponding to the consumption associated only with the operation of the observed module. The noise in side-channel attacks, also modelled as Additive White Gaussian Noise (AWGN), can come from several sources such as:
  - Power consumption related to operations other than the one being attacked, either on the module being attacked or on other modules in the circuit
  - Noise inherent to the measurement tool

  Some efforts have to be made in order to lower or remove the noise in the recorded signals for a better efficiency of the attack. It can be done by averaging the traces since the noise is supposed to be of zero mean. Once it is done, one can compute the Signal to Noise Ratio (SNR) which allows to compare the level of the noise part in a signal to the level of the informative part.

- **Leakage modelling:** After estimating the noise part in the signals, the attacker now gets the power consumption related to the actual computation. The next step is to model that signal. A function, also called leakage model, is used to link the target intermediate variable to the physical leakage. Hamming weight (HW) and Hamming distance (HD) are the most used leakage models. Yet there exist other models such as the Most Significant Bit (MSB) of the variable [Tim19], its direct value or the "signed HD" [CLH19]. The more accurate the model, the more successful the attack.

- **Classification of guesses:** This is the most important part of the attack. After the prior work that has been done on noise lowering and leakage modelling, the attacker will make hypotheses on the target value and process the resulting data by running some statistical analysis (T-test [Sta18], CPA [BCO04], LRA [Dog+11], SM [SLP05]...) or information theory analysis (MIA) [Bat+11] for all the hypotheses in order to rank them according to their scores that are given after the

analysis. The risk here is to end up with the correct guess not being at first position after the ranking. In order to lower that risk, instead of looking at only the hypothesis ranked first, the ones ranked at the first n positions are examined in order to reduce the number of possibilities and then to carry out other tests to finally come up with a single value that will be the correct one.

These attacks, as mentioned at the beginning of this section, were initially aimed at recovering encryption keys that are not easily accessible by means of classical cryptanalysis. The practical implementation of the attacks that is described above allows not only to retrieve encryption keys but also any other information processed in the remaining modules of a SoC such as registers, interconnect bus or cache memories. This approach broadens the scope of side-channel attacks and makes them even more dangerous for users of embedded systems and connected objects. Proprietary code could be disassembled [Str+15; CLH19], sensitive user data disclosed, authentication data obtained by a third party.

In the rest of this manuscript, we focus on this use of SCA attacks to retrieve any information about a SoC through its power consumption.

2.2 Attack strategies

The final goal of an attacker is to retrieve a security critical data through the power leakage at his disposal, due to the statistical relationship between the power consumption and the data. Once the leakage has been recorded, many statistical or information theory methods are available for extracting the desired information from the record. Researchers and industrials have worked to enhance the efficiency of these attacks leading to two main attack strategies: profiling and non profiling attacks.

2.2.1 Preliminaries and notations

Throughout this chapter and the rest of this manuscript, we use calligraphic letters as $\mathcal{Y}$ to denote sets. For a finite set $\mathcal{Y}$, its cardinality, i.e. the number of elements in this set is denoted by $|\mathcal{Y}|$. Random variables will be denoted by upper-case letters Y (respectively $\vec{Y}$ for random vectors), and the corresponding lower-case letters y (respectively $\vec{y}$) for their realizations. The i^{th} entry of the realization vector $\vec{y}$ is denoted $\vec{y}[i]$. The bold letter $\mathbf{M}$ denotes a matrix.

We will represent a sensitive variable of size n by the random variable Z whose realizations are from the set $\mathcal{Z} = \{z_1, ..., z_{|\mathcal{Z}|}\}, |\mathcal{Z}| = 2^n$.

2.2.2 Profiling attacks

The leakage of a target device is analysed and characterised with the leakage of a reference device that has the same leakage property as the target. The reference device is used to build leakage profiles that will help attacking the target device. To this end, the reference device is assumed fully controllable by the attacker in order for him to build efficient profiles. The attack is divided into two steps.

- **Profiling step:** For each of the possible values of the target sensitive variable (e.g. an instruction, an encryption key, some data in any component of the microarchitecture), the corresponding leakage is exploited to build a profile. This is done using the reference device. At the end of this step, the couples variable/profile are obtained for each possible values of the variable.

- **Attacking step:** Here comes the real attack where the previously built couples are compared to the leakage of the target device in order to infer the value of the variable that matches the best to the leakage recorded on this device. This is done by classifying the newly recorded leakage among the previously constructed pairs. The variable corresponding to the matched leakage profile is then the correct value.

We can notice, from the description given above that this kind of attack, under the condition of having an open circuit, can theoretically lead to the best success rate with a single trace from the target device. However, many practical issues can undermine the success of such attacks in practice. These issues will be covered in the description of a well known profiling attack hereafter.

2.2.2.1 Example of template attack

As many methods based on statistical estimation, side-channel attacks need several data, the so-called *traces*, to reach accurate estimations of the secret data under attack. Template attack have been introduced in 2002 by Chari et al. and is a well mastered profiling attack. The authors noticed that many solution against side-channel attacks are designed with the purpose of limiting the number of traces available to a potential

attacker [CRR02]. It means that they reduce the number of possible repetitions of the critical operation that leads to the leakage of the sensitive variable. In order to bypass that restriction, template attack tries to exploit the maximum information from a side channel trace and thus, makes use of a multivariate noise model to classify side channel traces. Using those models the attack can theoretically succeed with only one trace. The profiles here are the so-called *templates*.

Let's assume that we have a device that performs a sequence of operations manipulating a sensitive variable Z of size n. Let $\{Op_i\}_{i\in\{1,\dots,2^n\}}$ be the set of possible operations manipulating each possible value z_i of the sensitive variable. The observed trace t_i is divided into a deterministic part (actual signal from computation of the real sequence) and a noise sample that is considered to be drawn from a probability distribution. The deterministic part is estimated by averaging the traces since the noise part is assumed to be independent and drawn from a centered Gaussian random variable. The latter part is estimated using a multivariate normal distribution. For each operation Op_i, its corresponding leakage trace is modelled as the realization of a multivariate random Gaussian variable X with mean M_i and covariance matrix C_i, where M_i and C_i depend on the operation Op_i. From the leakage x_{target} recorded on the target device, the attacker tries to guess whether it originates from the operation Op_j by maximizing the probability $Pr(Op_j|x_{target})$.

- **Templates building or profiling phase**: This step consists of building a multivariate Gaussian noise model from the recorded side-channel traces on the reference device. The multivariate normal distribution of the N-dimensional random variable X is described by its mean vector M_i and its covariance matrix C_i and the Probability Density Function (PDF) is described by equation (2.1).

$$p(X|Op_i) = \frac{1}{\sqrt{(2\pi)^N|C_i|}}\exp(-\frac{1}{2}(X-M_i)^\intercal {C_i}^{-1}(X-M_i)) \tag{2.1}$$

  A given operation Op_i related to some data z_i is characterized by its template which is the couple (M_i, C_i). To compute the template, the attacker records a large number of traces having a dimension D, for each operation Op_k (l traces per operation). The mean trace and the covariance matrix are computed with equation

(2.2) over the recorded traces for each operation.

$$M_i = (m_i[1], ..., m_i[D]), \quad m_i[k] = \frac{1}{l}\sum_{j=1}^{l} x_j[k],\ 1 \leq k \leq D$$
$$C_i = \frac{1}{l-1}\sum_{j=1}^{l}(x_j - M_j)^\intercal(x_j - M_j) \tag{2.2}$$

In this equation, x_j is the j^{th} leakage trace recorded from operation Op_i and $m_i[k]$ is the k^{th} sample of the mean trace M_i.

In practice, the success of the attack depends on the quality of the templates. This implies an efficient calculation of the mean and covariance matrix for each operation. A high sampling rate is usually needed in order to get as much information as possible in a clock cycle of the running device. This leads to a large number D of samples per trace, typically around 100k samples. Using the whole trace leads to large matrix inversion issue since the covariance matrix size is quadratic in D. Furthermore, an excessive computational overhead and memory usage is induced. The better approach is then to only consider samples that leak the maximum information on the data before any further processing. Those samples are the so-called *Points of Interest* (PoI). Many methods have been proposed for the PoIs selection from simplistic and intuitive ones to data analysis related ones, including sum of difference [RO04], t-test [DS16], Principal Component Analysis [Arc+06], Linear Discriminant Analysis [SA08]. Templates can also be built on different power models like HW, HD or reduced parts of intermediate variables [HTM11].

- **Template matching or attacking phase**: After building the template, the attacker tries to identify an unknown operation running on the target device by recording its power leakage. With the assumption that the templates are perfectly constructed (perfect estimation of noise and leakage model), the attack is theoretically expected to succeed with a single trace from the target device [CRR02]. This is not the case in practice. The attacker measures a small number of traces (n_a) leading to the leakages $\{x_{target,k}\}_{k\in\{1,...,n_a\}}$ to increase his chances to find the correct value. For each measurement, he computes what we will refer to as a "matching probability density" $p_{k,i}$. This is the probability that the leakage $x_{target,k}$ matches

with the template (M_i, C_i). It is computed with equation (2.1). Under the assumption that the obtained power leakages are independent, the matching probability densities for each single trace can be multiplied using equation (2.3) to have the overall probability of classifying the operation leading to x_{target} from the target device as coming from the secret data z_i.

$$p_i(x_{target}|(M_i, C_i)) = \prod_{k=1}^{a} p_{i,k} \tag{2.3}$$

To each possible operation Op_i, a classification probability $p(Op_i|x_{target}) = p(z_i|x_{target})$ is associated. It is computed with Bayes' theorem:

$$p(Op_i|x_{target}) = \frac{p_i(x_{target}|Op_i)Pr(Op_i)}{Pr(x_{target})} \tag{2.4}$$

The correct operation $\hat{Op_j}$is obtained by means of equation (2.5).

$$\hat{Op} = \underset{j\in\{1,\ldots,2^n\}}{argmax}\ (p(Op_j|x_{target})) = \underset{j\in\{1,\ldots,2^n\}}{argmax}\ \frac{p_j(x_{target}|Op_j)Pr(Op_j)}{Pr(x_{target})} \tag{2.5}$$

It is worth recalling that the marginal probability $Pr_{x_{target}}$ does not depend on the hypothesis on the executed operation and hence, is usually neglected. In addition, the prior probability $Pr(Op_j)$ is assumed to be uniform so it does not influence the ranking of the data hypothesis.

2.2.2.2 Stochastic model

Stochastic Model (SM) has been originally proposed by Schindler et al. [SLP05] and differs from Template Attack (TA) in the way the leakage is characterized. They both exploit the signal and noise part of the side-channel traces. SM uses linear regression to model the signal part while the mean vector is used in TA. In this condition, we assume that at a given sample, the signal can be expressed as the linear combination of each bit value of the target variable. On the other hand, the noise part is also characterized by the covariance matrix but we do not compute a covariance matrix for each possible value of the target variable like it is the case in template attack. Only one covariance matrix is computed with all noise samples.

Let suppose an attacker that records a set $\mathcal{T} = (t_1, \cdots, t_{n_1})$ of n_1 side-channel traces. The stochastic model is used to approximate the leakage function L that describes the recorded side-channel measurements ($L = L_t + N$, with N a Gaussian distributed noise). We consider l_t as the deterministic part of a single leakage measurement of the sensitive variable z of size n at the time instant t. l_t is expressed as the linear combination of the leakage of each bit value of the sensitive data, having different leakage coefficients. In equation (2.6), $\alpha_{i,t}$ is the leakage coefficient associated to the i^{th} bit of z.

$$l_t = \alpha_{0,t} + \sum_{i=1}^{n} \alpha_{i,t} z[i],\ \alpha_{i,t} \in \mathbb{R} \tag{2.6}$$

This is also expressed using matrices like shown in equation (2.7).

$$\begin{bmatrix} l_{1,t} \\ l_{2,t} \\ \cdots \\ l_{n_1,t} \end{bmatrix} = \begin{bmatrix} 1 & z_1[1] & \cdots & z_1[n] \\ 1 & z_2[1] & \cdots & z_2[n] \\ \cdots & \cdots & \cdots & \cdots \\ 1 & z_{n_1}[1] & \cdots & z_{n_1}[n] \end{bmatrix} \begin{bmatrix} \alpha_{0,t} \\ \alpha_{1,t} \\ \cdots \\ \alpha_{n,t} \end{bmatrix} \tag{2.7}$$

$l_{k,t}$ represents the leakage in trace t_k at the time instant t. Let denote the following matrices:

$$\mathbf{A} = \begin{bmatrix} 1 & z_1[1] & \cdots & z_1[n] \\ 1 & z_2[1] & \cdots & z_2[n] \\ \cdots & \cdots & \cdots & \cdots \\ 1 & z_{|n_1}[1] & \cdots & z_{n_1}[n] \end{bmatrix}, \mathbf{l_t} = \begin{bmatrix} l_{1,t} \\ l_{2,t} \\ \cdots \\ l_{n_1,t} \end{bmatrix}, \alpha_{\mathbf{t}} = \begin{bmatrix} \alpha_{0,t} \\ \alpha_{1,t} \\ \cdots \\ \alpha_{n,t} \end{bmatrix} \tag{2.8}$$

The goal of the adversary is to compute the leakage coefficients matrix $\alpha_{\mathbf{t}}$. This is done by solving equation (2.9).

$$\mathbf{l_t} = \mathbf{A}\alpha_{\mathbf{t}} \Rightarrow \alpha_{\mathbf{t}} = (\mathbf{A}^\intercal \mathbf{A})^{-1} \mathbf{A}^\intercal \mathbf{l_t} \tag{2.9}$$

The obtained coefficients help characterize the deterministic part of the leakage. The next step is to compute the covariance matrix to model the noise part of the leakage. To do this, another set of n_2 traces is recorded from the reference device. The noise samples are then computed using equation (2.10) where $s_{j,k}$ is the noise sample in trace

t_j at dimension k.

$$s_{j,k} = t_{j,k} - \left[\alpha_{0,k} + \sum_{i=1}^{n} \alpha_{i,k} z[i] \right], \; j \in [1, n_2] \tag{2.10}$$

From these samples, the covariance matrix $\mathbf{C}$ is computed with equation (2.11) where $\vec{s_k}$ denotes the n_2-dimensional noise vector at the k^{th} dimension.

$$\mathbf{C} = \begin{bmatrix} Cov(\vec{s_1}, \vec{s_1}) & Cov(\vec{s_1}, \vec{s_2}) & \cdots & Cov(\vec{s_1}, \vec{s_D}) \\ Cov(\vec{s_2}, \vec{s_1}) & Cov(\vec{s_2}, \vec{s_2}) & \cdots & Cov(\vec{s_2}, \vec{s_D}) \\ \cdots & \cdots & \cdots & \cdots \\ Cov(\vec{s_D}, \vec{s_1}) & Cov(\vec{s_D}, \vec{s_2}) & \cdots & Cov(\vec{s_D}, \vec{s_D}) \end{bmatrix} \tag{2.11}$$

In the attacking phase, the same process as in template attack is performed. The matching probability densities are computed for each hypothesis on the target sensitive variable, using the same equations as in the template attack. Then, using the Maximum Likelihood Principle (MLP), the hypothesis leading to the maximum matching probability density is the correct one.

2.2.3 Non profiling attacks

In most situations, no fully controllable clone device is available. Hence, profiling attacks are no longer practicable. The attacker is limited to making hypotheses on the leakage model of the chip instead of characterizing it like it is the case in profiling attacks. The success rate of the attack is then impacted by the underlying hypothesis on the leakage model. Under these conditions, one can rely on leakage models that are commonly adopted in the literature (Hamming weight, Hamming distance, MSB, LSB [CKN00]) which suit quite well to most target modules. In light of these models, some distinguishers such as the correlation are used to assess the soundness of the model, leading to the well-known Correlation Power Analysis (CPA) [BCO04]. In case no common model is sound to explain the leakage acquired on a device, more generic distinquishers such as mutual information are used [Bat+11].

- **Simple and Differential Power Analysis**
  A first and simpler way to perform a side-channel attack is to infer a running operation just by observing the side-channel information of a device on a scope

directly with "*naked eyes*". This is, for example, try to identify the processing of a sensitive variable with a simple look at the power consumption variations on the device. It leads to the so-called Simple Power Analysis (SPA) performed by Kocher et al. [KJJ99] on an RSA implementation. This attack succeeds if the power consumption during processing of the secret data is clearly distinguishable from other data in the circuit.

From this simple attack, more elaborated ones appeared. Instead of using the variation in the observed raw consumption as distinguishing method, statistical tests are done to remove the dependence to the "*observer's eyes*" and get better success rates, especially when there is no directly observable variation in the power consumption according to the manipulated data. In addition, it allows to perform a larger number of measurements in order to optimize the success rate of the attack. The Difference of Means (DoM) [CRR02] appeared to be the adequate method, leading to the so-called Differential Power Analysis (DPA) [Koc+11].

Let consider an attacker willing to retrieve a sensitive variable z of size n manipulated in a target module on the chip. After recording n_1 traces where the same operation on the variable is repeated on different possible values of z, he can proceeds with a *divide-and-conquer* approach to retrieve each bit of the sensitive variable. The traces are divided into two subsets according to the hypothesis on the leakage of the device. A difference is computed between averages of the two subsets. If the partitioning is uncorrelated to the measurements, the difference will approach zero when the number of traces grows. It will rather approach a non zero value if there is a correlation between the partitioning and the measurements.

- **Correlation Power Analysis** In this attack, instead of computing DoM between subsets, the goal is to use the Pearson's correlation coefficient as distinguisher [BCO04] to rank different hypothesis on the secret variable. The hypothesis leading to the highest correlation is kept as the correct one.

- **Mutual Information Analysis** The mutual information between two random variables is defined as the quantification 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 [Bat+11] instead of the correlation coefficient that only points out linear dependencies between variables.

More details on the using of correlation coefficient and mutual information will be given in section 2.3 that recall their usage in side-channel leakage assessment.

2.3 Side-channel leakage assessment

Since the appearance of side-channel attacks, many countermeasures have been proposed to mitigate them. Before applying any countermeasure, leakage analysis methods are used to check whether there are any leaks that could lead to an attack. After detecting the leakage and localising the sources of that leakage in the chip, a suitable countermeasure can be defined and implemented. To achieve that goal, a trivial and intuitive evaluation method is to carry out a large set of known side-channel attacks to check the resistance of the device. This method is interesting in such a way that it gives a good idea of the security level of the component but the process is time consuming and computationally heavy due to the large number of attacks having different computing complexities. Coron et al. introduced leakage assessment [CKN00] as a procedure to assess side-channel information leakage. The goal is to check whether the information leaked by a device can be exploited by a side-channel attack. Hence, there is no more need to perform several attacks since it gives generic tests. The side aspect of that method is that, due to its generic approach, it could lead to false negative results. However, that evaluation method has been widely adopted by the research community after the work of Becker et al. [Bec+13] where a practical application of leakage assessment methodology referred to as Test Vector Leakage Assessment (TVLA) is given. It makes use of statistical tests (t-test, correlation coefficient...) to assess the exploitable nature of the leaked side-channel information. There are also information theory based tests that estimate the probability distribution of the leakage. These evaluation tests usually need some assumptions about the leakage model or if not, try to estimate the leakage distributions. They could therefore be biased by wrong assumptions on the model or bad estimations of density. Durvaux et al. give a methodology to observe the impact of assumptions errors on the security and the confidence level in the performed test [DSV14]. It is also recommended to not stuck at only one method but to confirm the results of one method with another one in order to increase the confidence level in the results.

2.3.1 Statistical (empirical) tests

Assessing the vulnerabilities of a chip can be done through statistical techniques that can help quantify the leakage of the given chip.

2.3.1.1 Welch's t-test

Welch's t-test is an extension of the Student's t-distribution based test [Mat+13]. Given two sets and the null hypothesis H_0: "the two sets are not distinguishable", the purpose of the t-test is to check whether H_0 is satisfiable or not and compute the threshold of confidence in the result. Gilbert Goodwill et al. [GJR11] give an overview of how to apply this test for leakage assessment on cryptographic devices. Beyond cryptographic devices, the leakage of any module on the chip can be characterized using t-test. The test statistic and the degree of freedom are given in equation (2.12).

$$t = \frac{\mu_0 - \mu_1}{\sqrt{\frac{\sigma_0^2}{n_0} + \frac{\sigma_1^2}{n_1}}}, \quad v = \frac{(\frac{\sigma_0^2}{n_0} + \frac{\sigma_1^2}{n_1})^2}{\frac{(\frac{\sigma_0^2}{n_0})^2}{n_0 - 1} + \frac{(\frac{\sigma_1^2}{n_1})^2}{n_1 - 1}} \tag{2.12}$$

Where μ_0 and μ_1 (respectively σ_0 and σ_1) are the means of the two subsets (respectively the variances), n_0 and n_1 being the subsets sizes. The common threshold for t is ± 4.5 meaning that when $|t| > 4.5$ the null hypothesis is rejected. In side channel analysis context, according to the way set of traces are built we can distinguish two types of t-test [SM15]:

- **Specific t-test**: The collected dataset is divided into two subsets according to the possible values of the sensitive information that is being tested ('0' and '1' for a single bit of an intermediate value). This method relies on some hypothesis on the performed attack, the leakage model... and therefore is linked to the a priori hypothesis on the type of attack [GJR11].
- **Non specific t-test**: One can decide to run more generic test and be independent of specific attacks, intermediate values and leakage models. Here, the device under test is fed with two type of data: a selected one and a random one. The two types of data are sent to the device in a random manner and then the two subsets are built for the t-test.

In the initial description of TVLA, the Difference of Means of the two datasets is used.

Schneider and Moradi proposed an extension to higher orders leakage in [SM15] for univariate and multivariate leakage. In the end, we can retain that TVLA uses almost the same distinguisher as standard DPA attack. The main difference between them is that the TVLA is not an attack and acts like a *pass-fail* test with a given confidence level (the null hypothesis is rejected with a confidence of 99.99% for a test score $|t| > 4.5$ [GJR11]) since it could lead to false negatives/positives.

2.3.1.2 Correlation test

Pearson's correlation coefficient defines the linear dependence between two random variables. Given X and Y, two random variables (discrete or continuous), the correlation coefficient ρ is defined by:

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}, \ -1 \leq \rho \leq +1, \ Cov(X,Y) = E[(X - E(X))(Y - E(Y)] \quad (2.13)$$

Where $Cov(X,Y)$ is the covariance between X and Y, σ_X and σ_Y (respectively $E(X)$ and $E(Y)$) the standard deviation of X and Y (respectively the means of X and Y). The use of the correlation coefficient has been introduced by Brier et al. for the CPA attack[BCO04].

Finding the appropriate leakage model. In the context of leakage assessment, the correlation is computed between the recorded leakage and the model that has been priorly assumed for that leakage in order to check the soundness of the model. Building a model of the observed power consumption is not a trivial work since there are several unknown parameters that can impact the consumption such as the ambient temperature, duty cycles or the clock's shape, and most of the time, the underlying physical implementation details (component libraries and consumption profiles) are not known to the attacker. However, several works have enabled a convergence towards a simple and functional modelling of side-channel leakage which consists in splitting it into a deterministic part, coming from the actual computation on the target variable, and a random part which groups together all the noise occurring during the measurements and on the circuit. According to this model, the leakage of a sensitive variable z is expressed in equation (2.14)

$$L(z) = \psi(z) + B \quad (2.14)$$

$\psi : \mathcal{Z} \rightarrow \mathbb{R}$ is the deterministic function that links the value of z to the observed physical leakage, namely the leakage model. B is a Gaussian zero-mean random variable independent from z and represents the noise. A comprehensive work on the power consumption of CMOS circuits has been done by Mangard et al. in [MOP08]. From that book, we can see that the most intuitive leakage model for a Flip-Flop is the Hamming Distance (HD) model because of the dynamic power consumption of those elements when '0' to '1' and '1' to '0' transitions occur. From that model, we can also derive the Hamming Weight (HW) model which matches well the leakage of interconnect buses for example. Hence, HD and HW models are widely used for leakage analysis due to their simplicity. Nonetheless, they are not the only leakage models possible [Dog+11]. There are also some leakage models based on data value instead of data transition. Value-based leakage functions model the leakage of a variable assuming that the values of the variable is directly leaked [Cor+12; Bal+15]. One can also consider the monobit leakage model (for example $\psi(z) = MSB(z)\ or\ LSB(z)$, used in [Ché+19b]). Another possibility is to consider the leakage to be a linear combination of the bits of z:

$$\psi(z) = \sum_{i=1}^{N} \alpha_i z[i],\ \alpha_i \in \mathbb{R} \tag{2.15}$$

α_i is the leakage coefficient associated to the bit $z[i]$. We can notice that in the particular case where the coefficients are the same for all bit ($\forall i\ \alpha_i = \alpha$), we get a Hamming weight model. Since the goal is not to attack a device but to determine its leakage behaviour, one may try all the available leakage models in order to find the one that suits better the measured leakage. Hence, the focus is not on discriminating different hypothesis on the sensitive variable (which is unknown) like it is in the CPA, but to find the leakage behaviour that leads to the highest correlation coefficient for a known variable.

Practical characterization. From a practical point of view, the characterization can be done in the following steps:

- generate a random value that will serve as the sensitive data
- acquire l side-channel traces $(t_i)_{i=1...l}$ corresponding to the leakage of l different random values
- define the leakage model ψ

- compute the leakage of the sensitive data using the previously defined leakage model
- compute the correlation coefficient between the actual leakage from the traces and the leakage model
- confirm the soundness of the assumed leakage model or look for another model according to the obtained correlation factor.

2.3.2 Signal-to-Noise-Ratio (SNR) test

Side channel attacks are based on measurements of a chip's physical characteristics through a scope. The physical leakage contains some noise that is either related to the executed program or independent noise from other parts of the circuit. The SNR is a ratio of the power of the desired signal to the power of the noise. It indicates the level of useful signal compared to noise. This ratio is commonly used in electrical engineering and signal processing.
In a digital environment, it is defined by the formula 2.16

$$SNR = \frac{Var(signal)}{Var(noise)} \tag{2.16}$$

In the context of side channels, the success of the assessment depends on the ability to capture the deterministic leakage in a noisy environment. Estimating the SNR helps knowing the expected success rate of an attack and also guide the designers when it comes to implementing protection mechanisms. A low SNR will make the attack less successful than a high one. This is consistent with the fact that a widely used countermeasure against side-channel attacks is the noise addition. The need to have a high SNR value has been highlighted by Mangard *et al.* in [MOP08] where they established a relationship between the correlation coefficient and the SNR in equation (2.17).

$$\rho = \frac{1}{\sqrt{1 + \frac{1}{SNR}}} \tag{2.17}$$

This shows that a high SNR leads to a better estimation of the correlation factor.

