Contributions to white-box cryptography : models and algebraic constructions

Pierre Galissant

université
PARIS-SACLAY

Contributions to white-box cryptography: models and algebraic constructions

Contributions à la cryptographie boîte blanche : modèles et constructions algébriques

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580, Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Informatique
Graduate School : Informatique et Sciences du Numérique
Référent : Université Saint-Quentin en Yvelines

Thèse préparée au LMV (Université Paris-Saclay, UVSQ, CNRS)
sous la direction de **Louis Goubin**, Professeur, Université Versailles Saint-Quentin

Thèse soutenue à Versailles, le 21 décembre 2023, par

Pierre Galissant

Composition du Jury

Membres du jury avec voix délibérative

Jean-Sébastien CORON Professeur, Université du Luxembourg	Président
Pierre-Alain FOUQUES Professeur, Université Rennes 1	Rapporteur & Examinateur
Marine MINIER Professeur, Université de Lorraine	Rapporteure & Examinatrice
Nadia EL MRABET Professeure, École des Mines de Saint-Étienne	Examinatrice
Henri GILBERT Responsable du laboratoire de cryptographie, ANSSI	Examinateur
Matthieu RIVAIN Senior Cryptography expert	Examinateur

THÈSE DE DOCTORAT

NNT : 2023UPASG099

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Contribution à la Cryptographie Boîte Blanche : Modèles et Constructions Algébriques

Mots clés : Cryptographie Boîte Blanche - Implémentation - Sécurité Logicielle

Résumé : Du fait de la démocratisation de technologies telles que le paiement par mobile ou l'essor des technologies basées sur la blockchain, les besoins d'implémentations sécurisées dans le modèle boîte blanche d'algorithmes standardisés sont de plus en plus conséquents dans l'industrie. Malgré ces besoins, très peu de nouveaux designs sont proposés dans la littérature. Pour ne pas avoir à utiliser des implémentations aux designs non publics, de nouvelles techniques d'implémentations doivent être proposées et étudiées.

Ce manuscrit commence par un guide pour la cryptographie boîte blanche. Son but est de réviser, préciser ou corriger les modèles boîte blanche, les notions de sécurité et les constructions qui ont émergé dans l'état-de-l'art depuis l'introduction du concept. Notamment, nous clarifions les modèles 'Remote-Access' et 'Hardware-Module' et les contextualisons dans la littérature cryptographique au sens large.

Nous explorons ensuite les implémentations boîte blanche de l'AES en synthétisant tout d'abord les implémentations connues et leurs failles. Nous proposons ensuite une nouvelle implémentation de l'AES pour laquelle nous proposons une analyse de sécurité et un challenge. La dernière partie de cette thèse est consacrée à l'étude de l'implémentation des primitives à base de cryptographie multivariée. Après une introduction succincte à la cryptographie multivariée, nous motivons l'utilisation de la cryptographie multivariée dans le modèle boîte blanche. Nous proposons ensuite la première implémentation boîte blanche de la famille de signature HFE, pour laquelle nous proposons une analyse de sécurité et un challenge. Enfin, pour proposer d'autres idées basées sur la cryptographie multivariée, nous proposons un chiffrement à flot incompressible basé sur QUAD.

Title : Contributions to White-Box Cryptography: Models and Algebraic Constructions

Keywords : White Box Cryptography - Implementation - Software Security

Abstract : Due to the democratization of technologies such as mobile payment or the soaring of blockchain technologies, there is a growing need for secure implementations of standardized algorithms in the white-box model. In spite of this, there are too few secure designs published in the literature. To avoid relying on hidden design implementations to provide any security in the white-box model, more implementations designs and techniques have to be explored.

This thesis begins with a guide to white-box cryptography. Its goal is to revise, precise or correct white-box models, security notions and constructions that have emerged in the state of the art since the introduction of the concept. We notably clarify the Remote-Access White-Box model and the Hardware Module White-Box and contextualize them in the general cryptographic literature.

We then explore white-box implementations of the AES by first synthesizing the known implementations techniques and their flaws, and then proposing a new solution based on polynomial representations, for which we propose a security analysis and a challenge implementation. The last part of this thesis focuses on the implementation of multivariate cryptographic primitives in the white-box model. After introducing succinctly multivariate cryptography, we motivate the study of this branch of public key cryptography in the white-box context. We propose the first implementation technique of the HFE family of signature algorithms, for which we propose an extensive security analysis and a challenge implementation. Finally, to propose other perspectives on multivariate white-box cryptography, we also propose an incompressible stream cipher adapted from QUAD.

Remerciements :

Tout d'abord, je souhaiterais remercier Louis Goubin, mon directeur de thèse, de qui j'ai tant appris et qui m'a soutenu à travers les moments les plus difficiles de cette thèse. Merci de m'avoir donné l'opportunité de démarrer cette aventure et donné goût à la cryptographie. Merci pour toutes les discussions que nous avons pu avoir, qu'elles concernent les mathématiques ou pas. C'est une des choses qui me manqueront le plus.

J'aimerais aussi remercier Marine Minier et Pierre-Alain Fouques d'avoir accepté d'être mes rapporteurs d'avoir accompli leur tâche avec attention et intérêt. Je remercie aussi Jean Sébastien Coron, Nadia El Mrabet, Henri Gilbert et Matthieu Rivain d'avoir fait partie de mon jury.

Comment ne pas remercier les membres de l'équipe CRYPTO de Versailles pour tout ce qu'ils m'ont apporté et d'avoir fait du laboratoire un environnement de vie des plus agréables. Merci donc à Axel, Edouard et Yann d'avoir été là dès le début, de m'avoir montré tant de choses et de m'avoir compris quand il le fallait. Merci Michael pour les discussions toujours passionnantes, passionnées et éclectiques : je sais maintenant que le matelas est l'investissement le plus rentable à faire dans sa vie. Merci Christina pour les bons conseils et le soutien pendant les moments difficiles. Je remercie les membres du bureau 309Amazing, Margot et Rachelle pour m'avoir supporté et rendu ce bureau joyeux et chaleureux. Je remercie aussi les membres du projet SWITECH avec lesquels j'ai pu travailler. Aleksei, Junwei et Matthieu, merci pour les échanges toujours intéressants et le partage de vos connaissances.

Je tiens au passage à non-remercier le Covid, qui a décidé de nuire au début et à la fin de cette thèse avec une attention particulière.

Il est impensable de ne pas remercier les amis qui m'ont soutenu pendant ces années. Merci Doriann et Théo pour les maths, les bières, les jeux, les bières, les randonnées, les bières et le reste. Merci Thomas, Lucien, Stefano et JB pour les grandes bouffées d'air frais en jouant à autre chose que LateX. Merci Jacky, d'être toujours à l'écoute et disponible pour pourfendre des dragons, jouer aux vampires ou construire des cabanes. Merci Guillaume d'être là depuis si longtemps avec un soutien indéfectible et d'être un expert en diversions divertissantes.

Carole et Florent, merci de votre écoute, votre amour et votre soutien à toute épreuve. Merci de m'avoir ramassé en (très) petits morceaux parfois. Merci Paul d'être mon frère, tout simplement. Merci à mes grands-parents pour leur attention constante et leur gentillesse. Merci à mes oncles, mes tantes, mes cousines et mes cousins pour les moments de légèreté ensemble.

Les derniers mots de ces remerciements sont pour toi Claire. Merci pour tout, sans toi je ne serais pas là.

Résumé Étendu :

Le but de la cryptographie est d'étudier la sécurité des communications et de l'information. L'analyse de sécurité d'un cryptosystème dépend d'une clé secrète. Dans les premières étapes de l'analyse de sécurité, le stockage et l'utilisation de la clé par un cryptosystème sont omises par abstraction dans ce que l'on appelle le modèle boîte-noire.

Si ce modèle est satisfaisant pour étudier la sécurité d'un algorithme à travers son comportement entrée-sortie, ce n'est pas le cas lorsque l'algorithme est déployé sur un hardware. En effet, un adversaire peut alors récupérer des données physiques provenant de l'exécution de l'algorithme, comme son temps de calcul ou sa consommation de puissance. Ce modèle d'attaque est appelé modèle boîte-grise. L'étude de la sécurité dans ce modèle est un sujet actif et bien étudié et les preuves de sécurité dans ce modèle sont conditionnelles à certaines propriétés du hardware sur lequel l'algorithme est déployé. La cryptographie boîte blanche a pour but de dépasser cette limite et d'étudier la sécurité des algorithmes lorsqu'aucun hardware sécurisé n'est disponible. Dans ce contexte, l'implémentation software est la dernière barrière entre la clé et un adversaire boîte blanche, qui a un contrôle complet de l'implémentation.

Du fait de la démocratisation de technologies telles que le paiement par mobile ou l'essor des technologies basées sur la blockchain, les besoins d'implémentations sécurisées dans le modèle boîte blanche d'algorithmes standardisés sont de plus en plus conséquents dans l'industrie. Malgré ces besoins, très peu de nouveaux designs sont proposés dans la littérature. Par exemple, le problème initial d'implémentation de l'AES est toujours ouvert après deux décennies et les récents efforts de la communauté cryptographique pour implémenter ECDSA se sont conclus par un échec. Pour ne pas avoir à utiliser des implémentations aux designs non publics, de nouvelles techniques d'implémentations doivent être proposées et étudiées.

Ce manuscrit commence par un guide pour la cryptographie boîte blanche. Son but est de réviser, préciser ou corriger les modèles boîte blanche, les notions de sécurité et les constructions qui ont émergé dans l'état-de-l'art depuis l'introduction du concept. Dans le modèle boîte blanche standard - 'Plain White Box Model' - nous révisons les notions d'incompressibilité, notamment pour les signatures, les algorithmes à clé publique ayant été peu étudiés dans la littérature. Nous exposons ensuite des liens entre des notions classiques de cryptographie, telles que le chiffrement à clé publique et les signatures basées sur l'identité, et les implémentations dans le modèle boîte blanche. Enfin, nous clarifions deux modèles issus de la littérature, les modèles 'Remote-Access' et 'Hardware-Module', et les contextualisons dans la littérature cryptographique au sens large. Notamment, nous montrons à travers des exemples de l'état de l'art que certaines de ces problématiques sont liées à celles rencontrées dans le domaine de l'obfuscation indistinguable.

La seconde partie est consacrée à l'étude des implémentations d'AES en boîte blanche. Nous proposons d'abord une synthèse des techniques de l'état de l'art ainsi que de leur cryptanalyse historique. Nous proposons ensuite une implémentation de l'AES dont le but est de résister aux attaques de type Linear Decoding Attack (LDA) de degré supérieur. Cette technique est basée sur le rationnel d'encodage interne et utilise une représentation polynomiale des calculs. La sécurité de l'implémentation est ensuite estimée contre les attaques de l'état de l'art conditionnellement au degré de certaines relations de décodage. Pour motiver l'étude de cette technique, nous proposons une implémentation challenge. Enfin, nous extrapolons une méthode d'implémentation pour les algorithmes de type Substitution Permutation Network (SPN) appelée Randomized Circuit Knitting (RCK). Pour illustrer sa versatilité, nous montrons comment implémenter l'algorithme PRESENT dans le modèle boîte blanche avec RCK.

La dernière partie de cette thèse est consacrée à l'étude de l'implémentation des primitives à base de cryptographie multivariée. Après une introduction succincte à la cryptographie multivariée, nous motivons l'utilisation de la cryptographie multivariée dans le modèle boîte blanche, notamment pour primitives fondées sur les problèmes d'Isomorphismes de Polynômes (IP). Nous proposons ensuite la première implémentation boîte blanche de la famille de signature HFE. Cette technique repose sur le calcul de multiples affine permettant d'inverser la clé publique d'HFE sans utiliser la trappe traditionnelle. Après avoir étudié l'existence et la calculabilité des multiples affines en fonction de divers paramètres, nous proposons une analyse de sécurité de notre implémentation relative à une conjecture sur une variation du problème IP. Enfin, pour illustrer la flexibilité de la cryptographie multivariée, nous proposons un chiffrement à flot incompressible basé sur QUAD

Contents

General Introduction: Towards White-Box Cryptography

What is Cryptography ?

Our society relies more and more on digital communication for most of its activities. Humans now share infomation in their everyday life through many devices and communications lines. In central places of our society, such as hospitals, banks and social networks, huge amount of sensitive data are stored and exchanged. The security of this information is central for the good functioning of our society and the well-being of individuals. It is therefore crucial to establish reliable methods to secure data: this is the goal of cryptography.

Cryptography is usually defined as the science of protecting communication and information in the presence of a malicious adversary. The two parties communicating are often abstracted as Alice and Bob and their adversary Charlie. For a better flow of this short introduction to cryptography, we will assume that Alice and Bob want to confidentially exchange information over a canal. Using cryptography, Alice and Bob will apply cryptographic algorithms to the data they exchange. Such algorithms are based on mathematical structures that we assume the reader is familiar with.

In a cryptographer's toolbox, Alice and Bob would need an encryption scheme to ensure the confidentiality of their communication. If Alice wants to send a message to Bob, we assume that they share a secret piece of information called the secret key. With the secret key, Alice can encrypt her message with an encryption algorithm that will render it incomprehensible to Charlie. Bob can then decrypt the encrypted message send by Alice to recover the message. In cryptographic jargon, the original message of Alice is called plaintext and the encrypted message the ciphertext. Formally an encryption scheme is a set of three algorithms $\mathbf{G}$,$\mathbf{E}$ and $\mathbf{D}$ such that :

- G is the key generation algorithm. It generates a private-key k.
- If a user knows k, the encryption algorithm E provides a ciphertext $c = E_k(m)$ of a message m.
- The decryption algorithm D uses the key k to decrypt the ciphertext so that:

$$c = E_k(m) \iff D_k(c) = m$$

The goal for us is now to very succinctly overview the methods available to build such algorithms and how to establish their security.