In practice there are several methods for measuring the SNR. Since it is impossible to measure separately the signal part and the noise component of the leakage, we can

use two plaintext configurations[Yan+17]: fixed plaintext and varied plaintext sets. With a constant plaintext the data part does not change so $Var(data)$ is null and we can extract $Var(noise)$. With a varied plaintext, $Var(data + noise)$ is observed. With the assumption of Gaussian noise, the variance of noise can be decreased with a high averaging of the side-channel acquisitions. This will help extract $Var(data)$. Another possibility is to estimate the SNR from a single acquisition campaign using the equation 2.18

$$SNR = \frac{Var(E(L/Z))}{E(Var(L/Z))} \tag{2.18}$$

Where L is the collected physical leakage and Z the intermediate variable.

2.3.3 Information theory test: mutual information

More than computing statistics on the sensitive variables and comparing them to the traces in order to define the leakage behaviour of a module, a direct comparison between their probability distributions can lead to more accurate characterizations. This is the goal of information theory tests. The most suitable tool for carrying out these tests is the well-known Mutual Information. It is defined as the quantification of the information learned on one random variable when observing the another one. It has been introduced as a generic distinguisher to exploit all kind of side-channel information available in a trace with respect to a given variable [Gie+08] as opposed to the correlation that highlights only linear dependencies between them. Let X and Y be two random variables that are jointly distributed. The mutual information $I(X, Y)$ between them is defined in equation (2.19) where p_{XY} is the joint probability distribution function of X and Y.

$$I(X, Y) = H(X) - H(X|Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p_{XY}(x, y) \log_2 \frac{p_{XY}(x, y)}{p_X(x) p_Y(y)} \tag{2.19}$$

$H(X)$ represents the entropy of the variable X. It can be understood as the average amount of information one can get by observing a realization x of X.

$$H(X) = \sum_{x \in \mathcal{X}} p_X(x) log_2(\frac{1}{p_X(x)}) \; ; \; H(X|Y) = \sum_{y \in \mathcal{Y}} p_Y(y) H(X|Y = y) \tag{2.20}$$

In the context of side-channel leakage assessment the MI $I(Z, X)$ between the sensitive variable Z and the trace X is computed.

The main advantage of using MI is the relaxation of the constraint on modelling the leakage prior to running the test. But this comes with some side effects. The computation of the MI requires the estimation of some probability densities which needs a large amount of traces, and this can become practically difficult to carry. Correct estimation of these PDFs is hence a critical step in computing the MI. In this regard, instead of talking about MI, Renauld et al. introduced the notion of "Perceived Information" (PI) [Ren+11] as a generalisation of the notion of mutual information that takes into account the impact of process variability of deep submicron technologies on the PDFs estimation. The PI is up-bounded by the MI. They are equal if the assumed leakage model is the correct one. Hence, for leakage evaluation, it can be interesting to think of the PI while keeping in mind that the mutual information reveals the worst case security level [Poz+15]. The most common ways to estimate MI are the histogram method, kernel density and parametric estimation both described in [PR10]. There exist also estimation methods based on more advanced techniques based on deep learning [Bel+18].

Mutual information estimation with neural networks. The MI is equivalent to the Kullback-Leibler (KL) divergence between the joint probability distribution p_{ZX} and the product of the marginal ones p_Z and p_X.

$$I(Z, X) = D_{KL}(p_{ZX} \mid\mid p_Z \otimes p_X) \tag{2.21}$$

The KL-divergence is defined as,

$$D_{KL}(p||q) = E_p\left[log(\frac{p}{q})\right] \tag{2.22}$$

With p absolutely continuous with respect to q. Equation (2.21) shows another method to estimate the MI. A neural estimator of mutual information based on Donsker-Varadhan's representation of KL-divergence [DV83] has been proposed lately [Bel+18]. Using this representation, the KL-divergence is expressed in equation (2.23) where the supremum is taken over all functions T such that the two expectations are finite.

$$D_{KL}(p||q) = \sup_{T:\Omega\to\mathbb{R}} E_p[T] - log(E_q[e^T]) \tag{2.23}$$

From this representation, we can see that for all set $\mathcal{F}$ of functions $T : \Omega \to \mathbb{R}$ leading to finite expectations the following lower bound can be established:

$$D_{KL}(p||q) \geq \sup_{T \in \mathcal{F}} E_p[T] - log(E_q[e^T]) \tag{2.24}$$

Hence, using equation (2.21), we obtain the lower bound for the mutual information $I(Z, X)$:

$$I(Z, X) \geq \sup_{T \in \mathcal{F}} E_{p_{ZX}}[T] - log(E_{p_Z \otimes p_X}[e^T]) \tag{2.25}$$

The idea behind this neural estimator (MINE) is to parametrize the set $\mathcal{F}$ with a parameter θ and define a loss function $f : \Theta \to \mathbb{R}$ that is equal to the previously defined lower bound of the MI. The next step is then to look for the parameters that maximize that loss function in order to get inequality (2.25) as tight as possible. 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 is up-bounded by the MI.

This estimation has been proven to be more accurate than classical methods especially with large-sample traces as the sampling frequency of scopes is increasingly improved to cope with the working frequency of today's SoCs. In addition, the efficiency of MINE in side-channel leakage assessment context has been proved in [CLM20] by showing its ability to deal with high dimensional variables as they are often encountered in side-channel analysis campaigns. It can be fed with the whole trace, without any pre-processing for PoIs selection and is well suited for conservative leakage evaluators or countermeasure designers who are interested by the amount of information leaked by their design even if there is a protection mechanisms that would require a further processing if they were using classical leakage assessment methods. Due to its proven efficiency, MINE estimator will be used in Chapter 4 for leakage assessment of the proposed countermeasure for protection of memory hierarchy.

2.3.4 Exploiting leakage assessment results

Side-channel leakage assessment is now understood as a time saving process that helps avoid the need to run full attacks to assess the security of a design. From a designer's viewpoint, this process should lead to a threat quantification on how likely his design could be attacked through its side-channel leakage. The formal question that arises is how to link the results of the leakage assessment to the success probability

of an attack exploiting that leakage. A first attempt to answer this question has been proposed by Mangard [Man04] who provided a link between the success rate and the number of traces to succeed a CPA. The author first determines the sampling distribution associated to each correlation coefficient. He observed that the success probability increases as the distance between the distribution of samples leading to $\rho = 0$ and $\rho = \rho_{max}$ increases. That distance in turn, increases when the number of traces increases. He then derived the formula that links the number of traces to the efficiency of the CPA attack, which is recalled in the following equation, where d_α represents the distance between $\rho = 0$ and $\rho = \rho_{max}$ for a given the success probability α:

$$N_{traces} = 3 + 8 \left[\frac{d_\alpha}{log(\frac{1+\rho_{max}}{1-\rho_{max}})} \right]^2 \tag{2.26}$$

Unfortunately, the use of Pearson correlation coefficient which is a linear dependency distinguisher makes it relevant only for CPA attacks which are not the most generic attacks. Rivain proposed the estimation of other distinguisher in the Gaussian leakage model (focusing on template and correlation attacks) [Riv08]. Moving towards more generic approaches has been of great interest to the community, specifically, how to link mutual information (MI) to the efficiency of an optimal attack. Guilley et al. estimated the success rate using the MI as distinguisher with the assumption that the number of traces is infinite. An extension to masked implementations has been a further concern since it fits the context of security evaluations where the robustness of a protected device is analysed. The work of Prouff and Rivain [PR13] on the soundness of higher order masking schemes has been exploited Duc et al. [DFS19] to propose a lower bound on the number of traces using the MI between a leakage L and one share Z_i of the target sensitive variable in a d^{th}-order masking, for a given success rate SR:

$$\frac{cte \times SR}{MI(L, Z_i)^{d/2}} \leq N_{traces}(SR) \tag{2.27}$$

With the goal of extending this bound to an arbitrary leakage model and for any level of noise, Chérisey et al. [Ché+19a] proposed the lower bound on the number of traces needed to reach a given success rate recalled in Equation (2.28). A recent work [MDP20] recall the link provided in [Ché+19a] and argue that it can be used for both low and

high number of traces, which was not the case in the initial work.

$$\frac{f(SR)}{MI} \leq N_{traces} \tag{2.28}$$

In this inequality, f is a known, invertible, strictly increasing function defined in [Ché+19a] and that depends on the success rate SR. Chérisey et al. demonstrated the accuracy and tightness of their bound to the success rate of a real attack (using Maximum Likelihood distinguisher) compared to that of Duc et al.

2.4 Countering side-channel attacks

The efficiency of side-channel attacks is due to the existing relation between the power consumption and the data being processed in a hardware module. Countermeasures are hence designed to reduce, to break or to limit the influence of this link on the success rate of these attacks. According to these main objectives, we can distinguish two main categories of countermeasures that will be described in this section.

2.4.1 Hiding the processing of sensitive data

One approach is to undermine the success of attacks by hiding the manipulation of sensitive data without changing it. This prevents the attacker from being able to distinguish when the data is being processed in his observations, making the leakage exploitation more difficult. There are several options to achieve this goal. They can be applied at different levels in the hardware design flow.

Desynchronizing the power consumption. Randomization of the power consumption has been studied in many works. This is done either by changing the order in which basic operations are executed without modifying the function of the module Hence, the acquired side-channel traces are misaligned and desynchronized with the predictions of the attacker on the interesting time instants in the trace where the target variable is processed. Practical methods to achieve it is the random delay insertion [LOM10], dummy instructions insertion [CK10] or instructions shuffling [Vey+12]. Note that these solutions are mostly done at software level. Applying the same processes at hardware level is more challenging but there are some possibilities using non-deterministic

processors. Instead, people mostly rely on randomizing the clock jitter which leads to a random frequency [Sin+18; GM11; JIP19], or asynchronous logic styles [Moo+03].

Balancing the power consumption. The rise of side-channel attacks is due to the ability to distinguish bit values transitions in the side channel information. In standard CMOS gates that use static logic, power is consumed only for '0' to '1' transition. For the inverse transition, the previously charged capacitance is discharged. No power is consumed for '0' to '0' or '1' to '1' transitions. In this context, it is possible to hide the computation of the sensitive variable by balancing the power consumption during data transitions and making it independent. Many resistant logic styles have been proposed to reach that goal. This includes the use of differential and dynamic logic styles, masked logic gates or the combination of both. Tiri et al. proposed the Sense Amplifier Based Logic (SABL) which is a family of full custom logic gates that incorporate both dynamic and differential logic [TAV02]. The main idea was to include dynamic logic to consume power independently of the input switching. In fact, the load capacitance of the gate is charged no matter the value of the input during the setup phase where the clock is low. During the evaluation phase, the output phase, the output can stay high or pulled low. In both cases, the load capacitance is charged at each clock cycle, making the power consumption independent of the switching activity ('0' to '0' and '0' to '1' transitions are not distinguishable as well as '1' to '0' and '1' to '1'). Including differential logic helps to avoid differentiation between '0' to '1' and '1' to '0' transitions. An improvement of the SABL have been proposed by the authors where Wave Dynamic Differential Logic (WDDL) was proposed to create gates with similar properties to SABL while making use of standard cells or field programmable gate array (FPGA) slices [TV04]. Unfortunately, these solution come with area overhead since there is a doubling of internals of each gate in order to obtain the true and complementary network. Moreover, getting a perfectly balanced power consumption is still difficult because of the need to have the same total capacitance on the two networks. Many improvements have been made to address this problem. Three-phase Dual-Rail Pre-charge Logic (TDPL) [Buc+06] and Masked Dual-Rail Pre-charge Logic (MDPL) [PM05] (which is resistant to unbalanced routing) have been proposed. The MDPL has been evaluated in [ST07] and the authors were able to exhibit the value of the mask, leading to a successful DPA on an AES implementation with 2,000 measurements. In order to get a more robust logic style Razafindraibe et al. proposed the Secure Triple

Track Logic (STTL) [RRM07] that includes a third rail that act as a valid signal that indicates whether the current output value is valid. This have the advantage of making the propagation time delay independent of the arriving signals, hence fixing the unbalanced routing problem but comes with a large overhead: compared to single-ended logics, the number of slices on FPGA is multiplied by 5.5. Delay-based Dual-rail Precharge Logic (DDPL) [Buc+10] has been proposed to overcome the large overhead issue in STTL by using time domain signaling (TEL [BST18]). A further implementation of DDPL using standard cells (SC-DDPL) has also been proposed by the authors [Bel+20].

Constant weight coding. This solution is meant to be implemented at software level or logic level (RTL) in hardware design process. The idea is to prevent the exploitation of power consumption leakage by using constant Hamming weight codes. These are specific coding schemes in which all codewords have the same HW. To achieve this, one can think of applying dual-rail techniques that exist in hardware to software implementations. This has been investigated in the work of Hoogvorst et al. [HDD11]. The author proposed to encode one bit *s.t.* the logical value 0 is represented as 01 and the logical value 1 is represented as 10 or the inverse. An extension of this idea has been proposed in [Ser+14]. Instead of encoding the data bit per bit, they encode the whole sensitive data with a fixed HW value. They applied it to implement an AES encryption scheme and proved that their proposal is leak-free in the HW leakage model. The main weak point of such countermeasures is the prior assumption of HW leakage model. Considering only HW as potential leakage model is not realistic as there are several leakage models such as Hamming distance that can be exploited by an attacker. To address this issue, Maghrebi et al. proposed a framework [MSB16] that builds a customized encoding function according to the leakage characteristics of a circuit, based on the model of linearly combined leakage of each bit of the data which is more generic than HW model. A recent study proposed a dynamic encoding of data taking into account both HW and HD leakages and has been implemented on shift registers [Mon+20]. In addition to protecting against side-channel attacks, the authors claim an inherent security against fault injection attacks due to the parity check and fault detection performed by the decoding function at the output. However, this solution incurs a large overhead as the size of the codewords is 4 times the size of information words.

2.4.2 Masking sensitive data

This family of countermeasures aims at removing the link between the side-channel information contained in the measurements and the processed data by adding a random value, the so-called *mask*, to the sensitive data. Hence, all operations manipulating the sensitive data are modified in order to process the masked one. In this context, executing a side-channel attack is no longer straightforward because the trace does not contain the direct leakage of the secret data as it is the case for unprotected modules. It can be more efficient than solutions such as secure logic styles in terms of overheads, making it the most studied countermeasure, but it is more case specific and hardly applicable to other modules. The most applied masking methods are the one proposed by Messerges [Mes01].

- Arithmetic masking: $z_m = (z - m)\ mod\ 2^n$
- Boolean masking : $z_m = z \oplus m$

The idea of masking is similar to secret sharing concept that consists in sharing a secret between a group of actors in order to keep it safe. Therefore, the mask values are considered as shares of the secret variable. Masking a sensitive variable z can be seen as randomly sharing it into multiple independent shares like shown in equation (2.29), where $z_1, ..., z_d$ represent the d shares of the secret.

$$z = z_1 \oplus z_2 \oplus ... \oplus z_d \tag{2.29}$$

Since the first idea of applying secret sharing to mitigate side-channel attacks [Cha+99], many improvements have been made on masking schemes to protect various implementations [ISW03; Mes01; JS17; GMK18; GM18]. Once the sensitive data has been split into shares, the next step is to refine the instructions previously implemented to handle the secret in order to process all the shares. For linear operations no modification of the original implementation is required and the masking is straightforward. Non linear operations (e.g. AND) require a special treatment since we can not apply the same functions as the original ones. The soundness of masking has been demonstrated by showing that the complexity of recovering some data by means of side-channel analysis grow exponentially with the number of shares with the assumption of presence of noise in the leakage [PR13]. As a consequence, it is common practice to consider the number of shares as a security indicator. The expression d^{th}-order masking is often

used to illustrate the fact that d masks are used to obtain the relationship described in the equation (2.29). In the rest of this thesis we will focus on boolean masking since a method has been proposed to switch from one type of masking to another in [Mes01]. The d-probing model introduced by Ishai et al. has been widely adopted to analyse the security of a masked implementation. In this model [ISW03], an attacker having the ability to record up to d instantaneous leakages of intermediate computations (e.g. an attacker using d EM probes) corresponding to d shares of the secret variable, cannot retrieve the sensitive information. The circuit is then said to be d^{th}-order secure. It was demonstrated in [RP10] that there is in fact a relationship between the number of probed wires and the attack order for a DPA attack. In addition to the independence requirement between the shares, there is also a need to have intermediate variables that are independent from input shares. This leads to an increasing need for fresh randomness in all masking schemes in order to ensure secure processing of the shares to avoid leaking information during the computation that could be harnessed to retrieve the secret without needing to retrieve all the shares [NRR06].

Classical boolean masking. In order to address the masking of non linear operation issue, many proposals have been made. We will describe the proposal of Trichina [Tri03] to compute a masked binary AND operation for a first order masking. Let consider two inputs x and y that are intended to be multiplied in a first order masking scheme. Both inputs can be defined as sums of their respective shares:

$$x = x_0 \oplus x_1 \ , \ y = y_0 \oplus y_1 \tag{2.30}$$

We have:

$$z = xy = (x_0 \oplus x_1)(y_0 \oplus y_1) = x_0y_0 \oplus x_0y_1 \oplus x_0y_1 \oplus x_1y_1 \tag{2.31}$$

In this equation, the sum of the output terms of each multiplier is not independent of the inputs even if it is the case between each term and the inputs. The addition of a new random value becomes necessary. This results in the diagram shown in Figure 2.1, proposed in the same work, where r is the new random value. The effectiveness of this masking scheme has been challenged in real-world situations where there are different propagation times for each signal incurred by glitches, allowing an adversary to successfully combine the outputs to reach sensitive data [MS06; MPG05]. There exist several proposals to overcome the issue introduced by glitches on logic gates due

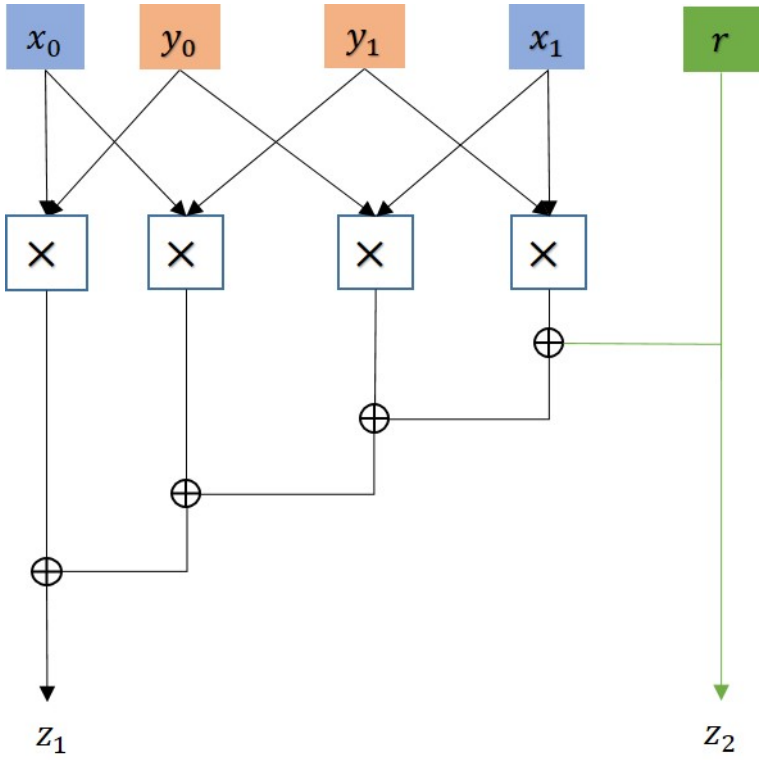


Figure 2.1: Masked AND operation proposed in [Tri03].

to terms addition by reordering the signals.

Threshold Implementation (TI). It has been proposed as solution to prevent glitches when masking logic gates [NRR06]. As mentioned earlier, non-linear functions are the most tricky ones to mask. In order to reach the same security level as the one of linear transformations for non-linear transformations, the authors rely on the so-called "composing functions". The idea is to define for each non-linear function, a set of functions satisfying two properties: non-completeness and correctness. Let $z = f(x, y, ...)$ be a non linear transformation. Let $\mathcal{F} = \{f_1, f_2, ..., f_n\}$ be a set of functions.

- Non-completeness property: Each function f_i is independent of at least one share of each of the input variables. For example, take f_i independent of at least $x_i, y_i, ...$

- Correctness property: The sum of the output shares gives the desired output.

$$z = f_1(...) \oplus f_2(...) \oplus ... \oplus f_n(...) = f(x, y, ...) \tag{2.32}$$

The non-completeness property is the one that guarantee the independence of the intermediate variables with regards to the inputs, which leads to mitigating glitch issues observed in classical masking schemes. The correctness property ensures the non-alteration of the initial function. The minimum number n of shares required to compute a product of s variables has been proved to be: $n \geq s + 1$. Hence, to compute the following product $z = xy$, 3 shares are needed as well as 3 composing functions. This can be done as follows:

$$\begin{aligned} z_1 &= f_1(x_2, x_3, y_2, y_3) = (x_2y_2) \oplus (x_2y_3) \oplus (x_3y_2) \\ z_2 &= f_2(x_1, x_3, y_1, y_3) = (x_1y_3) \oplus (x_3y_1) \oplus (x_3y_3) \\ z_3 &= f_1(x_1, x_2, y_1, y_2) = (x_1y_1) \oplus (x_1y_2) \oplus (x_2y_1) \end{aligned} \tag{2.33}$$

Many first-order protections have been developed from this scheme [Mor+11; Bil+14a]. An extension of TI implementation to higher order masking has also been introduced in [Bil+14b]. The authors gave a generalisation of the minimum number of shares equation for the computation of a product, which is recalled in equation (2.34) where t represents the degree of the non-linear function and d is the desired protection order.

$$n \geq t \times d + 1 \tag{2.34}$$

Although the security provided by threshold implementation in the probing model and in presence of glitches, this solution increases the area overhead. While a traditional masking of an AND gate requires 4 AND gates, the TI variant requires 12 AND gates.

Domain Oriented Masking (DOM). After the adoption of Threshold Implementations as a sound proposal for reaching secure masking of non-linear transformations, the focus was on how to lower the cost for protecting digital chips against side-channel attacks with masking. Groß et al. presented the Domain Oriented Masking [GMK16], a new masking scheme that provides the same security level as TIs while reducing the required randomness and area. The authors claim that their solution only requires $d + 1$ shares to reach d^{th}-order security. In DOM, the shares of variables form different domains that can be totally independent (for linear operations) or not (non-linear operations) and hence would require domains borders crossing. The goal of DOM is then to secure domain crossings by adding fresh randomness. To achieve this, from a classical shared multiplication of two inputs x and y, the shares are considered to be

part of different share domains. The DOM multiplication is realized using the random share z_0 like shown in equation (2.35).

$$\begin{aligned} x &= A_x \oplus B_x \\ y &= A_y \oplus B_y \\ q = xy &= [A_x A_y \oplus (A_x B_y \oplus z_0)] \oplus [(B_x A_y \oplus z_0) \oplus B_x B_y] \end{aligned} \tag{2.35}$$

The shares A_x and A_y (respectively B_x and B_y) form the domain A (respectively B). The terms $A_x B_y$ and $B_x A_y$ cause domain crossing and can lead to recovering of the inputs, hence the need for a fresh random share z_0 to secure the domain crossing. This is illustrated in Figure 2.2. Registers are added in each domain to prevent glitches that may lead to first-order weakness of the masking scheme.

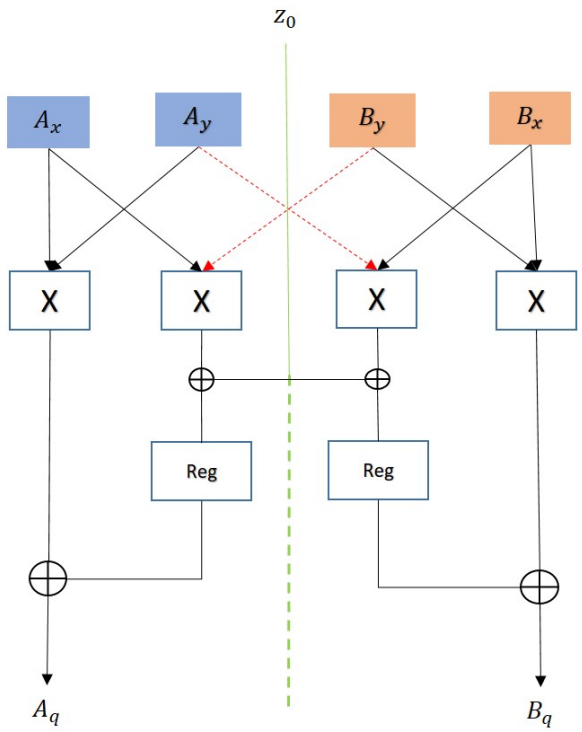


Figure 2.2: Masked AND gate with DOM.

A generalisation of the scheme to higher orders has also been provided by the authors, following the same principle of securing domain crossings.

In addition to the masking schemes described above, several other ones have been proposed in the literature such as Unified Masking Approach (UMA) [GM18], LUT-

Masked Dual-rail with Pre-charge Logic (LMDPL)[LMW14], Generic Low-Latency Masking (GLM) [GIB18].

2.5 Side-channel leakage of microarchitecture

The microarchitecture can be defined as a set of conceptual elements of the physical implementation of a circuit [ABB64]. Memory elements (main memory, cache memories, Flash memory...), registers, interconnect bus and ALU are considered as microarchitectural components of an integrated circuit.

At the beginning of this Chapter, we have seen that encryption keys and any internal data manipulated by cryptographic modules, while being the initial motivating factor for side-channel attacks, are not the only targets of these attacks. Since side-channel leakage is mainly caused by leakage in the microarchitecture, any information manipulated therein is potentially valuable to an attacker. In this section, we review some side-channel attacks on microarchitectural elements that are found on a System on Chip to show that besides encryption keys, many other information can be critical in a SoC.

2.5.1 Leakage assessment and attacks on microarchitecture

Returning to the origin of the side channel leaks, it becomes noticeable that key recovery attacks either on software or hardware implementation of cryptographic algorithms, actually exploit the leakage of micro-architectural components. Attacks on software implementations usually leverage leakage due to the mobilisation of memory hierarchy (interconnect bus, registers, caches) and pipeline stages of the CPU when the keys and intermediate variables are loaded or manipulated. As example, we can think of attacks on software implementations of AES [OD15; SBB18]. Another example is the successful CPA attack with HD leakage model on the Subbytes operation of an AES-128 running on a board embedding a Ubuntu 16.04 which has been presented as the result of a study of the micro-architectural leakage of a superscalar CPU (Cortex A7) carried by Barenghi and Pelosi [BP18].