Public Key and Private Key Cryptography

Cryptography has historically been considered in the symmetrical setting, that is, a setting where Alice and Bob both have a secret key that allows them to use cryptographic algorithms. However, Diffie and Hellman showed in 1976 that it was in fact

possible to study cryptography in an asymmetric setting. In this setting Alice owns a private-key and can broadcast its public key. With it, Bob can communicate with Alice, but Alice only can decrypt Bob's message.

The historical context also made these two settings different in the techniques they use to provide security. In the Private-Key setting, algorithms are composed of several rounds of a function that has, paraphrasing Shannon, local diffusion and confusion properties. The DES is the first standardized algorithm of this kind using the Feistel Network structure, but Substitution Permutations Networks (SPN) like the AES also share this school of design. We describe in more details the AES in the second part of this thesis. By design, these algorithms are fast and simple to evaluate on hardware.

The security of private-key algorithms is evaluated by submitting them to the best known attack. The state-of-the-art of attacks against cryptogtaphic is always improved by cryptographers and helps to constantly reevaluate the concrete security of the cryptography deployed.

In the public key setting, algorithms are often based on hard mathematical problems. The well known examples, RSA or ECDSA , are respectively based on finding roots over a RSA group and the problem of finding discrete logarithms over elliptic curves. Their security is proved by a reduction to said mathematical problem, which has often been studied by mathematicians before. The goal of cryptographers is then to break the underlying mathematical problem. Due to the mathematical structure needed to achieve public key functionnalities, asymmetric algorithms are often more costly than symmetric ones.

These two techniques usually work in tandem in our everyday life, in what we call hybrid encryption. The goal of this method is to use fast algorithms for communication while using the practical broadcasting of key provided by public key cryptography. To do so, Alice will ask Bob to send a private key using a public key algorithm. If Alice decrypts the encrypted private-key sent by Bob, they can now use a symmetric algorithm to communicate.

Other Cryptographic Functionalities

From its historical goal to protect communications, cryptography has then evolved into a mature science that tackles many problems related to information security. Modern cryptographic algorithms satisfy properties such as :

- Confidentiality : Alice and Bob can be sure that no information about their communication is revealed by their ciphertexts.
- Authenticity: Alice can be sure that the communication comes from Bob and Bob only.

- Integrity: Alice can be sure that the message send by Bob has not been altered by an adversary on the line
- Non-Repudiation: Alice cannot deny being the sender of messages she has send before
- Non-Falsifiability: Alice cannot tamper commitments she made in the past.

Many other functionalities can by assured by cryptographic algorithms. For some of them, such as Fully Homomorphic Encryption, Indistinguishability Obfuscation or Zero-Knowledge Proofs, their existence is already a marvel. We will encounter some of them in this thesis, and encourage the reader to refer to the sources we provide for more details about them.

As one of the part of this manuscript is dedicated to a signature algorithm, we briefly define it here. The goal of a signature scheme is to allow to Alice to prove to Bob the authenticity of a document she owns. A signature scheme is a set of three algorithms $\mathbf{G}$,$\mathbf{S}$ and $\mathbf{V}$ such that :

- G is the key generation algorithm. It generates a public key k_v and a private-key k_s.
- If a user knows k_s, the signature algorithm S provides a signature $S_{k_s}(m)$ of a message m.
- The verifying algorithm V uses the public key k_v to verify the signature so that:

$$s = S_{k_s}(m) \iff V_{k_v}(s, m) = 1$$

Signature algorithms are by definition public key algorithms, but authenticity can also be ensured in the private-key setting by Message Authentication Codes (MAC), provided that Alice and Bob share the same key.

Evaluating the Security of Cryptographic Algorithms

The security of the communications of Alice and Bob is only relative to the capabilities of Charlie. In cryptography, the role of the attacker is taken by cryptanalysts and the security of algorithms is evaluated and discussed in the literature. The goals of an attacker can be multiple. They can attempt to recover the secret key, decrypt a message or recover partial information about it. Key recovery is obviously the most the potent of them all, and modeling the capabilities of an attacker toward this goal is crucial for a precise evaluation of the security of communication.

A First Security Analysis : the Black-Box Model

The security of an algorithm is usually evaluated by using its input-output behavior in what is called the black-box model. In this model, Charlie has access to the line of communication of Alice and Bob and tries to recover their secret key. This first analysis is crucial as it will determine if an algorithm can be useful at all for cryptography.

Depending on the context, one can allow an adversary to make different types of queries to the encryption algorithm it studies. Among them, the most common ones are :

- Ciphertext Only Attack (COA): This is the most basic model of attack. Charlie only has access to the encrypted data of Alice and Bob to conduct the attack
- Known Plaintext Attack (KPA): In this model, the attacker also knows the plaintext for each ciphertext it recieves. In this case, the goal is to decrypt future messages or recover the key.
- Chosen Plaintext Attack (CPA): Relaxing the previous model, the attacker can now chose particular plaintexts to help their cryptanalysis. It is particularly relevant in the public key setting.
- Chosen Ciphertext Attack (CCA): In this model, Charlie can request decryption of chosen ciphertexts.

If these models of attack grasp various use cases, they abstract the fact that the key used by the algorithm is itself vulnerable to attacks. In our everyday life, cryptographic algorithms are computed by hardware and attackers can have access to more information than the input-output behavior of the algorithm. As the algorithms is implemented on hardware, operations depend on the physical restraints of the device. An attacker can now plant sensors or steal the device to attempt to recover the key.

Cryptography on Hardware: The Grey-Box model

The grey-box model of attack attempts to use the leakage from hardware to conduct a cryptanalysis. A grey-box attacker can use physical information of the device during executions, such as its power consumption, computation time or electromagnetic emanation to conduct its cryptanalysis. Hardware is indeed subject to the laws of physics and the behavior of operations varies depending on the computation it makes. The discovery of such methods by Kocher [74] in 1996 changed the way cryptographers envisioned security and forced the community to provide new solutions against this enhanced adversary.

In the grey-box model, we can classify attacks in roughly two categories : the passive attacks and the active attacks. In passive attacks, the attacker measures physical properties of the hardware during the execution of a cryptographic algorithm. In active attacks, the adversary tries to tamper the hardware to provoke a faulty behavior that will help the analysis.

The most common examples of side-channel attacks are Differential Power Analysis (DPA) and Differential Fault Analysis (DFA). DPA is a passive attack whose goal is to analyze the power consumption to build two distributions that are based on the input-output of the algorithm and key-guesses. If the distributions are distinguishable, the attacker made the correct key-guess. The DFA is an active attack in which the attacker aims to precisely fault the device. If the the fault is precise enough, information about the secret key can be recovered by comparing the correct input-output behavior with the faulty one.

If these attacks were a threat to original implementations, cryptographers have since proposed countermeasures to these attacks. Among them, we can notably find masking countermeasures [36, 61]. The goal of masking is to share a sensitive value into a collection of random ones, so that if all the shares are leaked but one, no information is leaked about the sensitive value. The computations are then made on the shares instead of the real value. For fault attacks, redundancy or noise can be introduced in computations to render the exploitation of faulted encryptions harder.

However, most of the countermeasures to these attacks are supplemented by the fact that hardware can be made a secure environment for computation. The epitome of this rationale is the usage of smart cards for sensible computations such as payment functionalities. Smart cards are a restrained environment where direct access is limited and the structure of the hardware can be hidden. In this context, security can be reasonably proved conditionally the limited leakage of the device.

Towards the White-Box Model

However, the usage seems to move away from the smart card dogma. In the recent years, more and more sensible information are held by less secure devices such as smartphones. One glaring example is the soaring in mobile payments. In such context, the hardware is of limited use to protect information. Indeed, smartphones for instance are way more malleable than smart cards and one can plant malware on it to directly extract the key. Software is now the last barrier between and attacker and a user's key. This context of attack is called the white-box model and is the focus of this thesis.

Summary of the Thesis and Contributions

To help the reader find logic in what follows, we now detail the content of this thesis and our contributions.

Part I: A Guide To White-Box Cryptography

The first part of this thesis consists of a guide to white-box cryptography. Its goal is to revise, precise or correct white-box models, security notions and constructions that have emerged in the state of the art since the introduction of the concept. This part is composed of 4 chapters. In the first chapter, we summarize the state of the art of white-box cryptography and extensively motivate the study of this model. The second chapter formally introduces the Plain White-Box model, the security notions discussed in this model, and the cryptographic notions that are linked to it. The third chapter focuses on two variations of the Plain White-Box model : The Remote-Access White-Box model and the Hardware-Module White-Box Model. We expose their study in the same fashion as chapter 2. Finally the fourth chapter closes this part with a state-of-the-art synthesis of generic white-box attacks.

This part is an extension of the chapter of Embedded Cryptography : P. Galissant and L. Goubin: "Introduction to White-Box Cryptography", in Embedded Cryptography, E. Prouff, G.Renault, M. Rivain and C. O'Flynn editors, Sciences Collection, Wiley, 2023.

Part II: Implementation of AES

The second part is dedicated to the study of white-box implementations of the AES. The first chapter offers a synthesis of the known implementations techniques and the attacks that broke them. The second chapter details our new method of implementation designed to resist algebraic attacks based on polynomial representations and propose a challenge implementation to motivate its study. The chapter 3 is a detailed correctness and security analysis of our construction. Finally, chapter 4 shows that this technique can be applied to any SPN for similar security results.

The chapter 2 to 4 are extracted from our joint publication with Louis Goubin of our implementation technique that is currently in submission. This publication comes with a challenge implementation that can be found at: `https://github.com/p-galissant/RCKAES`

Part III: White-Box Techniques for Multivariate Cryptography

The last part of this thesis focuses on the implementation of multivariate cryptographic primitives in the white-box model. In chapter 1, after succinctly introducing multivariate cryptography, we motivate the study of this branch of public key cryptography in the white-box context. In chapter 3, we propose the first implementation technique of the HFE family of signature algorithm, for which we propose an extensive security analysis and a challenge implementation. Finally, in chapter 4, to propose other perspectives on multivariate white-box cryptography, we also propose an incompressible stream cipher adapted from QUAD.

The chapter 3 is an adaptation of our joint publication with Louis Goubin of our implementation technique, currently in submission. This publication comes with a challenge implementation that can be found at: `https://github.com/p-galissant/WBHFE`. A previous version of this work can be found `https://eprint.iacr.org/2022/138.pdf`. The adaptation of QUAD to the white-box model is original to this manuscript.

Part I

A Guide To White-Box Cryptography

Chapter 1

The White Box Model: History and Context

In this chapter, we introduce the white-box model, first through a short summary of its development in the state-of-the-art. We then explore some of the problems that motivate the study of the white-box model of attack, as well as designing algorithms secure against a white-box adversary. We then formalize the white-box framework we will use for the rest of the thesis and discuss some design rationales that are common in the state-of-the-art and what they offer to the study of the model.

1.1 A Short Story of White-Box Cryptography

The white-box model has been introduced by Chow et al. in 2002 in their paper [38] as a setting to answer the following question: 'What security can cryptographers provide to implementations of cryptographic algorithm that are executed on non-secure - or worse, non-trusted - hardware ?'. In this setting, an attacker can get a complete access to the device it targets, read computed values and modify the implementation to its liking.

The white-box model is a natural extension of the grey-box model that aims to use secure hardware to provide security against an attacker that could use 'side-channel' information that are leaked from said hardware, such as execution time, power consumption or electromagnetic emissions, to get an advantage that he could not get in the black-box model of attack. If providing security in the grey-box model of attack is a hard problem on its own, providing security in the white-box model seems like a whole new ball game. Indeed, if information leaked by a hardware in the grey-box model are noisy and partial, the data gathered by a white-box adversary is exact and each computation made by the execution of the implementation can be retrieved.

For the cryptographer implementing an algorithm in the white-box model, goals are usually weaker than in the black-box model. The minimum for a white-box implementation is to guarantee the integrity of its key, that is, it should not be possible for an attacker to extract the secret key from its implementation. Then, if the implementation is key-extraction resilient, one can also ask the question if the code of the implementation can be compressed, or if the functionality it computes can be inverted. We discuss in more details the security goals of the white-box model in chapter 2.

Even if the transition from the grey to the white-box model of attack was ambitious, Chow et al. proposed in their seminal publication two white-box implementations of two widely used standards, DES and AES. However, their implementations were quickly broken by structural attacks like the BGE attack [15]. Some other candidates with interesting properties were also proposed, but all broken with adaptations of the same attacks, or known general methods.

If the state-of-the-art of implementation was dire at this point, it was only going to be worse. The introduction of Differential Computation Attack (DCA) by Bos et al. at CHES 2016 [28] and Sanfelix et al. at Black Hat Europe 2015 [99] showed that adaptations of attacks in the grey-box model are really efficient to break white-box implementations alike. In this context, two contests, the Whibox 2017 and 2019 [106, 107] contests were launched. As most of the papers before, the goal was to implement the AES so that the key could not be extracted from the implementation, up to the difference that the designs were hidden to the attackers, i.e there was no publication explaining the technique used for each implementation. At the end of both contests, the conclusion is without appeal the same: among hundreds of participants not a single implementation could stand attackers in the white-box model. However, these contests were not without any lessons. The new submitted implementations stimulated attackers to improve or develop new automated attacks. We detail these attacks in chapter 4.