Beyond the specific case of exploitation of micro-architectural leakage of cryptographic modules, that leakage enables many other kind of exploitations. The leakage of CPU registers have been exploited to extract 16 bits at a time on an 8-bit microcon-

troller executing two consecutive load operations [CK14a] using stochastic model. A first-order side-channel vulnerability has been reported on SRAM cells and embedded memories while recording their power consumption [GKF18]. Side-channel based disassembly of CPU instructions have also been exhibited in several works [Par+19; CLH19; Str+15].

These various exploitations have been made possible after leakage assessment of micro-architectural components. McCann et al. addressed this topic by suggesting a tool for modelling power leakage at the instruction level on an ARM Cortex-M0 and Cortex-M4 and that is able to point out leakage between couples of adjacent instructions [MOW17]. They further introduced a leakage simulator for ARM Cortex-M0 (ELMO). This has the benefit of allowing to model the expected leakage in the micro-architecture before having the physical device. The proposed leakage simulator has the weakness of only suiting to ARM Cortex M0 device. To go beyond this limitation to a specific family of devices, a proposal has been made with the purpose of comparing the impact of micro-architecture on side-channel leakage of all devices sharing the same Instruction Set Architecture (ISA) [Aro+21]. In the same context, Marshall et al. carried out a broad leakage assessment on 14 devices from 4 different ISAs and aiming to capture any leakage effect from sources within the micro-architectural implementation [MPW21]. A framework (PLAN) has been proposed to identify leaking modules in a microprocessor without needing a physical chip, as it takes a pre-synthesized netlist as input [Ars+20]. 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 [Gal+16]). HD leakage has also been obtained on registers of a RISC-V CPU [SR16]. The authors also reported unwanted correlation between different registers due to multiplexers used at the input of the register bank, stating that it could lead to breaking masked implementations if a correlation can be found between the data in the register containing the mask and the one containing the masked value. HW correlation has also been pointed out during data transfer on the interconnect bus of a SoC [Yao+20]. In most analysis, the assumed leakage models are the Hamming weight (HW) and Hamming distance (HD) as these are the most intuitive ones. An effort has been made on varying the underlying leakage model assumption instead of sticking to HW and HD. It can be seen in [MGH19] where the authors categorize data leakage of a microprocessor into direct-value leaks, data-overwrite and circuit-level leaks. There is also the work of You and Kuhn where the inputs of a SHA-3 implemented on an 8-bit microcontroller

have been reconstructed using template attacks on a single power trace to retrieve the value of each byte of the data (600 bytes), hence exploiting direct-value leakage [YK20].

2.5.2 Existing mitigations

The countermeasures to side-channel attacks described in Section 2.4 have been designed to specifically protect cryptographic algorithms. Some of these countermeasures (e.g. dual-rail, masking and noise addition) can also be applied to reduce microarchitectural leakage of all the components of the SoC (ALU, registers, caches, interconnect...). However, applying these solutions to protect the entire circuit will result in considerable additional costs, as it is no longer a matter of protecting a cryptographic module but the entire circuit. This will decrease the overall performances of the circuit (latency and area overhead, increased memory size), which is not desired. Hence, there is a need for countermeasures that take into account the whole micro-architecture, and not only cryptographic modules, with low impact on the system's performances. In the following lines, some solutions that have been suggested in the literature to address this problem are recalled.

A Power Attack Resistant Microprocessor (PARAM [Ars+20]) has been proposed to protect all leaking module in the processor by encrypting data and their addresses using a 4-round Feistel structure. In addition to the protection of data, another approach [SEH20] suggests the encryption of instructions by mean of authenticated encryption using a lightweight encryption primitive (ASCON). The authors were also able to build a control flow integrity countermeasure and implemented it on a RISC-V core (RISCY [Gau+17]). Some other works propose to reduce these leaks by directly modifying the software code before implementing it on a chip. An example of this is ROSITA, an automatic engine that reduces microarchitectural leakage by rewriting the software code [She+19]. The authors put forward the relevance of their proposal in making a masked implementation more resilient to such leakage. Guided by a leakage characterization of vulnerable instruction sequences, Seuschek et al. proposed the use of leakage aware code generation by re-scheduling (re-ordering) instructions and register allocation to prevent leakage [SDG17]. In the same line, FENL has been introduced as an instruction set extension to avoid leakage caused by instruction sequences [Gao+20]. All these solutions can be grouped in the two main countermeasure strategies to prevent side-channel attacks: hiding and masking.

2.6 Conclusion

Throughout this chapter, some side-channel attacks and their principles are reviewed, as well as various approaches to prevent them. Starting with attacks on encryption algorithms in an effort to retrieve secret keys or intermediate computational variables, we have seen that in the end, encryption keys are not the only information to be protected on a SoC. Several components of the microarchitecture process data that at first sight are not sensitive, but which can be crucial for an attacker (executed instructions and their sequences, CPU control data, user data). This wealth of information can be exploited by an attacker through side-channel attacks too. Some attacks along these lines are recalled, as well as solutions that have been developed to counter these attacks which take advantage of the leakage of components such as registers, ALU, cache memories, interconnect bus, etc. The memory hierarchy of SoCs being a privileged target for attackers, particular emphasis is placed on it in order to have an insight into the various attacks it suffers from and suggested solutions. The classical side-channel countermeasures on cryptographic modules would incur large overhead and performance losses if applied to protect the entire micro-architecture. This yields the need of solutions that can be applied to all the micro-architecture with a low impact on the performances. As a first step to achieve this, efforts have been made to study first order side-channel leakage of the memory hierarchy and in particular those of the registers and the interconnect bus in order to propose solutions to protect data in the whole memory hierarchy. This study of registers and interconnection bus leakage is the subject of the next chapter, which presents the contributions of the first part of the thesis.

Résumé du chapitre

Ce chapitre dresse un état de l'art des différentes attaques par canaux auxiliaires en se focalisant sur les attaques visant toute la microarchitecture dans le but de retrouver différentes sortes d'informations en plus des clés de chiffrement. Certaines attaques par canaux auxiliaires et leurs principes sont passées en revue, ainsi que diverses approches pour les prévenir. En commençant par les attaques sur les algorithmes de chiffrement dans le but de récupérer des clés secrètes ou des variables de calcul intermédiaires, nous avons vu qu'au final, les clés de chiffrement ne sont pas les seules informations à protéger sur un SoC. Plusieurs composants de la microarchitecture traitent des données a priori non sensibles, mais qui peuvent être cruciales pour un attaquant (instructions exécutées et leurs séquences, données de contrôle du CPU, données utilisateur). Cette richesse d'informations peut également être exploitée par un attaquant par le biais d'attaques par canaux auxiliaires. Quelques attaques de ce type sont rappelées, ainsi que les solutions qui ont été développées pour contrer ces attaques qui profitent de la fuite de composants tels que les registres, l'ALU, les mémoires cache, le bus d'interconnexion, etc. La hiérarchie mémoire des SoCs étant une cible privilégiée des attaquants, un accent particulier est mis sur elle afin d'avoir un aperçu des différentes attaques dont elle est victime et des solutions proposées. Les contre-mesures classiques de canaux auxiliaires sur les modules cryptographiques entraîneraient une surcharge importante et des pertes de performance si elles étaient appliquées pour protéger l'ensemble de la micro-architecture. Il est donc nécessaire de trouver des solutions qui peuvent être appliquées à l'ensemble de la micro-architecture avec un faible impact sur les performances. Dans un premier temps, des efforts ont été faits pour étudier les fuites de canal latéral de premier ordre de la hiérarchie mémoire et en particulier celles des registres et du bus d'interconnexion afin de proposer des solutions pour protéger les données dans toute la hiérarchie mémoire. Cette étude des fuites des registres et du bus d'interconnexion fait l'objet du chapitre suivant, qui présente les apports de la première partie de la thèse.

Chapter 3

Leakage characterization on interconnect bus and registers: threat quantification

In this chapter, investigations are made on the leakage characteristics of interconnect bus and general purpose registers of a 32-bit RISC-V CPU. Template attacks are also performed in order to assess the relevance of the attacker model defined in the previous chapter, with a focus on the retrieval of data value. The results presented in this chapter have been published in the proceedings of PAINE 2021 [*Tal+21*].

Contents

3.1 Introduction

Leakage assessment consists in using some evaluation tests to find and/or confirm the presence of leakage on the targeted part of the device, define the leakage model and its characteristics and also how harmful it could be to exploit that leakage. Several tests and methods have been developed for side-channel leakage assessment as recalled in chapter 2. In the design process of System on Chips (SoCs), we cannot do without interconnect buses and CPU registers. Interconnect bus is the communication channel between all the components of the device and most data exchange is done through it. In a SoC, the data needed by the CPU is transmitted from the main memory through the interconnect bus. This explains its key role in the whole system and its impact on the overall efficiency of the chip. Many types of on-chip interconnect buses have been developed with the ultimate goal to enhance the communication between on-chip components. The most popular ones are AMBA, Avalon, CoreConnect, STBus, Wishbone. CPU general purpose registers are the nearest and most quickly accessible memory of the chip. They are used to store the results of ALU operations, the data requested by the CPU to the main memory, intermediate variable of large computations... They are organised in register banks and their size is set by the Instruction Set Architecture (ISA). According to the running program, they can carry some sensitive data. The potential sensitivity of the handled data in the two components motivates the need to protect them from information theft, especially through side-channel attacks. In the first part of this chapter, the focus is on assessing the side-channel leakage of a RISC-V based System on Chip. The main goal is to describe the side-channel leakage of the studied device, find its characteristics and show how it can be exploited for a practical attack. This will lead to a better quantification of the threats and the definition of suitable countermeasures against the attacker model that has been defined at the end of Chapter 1. The ultimately implemented countermeasure should then be applied to any other device having the same leakage behaviour.

3.2 Setup and process for data acquisition

To identify and characterise register and bus leakages, random data is generated and transferred over the bus and stored in the registers. The intent is to collect the power consumption of the chip during these different operations and then analyse it. The

power consumption of the target device has been characterized due to the facilitated access to the chip's power consumption through a dedicated voltage probe connected to a resistor. This is supported by numerous published attacks exploiting power leakage [MSS07; RD20]. The leakage is recorded with a digital scope that has a bandwidth of 2 GHz and a sample rate set to 10 GS/s, allowing to capture the leakage on many samples within a single clock cycle. Figure 3.1 depicts the experimental environment.

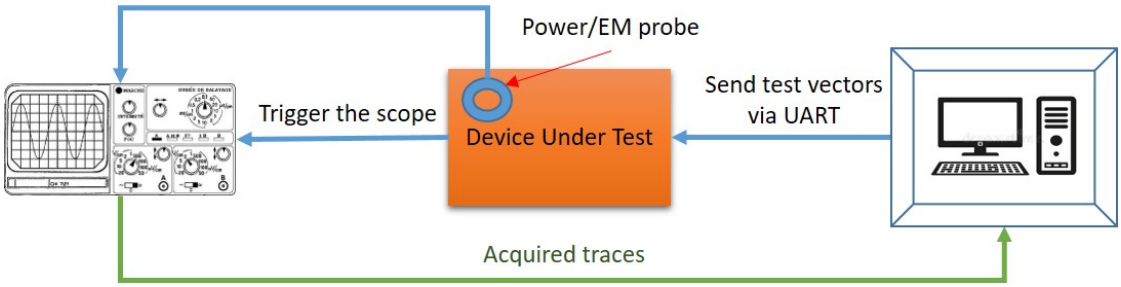


Figure 3.1: Side channel analysis bench.

A Python program running on a computer controls the overall acquisition process. It generates the test vectors and sends them to the Device Under Test (DUT) via UART. The power leakage that occurs during the processing of the data is captured by the scope which is triggered by setting a GPIO high. The traces are then sent back to the computer and stored for further processing.

3.2.1 Studied architecture

The experimental work is done on an ASIC SoC built around a 32-bit RISC-V CPU which is a 4-stage in-order processor (RI5CY [TGS]) that has a direct connection to separate instruction and data memories and works at the frequency of 100 MHz. Some specific Intellectual Properties blocks (IPs) like AES accelerators, Trivium accelerator, TestRAM have been added for security assessment purpose. They are connected through a hierarchical interconnect based on AHB and APB bus [Ltd] as shown in Figure 3.2. The circuit has been designed and manufactured using 28 nm bulk technology and was already available at the beginning of the Ph.D. work.

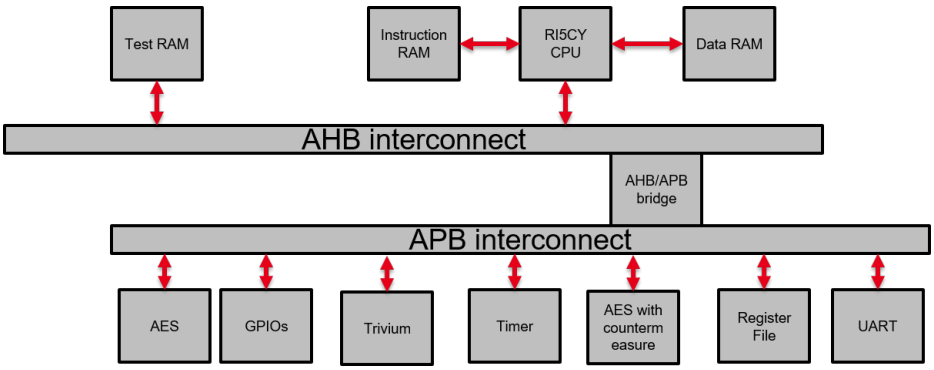


Figure 3.2: Target architecture.

3.2.2 Leakage measurement

3.2.2.1 Interconnect bus

To study the leakage behaviour of the interconnect bus, the power consumption of the chip is recorded while random data is transferred over the bus. The following process is therefore employed:

- Generate pseudo-random numbers (4 bytes) on the computer and send them to the chip via UART
- On the chip: store the received number in a general purpose register and send it to the "testRAM".
- Collect the power consumption of the chip during the transfer.

From figure 3.2 we can see that the "testRAM" is memory mapped on the AHB bus. Hence the observed power consumption during the store operation could come from either AHB bus or the testRAM. In order to make sure that we collect only the leakage of the interconnect bus (AHB), we also send pseudo-random data to other memory mapped components like the timer or the register file which is memory mapped on APB bus, and compare the result of the assessment tests.

3.2.2.2 Registers

With the purpose of studying the leakage characteristics of registers, power leakage of general purpose registers of the CPU is collected while storing random data in a given register. The used process is hereby described:

- Generate pseudo-random numbers (4 bytes) on the computer and send them to the chip via UART.
- On the chip: store the received number in a general purpose register and move it to the target register after pre-loading a constant value in it (we chose "$0xAAAAAAAA$") as constant value.
- Record the leakage during the move operation.

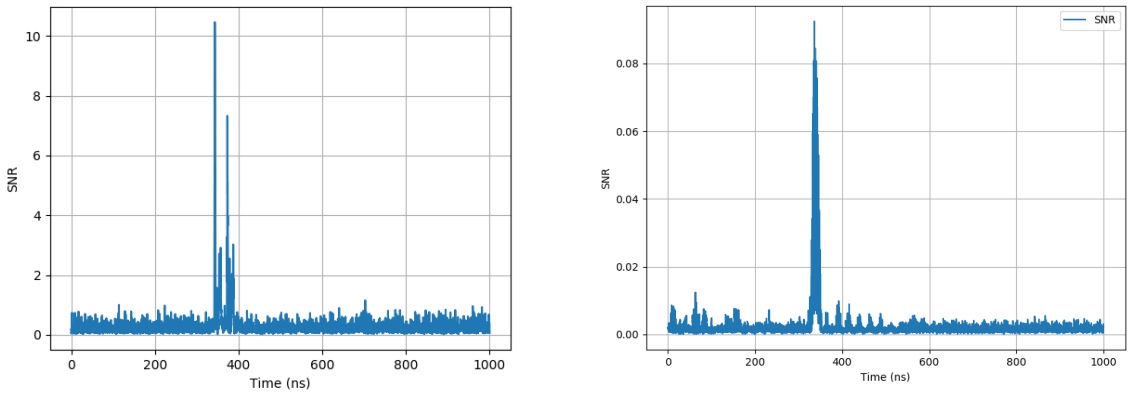
The processes described for interconnect bus and registers are repeated with different random numbers.

3.3 Leakage assessment results

3.3.1 SNR test

Equation 2.18 is used to evaluate the level of the signal part in the collected traces. For this test 2,000 power traces have been collected.

1. **Interconnect bus** The SNR observed on the interconnect is shown in Figure 3.3a. The value of the SNR is plotted at each time instant. We can notice that there seems to be "no activity" between 0 and 300 ns and between 400 and 1000 ns. From 300 ns we get higher values and reach almost 11, meaning that the desired signal (in our case, the data component in the leakage) is 11 times "more powerful" than the noise component. In a side-channel acquisition on the interconnect bus, an attacker can expect a low noise level and therefore be able to successfully perform his attack with a reduced number of traces if we refer to the SNR value plotted in Figure 3.3a. The time instants where the SNR is significantly low correspond to the *"nop"* instructions that have been added in the code running on the chip in order to ease the leakage localization.
2. **Registers** The peak of SNR (0.08) is observed between 300 ns and 400 ns (Figure 3.3b). Even though the peak value is not as high as seen for the interconnect bus, we can notice in the Figure that the instant where we have the highest SNR is distinguishable from the other time instants. These results can be explained in the same way as those on the interconnect bus. The *"nop"* operations led to lower SNR values. The useful operations led to higher values.



(a) Interconnect bus.

(b) CPU registers.

Figure 3.3: SNR observed.

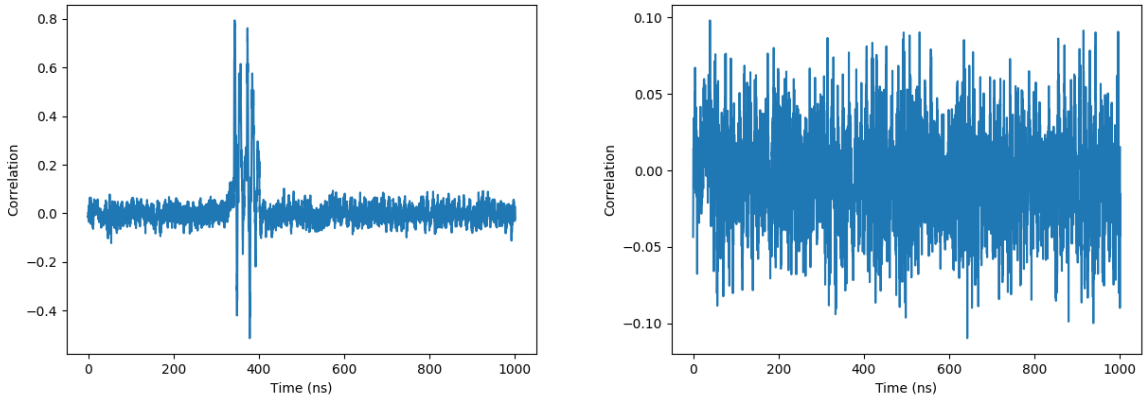
As a conclusion for this initial assessment, we saw that the strength of the useful signal is reduced in registers compared to interconnect bus. Yet, both of these modules allow to undoubtedly distinguish time instants where they handle the data, which will be valuable for any attacker.

3.3.2 Correlation test

The correlation test described in Chapter 2 has been used. From the set of pseudo-random data that was used for the acquisition, we built two leakage models based on Hamming weight (HW) and Hamming distance (HD). After that we computed the correlation between the two leakage models and the actual leakage that was recorded from the chip.

- **Interconnect bus** The results are depicted in Figure 3.4a and 3.4b. For the HW model, the correlation coefficient reaches 0.8 between 300 ns and 400 ns. We can see it goes close to 0.7 for almost 100 ns. When considering the HD model, the correlation varies between -0.1 and 0.1 all the time. We can see the time instants where the *"nop"* instructions are being executed, leading to low correlation values. The closer the value of the correlation is to ± 1 the better the leakage model is. The result with the HD model in Figure 3.4b shows a correlation between -0.1 and +0.1 meaning that this model does not fit with the collected leakage. We retain that the HW model is better fitted. This is consistent with the leakage behavior of

Flip-Flops (FFs) described in [MOP08] and confirmed in [MPW21] (See Chapter 2, Sect. 2.5.1). HD leakage could be expected since the interconnect bus is made of FFs. However, our view is that the bus lines are set to 0 after each data transfer, which explains why the HD model does not fit.



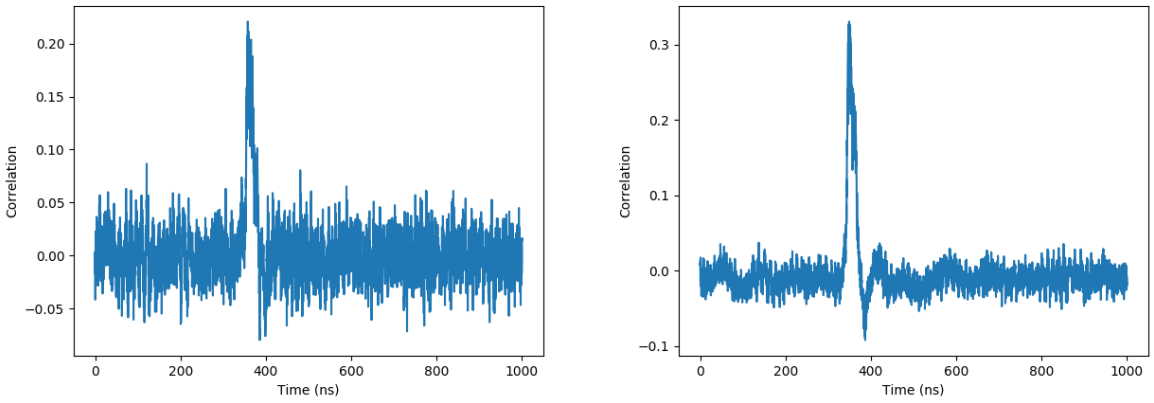
(a) HW correlation.

(b) HD correlation.

Figure 3.4: Correlation test on interconnect bus.

- **Registers** After running the correlation test on the acquisition from registers we got the results plotted in Figure 3.5a and 3.5b. Here we notice that right before 400 ns there is a peak of correlation for both HW and HD. The two peak values are located between 300 and 400 ns. We recall that 30 *"nop"* instructions (one cycle per *"nop"* instruction) are inserted after triggering the scope before sending the generated data from one register to another. The CPU works at 100 MHz thus a clock cycle lasts 10 ns. This explains why there are very low correlation values outside the time instant range of 300-400 ns. We get a peak with a level of 0.2 for HW and 0.3 for HD. In the two cases the correlation is not as strong as for interconnect bus, but it is easy to distinguish the time instants giving the peak values from other instants. We can hence establish that a register leaks the HW of a managed data and the HD between two data successively stored in it. The register's content doesn't change until an instruction is executed to update the content. It is anticipated that HD leakage will happen between the current data and the next one. The HW leakage observation could be explained by the transit of data on the internal bus of the CPU. These results are also consistent

with what has been explained in [Mor+08] about leakage of CMOS Flip-Flops. The FFs are made of two consecutive latches, leading to two operation phases upon each update of their content: sampling phase and holding phase. The first latch samples the input in the sampling phase and then updates the content of the second one during the holding phase. As reported by the authors, the HD leakage can be linked to the update of the second latch during the holding phase. Since their is an update of the content of the first latch, the power consumption is also affected. The authors in [Mor+08] link it to a toggle count model according to the changes of the input signal. We consider it to be the source of the HW leakage in our case if we assume that the internal bus lines of the CPU are set to 0 after each transfer.



(a) HW correlation.

(b) HD correlation.

Figure 3.5: Correlation test on registers

The correlation levels for interconnect bus and registers are also consistent with the results of the SNR test and underline the relationship between the SNR and the correlation.

3.3.3 Summary of the results & discussion

The leakage characterization that has been carried out led to the identification of HW leakage on the interconnect bus which is explained by a reset of the bus lines after each data tansfer. Regarding the registers, both HW and HD leakages have been found and also explained by the leakage behaviour of Flip-FLops. As they are composed of two

latches, the HW leakage is credited to the update of the first latch (sampling phase) and the HD leakage is linked to the update of the second latch (holding phase) [Mor+08].

Mangard et al. demonstrated the presence of these leakages when transferring different data from the internal memory to a register of an 8-bit CPU [MOP08]. It has also been evidenced by Barthe et al. [Bar+21]. We could have also tried to predict the results of our characterization by noticing that many papers that are applying side-channel attacks on software implementations of a cryptographic algorithm are utilizing the leakage of the interconnect bus and the CPU registers. Due to the underlying microarchitectural elements contained in interconnect bus and registers and the results in the previous works, one may question the relevance of this study. Despite the existing previous work, it has been found necessary to carry out a characterization work to confirm these leakage models (HW and HD) on a RISC-V based ASIC SoC, before exploring their exploitability and quantifying the threats. The rest of this chapter will hence focus on exploiting these leakages through template attacks and quantifying the capabilities of the attacker model defined in Chapter 1r.

3.4 Leakage exploitation through template attack

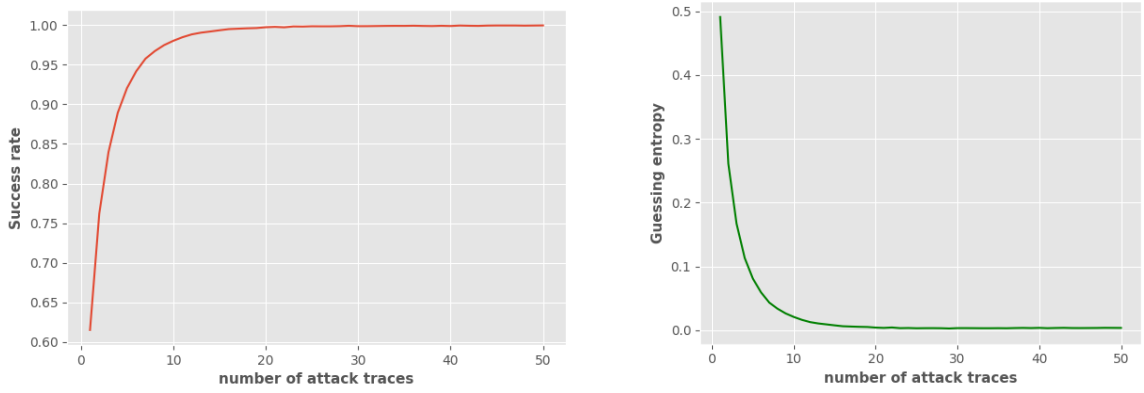
The road between characterization and attack is not so straightforward. Many leakage analysis suffer from not providing real world attacks that lead to retrieval of the leaked information that has been assessed in the analysis [SR15]. The attack completes the analysis and enhances the confidence in the proposed study. This is why some works focused on how to infer the robustness of an implementation after a leakage assessment without having to run an attack, by estimating, for example, the expected number of traces needed to succeed an attack [Ché+19a; MDP20]. Following the previous section where we pointed out the presence of side channel leakage on the target components, some effort has been invested in how to exploit it to gain information on manipulated data in interconnect bus and registers. A relevant method to achieve this is the template attack given that it is among the most powerful side channel attacks and also the most popular profiled attack.