In the meantime, the academic research pushed itself away from the initial problem of implementing the AES. A first line of work was to design algorithms adapted to the white-box model for which it is easier to prove security in the white-box model as well as the black-box model. We discuss shortly this line of work in the present chapter. Another line of work was to tweak the white-box model to get access to stronger security results. The main two example are the use of some hardware functionality or to consider only attacks characterized by malware. We discuss and formalize this variation of model in chapter 3. Lastly, the natural extension of the original AES problem is to implement another standard. The most important attempt is the community attempt at implementing ECDSA in the Whibox 21 [108] contest . As for the AES, all the implementations were broken even with unknown designs (see [6] for instance).

1.2 Motivations to the White-Box Model : Industrial and Societal Needs

Let us first summarize some of the main interests in the study of white-box designs, whether it is for industrial concerns or the individual security of citizens.

1.2.1 Digital Rights Management

One of the first motivations introduced by Chow et al. to study this model of security is the security of DRM (Digital Rights Managements) technologies. The goal of Digital Rights Management is to restrict the access and distribution of copyrighted products such as music, films or software. Avoid the distribution of illegal copies, locking certain functionalities to unlawful users or binding the content to a specific hardware are often the motivations to the use of such techniques.

Usually, content protected by DRM technologies is encrypted by its owner (often a company) and sent to a customer. The customer is then provided a license to use the content. They then have to use a specific device or client to properly use the content if the license is correct and then decrypt the content using a secret key. If the key is stored in a secure device, solutions in the grey-box model of attacks are often sufficient. However, if the key is stored in a software client, a malicious user can attempt to extract the key with the full knowledge of the implementation of the client. The security of such industrial solutions is often based on the obscurity of the construction, with hidden designs implementation. Designing solutions and understanding their security in the white-box model would greatly enhance the confidence in DRM technologies.

1.2.2 Security of Mobile Devices

The security of sensible data on mobile devices is a growing concern in the industry as well as for consumers. The ever growing usage of small devices in the Internet of Things gives rises to fears that personal data and especially cryptographic keys are exposed to private or institutional mistreatment. We motivate the use of white-box cryptography through two contemporary examples: the use of mobile payment and the development of mobile signature.

Mobile payment In the recent years, the payment industry has vested great interests in the extension of the EMV specifications [52] to mobile transactions via Near Field Communication (NFC). In that scenario, the usual contactless smart card is emulated by an NFC-compliant mobile phone or wearable device such as a smart watch. This is referred to as *Host Card Emulation* (HCE).

Unfortunately, mobile platforms do not provide access to a secure element to third-party applications: the SIM card belongs to the telecommunication operator and handset manufacturers keep any form of trusted hardware for their own needs. These emerging applications are therefore facing the challenge of being as secure as a tamper-resistant hardware although being totally based on software. White-box cryptography is currently the only approach to secure these applications and compensate the security risks inherent to common embedded operating systems such as Android.

By hard-coding the EMV keys into the application code itself, white-box cryptography tries to achieve a notion of *tamper-resistant software*. Improving white-box cryptography, in particular for signature algorithms, is therefore a powerful means to promote the rise of mobile payments. Note that the currently used signature algorithm is RSA, and EMVCo published a new specification in 2021, to enable Elliptic Curve Cryptography (ECC) on EMV contact chip payment cards.

Mobile contract signing The eIDAS regulation (EU Reg. N°910/2014) came into force on July 1st 2016 in the (then) 28 member states of the EU, and introduced the *end of the smart card dogma*, in the sense that the signing capability can now be implemented by purely software means as long as they fulfill specific requirements through a qualification procedure. Electronic signatures also became legal evidence that cannot be denied by sovereign authorities or in court.

By relaxing constraints on the signing utility, the eIDAS regulation opens the way to software-only solutions for digital signatures. As a result, a rapid emergence of mobile contract signing is anticipated in the near future. The user experience is straightforward: a contract (or any form of document in that respect) is downloaded on the mobile device, reviewed by the human user, digitally signed locally and the legally binding signature is returned to a back-end server where it is validated and archived.

Now, the need for the signing application to be eIDAS-qualified imposes (depending on the qualification level) to resist security threats pertaining to mobile platforms and most particularly logical attacks where some form of external control is exerted through malware, typically in an attempt to steal the signing key(s) stored on the device. White-box cryptography is the only approach that effectively puts the signing key(s) out of reach of logical attacks on the operating system. Combined with countermeasures against code lifting, white-box cryptography is expected to take a major role in the adoption and deployment of eIDAS-based services in the EU.

1.2.3 Cryptocurrencies and blockchain technologies.

Today, most solutions to store cryptocurrencies and perform transactions on the blockchain are based on a hardware token (USB stick, smart card) or on a mobile application.

While the former provide adequate security, it is inconvenient for the wider usage. For the latter case on the other hand, the security often relies on the operating system of the mobile device and the principle of application sandboxing. Given the wide variety of mobile OS on the field, strictly relying on the operating system to protect critical assets is very hazardous and must be always be avoided.

This raises a strong need for the design of security solutions for pure-software cryptocurrency wallet against all kind of threats such as stealing malwares. In order to protect the cryptographic keys intrinsically involved in cryptocurrencies and blockchain technologies, white-box cryptography is essential. As concerns digital signatures, ECDSA is currently the most used algorithm (for instance Bitcoin and Ethereum do), but alternatives are considered either for other cryptocurrencies or to prepare the post-quantum era.

1.2.4 Mass Surveillance and Malicious Hardware

If white-box cryptography can attempt to solve particular industrial problems, it can also propose solutions to offer better security for all citizens against monopolies and institutionalized attacks.

Malicious Hardware The basis for secure computation is a trustful hardware that correctly computes and does not leak the secret key it contains. But if the hardware is faulty - that is, outputs incorrect results on certain inputs - or at worst malicious, the security of the algorithms deployed on it is compromised.

In this context, the dangers implied by malicious hardware cannot be ignored, especially with the monopoly of few factories and the heavy cost of building new hardware factories. According to the IC Insight 2014 report, 90% of integrated circuits are made by 13 factories and none of them are in Europe. If any of said factories are found to be malicious, the repercussions could be tremendous for a large portion of users all around the world. Imagine the consequences of malicious hardware in the defense or in electrical infrastructures.

To limit the danger of such prospect, cryptographer have proposed methods such as the detection of Trojans [1, 2], or designing algorithms for this special case [32].