3.4.1 Template attack on interconnect bus

The HW leakage observed on the interconnect bus enabled to mount templates based on the HW of the data. To achieve this, 20K power traces have been recorded. Each trace contains 10K samples. For each value of HW ranging from 0 to 32 (32-bit data), we send 20K pseudo-random data corresponding to the given HW. We divided the traces into two sets, one for template building and the other for attacking, meaning that we used the same device for profiling and attacking.

In order to deal with dimensionality issues, a correlation analysis is done in order to keep samples corresponding to correlation values such that $|\rho| \geq 0.2$ and then we apply Linear Discriminant Analysis (LDA) [BG98]. At the end we retained 10 PoIs. For the attack phase we performed 2,000 attempts of classification using Quadratic Discriminant Analysis (QDA) [SGF07], which is a form of implementation of template attacks. The efficiency of the attack is evaluated through its success rate and the guessing entropy. The success rate is computed as the average rate of correct classification or the average number of times the correct data is sorted first on the 2,000 trials. On the other hand, the guessing entropy can be understood as the remaining average number of data to test before finding the correct one [Wu+20].

Figure 3.6a and 3.6b show the success rate and the guessing entropy computed after the attack.



(a) Success rate.

(b) Guessing entropy.

Figure 3.6: Template attack on HW of data on the interconnect bus.

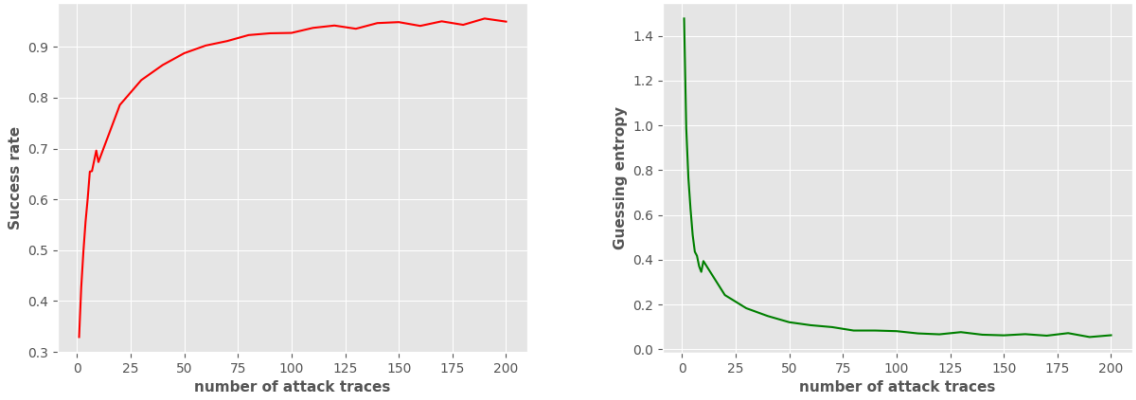
We can notice a success rate of 62% with a single attack trace. It increases rapidly

between 0 and 10 attack traces, reaches 99% for 15 traces and stabilizes there till the end. This behaviour is consistent with what is observed in guessing entropy plot where we have a very low value of guessing entropy for a single trace (0.5). It drops to zero around 15 traces. These results show that only 15 attack traces are needed by an attacker to successfully infer the HW of any data on the interconnect bus.

3.4.2 Template attack on registers

Unlike the interconnect bus, HW and HD leakage were identified on CPU registers. We thus carried out template attacks to recover these two types of information on the content of registers. The experimental protocol was similar to that of the interconnect bus with the same amount of traces for profiling and attacking.

Retrieving the Hamming weight. Figure 3.7a and 3.7b present the success rate and the guessing entropy of the attack that targets the HW of the data. A success rate of



(a) Success rate.

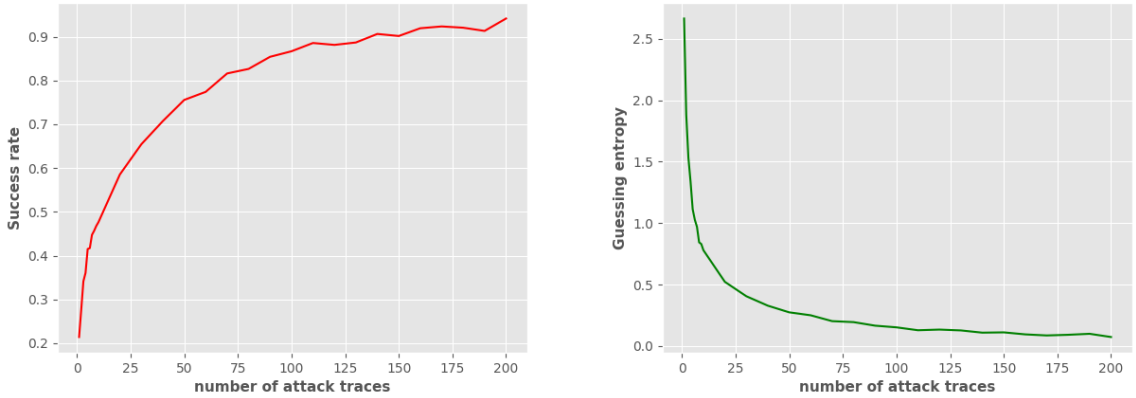
(b) Guessing entropy.

Figure 3.7: Template attack on HW of data in registers.

30% is reached with one trace. Further on, the success rate increases and reaches 90% around 50 attack traces with a peak rate at 95% around 200 traces. This is confirmed in the guessing entropy plot where the minimum value is 0.1. The reduced efficiency observed here compared to the attack on interconnect bus can be attributed to the difference of SNR and correlation levels observed between the two modules. However, we believe that the 95% success rate cannot inhibit the HW recovery provided that some

methods to reduce the noise in the measurements are applied in order to increase this success rate.

Retrieving the Hamming distance. The same process has been applied, this time, to recover the HD between two data consecutively stored in a register. The success rate and guessing entropy are plotted in Figure 3.8a and 3.8b. In this case we have a small



(a) Success rate.

(b) Guessing entropy.

Figure 3.8: Template attack on HD of data in registers.

decline in the results compared to the previous one (attack on HW on registers). The success rate is 20% for a single trace and reaches 75% for 50 traces. We can notice that more than 100 traces are needed to reach 90%. The guessing entropy starts reaching 0.1 after 125 traces, which is coherent with the success rates computed. Once again, the efficiency decline of this attack compared to previous ones can be explained by the low correlation and SNR values observed on registers during the leakage assessment but it can still be reverted with an appropriate method to reduce the noise, such as traces averaging as mentioned in Chapter 2.

3.5 Data value exposure on interconnect bus

Various analysis have examined side-channel leakage based on classical Hamming Weight and Hamming distance leakage models. These models work quite well when it comes to building a template attack to retrieve the key of an encryption algorithm

such as AES [HTM09]. They do not, though, allow the value of the data to be retrieved directly. Hence, we wondered whether there was more information to be gained from the side-channel power traces we collected. Our main hypothesis was that the power side-channel traces can lead directly to the value of the data. This means checking if the contribution of the different bits of the secret data to the overall leakage is different, allowing the value of each bit to be recovered.

3.5.1 Stochastic Model as starting point

The working principle of Stochastic Model (SM) is described in [ZZ19] and recalled in Chapter 2 of this manuscript. In order to motivate the choice to target directly data value in the leakage of interconnect, the least square estimation used in the profiling phase of SM attack has been used to estimate the leakage coefficients associated to each bit in the data. Finding different value for the coefficient of each bit will justify the investigation.

We recorded 300K power traces using the same equipment as in previous analysis and applied the analysis on each byte of the 32-bit word sent on the interconnect bus. Figure 3.9 shows the coefficients computed at the time instant where the highest correlation factor has been observed in the leakage assessment.

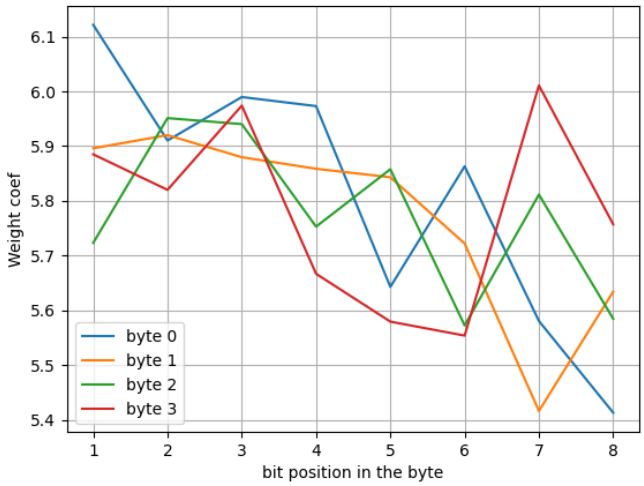


Figure 3.9: SM: weight coefficients.

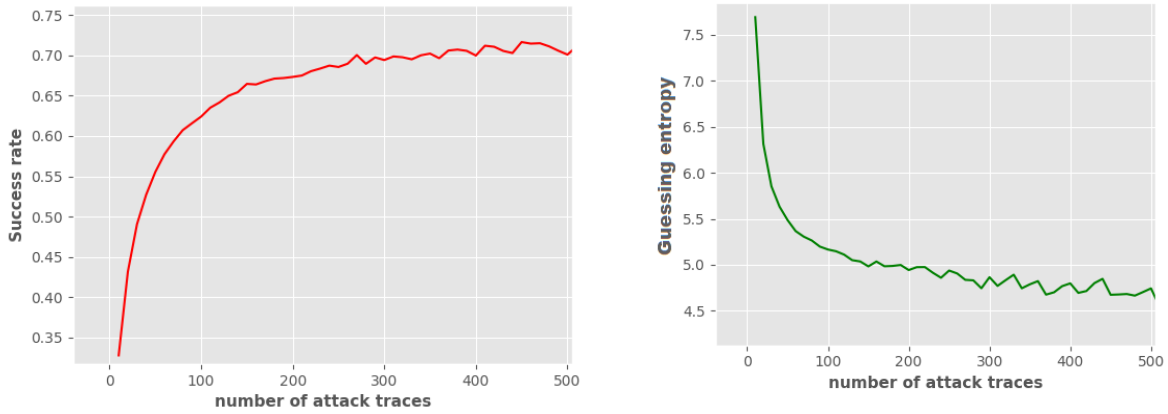
On these plots, we can see that for each byte, from one bit to another we obtain different coefficients showing unequal contributions of each bit to the global power

leakage. This allows to expect retrieving the value of a bit, a byte or the whole data via an attack. Based on this analysis, we set up a template attack with the goal of recovering the value of one byte of the data sent on the interconnect bus.

3.5.2 Recovering a byte value

The difficulty of attacking a 32-bit value lies in the need for a huge amount of traces to build the templates (at least 2^{32} traces). A divide and conquer strategy to attack one byte value at a time was thus necessary. We built templates on a given byte of the data while drawing the other bytes at random. Therefore, we recorded 5,000 traces for each class (256 classes) and divided the records into two sets for profiling and attack. For dimensionality reduction, we first run a correlation analysis with the HW model in order to remove non useful samples (samples corresponding to "nop" instructions) and then we used LDA on the remaining samples. We kept 10 PoIs after a prior research of the best number of PoIs.

Figure 3.10a and 3.10b show the guessing entropy and success rate of the template attack for 2,000 attack trials.



(a) Success rate.

(b) Guessing entropy.

Figure 3.10: Template attack on value of data on interconnect bus.

We can observe on those graphics that with a single attack trace we get 30% success rate and more than 7.5 as guessing entropy. We can also see that less than 30 traces are needed to reach 50% success rate which is a significant point if we consider a program running loading 30 times the same sensitive data trough the interconnect bus, we have

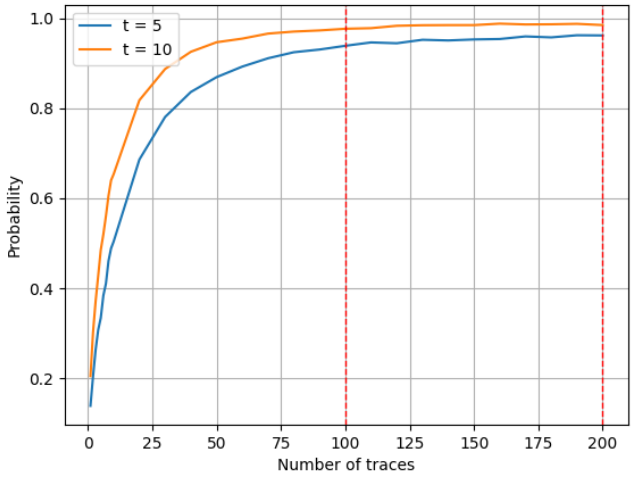


Figure 3.11: Template attack: probabilities for the correct byte to be among the t first values for different number of attack traces. Examples for $t = 5, 10$.

1 out of 2 chances to retrieve the byte value of the data. More than 300 attack traces are needed to reach 70% success rate while having a guessing entropy approaching 4.5.

3.5.3 Recovering the full value

The results of the attack on one byte value enable the use of a *divide-and-conquer* approach to retrieve the 32 bits of the data.

A template attack on the byte value only guarantees a success rate of 70% with 200 traces. This shows that we cannot only rely on a single template attack on each byte of our data. A common approach in this situation is to estimate the rank of the correct value among all the possible values of the byte. To do this, we computed the probability that the correct byte value is among the first t values. That is, computing equation (3.1) where $\{b_i\}_{0 \leq i \leq 255}$ are all the possible values of the byte, $\hat{b}$ represents the correct byte value and $rank(b_i)$ is the rank of the byte value b_i after classifying the posterior probabilities computed in the template attack.

$$Pr(\hat{b} \in \{b_i \; s.t. \; rank(b_i) \leq t\}) \tag{3.1}$$

We computed equation (3.1) for different values of t and got the results depicted in Figure 3.11.

$Size\ of\ Data$ / $Number\ of\ traces$	1B	4B	16B
100	0.98	0.92	0.72
200	0.99	0.96	0.85

Table 3.1: Success probabilities expected for different sizes of data

For $t = 1$, we fall back on the success rate of the attack. We can notice on this picture that we have high probabilities for more than 100 attack traces and even better ones for $t = 10$ (0.98 for 100 traces and 0.99 for 200 traces). Hence for the rest of our analysis we will focus on the two cases meaning that we suppose our attacker has access to either 100 or 200 recordings of the same data on the interconnect bus.

On the basis of the results shown in Figure 3.11 on one byte we extended the analysis by giving the success probabilities (with $t = 10$) one can expect when targeting a 4-byte data (e.g. a single word in our implementation) or a 16-byte data (e.g. an AES-128 key) and summarized it in Table 3.1. We derived the success probability for an n-byte data by raising the probability for a single byte to the power n.

Running a template attack on 32-bit data is not practical so the only approach to retrieve the value of a large data (more than one byte) is the *divide and conquer* method. That is, attack each byte separately instead of attacking the whole data at once. In this case, with the results of the single byte attack, we can estimate, for a fixed value of t, the number of byte values remaining to be tested to obtain the correct value. This procedure, known as key enumeration [MMO18], is used after a side-channel attack on an encryption algorithm but can be used in our case as well if we use a divide and conquer approach. For example, for $t = 10$ and with 200 traces there is a 99% chance that the correct byte value is among the first 10 values. Thus, the template attack helped us reduce the set of 256 possible values to 10. For a 4 byte data, we can estimate the number of byte values to be tested after the attack at 10^4 ($2^{13.2}$), which is computationally feasible. A summary of the number of remaining values to be tested is presented in Table 3.2 for different size of data. These numbers were calculated with equation (3.2) where x represents the size in bytes of data d.

$$N_{enum}(d) = 2^y,\ with\ y = log_2(t^x) \tag{3.2}$$

Number of traces \ *Size of Data*	1B	4B	16B
100	$2^{3.3}$	$2^{13.2}$	$2^{53.1}$
200	$2^{3.3}$	$2^{13.2}$	$2^{53.1}$

Table 3.2: Number of enumeration for a given number of traces and $t = 10$.

The information provided in Tables 3.1 and 3.2 allow us to conclude that, using a template attack on each byte in a divide-and-conquer approach, we can recover the value of a 32-bit data sent over the interconnect bus with a probability of at least 0.96 with 200 traces and time complexity of $2^{13.2}$ for the remaining research of correct values. The same attack, if launched during the transfer of a 128-bit AES key on the interconnect bus will require $2^{53.1}$ operations for the enumeration part (with $t = 10$ and a probability of 0.85 for 200 traces). The French National Agency for the Security of Information Systems (ANSSI) recommends that below 2^{70}, the cost of the remaining effort after an attack should be considered as null [Str19]. Hence, an AES key recovery attack with our approach can be considered as a serious threat. Yet, $2^{53.1}$ is a large number and obtaining the full key is still a challenge. In this context, a trade-off between the desired probability of success and the complexity of the attack may be considered. Table 3.3 shows the time complexity in terms of required number of operations for enumeration and success probability that is to be expected for $t = 5$.

Number of traces \ *Size of Data*		1B	4B	16B
100	Probability	0.94	0.78	0.37
	Complexity	$2^{2.3}$	$2^{9.2}$	$2^{37.1}$
200	Probability	0.96	0.85	0.52
	Complexity	$2^{2.3}$	$2^{9.2}$	$2^{37.1}$

Table 3.3: Number of enumeration and expected success probability for a given number of traces

We can observe that reducing the set of byte values from 256 to 5 after the template attack naturally reduces the complexity of the post-attack search but reduces the like-

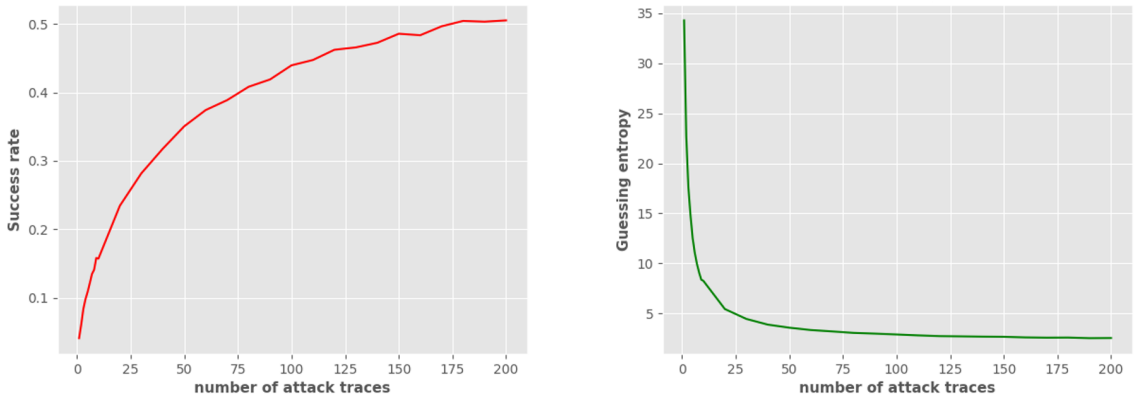
lihood of successfully retrieving the entire data, especially for a 16-byte data like an AES-128 key. Two possibilities emerge: choose $t = 10$, which results in a higher probability (0.85) of success but with a high computation cost $(2^{53.1})$, or choose $t = 5$ to reduce the amount of computation $(2^{37.1})$ to be done but with a lower probability of success (0.52). An adversary with limited computing power will be more likely to go for low values of t, seeking a balance between the computation complexity and the expected success probability, rather than a well-equipped one who will be comfortable with high values of t, looking for best success probability for his attack.

3.5.3.1 Discussion on the exploitation of the attack

The attack presented in the previous section leads to the direct recovery of data value on the interconnect bus using its side-channel leakage. One might think of another approach to extract data on interconnect bus which is by mean of physical probing attacks [SK08]. However, these attacks mostly require expensive equipment and are invasive, whereas template attacks do not incur any damage on the device to achieve the same goal. Many microcontrollers make use of software libraries that provide cryptograhic algorithms. The CPU executing such algorithms usually needs to load some critical data from the main memory through the interconnect bus multiple times. Using the presented attack, an adversary could retrieve those sensitive information, ranging from critical user data to encryption keys. In fault injection attacks, it is quite hard to determine the precise moment to inject the perturbation. Having the capacity to retrieve the data sent on the bus will make it possible. Due to the requirement of 200 traces for our attack, we believe that it will be difficult to use it as a synchronization tool for a fault injection attack. In addition, it can be noticed that none of the values of t leads to a probability of 0.99 for retrieving the byte value. A possible solution to this issue is to choose greater values of t, which will increase the chances to find the correct value in the obtained set. Again, the side effect of this is the increasing residual complexity. These few examples give an idea of how to exploit our attack either as a complement to other attacks or for the purpose of retrieving some sensitive data.

3.5.3.2 Can we find the data value on registers?

The goal here is to run the same experiments as for interconnect bus to check if retrieving the value of registers content is practicable. We got the results depicted in Figures 3.12a and 3.12b.



(a) Success rate.

(b) Guessing entropy.

Figure 3.12: Template attack on value of data in register

The highest value of success rate is 50% and is reached for 200 traces. Unlike the results on interconnect bus, it becomes difficult to reach lower guessing entropy or higher success rate values. What we can understand from this figure is that it will be a challenging task for an attacker to retrieve one byte of some data stored in a register with a great success rate value through a template attack. In this situation, we consider that trying to retrieve the entire data value is not relevant if we cannot retrieve one byte. In the design of countermeasure this threat could be less considered due to the low success rate of the attack.

3.6 Attacker capabilities quantification

The earlier sections of this chapter outlined the capabilities of an attacker having the leakage of the data as it passes through the interconnect bus or registers. These results provide an adequate baseline to estimate the extent of the threats identified in the attacker model(Chapter 1). Eventually, the relevance of such a model can be examined.

The study presented in this chapter reports the possibility of directly retrieving the value of the data with a probability of 0.96 for 200 attack traces and a residual enumeration complexity of $2^{13.2}$. Even if the same attack was not successful on registers, we were able to retrieve HD between data. Many observation attacks have been demonstrated on all the parts of the memory hierarchy. Cache memories vulnerability to first-order power analysis attack has been exhibited by Giterman et al. [GKF18]. As it can be no-

ticed, the lack of security of data in the memory hierarchy regarding power side-channel attacks have been shown in many works [Wei+21; LLW18]. It is also worth observing that the protection of data requires to define a scheme that considers the memory hierarchy as a whole instead of trying to protect each module of the hierarchy.

3.7 Conclusion

Microarchitectural side-channel leakage has been exploited for a while, either for retrieving sensitive data such as encryption keys or finding internal variables stored in the memory hierarchy. The need for protection of such components implies an initial leakage assessment with some methods to exhibit how this leakage could be exploited in order to provide hardware designers with sound analysis and an attacker model with a threat that is quantified through practical attacks. This rationale motivated the work presented in this chapter.

There are many studies that focus either on attacks assuming a pre-existing leakage or on attacks to retrieve specific data based on microarchitectural leakage. This study, instead, proposes a complete leakage analysis in which the classical Hamming Weight (HW) and Hamming Distance (HD) leakage models evidenced in [MPW21] are confirmed. To complete the analysis, we propose a means to exploit them with a template attack to retrieve the HW or the HD information on any data sent through the interconnect bus or stored in a register. Furthermore, a leakage assessment has been conducted on interconnect bus and registers of a 28 nm ASIC SoC embedding a RISC-V CPU using another leakage model which is the data value. A practical attack to exploit that leakage and that can lead to a recovery of any data transmitted on interconnect bus is presented.

From these results, the relevance of the defined attacker model has been demonstrated in order to enable the definition and implementation of a suitable countermeasure against the specified attacker. This will be the focus of the next chapter where the investigation of the different possible countermeasures is presented, as well as the adopted solution and its analysis regarding some security and performance requirements.

Résumé du chapitre

Les fuites par canaux auxiliaires de la microarchitecture sont exploitées depuis des décennies, soit pour récupérer des données sensibles telles que des clés de chiffrement, soit pour trouver des variables internes stockées dans la hiérarchie de la mémoire. La nécessité de protéger de tels composants implique une évaluation initiale des fuites avec certaines méthodes pour montrer comment ces fuites pourraient être exploitées afin de fournir aux concepteurs de matériel une analyse solide et à un modèle d'attaquant une menace quantifiée par des attaques pratiques. Ce raisonnement a motivé le travail présenté dans ce chapitre. De nombreuses études se concentrent soit sur des attaques supposant une fuite préexistante, soit sur des attaques visant à récupérer des données spécifiques à partir d'une fuite microarchitecturale. Cette étude, au contraire, propose une analyse complète des fuites dans laquelle les modèles classiques de fuite par poids de Hamming (HW) et distance de Hamming (HD) mis en évidence dans [MPW21] sont confirmés. Pour compléter l'analyse, nous proposons un moyen de les exploiter avec une attaque par profilage pour récupérer les informations HW ou HD sur toute donnée envoyée par le bus d'interconnexion ou stockée dans un registre. En outre, une évaluation des fuites a été menée sur le bus d'interconnexion et les registres d'un SoC ASIC de 28 nm intégrant un CPU RISC-V en utilisant un autre modèle de fuite qui est la valeur des données. Une attaque pratique pour exploiter cette fuite et qui peut conduire à la récupération de toutes les données transmises sur le bus d'interconnexion est présentée.

A partir de ces résultats, la pertinence du modèle d'attaquant défini a été démontrée afin de permettre la définition et la mise en œuvre d'une contre-mesure appropriée contre l'attaquant spécifié. Ce sera l'objet du chapitre suivant, qui présente l'étude des différentes contre-mesures possibles, ainsi que la solution adoptée et son analyse au regard de certaines exigences de sécurité et de performance.

Chapter 4

Ensuring data confidentiality in memory hierarchy

This chapter outlines the investigations that have been done in the research for suitable countermeasure to protect interconnect bus and CPU registers against first-order side-channel attacks using their power consumption. A systemic approach is adopted in order to protect the whole memory hierarchy using a combination of encryption and lightweight masking based on a new mask generator that is proposed. The work presented in this chapter has been published in MDPI Cryptography journal [*Tal+22*].

Contents

4.1 Introduction

Micro-architectural leakages have increased the vulnerabilities of communication infrastructures inside a SoC. Specifically, interconnect bus and registers leakage have been exploited for the sake of extracting information on the data that they contain. After the leakage assessment work presented in the previous chapter, it turned out that these components' leakage can be explained with classical leakage models such as Hamming weight and Hamming distance. Furthermore, the successful recovery of these informations through template attacks, and even the data value on the interconnect bus, confirmed the soundness of the attacker model that has been defined at the beginning of the manuscript. To face these threats that undermine data confidentiality in such components, we investigated different solutions for ensuring protection of data against first-order side-channel attacks based on power consumption. In addition to the need for security, emphasis is put on solutions that have a low impact on the processor size as well as on the performances of the chip (latency, maximum frequency).

Like it is done in most securing process, after a prior leakage assessment and threat quantification on these modules, it is quite obvious to look for solutions that protect them against side-channel attacks. Some solutions like constant weight encoding, masking, data encryption and dual-rail implementations have been investigated. However, protecting these two components separately will not allow us to achieve the desired security for the data. Indeed, one could as well protect the data exclusively in the registers and on the bus, but we believe that an attacker could exploit the leaks of the other components of the memory hierarchy which would not be protected in order to find the same data whose leaks have been limited on the bus and in the registers. Therefore, a systemic and monolithic approach in which the entire memory hierarchy is considered, is the best way out of the threats endangering the confidentiality of data from the main memory to the CPU. This approach ensures security of data in the memory hierarchy while having a reduced impact on the overall performances of the system. Protecting each component separately leads to the addition of individual overheads and the interfaces between all the components are potentially not considered, which is critical for performances purpose and even for security. The goal of this chapter is to detail our proposed protection scheme, which is based on a lightweight encryption and masking scheme in order to achieve first-order security against power side-channel attacks while keeping a low performance overhead on the implementation. We also present a secu-

rity analysis of our proposal and discuss its impact on the final SoC in terms of area, latency and maximum frequency.

4.2 Modular protection scheme

As a natural follow-up to the analysis presented in Chapter 3, some solutions have been investigated with the purpose of protecting data in interconnect bus and registers separately while keeping low impact on performances.

Constant Hamming weight encoding for interconnect bus. The template attacks performed on the interconnect bus has revealed the possibility to retrieve the HW of data with 15 traces. The first solution that has been investigated is the development of constant HW coding schemes to remove or at least reduce the link between the power consumption and the data on the bus. This can be done at logical/algorithmic level. The proposal of Montoya et al. [Mon+20] to dynamically switch between two coding schemes with an additional support for constant HD between codewords suffers from large overhead (size of information word $\times 4$) and does not suit our needs since no HD leakage has been found in interconnect bus. It would also require to increase the size of the bus by factor 4. Software-like dual-rail could also help achieve our goal but would lead to a double-sized bus. Knuth presented efficient solutions for building balanced codes [Knu86]. A balanced binary word of size n is defined as a word that has a Hamming weight of $\lfloor \frac{n}{2} \rfloor$. A balanced code is formed by balanced codewords that are all different even if they have the same HW. The total number of balanced words of size n can be referred to as $M(n)$ and is defined in equation (4.1).

$$M(n) = \binom{n}{\lfloor \frac{n}{2} \rfloor} \tag{4.1}$$

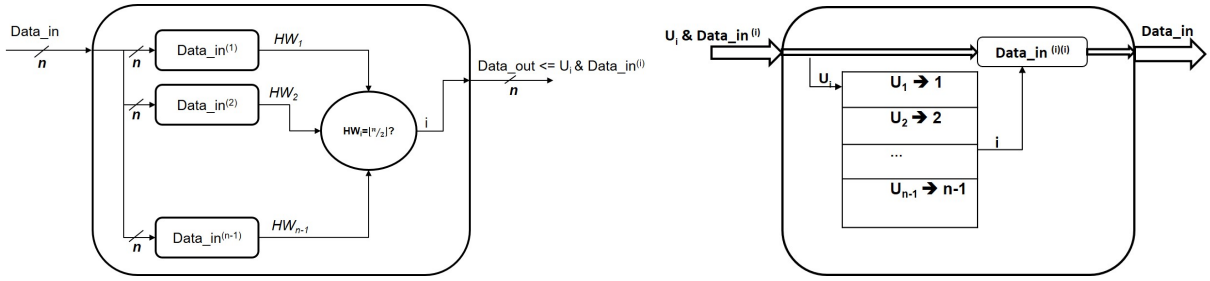
We can hence have a constant HW on the interconnect bus if we are able to build balanced code to encode all data going through the bus. Both the encoding and decoding can be done by adding two hardware modules (RTL level) to the interconnect bus. Let $X = x_1 x_2 ... x_n$ be a bit sequence of size n. Let's denote $X^{(i)}$ the binary word X with its first i bits complemented. Knuth proved that there is at least one integer k such that $X^{(k)}$ is balanced.

- *Encoding:* To encode an n-bit word X, one has to go through the following process:

  - Find $min(D)$, with $D = \{k : HW(X^{(k)}) = \lfloor \frac{n}{2} \rfloor\}$
  - Encode $min(D)$ in a balanced codeword u of length p
  - Finally obtain $uX^{(min(D))}$ as codeword for X

- *Decoding:* The decoding operation is done by retrieving the value of $min(D)$ knowing u and then computing $X^{(min(D))(min(D))}$.

Based on this construction, a hardware encoder and decoder has been developed. Figures 4.1a and 4.1b describe the implementation of the encoder and decoder.



(a) Encoder. (b) Decoder.

Figure 4.1: Hardware implementation of balanced coding scheme.

Let p be the size in bits of the codeword u used to encode the information on the number of bits that have been flipped in the initial word X. To obtain a balanced word from X, one has to flip at most $2 \times \lfloor \frac{n}{2} \rfloor$ bits. Hence, for this scheme to work, the following equation has to be satisfied:

$$M(p) \geq 2 \times \lfloor \frac{n}{2} \rfloor \tag{4.2}$$

For example, in order to encode a 64-bit data, one has to flip at most the first 63 bits of that data. $M(8) < 2 \times \lfloor \frac{n}{2} \rfloor < M(9)$ so, the minimum number of bits needed for the codeword u is 9. In the end, 73 bits (64 + 9) are needed to encode a 64-bit data, making an overhead of 14% (9/64) which is less than dual-rail solutions. The benefits of this solution is the reduced number of additional bits needed to ensure the decoding, as shown in this analysis for a 64-bit data. Nonetheless, implementing the serial encoding in hardware implies to duplicate the input data n-1 times like depicted in Figure 4.1a. This will increase the leakage of the input data, which is not wanted. In addition to this, this scheme only helps reducing HW leakage, leaving aside the direct value leakage evidenced on interconnect bus in the previous chapter.

Boolean masking for registers As a classical countermeasure against side-channel attacks, masking could be a relevant candidate for protecting registers. The content of registers would be added to random masks generated by a Random Number Generator (RNG). The masked data and the mask are then stored in different registers. It has the advantage of ensuring the confidentiality of data with a low implementation cost, but it reports the security of the data to the one of the masking scheme, especially the LFSR which in turn has been shown to be vulnerable to power side-channel attacks [BMV07] as well as classic cryptanalysis [Zen04]. Using a True Random Number Generator (TRNG) can lead to optimal first-order security but it implies a performance overhead since the TRNG will have to generate 64 bits in a few clock cycles. In addition, the size of register bank will be doubled to be able to store the masks.

4.3 Lightweight memory hierarchy encryption

The necessity to protect data during the transit in the memory hierarchy has appeared after several work exposing first-order side-channel attacks on different part of the memory hierarchy such as caches, interconnect bus, registers, ALU and FPU. Data encryption has then become among the most studied countermeasure against this threat [UWM19; YXH20], keeping in mind the need to reduce its impact on the performance of the processor. Wong et al. also proposed the design of a memory protection unit (MPU) to ensure confidentiality and integrity of data on a RISC-V SoC [WHC18]. 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. This does not take into account the protection of data in registers. The power attack resistant microprocessor (PARAM) designed by Arsath K F et al. [Ars+20] goes beyond the only protection of DRAM content by protecting all the leaking modules of the processor with encryption of data and their addresses using a 4-round Feistel structure, leading to a lightweight solution. The authors use a frequent re-keying strategy to prevent disclosure of the secret key using 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.

The primary incentive for this work is how to bring data securely from the main memory to CPU registers and to prevent data eavesdropping via side-channel attacks

up to the execute stage of the CPU by enabling secure execution of instructions in that stage. It is widely adopted in the literature that protecting such transfers can be done with encryption. Nonetheless, keeping encrypted data in cache memories 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 of the pipeline where instructions can be processed on masked data. Figure 4.2 illustrates the protection method we proposed, which will be applied to a SoC embedding a 64 bit RISC-V CPU. This Figure illustrates the data flow from RAM to CPU during the execution of a given program. The data from peripherals are not currently encrypted nor masked. This could lead to recovery of potential user data from a plugged in peripheral but this threat is not considered in our analysis at this time.

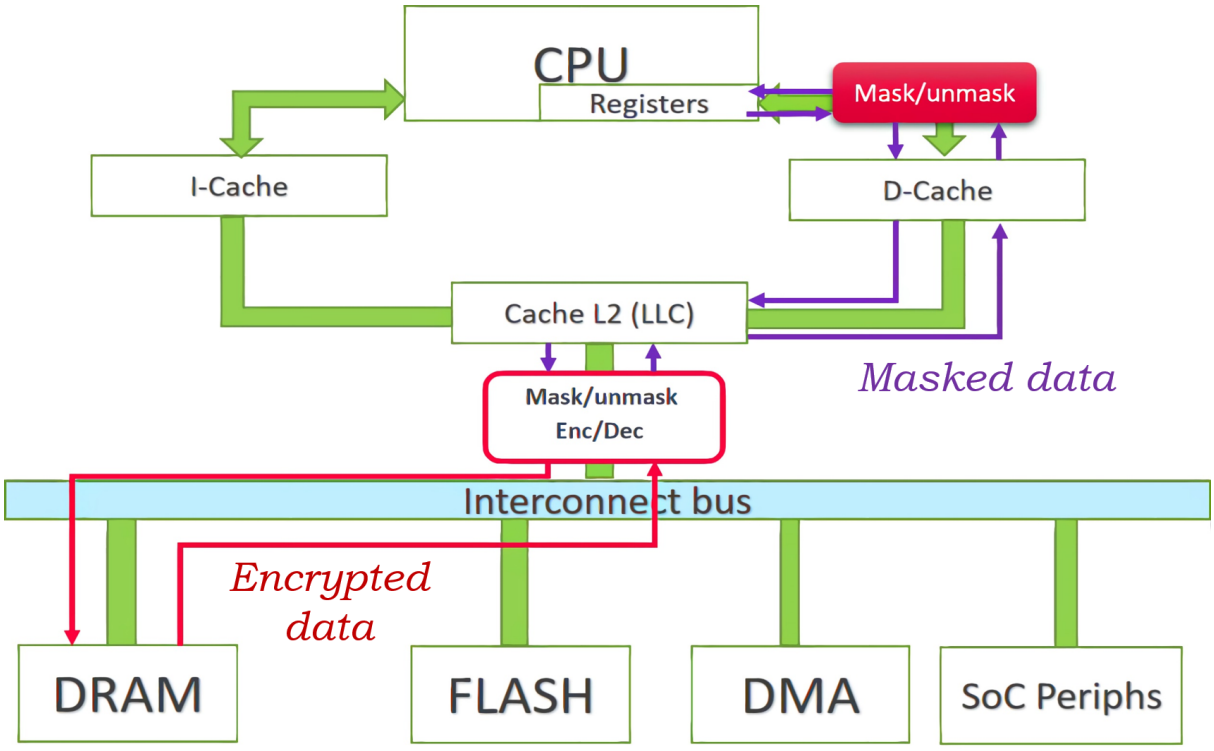


Figure 4.2: Overview of protected memory hierarchy.

4.3.1 Encrypting data on the interconnect bus

The architecture presented in Figure 4.2 shows the use of authenticated encryption of data on the interconnect bus up to the last level cache memory. In order to have the lightest solution, we have opted for lightweight cryptographic primitives. A key point to pay attention to, regarding the interconnect bus, is the throughput. Encrypting the data passing through the bus will therefore decrease this transfer rate. Hence the need for an encryption primitive with a high throughput in order to reduce the impact on the overall transfer rate of the interconnect bus. To do so, we have studied some of the algorithms submitted to the ongoing NIST Lightweight Cryptography Competition (LWC) [Com17] for authenticated encryption that offer protection to side-channel and fault injection attacks. The goal is to analyse the following parameters:

- Spatial footprint: the number of resources utilised by the hardware implementation of the primitive either on ASIC (Gate Equivalents) or FPGA (LUTs, FFs, slices)
- Latency overhead: the delays added on the interconnect bus due to the addition of encryption/decryption.
- Number of modules for encryption and decryption: it specifies whether the encryption and the decryption can be done with a single hardware module or not. Using a single module for the two operations reduces the resource utilisation and the implementation efforts.
- Throughput: amount of data encrypted or decrypted per second. According to the frequency of the module, the number of cycles needed to encrypt/decrypt is known from the throughput (in Mbit/s).
- Implemented mechanisms for side channel and fault injection protection

The size of the internal state and the security analysis of their underlying permutations are also taken into account. According to the given criteria, we focused on the following candidates: ACE, ASCON, DryGascon, Gimli, ISAP, Subterranean, and Xoodyak. All these algorithms provide authenticated encryption using sponge construction with public permutations. After looking at these candidates with regards to our pre-defined parameters, we summarized the results into two tables. Table 4.1 gives the different parameters related to the architecture of the ciphers and Table 4.2 gathers the result of

hardware benchmarking that has been reported on FPGA. For each primitive presented in Table 4.1, the first column gives the number of bits of the internal state of the used permutations. The second column informs about the possibility to encrypt and decrypt with the same hardware module. The third column gives an overview of the security analysis performed on the primitive, either by cryptanalysis or by physical attack. the last column summarises the different mechanisms put in place (or facilitated) by the authors of the algorithms to protect the latter against side-channel and fault injection attacks.

Candidates	*State size (bits)*	*Encrypt / Decrypt*	*Security analysis*	*Side-channel & fault protection*
ACE	320	Single module	Differential cryptanalysis on ACE permutation[LLQ20]; Master key recovered via fault attack on SIMECK whose permutation is used for non-linearity in ACE[LLG20]	No available feature against side channels or fault injection
ASCON	320	Single module	Several cryptanalysis realised on reduced versions for key recovery and state recovery; initialisation and finalisation phase vulnerable to DPA [AFM18]	Bit-sliced software implementation of S-boxes facilitated

DryGascon	320	Single module	differential cryptanalysis on the GASCON permutation with time complexity of $2^{61.28}$[Tez20]	Authors claim similar security level as ASCON permutation in addition to algorithmic level protection against physical attacks via the DrySponge construction; no support for *safe error attacks* and *blind side-channel analysis*
Gimli	384	Single module based on Gimli permutation	Various attacks for state recovery [LIM20b]; Distinguishing attack in time complexity of 2^{52}[LIM20a]	All non-linear operations are carried out in parallel, which can ease the implementation of masking countermeasure
ISAP	400/320 (Keccak-p/ ASCON-p)	Two distinct modules	Single trace side-channel attack reported on Keccak permutation[KPP20]; Also take into account the various attacks on ASCON permutation	IsapRk for generation of session keys enhances side-channel protection through the re-keying strategy; the re-keying function is hard to invert in order to mitigate fault injection attacks

Subterranean	257	Single module	Full state recovery through cryptanalysis with 2^{15} bytes of data and key recovery with time complexity of 2^{35}[LIM19]; All published attacks are on reduced versions of the algorithm (4 blank rounds instead of 8)[Son+21]	Authors point out the ability to derive a fresh key per session using Subterranean-deck; The low algebraic degree (two) enables protections such as masking or threshold implementations
Xoodyak	384	Single module based on Xoodoo permutation	Cryptanalysis on reduced version (6 rounds out of 12) to recover the 128-bit key in time complexity of $2^{43.8}$ in nonce misuse setting[Zho+20]	Authors propose key derivation to counter side channel attacks: a key is derived from the secret key and stored for use at the next invocation of the algorithm; The round function lends itself to efficient masking countermeasure

Table 4.1: NIST LWC candidates features comparison.

Many benchmarking have been published both on software and hardware implementations but we focus here on hardware benchmarking since the architecture presented in Figure 4.2 will be implemented in hardware. Mohajerani et al. published benchmarking of FPGA prototypes of these algorithms on 3 FPGA platforms: Xilinx Artix 7, Intel Cyclone 10 LP and Lattice Semiconductor ECP5 [Moh+20]. We summarize their results on Xilinx Artix 7 in Table 4.2 where the spatial footprint, the throughput and the maximum achievable frequency are presented for each candidate.

In the effort to find the most suitable lightweight primitive for our context, we examined the elements presented in the previous two tables. Due to its small memory

	Area			Max freq	Throughput for 64B plaintext (Mbit/s)
Candidates	*LUTs*	*Flip-Flops*	*Slices*		
ACE	1,229	894	400	200	55.8
ASCON	2,797	666	785	150	1,181.5
DryGascon	2,074	1,220	596	238	902.6
Gimli	1,747	1,169	502	175	452.5
ISAP	3,491	1,177	937	193	189.3
Subterranean	891	610	253	190	1,496.6
Xoodyak	1,355	555	407	234	1,079.4

Table 4.2: FPGA benchmarking on Xilinx Artix-7.

footprint and high throughput, we decided to use Subterranean. Regarding security, all cryptanalysis attacks on Subterranean have been performed on versions with a reduced number of rounds, as reported in Table 4.1. Since we use the complete version of the algorithm for encrypting data in the main memory, we expect the encryption module to not suffer from such attacks. Given the results in Table 4.2, we might as well go for Xoodyak, which operates at a better frequency than Subterranean. However, the throughput of the latter is higher than that of Xoodyak. Although its lower frequency compared to Xoodyak's, we believe that this will not have a significant impact on latency performances of the bus. Another benefit of Subterranean over Xoodyak is its small size.

4.3.2 Masking data in caches

Encrypted data with Subterranean will need to be decrypted at the output of the bus before being masked to be brought into the cache. 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 doubles the size of the data to be stored in caches since each masked data need to be stored along with its mask which has the same size. 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 [MAD11], 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. It thus implies to be able to recompute the masks at any time since they are not stored. Classical random number generators do not suit for this purpose as they require to store the masks or a given seed. A new mask generator has to be designed. That mask generator, while enabling first order security, should be able to output 64-bit masks without the need to store them while keeping low area overhead and low impact on performances of the CPU. Our goal is hence to define a mask generator that respects these security and performance constraints. Therefore, LightMaG, a lightweight mask generator is proposed. It can output 64-bit masks at each clock cycle using a lightweight cipher. With this generator, only 8 informative bits (referred to as IVs) are stored in the cache instead of the whole 64-bit masks. Memory and delay overheads are hence reduced and the lightweight cipher used helps keeping a low area overhead on the overall spatial footprint of the CPU. Being aware of this, it is reasonable to question whether a masking solution based on such a mask generator will be able to protect the data in the caches against first-order power side-channel attacks. We address this concern in the next section with a leakage assessment on simulated traces which will lead to the definition of security requirement for masks generation in order to get first-order secure memory hierarchy.

4.3.2.1 Security requirement for mask generation

Several methods are available for assessing the robustness of circuits [GJR11; Bat+11; BCO04] against side-channel attacks and the efficiency of most of them has been proven for first order leakage analysis. In case of masked implementations, the higher order extensions of these tools are also available. We can think of higher order t-test which uses high-order statistical moments [SM15]. Due to its ability to capture all kinds of dependencies between side-channel traces and sensitive data, we believe that Mutual Information (MI) [SMY09] is a relevant tool for assessing the security level of any implementation as no assumption on the leakage (it can be applied with both univariate and multivariate leakage models) is needed while using it. Besides using MI only for leakage assessment, it can also serve as metric to compare different mask generation schemes firstly with each other and afterwards with an ideal one that outputs uniformly distributed masks. This is shown by Grosso et al. when trying to study the impact of randomness on the security of masking schemes [GSF14]. 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. To complement their analysis, we suggest the use of the number of traces needed to successfully attack an implementation (either protected or not) as security metric. 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 considered to be among the most harmful observation-based attacks on a device. The threshold of 10,000 attack traces is due to the fact that most attacks published in the literature use less than 2,000 traces for the attack phase [CK14b; HTM09; CK13], 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 10,000 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. The key role of noise in the masking countermeasure has been identified from the earliest days of side-channel attacks mitigation [Cha+99], and has been widely acknowledged. In this section, experiments are presented with the goal of assessing whether it is still possible to reach our security objective with a non-uniform mask distribution instead of uniform masks, and within what range of noise level this could be done. To do this, MI is estimated using MINE [1] with simulated side-channel traces of masked implementation in which we vary the distribution of the masks and the noise level.

Generating simulated traces. Some random data d are generated and *"XORed"* with the masks m to obtain the masked data $d_m = d \oplus m$. We then construct traces containing two samples with the Hamming weight (HW) as leakage model. The first sample represents the leakage of the mask and the second represents the leakage of the masked data. The leakage trace is hence the tuple $(HW(d_m), HW(m))$. 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.

[1] The mutual information neural estimator described in Chapter 2.

Mutual Information estimation. Depending on the type of distribution from which the masks are drawn, a high or low MI can be obtained. In order to better understand this effect, we aim to characterize different masks distributions. The data and the masks are both 8-bit sized. 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 method 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 4.3 explains the experimental process we have been through for 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 the previous paragraph. Then we compute the MI between the trace and the unmasked data using the HW leakage model. 1,000 MI estimations per noise variance were ran to remove measurement fluctuations due to noise. The simulations were done

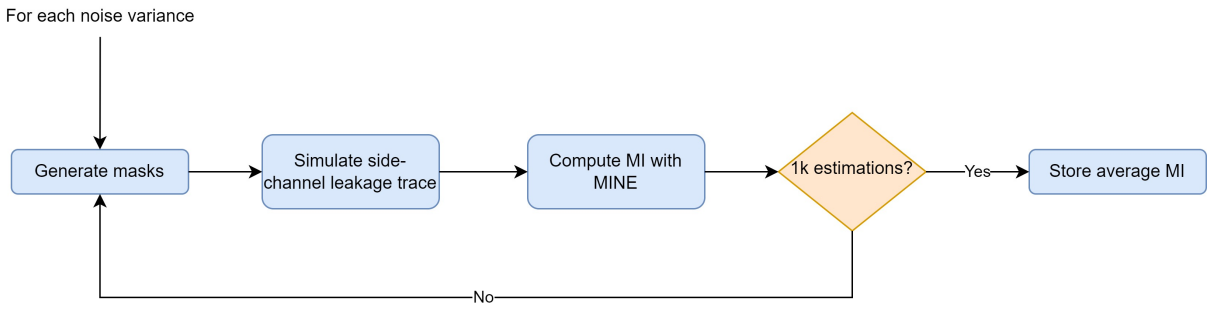


Figure 4.3: MI estimation process.

for different noise levels reflected in the SNR values. For a n-bit data and a noise variance σ^2, the SNR is computed with Equation (4.3) which has been derived from Equa-

tion (2.16) with the Hamming weight of the sensitive data as useful signal.

$$SNR = \frac{n}{4\sigma^2} \tag{4.3}$$

For the biased masks, we used $x = 2, 4, 8, 16$. The result is depicted in Figure 4.4. With

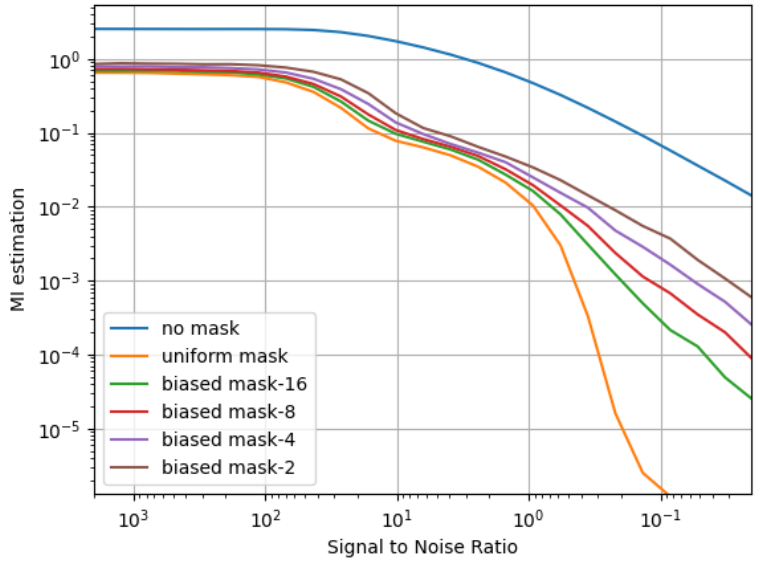


Figure 4.4: MI estimation for different noise levels and different types of bias

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, confirming the soundness of masking when combined with noise. Following our proposal to consider the number of traces needed for an attack as security goal, the next step is to determine how many traces it will take to carry out a successful attack based on the MI analysis. This will help conclude on the possibility to use biased masks in case no uniform mask generator is available. Using the inequality proposed by Chérisey et al. and recalled in Equation (2.28), we can compute the expected minimum number of traces for a successful attack using the MI depicted in Figure 4.4. The results are summarized in Table 4.3 where the success rate parameter (SR) in Equation (2.28) is set to 0.99. We can quickly realize that a potential attack would need at least 10,000 attack traces to succeed if biased masks are used (biased mask-16 and biased mask-8) on a device where the SNR is less than 10^{-2}.

	MI	$Min(N_{traces})$
biased mask-8	10^{-4}	25,900
biased mask-8	5×10^{-5}	59,000

Table 4.3: Number of traces for to reach a success rate of 0.99 for SNR less than 10^{-2}.