Even if these methods are efficient to a certain degree, they all have some disadvantage : the more complex the hardware is, the harder the detection of the Trojan, and using dedicated algorithms greatly diminishes the interoperability of the protected device in its ecosystem. One other solution is to use white-box cryptography. Indeed, if an implementation is secure in the white-box model, regardless of the hardware, a malicious or faulty hardware cannot break its security properties, as it could be seen as an adversary in the white-box model.

Mass surveillance In their 2016 paper [10], Bellare et al. propose solutions to mass surveillance by Advanced Persistent Threats (APTs), i.e. malware implanted in a device that attempts to extract informations, especially secret keys. The start of their reflexion is that APTs are almost impossible to entirely remove from one's system. They argue that leaks from Snowden show that the NSA and other organizations have enough means to plant APTs if they want to.

They propose a solution with the 'Bounded Retrieval Model' which consist in making the keys so big that their extraction cannot go undetected. This goal is somewhat similar to the goal of an 'incompressible' white-box implementation. In this setting, the fact that the entire implementation has to be retrieved to get the functionality ensured by the secret key would hinder attempts of extractions.

The possibility for citizens to protect their secret keys in software implementation with white-box techniques and the open access to white-box implementations that are publicly studied is an interesting track to follow to ensure overall privacy against the ever growing malware and surveillance adversaries. This is especially important when most of the solutions in these models are designed by companies, and hence often copyrighted and unavailable to the majority.

1.2.5 Ecological Concerns

A certain advantage of security through software is a diminishing reliance on dedicated hardware for security in our everyday life. Without a strong reliance on hardware for dedicated tasks, discussions at the societal and consumerist level can be started to elect more reasonable consumption of hardware relatively to our needs and the stress they put on our ecosystem. That is not to say that abandoning all hardware is a good solution, but having the opportunity to use secure software can open new discussions.

One of the striking examples of how hardware production is detrimental to our ecosystem is the recycling and production of secure cards such as payment cards. We take this example as it is one of the most common, but it applies to other dedicated secure hardwares. Mainly composed of PVC and rare minerals such as copper, palladium, nickel, silver and gold, payment cards are often not recycled due to security concerns and the hardness of recycling components. According to the 2020 report of the french 'Observatoire de la sécurité des moyens de paiement' (Observatory for the security of means of payment), there are currently 94 millions payment cards in circulation, more than one per citizen, which represents about 400 tons of non-recycled wastes. Considering that the mining of rare mineral is one of the most polluting industry, with the most diminishing returns as reserves expire, exploring solution to this problem seems of prime interest.

If the recycling of hardware is certainly an excellent solution to extend the life cycle

of metals and plastics alike, not using dedicated hardware seems like a more radical and efficient solution, that can be discussed in our current technological ecosystem. As the 'Observatoire de la sécurité des moyens de paiement' remarks, we have seen a surge in mobile payment post-COVID crisis, which supports the idea that consumers as well as industrial are ready for such a transition. While we note that it is of course due the mainstream use of smartphones, that contains more rare minerals than smartcards, the social cost of abandoning smartphones would be way heavier than changing a mean of payment.

The study of security in the white-box model can only put more trust in software-based solutions and incline the public opinion to change their usage or discuss such changes.

1.3 Formalization of the Context

The goal of this section is to set a formalism to study the white-box model. We first set the boundaries for programs and then formalize white-box compilers.

1.3.1 Programs

As cryptographic algorithms are implemented, they leave their realm of abstraction to become grounded programs. Programs are very formally defined in a language theoretic sense and interpreted in the context of a programming and execution model. We will however not need this level of detail for our uses. Cryptographic algorithms are usually a composition of operations on mathematical structures and often agnostic to the programming language. This is the case for most of the white-box designs proposed in the literature.

For the following, we assume that a program is a sequence of operations over data that can be implemented by any programming language such as C, Java or Python. We describe programs as operations over mathematical structures and do not specify them for any programming language. Programs can be executed, copied or modified at will, which separate them from oracle calls to an algorithm. This definition is consistent with the white-box model in which an adversary can interrupt computations at will, read memory at any point, modify values in the execution and so on.

For any program P, we note $Size(P)$ the size of its code ignoring the constant overhead of its programming language. This definition is consistent with the attempt to propose incompressible structures regardless of the language they are implemented on. For any input x of P, we note $Time(x)$ its running time on x measured in number of elementary operations on data structures, similarly to the definition of complexity of an algorithm. We note $Time(P)$ if the running time is identical for all inputs.

Remark: We could be tempted to define $Size(P)$ to be the minimal size of code that implements P, but as we deal with white-box implementations, this minimal size is often close to the size of the key and hence not a good measure of the size of a program that wants to hide its key.

1.3.2 White-Box Compiler

The goal of a white-box designer is to implement a cryptographic algorithm, that is defined by a description of the algorithm and its secret key, into a secure implementation in the white-box model. Depending on use cases, these algorithms can be encryption schemes or signature algorithms. To do so, a designer can draw randomness to 'inject' into the implementation and try to hide the key or use the structure of the algorithm itself to bury the key. To synthesize this approach, we follow [43] and use the framework of white-box compiler.

Definition 1 (White-Box Compiler). *A white-box compiler $\mathcal{C}$ is probabilistic algorithm that on the input of a keyed-algorithm A and a key k, outputs an implementation of A_k noted $\mathcal{C}_A(k)$ that aims to achieve security properties in the white-box model.*

The Grail for a cryptographer would be to propose a compiler for entire classes of algorithms that can be applied to an algorithm without concerns for its structure. In practice, white-box compilers are dedicated to attempt to white-box a particular algorithm. The security notions that a compiler must achieve are discussed in detail in chapter 2 and 3.

The framework of white-box compiler allows to consider a setting where two or more implementations come from a same compiler and can be used to mount more complex attacks based on the similarities between the implementation. The author of [43] define these attacks as 'recompiling attacks'. While they certainly need to be addressed, there is no mention of these attacks in the literature : this is probably due to the fact that regular attacks on one implementation already break most white-box candidates. As recompiling attacks are not necessary to understand the state of the art of white-box cryptography, we do not study them further. For this reason it is common to find security claims for an implementation instead of a compiler.

1.4 Hidden and Open Designs, Chosen or Fixed Algorithms

As the seminal problem of implementing the AES in the white-box model quickly appeared to be a hard problem, cryptographers turned themselves into variations of

the initial problem to better understand the white-box model of attacks. While we thoroughly discuss the variations of the initial model of security, we want here to discuss two variations in the practice of white-box cryptography. The first one is the degree of freedom that hidden designs of implementation allow for the security of implementations. The second one is to build algorithms designed for white-box instead of working in implementations of standardized ones.