From these results, we move on to provide a mask generator architecture while keeping in mind that even if the output masks are biased, the security goal is still reachable within a given SNR range.

4.3.3 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 the mask generator, we believe, should satisfy are defined hereafter.

- **Security constraint:** the generated masks have to enable the protection of the data in caches against an attacker with up to 10,000 attack traces using a profiling attack.
- **Architectural constraints:**
  1. The mask generator must have a low area footprint since it is intended to be placed near the cache memories.
  2. A new mask has to be available at each clock cycle.
  3. It must be possible to recompute the masks as needed.
  4. No latency penalty has to be incurred by the mask generator.

A TRNG is the optimal solution that would respect the security constraint as it produces 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 previous section 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 Subterranean 2.0 [Dae+20]. The comparative study of cryptographic primitives presented in Table 4.2 highlighted that Subterranean had the lowest resource utilisation on FPGA. The same applies to its spatial footprint on ASIC as presented in the benchmarking study of Aagaard and Zidaric [AZ21] where Subterranean ranked first. 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 where the bias level in the masks is characterized via MI estimations.

4.3.3.1 Starting with Subterranean round function

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 i^{th} 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 K, the associated data A, the nonce N, and the message M, like depicted in Figure 4.5 . Although the limited spatial footprint of the whole algorithm demonstrated by Mohajerani et al. [Moh+20] (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

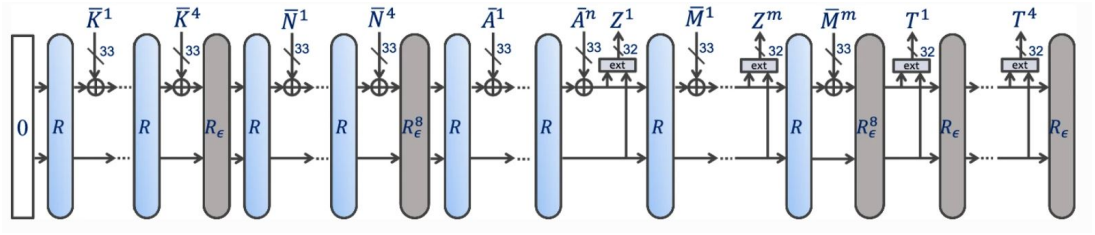


Figure 4.5: Subterranean Authenticated Encryption [Son+21].

up to two rounds. This choice is argued later in this section with the security analysis of the produced masks.

4.3.3.2 Modified absorption and extraction function

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).

Absorption function In the initial function, each data to be absorbed in the internal state is injected by blocks of 33 bits through a XOR operation between each bit of the block and the bits at the first 33 positions forming the set $A = \{12^{4i}\ mod(257),\ 0 \leq i < 33\}$. In order to absorb 128 bits, we use the set A' which is defined from A in the following way: $A' = \{12^{4i+j}\ mod(257),\ 0 \leq i < 32,\ j = 0,\ 1,\ 2,\ 3\}$.

Extraction function The initial extraction function combines two internal state bits at 64 different positions to obtain the extracted block D of 32 bits. It computes the following formula, where S represents the internal state of Subterranean:

$$D_i = S_{12^{4i}} \oplus S_{-12^{4i}},\ \ 0 \leq i < 32 \tag{4.4}$$

We modified this extraction function to extract 128 bits using the following Equation (4.5).

$$D_i = S_{12^{4i+j}} \oplus S_{-12^{4i+j}},\ \ 0 \leq i < 32,\ j = 0,\ 1,\ 2,\ 3 \tag{4.5}$$

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.

4.3.3.3 Architecture of LightMaG

The architecture of the proposed mask generator is shown in Figure 4.6.

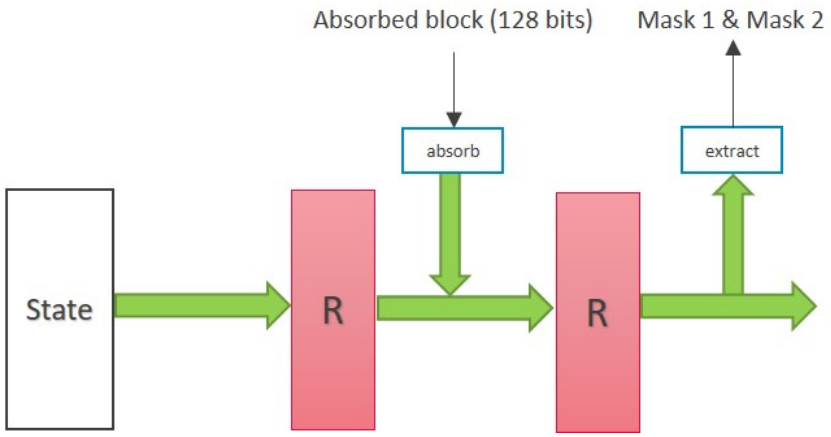


Figure 4.6: Architecture of LightMaG.

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 [Nas+21]

$$State \leftarrow K||ASID||IV||Ptr\_id||address||0^{16}||1^{16} \tag{4.6}$$

The 256 LSB bits of the initial state are permuted by the mean of DES permutation according to the scheme in Figure 4.7 before applying the first round. This allows a better mixing of the input state, thereby improving the entropy of the mask distribution.

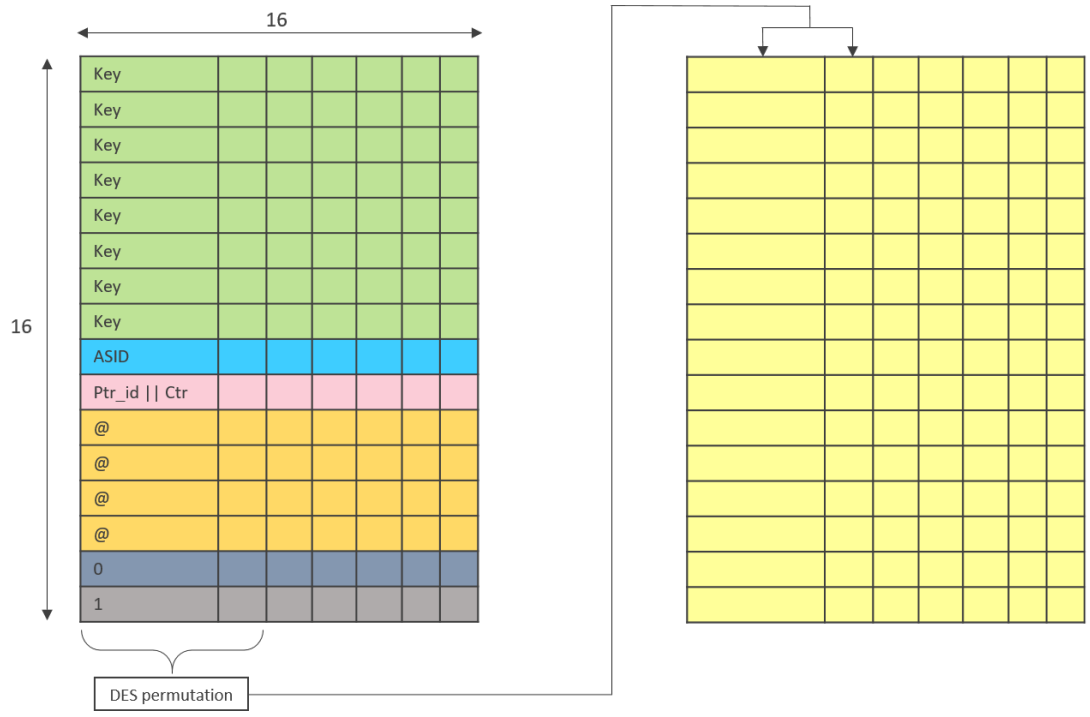


Figure 4.7: Permutation of initial state.

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 (4.7) where $@(48)$ denotes the 48 LSB bits of the address.

$$Absorbed\ block \leftarrow Ptr\_id||IV||@(48)||Perm\_DES(\overline{Ptr\_id||IV||@(48)}) \tag{4.7}$$

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. Figure 4.8 confirms these choices by showing the cumulative distribution functions of each byte of the masks (Mask 1). 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).

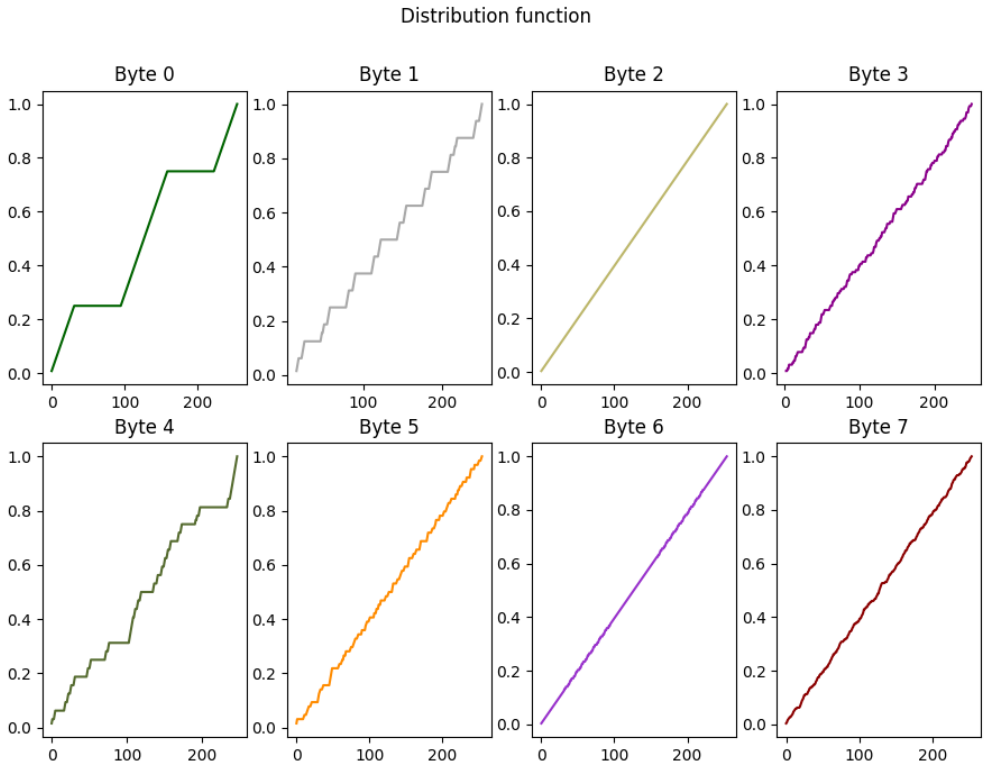


Figure 4.8: Cumulative distribution function of the mask values.

4.3.3.4 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 values meaning that we have highly biased masks. Figure 4.9 shows the MI as a function of the SNR. 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

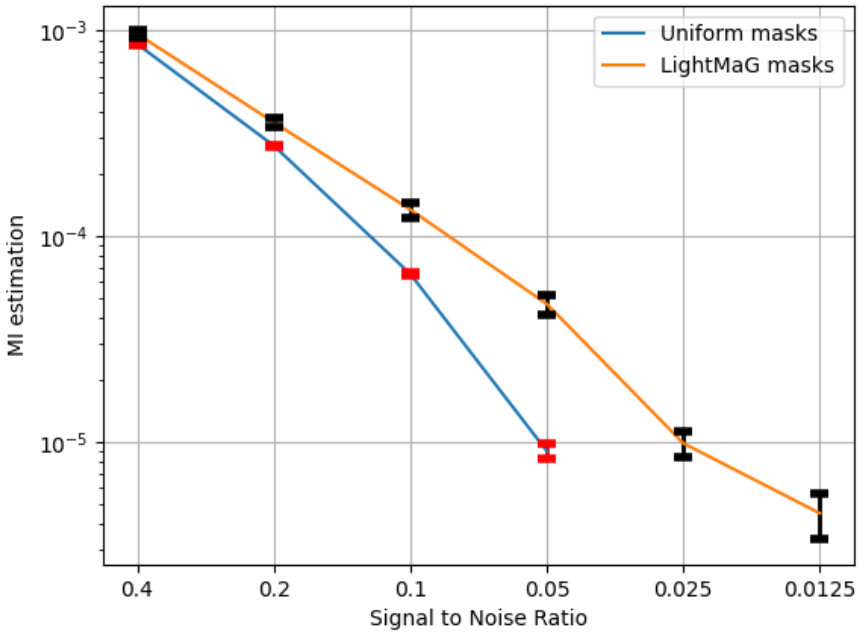


Figure 4.9: Comparing MIs: Uniform vs. LightMaG masks.

to first-order secure masking scheme. Using inequality (2.28), we compute this number for different success rates and a fixed MI value. The results are presented in Table 4.4. We choose the MI corresponding to a SNR of 0.02 as example, which is consistent with

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

Table 4.4: Number of traces needed to for different success rates. MI set to 10^{-5} (SNR = 0.02). Data are masked with LightMaG masks.

the SNR values observed on consumer application processors [Yan+13]. Figure 4.10 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.

The results in Table 4.4 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

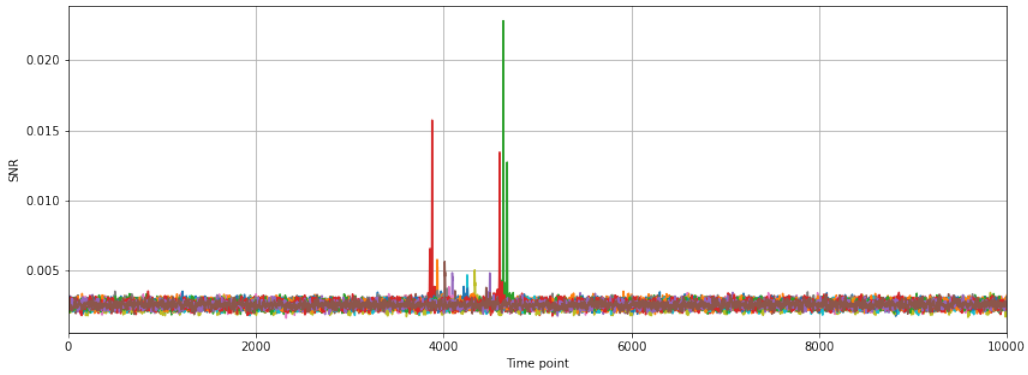


Figure 4.10: SNR on ARMv7 platform with Cortex-A7 dual core.

traces in the profiling phase to build perfect profiles, which is hardly the case most of the time. These results show that our mask generator satisfies the security requirement if the SNR on the chip is less than 0.02, meaning that it provides better security than unprotected systems where less than 10,000 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 and application processors for examples. Table 4.4 can as well be computed for other values of SNR using inequality (2.28). 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 10,000 traces (with (2.28)) and then derive the upper-bound on the SNR from the results depicted in Figure 4.9. 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 being said, the typical SNR on application processors has been shown to be mostly of 0.02 which is less than 0.05.

4.3.3.5 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 maximum frequency of 150 MHz. The original design of the CVA6 SoC [ZB19][2], used 66,510 LUTs [Koe21]. 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.

4.3.4 Discussion and analysis of LightMaG

The mask generator proposed 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 mostly implemented [KJJ10]. 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 4.3.3.4 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

[2]The CVA6 is used in the Nanotrust project whose final processor will contain LightMaG. Hence, comparing its resource utilization with that of LightMag is relevant to assess the area overhead that is expected upon the integration of LightMaG.

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 as 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 [Med+10] 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 security assessments carried out on subterranean and summarized in Table 4.1 report cryptanalysis attacks on reduced-round versions of the algorithm. This introduces a concern about the security of our mask generator against cryptanalysis attacks since we only use two rounds of subterranean. However, the attacker will only have access to the inputs of the module and not the outputs (the masks) which are internal to the circuit. This creates challenges for cryptanalysis since most of these attacks require at least an access to the outputs of the attacked module [Sch09].

4-rounds variant of LightMaG. The decision of using two rounds of subterranean allows to generate the masks in one cycle while having a low spatial occupation (400 LUTs). We could try to increase the number of rounds (say to 3 or 4) to have a lower MI. The architecture for the 4-rounds variant and the cumulative distribution functions of each byte of the output mask are given respectively in Figures 4.11 and 4.12. The state and the absorbed block are built like it is the case in the initial proposal. The only changing parameter is the number of rounds that are applied to the state before extracting the mask values. Depending on the whether the data absorption is done after the first, second or third round, it may lead to different distributions of mask and

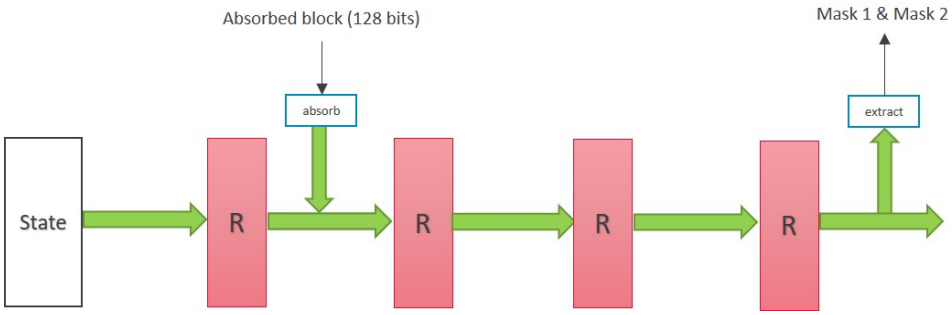


Figure 4.11: Architecture of the four-rounds variant of LightMaG.

possible increase or decrease of entropy. Only one possibility has been illustrated in this section in order to give an insight on what could be expected when increasing the number of rounds and compare the results to those of the initial proposal. This

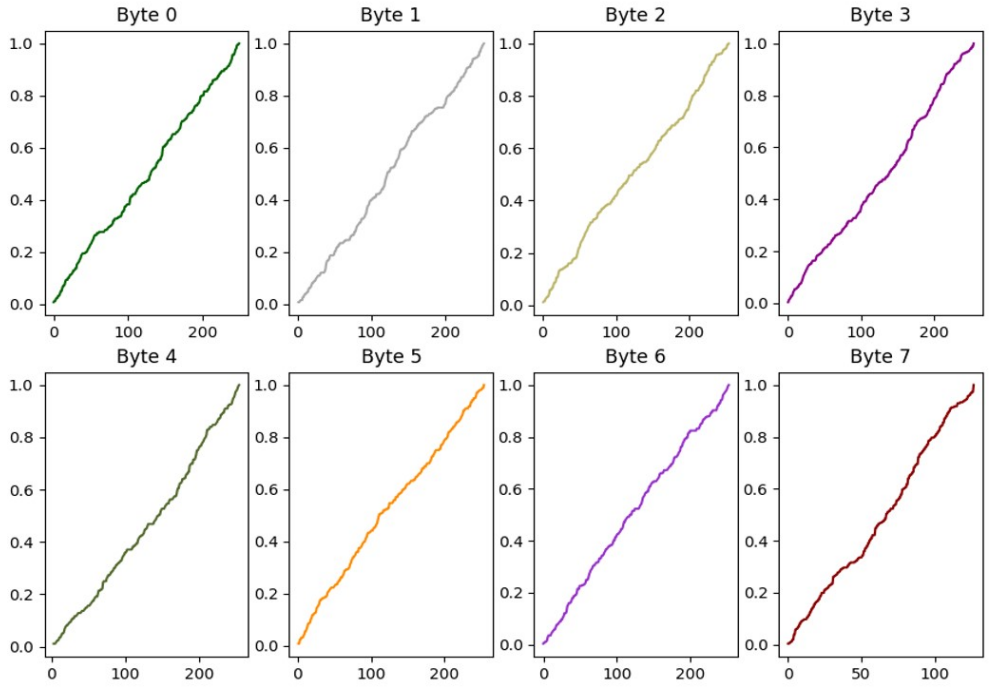


Figure 4.12: Cumulative distribution functions of the four-rounds variant of LightMaG.

analysis shows cumulative distributions approaching the uniform one but in this case, two clock cycles would be needed to generate the masks in addition to 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 4.3.3.4, 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. In addition, with more changing parameters than only the IV, the set of possible values of masks will be broadened leading to the reduction of the bias in the masks distribution. We also recall that according to the memory protection architecture illustrated in Figure 4.2, 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.

Masking integrity tags with LightMaG. The protection of data can also be extended to that of potential integrity tags. This helps ensuring confidentiality and integrity of data in the whole memory hierarchy as it can be used in continuity with authenticated encryption of data from main memory to last level cache. The content of the cache memory is hence obtained with the concatenation of the masked data and its masked integrity tag like follows:

$$Masked\_data||Masked\_tag \leftarrow data \oplus Mask_1||Integrity\_tag(data) \oplus Mask_2 \quad (4.8)$$

In this case we can even get rid of the IV by replacing it with integrity tags of each data. 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 any more. The well-known tradeoff between security and performance shows up here and the designers have to choose to keep the IV and have better security and some memory overhead or replace it with integrity tags 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 size of 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.

4.4 Integrating encryption & masking schemes on a System on Chip

The encryption and masking solutions described in this Chapter are meant to be implemented on the processor of the Nanotrust project. To this extent, we briefly present the various implementation choices that are put in place in order to get an operational system consistent with our countermeasures.

At the reset of the processor, the keys used for encryption (Subterranean) and masking (LightMaG) are initialized (they are fixed in hardware). The user program which is in clear in the Flash memory, is read by the CPU. Since this program is not encrypted, it will only be masked (no decryption is performed) before going to caches and registers. Then, this program is sent to the DRAM, this time being encrypted. At this stage, the user program is encrypted in the main memory. Instead of encrypting the 64-bit data one by one, 16 words (128 bytes) are encrypted at a time to match the size of the cache lines. This allows for an entire cache line to be decrypted at once on each memory access after a cache miss. During program execution, only DRAM accesses involve the encryption engine and masking unit. If the address corresponds to a peripheral, the caches are bypassed. It should be kept in mind that caches only contain masked data along with information to unmask them (Ptr_id, IV). Similarly, DRAM contains only encrypted data and information for decrypting it and checking their integrity (Message Authentication Code).

Store operation. Once a store instruction is received, the data to be stored is masked at the output of the CPU and then stored in the cache with its Ptr_id and IV. Each data in a cache line is stored with its Ptr_id and IV. They allow to regenerate the mask that was used to obtain this data. To store a cache line in memory, the data contained in the line are unmasked at the output of the cache and then encrypted. At the end, we obtain a MAC which ensures the integrity of all the content of the line. The encrypted data, the MAC and some meta data are sent to memory through the interconnect bus.

Load operation. When a load instruction is received, considering that we have a cache miss, all 16 words belonging to the line where the data requested by the CPU is located are sent to the decryption engine at the output of the bus. As the data is decrypted and then masked, the cache line is filled with the masked data, their Ptr_id and their respective IVs. During this process, the cache line is kept invalid. Once the MAC is verified, the cache line is validated and the data requested by the CPU is sent to a register. The mask corresponding to this data will be computed from its Ptr_id and IV. It is then provided to the CPU in another register.

In order to perform the encryption, a random nonce will be needed. For this purpose, an internal register of the encryption module is initialized with a 64-bit random seed at the reset of the SoC. That seed will be used as nonce for the first encryption operation. Let refer to that internal register as r_{intern}. After each decryption, the MACs used will be accumulated in this register through a XOR operation: $r_{intern} = r_{intern} \oplus MAC$. The content of r_{intern} will be used as nonce for the next encryption

4.5 Conclusion

Following the work presented in Chapter 3, the main focus in this chapter has been on mitigating first order side-channel attacks exploiting the power consumption of interconnect bus and registers. A first attempt to protect them separately has been investigated. The emerging solutions was constant weight coding to reduce Hamming weight leakage on interconnect bus and boolean masking for CPU registers. Despite the practicability of these countermeasures, the confidentiality of data wouldn't be ensured because an attacker can target other components such as cache memories to retrieve some data that do not leak in registers or interconnect bus. This led us to considering a systemic approach and protect the entire memory hierarchy using lightweight masking in caches and registers combined with encryption in interconnect bus. The data are encrypted on the interconnect bus, decrypted and masked before entering the caches, and stay masked up to the execute stage of the CPU where computations could be done on masked data.

In order to have a secure masking scheme while keeping low overhead on caches performances, we proposed a lightweight mask generator (LightMaG). This has been made possible after finding that biased masks distribution could also contribute to a secure masking scheme if there is enough noise on the chip. 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. However, the work of Chérisey et al. [Ché+19a] allowed us to assess the number of traces needed for a successful attack. More than 592 K attack traces are needed 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 benefit 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. Integrating the mask generation module in a complete SoC design and run experimentations on a real ASIC circuit would be a good follow-up to our work. However, we believe that the simulations realised in a worst case context already give a relevant insight on the amount of information that can be obtained by a potential attacker.

Résumé du chapitre

Suite au travail présenté dans le chapitre 3, l'accent a été mis dans ce chapitre sur l'atténuation des attaques de canal secondaire de premier ordre exploitant la consommation de puissance du bus d'interconnexion et des registres. Une première tentative pour les protéger séparément a été étudiée. Les solutions émergentes étaient le codage à poids constant pour réduire la fuite en poids de Hamming sur le bus d'interconnexion et le masquage booléen pour les registres du CPU. Malgré la praticabilité de ces contremesures, la confidentialité des données n'est pas garantie car un attaquant peut cibler d'autres composants tels que les mémoires caches pour récupérer certaines données qui ne fuient pas dans les registres ou le bus d'interconnexion. Cela nous a conduit à envisager une approche systémique et à protéger l'ensemble de la hiérarchie des mémoires en utilisant un masquage léger dans les caches et les registres combiné à un chiffrement dans le bus d'interconnexion. Les données sont chiffrées sur le bus d'interconnexion, déchiffrées et masquées avant d'entrer dans les caches, et restent masquées jusqu'à l'étape d'exécution du CPU où les calculs peuvent être effectués sur les données masquées.

Afin de disposer d'un schéma de masquage sécurisé tout en maintenant une faible surcharge sur les performances des caches, nous avons proposé un générateur de masques léger (LightMaG). Cela a été rendu possible après avoir constaté que la distribution de masques biaisés pouvait également contribuer à un schéma de masquage sécurisé s'il y a suffisamment de bruit sur la puce. Une analyse de sécurité des masques produits par LightMaG comparés aux masques uniformes. Les résultats montrent un écart entre les MI obtenus par les deux méthodes. Cependant, le travail de Chérisey et al. [Ché+19a] nous a permis d'évaluer le nombre de traces nécessaires pour une attaque réussie. Plus de 592 K traces d'attaque sont nécessaires pour avoir 99% de chances de récupérer les données non masquées sur une puce avec un SNR de 0,02, ce qui est un grand nombre de traces. Au-delà, bien que la pertinence de l'analyse de sécurité soit donnée pour un SNR de 0.02, nous avons surtout voulu proposer aux concepteurs une plage de SNR ($SNR \leq 0.05$) dans laquelle notre solution répond mieux à leurs besoins tant en termes de sécurité que de performances.