1.4.1 What do hidden designs offer ?

One of the key principles of modern cryptography is Kerckhoff's Principle. In its 1883 article 'La Cryptographie Militaire', Kerckhoff states in the jargon of its time that if a cryptographic machine should fall to the hands of the ennemies, the security of the rest of the communications should not be impacted. In modern terms, cryptographers want that cryptographic algorithms to be secure, even if everything is known about it, except its secret key. While cryptography has drastically changed since the late 19th century, this rationale of design is still core to modern cryptography. In the white-box context, the natural way to extend Kerckhoff's Principle is to publicly describe the compiler that produces our white-box implementation, even if there are secret random data drawn by the compiler to produce the final implementation.

If this rationale of design was followed by the seminal paper of Chow et al., the idea of working with implementations with hidden design began to emerge to allow degrees of freedom to designers. For instance, the biggest contests for candidate white-box implementations, Whibox17 and 19, were working with hidden designs. While this idea might seem daring, this change of rationale is supported by multiple factors. First, the hardness of the initial problem,that is implementing the AES, did not get many advances with usual cryptographic methods. Some of the best proposed solutions such as [16], relied on obscurity of the code to get concrete security, and code obfuscation techniques were seen as an opportunity to patch what state-of-the-art implementation techniques were lacking. Secondly, a setting of hidden design is closer to industrial concerns and can help bridge the gap between communities. Understanding the security of white-box products on the market is essential for the overall security, especially when very few proven solutions exist, and the 'white-box' tag is used as a marketing advantage.

However, from this change of paradigm, it seems that very little security can be gained. If we take WhiBox17 and 19 as a case of study, the observation is clear: among the hundreds of proposed designs, none did stand in the end. The reports of attacks such as [62] show that after a meticulous reverse engineering work, the power of automated attacks often trivialise a key-recovery from the implementation.

1.4.2 Choosing the algorithm to have better implementations

The holy grail of white-box cryptography would be to have secure white-box implementation of popular standards algorithms like the AES, RSA or ECDSA. However, proposing secure implementations of said algorithms has been found to be a hard problem, with very few candidate solutions. One of the fruitful alternatives for cryptographers has been to change the algorithms to implement. If the most commons algorithms are not well-suited for certain context or problem, it is common to design new algorithms adapted for a specific context. One can note algorithms designed with minimal multiplicative depth in mind for FHE or algorithms that are designed easy to mask to improve their security against side-channel attacks. In the same fashion, it is natural for cryptographer to design algorithms tailored for the white-box model.

The line of work dedicated to a better understanding of the notion of 'incompressibility' or 'big-key' ciphers in the white-box is the most notable one. In [23] and [57], the authors describe symmetric encryption algorithms that are proven to be secure in the white-box model. In [43], the authors transform the public-key algorithm RSA into a private-key algorithm and prove an implementation of this algorithm to be secure in the white-box model. The proven security of these implementation is a striking advantage of these methods, compared to all the broken candidates implementations of the AES.

While it does provide practical secure white-box implementation for these chosen algorithm, the designs exposed in this line of work did not help to get a better grasp of what was needed to achieve the implementation standardized algorithms. This gap between algorithms designed with the white-box model in mind, and the common standards has yet to be bridged. Furthermore, the success of chosen algorithm constructions in the white-box model strongly highlights that designing an implementation of a standard that is not inherently adapted to white-box model - such as the AES to name but one - is a way harder problem than choosing algorithms adapted to it.

Chapter 2

The Plain White-Box Model

In this chapter, we study the plain white-box model. We contrast it with models that differ from Chow et al. initial model and that are motivated by specific use cases. We describe them in chapter 3. After defining its boundaries, we describe security notions that are relevant to this model, in the public-key and private-key setting. We then discuss of links between white-box cryptography and other notions from the state of the art of cryptography, including identity based cryptography and cryptographic obfuscation.

2.1 What is it ? For What use cases ?

The plain white-box is the model introduced by Chow et al. in their seminal paper [38]. It studies the security notions that cryptography can offer when the program computing a cryptographic algorithm is the last barrier before the attacker. This model is the most demanding since an attacker has the complete control over an implementation and each of its executions. Hence, the security notions that can be studied in this model are way weaker than the ones studied in the black-box model.

However, the notions in the white-box model are more resilient to a bad usage by its user: to void the security of an algorithm with security only in the black-box model, one only needs a user to store its key negligently, as for an algorithm implemented with security in the white-box model in mind, the security property persists even through misuse. As this model is the most generic, properties studied in this model provide security in any of the context motivated in the previous chapter.

If the model is rather simple to describe, defining the white-box attacker and understanding its power is a rather difficult problem. We dedicate the entire chapter 4 to the automated attack such attacker can perform.

The first intuition about implementations in the white-box model is that they should

guarantee the confidentiality of its key. This folklore notion has been formalized has 'unbreakability'. If achieving unbreakability is the main line of work for the implementation of the AES, it is clear that this notion is not satisfactory enough. Indeed, if the key cannot be extracted by an attack, he can then attempt to steal the entire implementation. To mitigate this, the implementation can be made as big as a desired threshold, and should be able to be compressed by the adversary. This notion is aptly known as 'incompressibility'.

Similarly to the black-box model, an adversary can attempt to inverse the implementation of our cryptographic algorithm, expect this time, he has access to the implementation instead of having an oracle access to the functionality. If a program, that is computing a bijection is hard to invert with the knowledge of its implementation, we say that it is 'one-way'.

In contexts similar to DRMs, one can also want to find the original owner of a stolen program. If an implementation can be bound to the user it has been delivered to, we say that the implementation is 'traceable'. As this notion is not one of the most studied, we do not detail it here ([43] for more details).

Remark on the Denomination: We add the qualifier 'plain' to the white-box model to distinguish it from variations that have been studied in the literature and to avoid any confusion that can come up when comparing models, security properties that can be achieved and techniques that are available . We study these variations of the plain white-box model in chapter 3.

2.2 Security Notions

The original work of Chow et al. did not formally introduce security notions. We adapt definitions from [43] or in the same fashion as we believe they are the closest to practical concerns and the goals of the state-of-the-art of implementations. While there have been other attempts at formalizing security notions for white-box cryptography, such as [100] or [57] - for often stronger notions than the ones we will discuss - we believe that the ones we propose here are the closest to the problems implementation designers have an that they are formal enough to be interesting for more theoretical designs.

2.2.1 Unbreakability

The first intuitive requirement for a white-box implementation is that is provides the privacy of the secret key embedded in the program. This the first developed notion by Chow et al. as for them, the choice of the implementation is the last standing line of defense against an attacker. This notion of resistance to key-extraction is formalized in

the state-of-the-art by "key-extraction resistance" or "unbreakablity". Unbreakability is the bare minimum a white-box implementation should satisfy.

If designing implementations that resist key-extraction was a seminal goal of white-box cryptography, especially of the AES as noted before, it is important to note that no publicly published AES implementation satisfies the unbreakability property. The two implementations contests [106] and [107] have been a community effort dedicated to implementing an unbreakable AES, which was not a success. It is hence a reasonable basic goal to achieve for new designs.