L'avantage de cette solution réside dans la réduction du surcout en taille du processeur (seulement deux tours de Subterranean résultant en 400 LUTs après l'implémentation de la conception sur FPGA) et le fait qu'aucun masque ne sera stocké pour un dé-

masquage ultérieur, à l'exception de l'IV 8 bits. L'intégration du module de génération de masques dans une conception SoC complète et la réalisation d'expérimentations sur un circuit ASIC réel constitueraient une bonne suite à notre travail. Cependant, nous pensons que les simulations réalisées dans un contexte de pire cas donnent déjà un aperçu pertinent de la quantité d'informations qui peuvent être obtenues par un attaquant potentiel.

Chapter 5

Conclusion and perspectives

Here comes the conclusion of this manuscript whose purpose was to describe the research work on how to build a secured memory hierarchy with regards to data confidentiality against first-order side-channel attacks. A summary of the works and contributions is given in this chapter. Then we outline some limitations from which they may suffer. Some perspectives and tracks to improve the presented work are ultimately given as suggestions to the community.

Contents

5.1 Summary of the contributions

The research project whose achievements have been presented in this manuscript is the result of a growing interest in the protection of all the microarchitectural components against side-channel attacks. Much more than the classical approach of protecting cryptographic operations, the main motivation is how to secure any data transfer between the main memory and the CPU against first-order side-channel attacks.

The leakage analysis presented in Chapter 3 provided an insight into the vulnerabilities of the interconnect bus and registers, which could eventually lead to the exposure of the data values passing through them. As outlined and explained in some previously published works [MPW21; MPG05], Hamming weight and Hamming distance leakage models have first been assessed on these components and confirmed on a RISC-V based System on Chip. Their power consumption has been used for these experiments. The observed correlation peaks being quite clear and distinctive on the curves allowed us to confirm the validity of these models without uncertainty. Subsequently, the analysis was taken further by exploring the possibility that data transmitted over the bus could leak directly. An initial analysis using a stochastic model demonstrated the relevance of this idea. A template attack was undertaken with the aim of recovering each byte of these data directly from the power consumption on the interconnect bus. Unfortunately, we were not able to reveal the value of the bytes with a success rate of 99% as 200 attack traces led to a success rate of 70%. A further enumeration step helps recover the 32-bit data with a research cost of $2^{13.2}$ and a probability of 0.96 to find the correct data. These results have been published at PAINE 2021 [Tal+21].

Following this, our first approach was to protect the two components separately (modular approach), as can be seen in Chapter 4. Constant Hamming weight coding has been investigated for use on interconnect bus. This solution has been found inadequate since data value could be exposed directly from the side-channel leakage. On the side of the registers, boolean masking was the retained solution. The goal was to use masks produced by an LFSR to mask each data in registers and store the masked data and the masks in different registers. This would ensure first-order protection of the register but the use of an LFSR would introduce new vulnerabilities. We finally went for a systemic approach which consists in protecting the data through the whole memory

hierarchy. We, therefore, proposed to use authenticated encryption of the memory hierarchy while lightening it at the level of the cache memories via lightweight masking. Many studies propose memory hierarchy encryption as a solution to mitigate eavesdropping on data [UWM19; YXH20]. The common drawback of this countermeasure is the time penalty incurred when decryption is needed, especially in cache memories and registers. To reduce that penalty we proposed to replace encryption in cache memories with lightweight boolean masking which consists of a simple XOR operation between the data and a mask value. This in turn implies the need to store the masks in order to be able to unmask the data later on. The size of the cache memory would be doubled, which will have a severe impact on the area cost of the chip. To overcome the performance, size, and memory overhead issues, we proposed a Lightweight Mask Generator (LightMaG). It produces 64-bit (or 128-bit) masks in one clock cycle at a maximum frequency of 150 MHz on FPGA, without requiring the storage of the masks. Only 8-bit IVs are stored and allow the re-computation of the masks. That frequency is for an application processor such as CVA6 which runs at 50 MHz. A security analysis based on Mutual Information helped show the relevance of the masking scheme in the protection of an application processor within a range of SNR ($SNR \leq 0.05$) which is consistent with the SNR level usually observed on application processors. The resulting architecture and the security analysis realized have been published in MDPI Cryptography journal [Tal+22].

5.2 Limitations and tracks for improvements

In this section, we discuss the various limitations we identified with some detached views of our work while giving perspectives on how to improve it.

The template attacks presented in Chapter 3 were performed on the same device which is used both for the profiling and attack phase. Even if a distinct set of traces were used for the profiling and attack phase, it can be expected that the success rates of these attacks decreases in a real scenario where the attack is done on a different device than the one used for profiling. This issue, referred to as *portability* issue, has been first identified by Elaabid and Guilley [EG12]. They suggested waveform realignment and acquisition campaigns normalization to overcome the portability issue. Some following works linked the reduced performance of the attack to the DC offset which is different for each device and depends on several campaign-dependent parameters such

as temperature and environmental noise [CK14b; CK18]. A compensation of that offset is proposed as a solution by either using a high-pass filter or removing the DC offset after estimating its value. Attacking two ASIC devices using those various proposals, the results of our template attacks could have been more realistic on the abilities of a potential attacker and not extreme results which give the maximum capacity of the attacker.

The retrieval of the data value on the interconnect bus has been demonstrated through a *divide and conquer* approach. We performed template attacks on byte values before estimating the residual computation complexity to retrieve the whole 32-bit data. This could be improved by deepening the *divide and conquer* strategy to attack each bit value of the data with a template attack. With a larger number of traces, it would considerably reduce the residual complexity to find the correct value. Combining this strategy with the exploitation of electromagnetic (EM) radiations of the chip instead of the power consumption may be a relevant alternative. This would allow observation of more localized leakage and may lead to better discrimination of bit values. Various studies on reverse engineering based on side channels take advantage of EM leakage to target each bit of the instruction's opcode [Str+15; Par+19; CLH19]. It leads to more than 90% instructions recognition. With the rising interest in side-channel based disassembling [GB22; Fen+22], applying such techniques to data value recovery on interconnect bus might be a promising track to reduce the residual complexity reported in our study.

On the other hand, the security analysis presented in Chapter 4 highlights the possibility of running a CPA attack on the mask generator (LightMaG) in order to retrieve the 128-bit key. This threat was not included in the analysis leading to the SNR range that has been given and within which our scheme provides first-order security. Let us assume that we have an ASIC circuit implementing the architecture illustrated in Figure 4.2 of Chapter 4. A CPA (or MIA) attack can be done on the mask generator if the attacker stores random data at the same address in the RAM memory in order to keep all parameters unchanged except the 8-bit IV. The second step would be to produce the masks using different combinations of other parameters to unmask the data in caches. Including the protection of the mask generator against side-channel attacks in the protection scheme definition would address that vulnerability. The challenging part of this task will be how to keep a low overhead on the size of the cache

after applying the countermeasure. Using SC-DDPL logic cells [Bel+20] would be an approach to balance the power consumption of LightMaG. It will not be necessary to adapt tools used in the design flow (synthesis, Place and Route ...) according to the changes made in the cells as SC-DDPL cells are built on top of standard cells. Although the needed simplified implementation because of the use of standard cells, the area overhead would be multiplied by 4 as four standard cells are needed to build a SC-DDPL cell. Re-keying strategy [Med+10] could address this issue with reduced area overhead as opposed to dual-rail based implementations. The need for fresh randomness will rise only when changing the key. Most recent, advances in masking schemes such as the Self-Synchronized Masking (SESYM), introduced by Nagpal et al., can also contribute to tackling the area and latency issue [Nag+22]. In their proposal, both asynchronous circuits (Muller C-elements [Mul59]) and dual-rail logic gates (WDDL [TV04]) are used to build a single-cycle masking scheme than holds for first-order and higher-order masking. Using WDDL gates helps avoid glitch issues due to precharge and evaluation phases. The Muller C-elements which are used for synchronization in clock-less circuits are employed here to convert WDDL logic to single-rail enabling the use of dual-rail logic only for critical operations such as non-linear ones. Beyond the reduced area overhead demonstrated by the authors when compared to other low latency masking schemes (GLM [GIB18] for example), their solution is flexible due to the possibility to apply it to other masking schemes. The *SESYM gadgets* developed by the authors can replace the XOR and AND gates without needing additional registers for synchronization. Applying this work to protect the round operations of Subterranean could lead to a secure implementation of the mask generator.

5.3 Long term perspectives

The various experiments and analyses realized in this thesis and presented in this manuscript were driven by the main concern of designing a secure data transfer in the memory hierarchy of a System on Chip. The focus has been made on first-order side-channel attacks using the power consumption of the device with the goal of providing a protection scheme taking into account the system's performances through the use of a lightweight mask generator. However, new research topics, as well as possibilities, can emerge from this work in order to improve its different achievements. Moreover, there is also a place for improvements in order to take the solutions proposed in this

work to the industry. One may think of SoCs and microprocessor providers that might plan to include security of internal data in their devices and especially in the memory hierarchy.

The first line of investigation that has been left open is on how to exploit jointly the Hamming weight and Hamming distance leakages observed on the registers. Indeed, the leakage analysis carried out on the registers shows a simultaneous occurrence of HW and HD leakages. Moreover, the template attacks that followed showed that such information can be recovered with a success rate of 90% for 200 attack traces. Conversely, it is more challenging to retrieve the contents of the registers directly via the same attack. In this context, one could think of exploiting jointly the HW and HD information of such data to improve the success rate of the attack on the content of the registers. Typically, knowing the Hamming weight of the data under attack reduces the set of possible data values to those having the same HW. The Hamming distance information can be exploited as well. Let $\mathcal{W}$ be the set of values that is initialized with all the possible values having the HW of the target data. Assuming that the attacker knows HD between the target data and a constant value C, that has been set by himself, he can reduce the size of $\mathcal{W}$ by removing the values that lead to a HD different from the one that has been observed. Repeating the same experiment with different values for C helps reduce the set $\mathcal{W}$. A potential final reduction could also be done through a template attack on the data value in order to rank the values contained in the set $\mathcal{W}$. The exploitation described above might require many traces acquisition since many trials have to be done typically for exploiting the HD information between the target data and various constant values. Still, it can lead to a better success rate than a template attack that directly targets the data value in the register since 90% success rate has been reported in Chapter 3 when trying to retrieve the HD in registers.

Concerning attacks on the microarchitecture, side-channel power analysis attacks have evolved from their traditional form where physical access to the circuit is required toward remote attacks. Martínez-Rodríguez et al. tried to give a formal definition of remote power analysis attacks by emphasizing that they only apply in cases where no specific equipment (EM probe, oscilloscope...) is used [MDB21]. We can mention particularly the works of Gnad et al. and O'Flynn and Dewar that exploit the noise in Analog-to-Digital Converters (ADC) to build power traces [GKT19; OD19]. Common targets of these attacks are cryptographic algorithms such as AES and RSA. One

could therefore think of broadening the potential targets to take into account the whole microarchitecture. This way, the constraint of the physical access to the circuit will be relaxed and an interconnect bus or a register can be attacked remotely for example. This could make it possible to conceive side-channel attacks on remote IoT devices using the networks through which they are connected. To avoid the need of exploiting ADCs to build power traces for power attacks, some attacks exploit the available interfaces on a chip that expose the CPU's power consumption. This is the case of PLATYPUS attacks that exploit unprivileged access to the Intel Running Average Power Limit (RAPL) interface that provides the power consumption of the CPU [Lip+21]. The authors reported their attacks on x86 architecture (Intel server, desktop, laptop) and were able, for example, to retrieve AES keys from Intel SGX and the Linux kernel, and break the Kernel Address-Space Layout Randomization (KASLR). The instantaneous power consumption is sampled at 20 KHz, which is low compared to the sampling rate of the classical scopes (magnitude of several GHz). Still, they could achieve AES key recovery in Linux kernel within 26 hours. The threat posed by such attacks is the possibility to have access to power consumption information remotely and without needing special equipment to acquire the power leakage. Moreover, these features offered by the chips providers can also be leveraged to target all the microarchitecture as demonstrated by Lipp et al. [Lip+21] when trying to infer secret instructions. These advances raise the need to investigate mitigation techniques to counter such harmful attacks.

Another challenge that needs to be faced is about implementing the countermeasures for use in the industrial context. The goal of most research works on side-channel attacks mitigation is to provide chip manufacturers with sound and performance-aware solutions that can be implemented in their products without incurring a large overhead on the performance. Proving the security level of a proposal is a relevant step but not the least if one wants the proposal to be adopted by industries. Hence, there is a need to identify and discuss the various adjustments that should be done to the system when implementing the countermeasure proposed in this thesis. The memory hierarchy protection mechanism that has been developed is meant to be implemented on a 64-bit RISC-V SoC such as the CVA6 SoC [ZB19]. A hardware module is inserted between the interconnect bus and the last level cache. It contains the encryption/decryption engine and the masking unit. Integrating that module in the SoC requires carefully defining how it will interact with the bus controller on one hand and the cache mem-

ory controller on the other. A brief analysis of the latency incurred by the encryption or decryption led to the choice to encrypt or decrypt 16 words of 64 bits at a time instead of doing it for each data. This generates a unique MAC for the 16 words, which helps reduce the memory overhead compared to the case where each 64-bit word is encrypted individually. There would be 16 MACs instead of one. This also has the benefit of reducing the latency of decryption. The implementation realized on Kintex-7 FPGA has resulted in the need for 40 cycles to perform the decryption of 128 bytes (16×64 bits), which is acceptable compared to the delay of reading data in main memory. This has been made possible because of the possibility to change the size of the cache lines to 128 bytes on the CVA6 processor. It may not be possible on another processor where the size of the cache lines cannot be changed because of some performance purposes. This is the case for some architectures such as ARM and x86 that have cache lines of 64 bytes. A possible track in this situation is to lower the size of data to encrypt/decrypt to match the cache line size at the cost of increased memory overhead (more MACs) but with reduced latency for encryption/decryption. Another adjustment is the support of burst mode transfers as well as single transfers on the interconnect bus. Some communication protocols on the bus allow burst transfers (AHB, AXI), which is not the case for some others such as APB [Ltd]. In the burst mode, according to the way the encrypted data is stored in the main memory, it can be possible to start data decryption as soon as it is read or wait till the end of the transfer to start decrypting it. Indeed, to start the decryption, one needs to absorb the nonce and the associated data in the internal state of Subterranean. For more efficient decryption these data need to be transferred first during the burst transfer. This way, they can be absorbed before the encrypted data arrives. Doing this requires arranging the data in the main memory in order to read the nonce and associated data first, which may not be possible on other processors. It can be done by specifying the order in which data is sent to the main memory upon a store operation in order to be able to read the nonce and the associated data first upon a load operation. Special care should also be taken to detect data transfers between the CPU and peripherals (I/Os) in order to not encrypt, decrypt or mask them as they are not sent to cache memories but directly sent to CPU registers. Support could also be added for those transfers when the peripheral carries sensitive data. That data could be masked by the masking module near the CPU (depicted in Figure 4.2). The main goal being to provide an intrinsically secure application processor, a future work is on embedding an operating system such as Linux that is customized to run on the CPU.

A relevant question to be also addressed is how to manage keys distribution inside the SoC since the encryption engine and masking unit use potentially different keys. Having different keys used at the same time by different modules increases the security as there are many keys manipulated in parallel. Yet, there is a need to implement a mechanism for secure key distribution. Using a cache of keys (a special cache that only contains keys) to store them and letting the OS enable key transfers only in machine mode can be a promising track to cope with this challenge. The few improvements discussed in this paragraph give an insight into the possible adjustment and future works to do for integrating the encryption and masking countermeasures proposed in this thesis.

5.4 Final words

Throughout this effort, various paths and methods have been investigated to ultimately build a secure application processor that does not require any action from a software developer to ensure the confidentiality of data in the memory hierarchy. We hope that the work presented in this manuscript can help the community by providing a better understanding of the capabilities of an attacker in retrieving data on the bus. We also hope that the proposed masking scheme will inspire the community towards increasingly lightweight solutions for better use in the industrial world. The support for integrity preserving solutions enabled by the masking scheme might also pave the way for future efforts to complete the architecture proposed in Chapter 4 to ensure the integrity of data across the memory hierarchy.

Résumé du chapitre

Voici donc la conclusion de ce manuscrit dont le but était de décrire le travail de recherche sur la façon de construire une hiérarchie mémoire sécurisée en ce qui concerne la confidentialité des données contre les attaques par canaux auxiliaires de premier ordre. Un résumé des travaux et des contributions est donné dans ce chapitre. Ensuite, nous en soulignons certaines limites pour ensuite fournir quelques perspectives et pistes pour améliorer le travail présenté, comme suggestions à la communauté.

Tout au long de ce travail, diverses voies et méthodes ont été étudiées pour finalement construire un processeur d'application sécurisé qui ne nécessite aucune action de la part d'un développeur de logiciel pour assurer la confidentialité des données dans la hiérarchie de la mémoire. Nous espérons que le travail présenté dans ce manuscrit pourra aider la communauté en fournissant une meilleure compréhension des capacités d'un attaquant à récupérer des données sur le bus. Nous espérons également que le schéma de masquage proposé inspirera la communauté vers des solutions de plus en plus légères pour une meilleure utilisation dans le monde industriel. Le support des solutions de préservation de l'intégrité rendu possible par le schéma de masquage pourrait également ouvrir la voie à de futurs efforts pour compléter l'architecture proposée dans le chapitre 4 afin d'assurer l'intégrité des données à travers la hiérarchie de la mémoire.

List of Figures

List of Tables

Bibliography

[ABB64] G. M. Amdahl, G. A. Blaauw, and F. P. Brooks. "Architecture of the IBM System/360". In: *IBM Journal of Research and Development* 8.2 (1964), pp. 87–101.

[AFM18] Alexandre Adomnicai, J. J. Fournier, and Laurent Masson. "Masking the lightweight authenticated ciphers ACORN and Ascon in software". In: *Cryptography and Information Security in the Balkans. Springer, Cham* (2018).

[ANS] ANSSI. *L'ANSSI et le BSI alertent sur le niveau de la menace cyber en France et en Allemagne dans le contexte de la crise sanitaire*. https://www.ssi.gouv.fr/actualite/lanssi-et-le-bsi-alertent-sur-le-niveau-de-la-menace-cyber-en-france-et-en-allemagne-dans-le-contexte-de-la-crise-sanitaire/. Accessed: 2022-02-24.

[Arc+06] C. Archambeau et al. "Template Attacks in Principal Subspaces". en. In: *Cryptographic Hardware and Embedded Systems - CHES 2006*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Oct. 2006, pp. 1–14. URL: https://link.springer.com/chapter/10.1007/11894063_1 (visited on 10/29/2019).

[Aro+21] Vipul Arora et al. *A Tale of Two Boards: On the Influence of Microarchitecture on Side-Channel Leakage*. Tech. rep. 905. 2021. URL: http://eprint.iacr.org/2021/905 (visited on 07/12/2021).

[Ars+20] Muhammad Arsath K F et al. "PARAM: A Microprocessor Hardened for Power Side-Channel Attack Resistance". In: *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. Dec. 2020, pp. 23–34.

[AZ21] Mark D. Aagaard and Nusa Zidaric. "ASIC Benchmarking of Round 2 Candidates in the NIST Lightweight Cryptography Standardization Process". In: *IACR Cryptol. ePrint Arch* (2021), p. 49.

[Bal+15] Josep Balasch et al. "On the Cost of Lazy Engineering for Masked Software Implementations". en. In: *Smart Card Research and Advanced Applications*. Ed. by Marc Joye and Amir Moradi. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 64–81.

[Bar+21] Gilles Barthe et al. "Masking in fine-grained leakage models: Construction, implementation and verification". In: *IACR transactions on cryptographic hardware and embedded systems* 2021.2 (2021). Publisher: Ruhr-Universität Bochum, pp. 189–228.

[Bat+11] Lejla Batina et al. "Mutual information analysis: a comprehensive study". In: *Journal of Cryptology* 24.2 (2011), pp. 269–291.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. "Correlation power analysis with a leakage model". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2004, pp. 16–29.

[Bec+13] George Becker et al. "Test vector leakage assessment (TVLA) methodology in practice". In: *International Cryptographic Module Conference*. Vol. 1001. 2013, p. 13.

[Bel+18] Mohamed Ishmael Belghazi et al. "Mutual information neural estimation". In: *International conference on machine learning*. PMLR, 2018, pp. 531–540.

[Bel+20] Davide Bellizia et al. "SC-DDPL: A Novel Standard-Cell Based Approach for Counteracting Power Analysis Attacks in the Presence of Unbalanced Routing". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 67.7 (July 2020). Conference Name: IEEE Transactions on Circuits and Systems I: Regular Papers, pp. 2317–2330.

[BG98] Suresh Balakrishnama and Aravind Ganapathiraju. "Linear discriminant analysis-a brief tutorial". In: *Institute for Signal and information Processing*. Vol. 18. 1998, pp. 1–8.

[Bil+14a] Begül Bilgin et al. "A more efficient AES threshold implementation". In: *International Conference on Cryptology in Africa*. Springer, 2014, pp. 267–284.

[Bil+14b] Begül Bilgin et al. "Higher-order threshold implementations". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2014, pp. 326–343.

[BMV07] Sanjay Burman, Debdeep Mukhopadhyay, and Kamakoti Veezhinathan. "LFSR Based Stream Ciphers Are Vulnerable to Power Attacks". en. In: *Progress in Cryptology – INDOCRYPT 2007*. Ed. by K. Srinathan, C. Pandu Rangan, and Moti Yung. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007.

[BP18] Alessandro Barenghi and Gerardo Pelosi. "Side-channel security of superscalar CPUs : Evaluating the Impact of Micro-architectural Features". In: *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. June 2018, pp. 1–6.

[BST18] Davide Bellizia, Giuseppe Scotti, and Alessandro Trifiletti. "TEL logic style as a countermeasure against side-channel attacks: Secure cells library in 65nm CMOS and experimental results". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 65.11 (2018). Publisher: IEEE, pp. 3874–3884.

[Buc+06] Marco Bucci et al. "Three-phase dual-rail pre-charge logic". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2006, pp. 232–241.

[Buc+10] Marco Bucci et al. "Delay-based dual-rail precharge logic". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 19.7 (2010). Publisher: IEEE, pp. 1147–1153.

[Cha+99] Suresh Chari et al. "Towards Sound Approaches to Counteract Power-Analysis Attacks". en. In: *Advances in Cryptology — CRYPTO' 99*. Ed. by Michael Wiener. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1999, pp. 398–412.

[Ché+19a] Eloi de Chérisey et al. "Best information is most successful". In: *Cryptology ePrint Archive* (2019).

[Ché+19b] Éloi de Chérisey et al. "An Information-Theoretic Model for Side-Channel Attacks in Embedded Hardware". In: *2019 IEEE International Symposium on Information Theory (ISIT)*. July 2019, pp. 310–315.

[CK10] Jean-Sébastien Coron and Ilya Kizhvatov. "Analysis and improvement of the random delay countermeasure of CHES 2009". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2010, pp. 95–109.

[CK13] Omar Choudary and Markus G. Kuhn. "Efficient template attacks". In: *International Conference on Smart Card Research and Advanced Applications*. Springer, 2013, pp. 253–270.

[CK14a] Marios O. Choudary and Markus G. Kuhn. "Efficient stochastic methods: Profiled attacks beyond 8 bits". In: *International Conference on Smart Card Research and Advanced Applications*. Springer, 2014, pp. 85–103.

[CK14b] Omar Choudary and Markus G. Kuhn. "Template attacks on different devices". In: *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2014, pp. 179–198.

[CK18] Marios O. Choudary and Markus G. Kuhn. "Efficient, Portable Template Attacks". In: *IEEE Transactions on Information Forensics and Security* 13.2 (Feb. 2018), pp. 490–501.

[CKN00] Jean-Sébasticn Coron, Paul Kocher, and David Naccache. "Statistics and secret leakage". In: *International Conference on Financial Cryptography*. Springer, 2000, pp. 157–173.

[CLH19] Valence Cristiani, Maxime Lecomte, and Thomas Hiscock. "A bit-level approach to side channel based disassembling". In: *International Conference on Smart Card Research and Advanced Applications*. Springer, 2019, pp. 143–158.

[CLM20] Valence Cristiani, Maxime Lecomte, and Philippe Maurine. "Leakage Assessment through Neural Estimation of the Mutual Information". In: *International Conference on Applied Cryptography and Network Security (ACNS)*. Vol. 12418. Lecture Notes in Computer Science. Rome, Italy, Oct. 2020, pp. 144–162. URL: https://hal.archives-ouvertes.fr/hal-02980501 (visited on 10/18/2021).

[Com17] Information Technology Laboratory Computer Security Division. *Lightweight Cryptography | CSRC | CSRC*. EN-US. Jan. 2017. URL: https://csrc.nist.gov/Projects/lightweight-cryptography (visited on 01/12/2022).

[Cor+12] Jean-Sébastien Coron et al. "Conversion of Security Proofs from One Leakage Model to Another: A New Issue". en. In: *Constructive Side-Channel Analysis and Secure Design*. Ed. by Werner Schindler and Sorin A. Huss. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, pp. 69–81.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. "Template attacks". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2002, pp. 13–28.

[Dae+20] Joan Daemen et al. "The Subterranean 2.0 cipher suite". In: *IACR Transactions on Symmetric Cryptology* (2020), pp. 262–294.

[DFS19] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. "Making masking security proofs concrete (or how to evaluate the security of any leaking device), extended version". In: *Journal of Cryptology* 32.4 (2019). Publisher: Springer, pp. 1263–1297.

[Dog+11] Julien Doget et al. "Univariate side channel attacks and leakage modeling". en. In: *Journal of Cryptographic Engineering* 1.2 (Aug. 2011), p. 123. (Visited on 10/17/2019).

[DS16] François Durvaux and François-Xavier Standaert. "From Improved Leakage Detection to the Detection of Points of Interests in Leakage Traces". en. In: *Advances in Cryptology – EUROCRYPT 2016*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2016, pp. 240–262.

[DSV14] François Durvaux, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. "How to certify the leakage of a chip?" In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2014, pp. 459–476.

[DV83] Monroe D. Donsker and SR Srinivasa Varadhan. "Asymptotic evaluation of certain Markov process expectations for large time. IV". In: *Communications on pure and applied mathematics* 36.2 (1983). Publisher: Wiley Online Library, pp. 183–212.

[DY83] D. Dolev and A. Yao. "On the security of public key protocols". In: *IEEE Transactions on Information Theory* 29.2 (Mar. 1983). Conference Name: IEEE Transactions on Information Theory, pp. 198–208.

[EG12] M. Abdelaziz Elaabid and Sylvain Guilley. "Portability of templates". In: *Journal of Cryptographic Engineering* 2.1 (2012). Publisher: Springer, pp. 63–74.

[Fen+22] Hedi Fendri et al. "A Deep-Learning Approach to Side-Channel Based CPU Disassembly at Design Time". In: *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Mar. 2022, pp. 670–675.

[Gal+16] Neel Gala et al. "SHAKTI processors: An open-source hardware initiative". In: *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*. IEEE Computer Society, 2016, pp. 7–8.

[Gao+20] Si Gao et al. "FENL: an ISE to mitigate analogue micro-architectural leakage". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020), pp. 73–98.

[Gau+17] Michael Gautschi et al. "Near-threshold RISC-V core with DSP extensions for scalable IoT endpoint devices". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.10 (2017). Publisher: IEEE, pp. 2700–2713.

[GB22] Jurian van Geest and Ileana Buhan. *A side-channel based disassembler for the ARM-Cortex M0*. Cryptology ePrint Archive, Paper 2022/523. `https://eprint.iacr.org/2022/523`. 2022. URL: `https://eprint.iacr.org/2022/523`.

[GHR15] Sylvain Guilley, Annelie Heuser, and Olivier Rioul. "A Key to Success". en. In: *Progress in Cryptology – INDOCRYPT 2015*. Ed. by Alex Biryukov and Vipul Goyal. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 270–290.

[GIB18] Hannes Groß, Rinat Iusupov, and Roderick Bloem. "Generic low-latency masking in hardware". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), pp. 1–21.

[Gie+08] Benedikt Gierlichs et al. "Mutual Information Analysis". en. In: *Cryptographic Hardware and Embedded Systems – CHES 2008*. Ed. by Elisabeth Oswald and Pankaj Rohatgi. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008, pp. 426–442.

[GJR11] Benjamin Jun Gilbert Goodwill, Josh Jaffe, and Pankaj Rohatgi. "A testing methodology for side-channel resistance validation". In: *NIST non-invasive attack testing workshop*. Vol. 7. 2011, pp. 115–136.

[GKF18] Robert Giterman, Osnat Keren, and Alexander Fish. "A 7T security oriented SRAM bitcell". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 66.8 (2018). Publisher: IEEE, pp. 1396–1400.

[GKT19] Dennis RE Gnad, Jonas Krautter, and Mehdi B. Tahoori. "Leaky noise: New side-channel attack vectors in mixed-signal IoT devices". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2019), pp. 305–339.

[GM11] Tim Güneysu and Amir Moradi. "Generic side-channel countermeasures for reconfigurable devices". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2011, pp. 33–48.

[GM18] Hannes Gross and Stefan Mangard. "A unified masking approach". en. In: *Journal of Cryptographic Engineering* 8.2 (June 2018), pp. 109–124. (Visited on 01/05/2022).

[GMK16] Hannes Groß, Stefan Mangard, and Thomas Korak. "Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order." In: *TIS@ CCS*. 2016, p. 3.

[GMK18] Hannes Groß, Stefan Mangard, and Thomas Korak. "Domain-Oriented Masking–". PhD Thesis. Graz University of Technology, Austria, 2018.

[GSF14] Vincent Grosso, François-Xavier Standaert, and Sebastian Faust. "Masking vs. multiparty computation: how large is the gap for AES?" In: *Journal of Cryptographic Engineering* 4.1 (2014). Publisher: Springer, pp. 47–57.

[HDD11] Philippe Hoogvorst, Guillaume Duc, and Jean-Luc Danger. "Software implementation of dual-rail representation". In: *COSADE* 51 (2011). Publisher: Citeseer, pp. 24–25.

[HTM09] Neil Hanley, Michael Tunstall, and William P. Marnane. "Unknown plaintext template attacks". In: *International Workshop on Information Security Applications*. Springer, 2009, pp. 148–162.

[HTM11] Neil Hanley, Michael Tunstall, and William P. Marnane. "Using templates to distinguish multiplications from squaring operations". In: *International Journal of Information Security* 10.4 (2011). Publisher: Springer, pp. 255–266.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. "Private Circuits: Securing Hardware against Probing Attacks". en. In: *Advances in Cryptology - CRYPTO 2003*. Ed. by Dan Boneh. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2003, pp. 463–481.

[JIP19] Darshana Jayasinghe, Aleksandar Ignjatovic, and Sri Parameswaran. "SCRIP: Secure Random Clock Execution on Soft Processor Systems to Mitigate Power-based Side Channel Attacks". In: *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. Nov. 2019, pp. 1–7.

[JS17] Anthony Journault and François-Xavier Standaert. "Very High Order Masking: Efficient Implementation and Security Evaluation". en. In: *Cryptographic Hardware and Embedded Systems – CHES 2017*. Ed. by Wieland Fischer and Naofumi Homma. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 623–643.

[KJJ10] Paul C. Kocher, Joshua M. Jaffe, and Benjamin C. Jun. "Cryptographic computation using masking to prevent differential power analysis and other attacks". Publisher: Google Patents. Feb. 2010.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. "Differential power analysis". In: *Annual International Cryptology Conference*. Springer, 1999, pp. 388–397.

[Knu86] DONALDE Knuth. "Efficient balanced codes". In: *IEEE Transactions on Information Theory* 32.1 (1986), pp. 51–53.

[Koc+11] Paul Kocher et al. "Introduction to differential power analysis". en. In: *Journal of Cryptographic Engineering* 1.1 (Apr. 2011), pp. 5–27.

[Koe21] Davy Koene. "Implementation and Evaluation of Packed-SIMD Instructions for a RISC-V Processor". en. In: (2021). URL: https://repository.tudelft.nl/islandora/object/uuid%3Ac4162ff8-9419-4434-852d-c1c3297df808 (visited on 03/08/2022).

[KPP20] Matthias J. Kannwischer, Peter Pessl, and Robert Primas. "Single-Trace Attacks on Keccak". en. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (June 2020), pp. 243–268. (Visited on 06/08/2022).

[LIM19] Fukang Liu, Takanori Isobe, and Willi Meier. "Cube-Based Cryptanalysis of Subterranean-SAE". en. In: *IACR Transactions on Symmetric Cryptology* (2019), pp. 192–222. (Visited on 06/08/2022).

[LIM20a] Fukang Liu, Takanori Isobe, and Willi Meier. *Automatic Verification of Differential Characteristics: Application to Reduced Gimli* (*Full Version*). Cryptology ePrint Archive, Paper 2020/591. https://eprint.iacr.org/2020/591. 2020. URL: https://eprint.iacr.org/2020/591.

[LIM20b] Fukang Liu, Takanori Isobe, and Willi Meier. *Exploiting Weak Diffusion of Gimli: Improved Distinguishers and Preimage Attacks*. Cryptology ePrint Archive, Paper 2020/561. https://eprint.iacr.org/2020/561. 2020. URL: https://eprint.iacr.org/2020/561.

[Lip+21] Moritz Lipp et al. "PLATYPUS: Software-based Power Side-Channel Attacks on x86". In: *2021 IEEE Symposium on Security and Privacy* (*SP*). IEEE. 2021.

[LLG20] Duc-Phong Le, Rongxing Lu, and Ali A. Ghorbani. *Improved Fault Analysis on SIMECK Ciphers*. Cryptology ePrint Archive, Paper 2020/1263. https://eprint.iacr.org/2020/1263. 2020. URL: https://eprint.iacr.org/2020/1263.

[LLQ20] Jingyi Liu, Guoqiang Liu, and Longjiang Qu. "A New Automatic Tool Searching for Impossible Differential of NIST Candidate ACE". In: *Mathematics* 8.9 (2020).

[LLW18] Bozhi Liu, Roman Lysecky, and Janet Meiling Wang-Roveda. "Composable Template Attacks Using Templates for Individual Architectural Components". In: *2018 IEEE 36th International Conference on Computer Design* (*ICCD*). ISSN: 1063-6404. Oct. 2018, pp. 1–8.

[LMW14] Andrew J. Leiserson, Mark E. Marson, and Megan A. Wachs. "Gate-Level Masking under a Path-Based Leakage Metric". en. In: *Cryptographic Hardware and Embedded Systems – CHES 2014*. Ed. by Lejla Batina and Matthew Robshaw. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2014, pp. 580–597.

[LOM10] Yingxi Lu, Maire O'Neill, and John McCanny. "Evaluation of random delay insertion against DPA on FPGAs". In: *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 4.1 (2010). Publisher: ACM New York, NY, USA, pp. 1–20.

[Ltd] Arm Ltd. *AMBA | Specifications*. en. URL: `https://developer.arm.com/architectures/system-architectures/amba/specifications` (visited on 01/08/2021).

[MAD11] Mehrtash Manoochehri, Murali Annavaram, and Michel Dubois. "CPPC: Correctable parity protected cache". In: *2011 38th Annual International Symposium on Computer Architecture (ISCA)*. June 2011, pp. 223–234.

[Man04] Stefan Mangard. "Hardware countermeasures against DPA–a statistical analysis of their effectiveness". In: *Cryptographers' Track at the RSA Conference*. Springer, 2004, pp. 222–235.

[Mat+13] Luke Mather et al. "Does my device leak information? an a priori statistical power analysis of leakage detection tests". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2013, pp. 486–505.

[MDB21] Macarena C. Martínez-Rodríguez, Ignacio M. Delgado-Lozano, and Billy Bob Brumley. "SoK: Remote Power Analysis". In: *The 16th International Conference on Availability, Reliability and Security*. ARES 2021. New York, NY, USA: Association for Computing Machinery, Aug. 2021, pp. 1–12.

[MDP20] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. "A comprehensive study of deep learning for side-channel analysis". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020), pp. 348–375.

[Med+10] Marcel Medwed et al. "Fresh re-keying: Security against side-channel and fault attacks for low-cost devices". In: *International Conference on Cryptology in Africa*. Springer, 2010, pp. 279–296.

[Mes01] Thomas S. Messerges. "Securing the AES Finalists Against Power Analysis Attacks". en. In: *Fast Software Encryption*. Ed. by Gerhard Goos et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2001, pp. 150–164.

[MGH19] Elke De Mulder, Samatha Gummalla, and Michael Hutter. "INVITED: Protecting RISC-V against Side-Channel Attacks". In: *2019 56th ACM/IEEE Design Automation Conference (DAC)*. ISSN: 0738-100X. June 2019, pp. 1–4.

[MMO18] Daniel P. Martin, Luke Mather, and Elisabeth Oswald. "Two Sides of the Same Coin: Counting and Enumerating Keys Post Side-Channel Attacks Revisited". en. In: *Topics in Cryptology – CT-RSA 2018*. Ed. by Nigel P. Smart. Lecture Notes in Computer Science. Springer International Publishing, 2018, pp. 394–412.

[Moh+20] Kamyar Mohajerani et al. "FPGA Benchmarking of Round 2 Candidates in the NIST Lightweight Cryptography Standardization Process: Methodology, Metrics, Tools, and Results." In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 1207.

[Mon+20] Maxime Montoya et al. "Dynamic encoding, a lightweight combined countermeasure against hardware attacks". In: *2020 23rd Euromicro Conference on Digital System Design (DSD)*. Aug. 2020, pp. 185–192.

[Moo+03] Simon Moore et al. "Balanced self-checking asynchronous logic for smart card applications". en. In: *Microprocessors and Microsystems* 27.9 (Oct. 2003), pp. 421–430. ISSN: 0141-9331. URL: https://www.sciencedirect.com/science/article/pii/S0141933103000929 (visited on 04/13/2022).

[MOP08] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards*. Vol. 31. Springer Science & Business Media, 2008.

[Mor+08] Amir Moradi et al. "Information Leakage of Flip-Flops in DPA-Resistant Logic Styles." In: *IACR Cryptology ePrint Archive* 2008 (2008), p. 188.

[Mor+11] Amir Moradi et al. "Pushing the limits: A very compact and a threshold implementation of AES". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2011, pp. 69–88.

[MOW17] David McCann, Elisabeth Oswald, and Carolyn Whitnall. "Towards Practical Tools for Side Channel Aware Software Engineering:'Grey Box'Modelling for Instruction Leakages". In: *26th ${$USENIX$}$ Security Symposium (${$USENIX$}$ Security 17)*. 2017, pp. 199–216.

[MPG05] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. "Side-channel leakage of masked CMOS gates". In: *Cryptographers' Track at the RSA Conference*. Springer, 2005, pp. 351–365.

[MPW21] Ben Marshall, Dan Page, and James Webb. "MIRACLE: MIcRo-ArChitectural Leakage Evaluation." In: *IACR Cryptol. ePrint Arch.* 2021 (2021), p. 261.

[MS06] Stefan Mangard and Kai Schramm. "Pinpointing the side-channel leakage of masked AES hardware implementations". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2006, pp. 76–90.

[MSB16] Houssem Maghrebi, Victor Servant, and Julien Bringer. "There Is Wisdom in Harnessing the Strengths of Your Enemy: Customized Encoding to Thwart Side-Channel Attacks". en. In: *Fast Software Encryption*. Ed. by Thomas Peyrin. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2016, pp. 223–243. ISBN: 978-3-662-52993-5.

[MSS07] Amir Moradi, Mahmoud Salmasizadeh, and Mohammad T. Manzuri Shalmani. "Power Analysis Attacks on MDPL and DRSL Implementations". en. In: *Information Security and Cryptology - ICISC 2007*. Ed. by Kil-Hyun Nam and Gwangsoo Rhee. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, pp. 259–272.

[Mul59] David Muller. "A theory of asynchronous circuits". In: *Proc. Int. Symp. on Theory of Switching, 1959*. Vol. 29. 1959, pp. 204–243.

[Nag+22] Rishub Nagpal et al. "Riding the Waves Towards Generic Single-Cycle Masking in Hardware". In: *Cryptology ePrint Archive* (2022).

[Nas+21] Pascal Nasahl et al. "CrypTag: thwarting physical and logical memory vulnerabilities using cryptographically colored memory". In: *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. 2021, pp. 200–212.

[NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. "Threshold implementations against side-channel attacks and glitches". In: *International conference on information and communications security*. Springer, 2006, pp. 529–545.

[OD15] Colin O'Flynn and Zhizhang David Chen. "Side channel power analysis of an AES-256 bootloader". In: May 2015, pp. 750–755.

[OD19] Colin O'Flynn and Alex Dewar. "On-Device Power Analysis Across Hardware Security Domains.: Stop Hitting Yourself." In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2019), pp. 126–153.

[Par+19] Jungmin Park et al. "Leveraging side-channel information for disassembly and security". In: *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 16.1 (2019). Publisher: ACM New York, NY, USA, pp. 1–21.

[PM05] Thomas Popp and Stefan Mangard. "Masked Dual-Rail Pre-charge Logic: DPA-Resistance Without Routing Constraints". en. In: *Cryptographic Hardware and Embedded Systems – CHES 2005*. Ed. by Josyula R. Rao and Berk Sunar. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 172–186.

[Poz+15] S. M. Del Pozo et al. "Side-channel attacks from static power: When should we care?" In: *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*. ISSN: 1558-1101. Mar. 2015, pp. 145–150.

[PR10] Emmanuel Prouff and Matthieu Rivain. "Theoretical and practical aspects of mutual information-based side channel analysis". In: *International Journal of Applied Cryptography* 2.2 (2010). Publisher: Inderscience Publishers, pp. 121–138.

[PR13] Emmanuel Prouff and Matthieu Rivain. "Masking against side-channel attacks: A formal security proof". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2013, pp. 142–159.

[RD20] Mark Randolph and William Diehl. "Power Side-Channel Attack Analysis: A Review of 20 Years of Study for the Layman". en. In: *Cryptography*

4.2 (June 2020). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 15. (Visited on 04/11/2022).

[Ren+11] Mathieu Renauld et al. "A Formal Study of Power Variability Issues and Side-Channel Attacks for Nanoscale Devices". en. In: *Advances in Cryptology – EUROCRYPT 2011*. Ed. by Kenneth G. Paterson. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, pp. 109–128. ISBN: 978-3-642-20465-4.

[Riv08] Matthieu Rivain. "On the exact success rate of side channel analysis in the gaussian model". In: *International Workshop on Selected Areas in Cryptography*. Springer, 2008, pp. 165–183.

[RO04] Christian Rechberger and Elisabeth Oswald. "Practical Template Attacks". en. In: *Information Security Applications*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Aug. 2004, pp. 440–456. URL: https://link.springer.com/chapter/10.1007/978-3-540-31815-6_35 (visited on 10/29/2019).

[RP10] Matthieu Rivain and Emmanuel Prouff. "Provably Secure Higher-Order Masking of AES". en. In: *Cryptographic Hardware and Embedded Systems, CHES 2010*. Ed. by Stefan Mangard and François-Xavier Standaert. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2010, pp. 413–427.

[RRM07] Alin Razafindraibe, Michel Robert, and Philippe Maurine. "Improvement of dual rail logic as a countermeasure against DPA". In: *2007 IFIP International Conference on Very Large Scale Integration*. IEEE, 2007, pp. 270–275.

[SA08] François-Xavier Standaert and Cedric Archambeau. "Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages". en. In: *Cryptographic Hardware and Embedded Systems – CHES 2008*. Ed. by Elisabeth Oswald and Pankaj Rohatgi. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008, pp. 411–425.

[SBB18] Petr Socha, Jan Brejník, and Matěj Bartik. "Attacking AES implementations using correlation power analysis on ZYBO Zynq-7000 SoC board". In: *2018 7th Mediterranean Conference on Embedded Computing (MECO)*. June 2018, pp. 1–4.

[Sch09] Edward Schaefer. *An introduction to cryptography and Cryptanalysis*. 2009.

[SDG17] Hermann Seuschek, Fabrizio De Santis, and Oscar M. Guillen. "Side-channel leakage aware instruction scheduling". In: *Proceedings of the fourth workshop on cryptography and security in computing systems*. 2017, pp. 7–12.

[SEH20] Olivier Savry, Mustapha El-Majihi, and Thomas Hiscock. "Confidaent: Control FLow protection with Instruction and Data Authenticated Encryption". In: *2020 23rd Euromicro Conference on Digital System Design (DSD)*. Aug. 2020, pp. 246–253.

[Ser+14] Victor Servant et al. "Study of a novel software constant weight implementation". In: *International Conference on Smart Card Research and Advanced Applications*. Springer, 2014, pp. 35–48.

[SGF07] Santosh Srivastava, Maya R. Gupta, and Béla A. Frigyik. "Bayesian quadratic discriminant analysis." In: *Journal of Machine Learning Research* 8.6 (2007).

[She+19] Madura A. Shelton et al. "Rosita: Towards automatic elimination of power-analysis leakage in ciphers". In: *arXiv preprint arXiv:1912.05183* (2019).

[Sin+18] Arvind Singh et al. "Improved power/EM side-channel attack resistance of 128-bit AES engines with random fast voltage dithering". In: *IEEE Journal of Solid-State Circuits* 54.2 (2018). Publisher: IEEE, pp. 569–583.

[SK08] Jörn-Marc Schmidt and Chong Hee Kim. "A probing attack on AES". In: *International Workshop on Information Security Applications*. Springer, 2008, pp. 256–265.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. "A stochastic model for differential side channel cryptanalysis". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2005, pp. 30–46.

[SM15] Tobias Schneider and Amir Moradi. "Leakage assessment methodology". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2015, pp. 495–513.

[SMY09] François-Xavier Standaert, Tal G. Malkin, and Moti Yung. "A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks". en. In: *Advances in Cryptology - EUROCRYPT 2009*. Ed. by Antoine Joux. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 443–461.

[Son+21] Ling Song et al. "Security analysis of Subterranean 2.0". en. In: *Designs, Codes and Cryptography* 89.8 (Aug. 2021), pp. 1875–1905. (Visited on 05/23/2022).

[SR15] Hermann Seuschek and Stefan Rass. "Side-Channel Leakage Models for RISC Instruction Set Architectures from Empirical Data". In: IEEE, Aug. 2015, pp. 423–430. URL: http://ieeexplore.ieee.org/document/7302305/ (visited on 10/24/2019).

[SR16] Hermann Seuschek and Stefan Rass. "Side-channel leakage models for RISC instruction set architectures from empirical data". In: *Microprocessors and Microsystems* 47 (Nov. 2016), pp. 74–81. URL: http://www.sciencedirect.com/science/article/pii/S0141933116000065.

[ST07] Patrick Schaumont and Kris Tiri. "Masking and dual-rail logic don't add up". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2007, pp. 95–106.

[Sta18] François-Xavier Standaert. "How (not) to use welch's t-test in side-channel security evaluations". In: *International Conference on Smart Card Research and Advanced Applications*. Springer, 2018, pp. 65–79.

[Str+15] Daehyun Strobel et al. "Scandalee: a side-channel-based disassembler using local electromagnetic emanations". In: *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 139–144.

[Str19] Vincent Strubel. *DECISION SUR LA SECURITE ALGORITHMIQUE RESIDUELLE (« REMAINING STRENGTH »)*. fr. 2019. URL: https://www.ssi.gouv.fr/uploads/2014/11/anssi-cc-note-23-remaining-strength_v1.0.pdf (visited on 07/27/2021).

[Tal+21] E. Bertrand Talaki et al. "Exposing Data Value On a Risc-V Based SoC". In: *2021 IEEE Physical Assurance and Inspection of Electronics (PAINE)*. Nov. 2021, pp. 1–8.

[Tal+22] Ezinam Bertrand Talaki et al. "A Memory Hierarchy Protected against Side-Channel Attacks". en. In: *Cryptography* 6.2 (June 2022). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 19. (Visited on 04/20/2022).

[TAV02] K. Tiri, M. Akmal, and I. Verbauwhede. "A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards". In: *Proceedings of the 28th European Solid-State Circuits Conference*. ISSN: null. Sept. 2002, pp. 403–406.

[Tez20] Cihangir Tezcan. "Analysis of Ascon, DryGASCON, and Shamash Permutations". en. In: *International Journal of Information Security Science* 9.3 (Sept. 2020). Number: 3, pp. 172–187.

[TGS] Andreas Traber, Michael Gautschi, and Pasquale Davide Schiavone. *RI5CY: User Manual*. URL: `https://www.pulp-platform.org/docs/ri5cy_user_manual.pdf` (visited on 01/08/2021).

[Tim19] Benjamin Timon. "Non-profiled deep learning-based side-channel attacks with sensitivity analysis". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2019), pp. 107–131.

[Tri03] Elena Trichina. "Combinational logic design for AES subbyte transformation on masked data". In: *Cryptology EPrint Archive* (2003).

[TV04] K. Tiri and I. Verbauwhede. "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation". In: *Automation and Test in Europe Conference and Exhibition Proceedings Design*. Vol. 1. ISSN: 1530-1591. Feb. 2004, 246–251 Vol.1.

[UWM19] Thomas Unterluggauer, Mario Werner, and Stefan Mangard. "MEAS: memory encryption and authentication secure against side-channel attacks". en. In: *Journal of Cryptographic Engineering* 9.2 (June 2019), pp. 137–158. (Visited on 01/14/2022).

[Vey+12] Nicolas Veyrat-Charvillon et al. "Shuffling against side-channel attacks: A comprehensive study with cautionary note". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2012, pp. 740–757.

[Wei+21] Yoav Weizman et al. "Low-Cost Side-Channel Secure Standard 6T-SRAM-Based Memory With a 1% Area and Less Than 5% Latency and Power Overheads". In: *IEEE Access* 9 (2021). Conference Name: IEEE Access, pp. 91764–91776.

[WHC18] Ming Ming Wong, Jawad Haj-Yahya, and Anupam Chattopadhyay. "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*. HASP '18. New York, NY, USA: Association for Computing Machinery, June 2018, pp. 1–8. (Visited on 01/18/2022).

[Wu+20] Lichao Wu et al. "On the attack evaluation and the generalization ability in profiling side-channel analysis". In: *Cryptol. ePrint Arch., Tech. Rep* 899 (2020), p. 2020.

[Yan+13] Yingjian Yan et al. "Utility analysis and evaluation method study of side channel information". en. In: *Journal of Electronics (China)* 30.5 (Oct. 2013), pp. 500–508. (Visited on 03/30/2022).

[Yan+17] Y. Yano et al. "Signal-to-noise ratio measurements of side-channel traces for establishing low-cost countermeasure design". In: *2017 Asia-Pacific International Symposium on Electromagnetic Compatibility (APEMC)*. June 2017, pp. 93–95.

[Yao+20] Yuan Yao et al. "Architecture Correlation Analysis (ACA): Identifying the Source of Side-channel Leakage at Gate-level". In: *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. Dec. 2020, pp. 188–196.

[YK20] Shih-Chun You and Markus G. Kuhn. "A Template Attack to Reconstruct the Input of SHA-3 on an 8-bit Device". In: *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2020, pp. 25–42.

[YXH20] Tianyu Yin, Guozhu Xin, and Jun Han. "HPME: A High-Performance Hardware Memory Encryption Engine Based on RISC-V TEE". In: *2020 IEEE 15th International Conference on Solid-State Integrated Circuit Technology (ICSICT)*. Nov. 2020, pp. 1–3.

[ZB19] Florian Zaruba and Luca Benini. "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". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27.11 (2019), pp. 2629–2640.

[Zen04] Erik Zenner. "Cryptanalysis of LFSR-based pseudorandom generators-a survey". In: *None* (2004).

[Zho+20] Haibo Zhou et al. "Practical Key-Recovery Attacks On Round-Reduced Ketje Jr, Xoodoo-AE And Xoodyak". In: *The Computer Journal* 63.8 (Aug. 2020), pp. 1231–1246. (Visited on 06/08/2022).

[ZZ19] Hailong Zhang and Yongbin Zhou. "Template attack vs. stochastic model: An empirical study on the performances of profiling attacks in real scenarios". en. In: *Microprocessors and Microsystems* 66 (Apr. 2019), pp. 43–54. URL: http://www.sciencedirect.com/science/article/pii/S0141933117304532 (visited on 02/10/2020).