We separate definitions in the private-key setting and in the public-key setting to highlight the differences between the two contexts and what they imply for the obtained implementations.

In the Private-Key Setting

The most common case of study of unbreakablilty is the symmetric key encryption algorithm, so we describe unbreakability in this context. Note that it can be adapted for any symmetric algorithm. For this paragraph, let $\mathcal{E}$ is an encryption scheme with encryption algorithm E

Let us describe the game for unbreakability of a compiler $\mathcal{C}_E$:

- Draw at random a key k in keyspace K
- The adversary $\mathcal{A}$ gets the program $\mathcal{C}_E(k)$ from the compiler
- The adversary $\mathcal{A}$ returns a key guess $\hat{k}$ in time τ knowing $\mathcal{C}_E(k)$
- The adversary $\mathcal{A}$ succeeds if $k = \hat{k}$

Definition 2. *Let E be an symmetric encryption algorithm, $\mathcal{C}_E$ a white-box compiler and let $\mathcal{A}$ be any adversary. We define the probability of the adversary $\mathcal{A}$ to succeed in the unbreakability game by:*

$$Succ_{\mathcal{A},\mathcal{C}_E} := \mathbb{P}[k \leftarrow K; \mathcal{P} = \mathcal{C}_E(k), \mathcal{A}(\mathcal{P}) = \hat{k}; k = \hat{k}]$$

We say that $\mathcal{C}_E$ is (τ,ϵ)-unbreakable if for any adversary $\mathcal{A}$ running in time τ, $Succ_{\mathcal{A},\mathcal{C}_E} \leq \epsilon$.

When we want to precise that the compiler is unbreakable against a certain class of attacks noted ATK, we note that the compiler is (τ,ϵ)-UBK-ATK.

Remark: Contrary to what can be done studying when asymptotic security, we do not set $\mathcal{A}$ to be a polynomial adversary depending on a security parameter λ and hope that

ϵ is exponentially small in λ, as we are interested in concrete security for our chosen parameters.
Remark: It is often the case that, due to fact that an adversary only get one implementation from one compiler and that not everybody uses the compiler formalism, we often say that an implementation is unbreakable by abuse.

The 'one-way function trick' The first remark a mischievous reader would make is that it is easy to transform an encryption algorithm - or any keyed cryptographic primitive - into another encryption algorithm for which an unbreakable implementation exists, provided one way function exists. As it is indeed the case, let us describe such a method. To do so, let us consider an encryption algorithm $\mathcal{E}$ = (M,C,K,Setup,Enc,Dec) and a one-way function $f : K \to K$. We define a new encryption algorithm $\mathcal{E}'$:

- The plaintext, ciphertext and key space are identical to $\mathcal{E}'$

$$M = M', C = C', K' = K$$

- The algorithm Setup' ,on the input of 1^k, outputs a call of Setup on 1^k :

$$Setup'(1^k) = Setup(1^k)$$

- The algorithms Enc' and Dec' are identical to respectively Enc and Dec except they operate on the key $f(k)$:

$$C = Enc'_k(M) = Enc_{f(k)}(M) \iff M = Dec'_k(C) = Dec_{f(k)}(C)$$

If the function f is one-way, it is then trivial that $\mathcal{E}'$ = (M',C',K',Setup',Enc',Dec') admits an unbreakable implementation. Indeed, by semantic definition, the output of Setup' k is the key of the algorithm, and $f(k)$ is sufficient to encrypt and decrypt. As a consequence, an unbreakable implementation of Enc' can be made with the value $f(k)$ and a regular implementation of Enc calling the value $f(k)$.

While this remark makes the unbreakability security property pretty trivial to achieve from the design of encryption algorithms, it also underlines the fact white-box construction have to be evaluated and compared in regard to a reference implementation. It also helps to get some perspectives on the state ofthe-art of implementations against stronger security notions that can be found in the literature : there isn't any unbreakable implementation of usual standardized cryptographic algorithm such as AES or RSA in the literature.

This remark however highlights that for practical as well as for theoretical concerns, the unbreakability notion is not enough. It is obvious that if the security notion was

enough for practical concerns standards could be adapted to just use the 'one-way function trick' to have unbreakable implementations.

In the Public-Key Setting

As we motivated in chapter 1, public-key algorithms and especially signatures in the white-box model are a topic of their own interest due to the technologies they facilitate. However, very few research on this topic has been made, theoretical and practical aspects as well. We adapt here the usual unbreakability notion to signature algorithms. The main difference here is that an attacker has the knowledge of the public key, i.e the verification algorithm, in addition to the implementation. For the rest of the paragraph, $\mathcal{S}$ is a signature algorithm.

Let us describe the game for unbreakability of a compiler $\mathcal{C}_{\mathcal{S}}$:

- Draw at random a key k in private key-space $K_{\mathcal{S}}$
- The adversary $\mathcal{A}$ gets the program $\mathcal{C}_{\mathcal{S}}(k)$ from the compiler
- The adversary $\mathcal{A}$ returns a key guess $\hat{k}$ in time τ knowing $\mathcal{C}_{\mathcal{S}}(k)$ - and the verifiying algorithm V
- The adversary $\mathcal{A}$ succeeds if $k = \hat{k}$

Definition 3. *Let $\mathcal{S}$ be an asymmetric signature algorithm, $\mathcal{C}_{\mathcal{S}}$ a white-box compiler and let $\mathcal{A}$ be any adversary. We define the probability of the adversary $\mathcal{A}$ to succeed in the unbreakability game by:*

$$Succ_{\mathcal{A},\mathcal{C}_{\mathcal{S}}} := \mathbb{P}[k \leftarrow K; \mathcal{P} = \mathcal{C}_{\mathcal{S}}(k), \mathcal{A}(\mathcal{P}) = \hat{k}; k = \hat{k}]$$

We say that $\mathcal{C}_{\mathcal{S}}$ is (τ,ϵ)-unbreakable if for any adversary $\mathcal{A}$ running in time τ, $Succ_{\mathcal{A},\mathcal{C}_{\mathcal{S}}} \leq \epsilon$.

White-Box Signatures: a 'Double Trapdoor'

It is common to have signature built from a secure trapdoor one-way function (OWF) and extended into complete signature algorithm following the so-called 'hash-and-sign' paradigm. This is the case for RSA, with FDH or PSS and for instance. It is natural to study the white-box properties of these functions and how they extend to their corresponding signature schemes. In this case, what is interesting is the implementation of the inversion of f, for which a secret is needed.

For the rest of the paragraph, f is a secure trapdoor one-way function $f : \mathcal{X} \times \mathcal{K} \rightarrow \mathcal{Y}$ and $\mathcal{C}_{f^{-1}}$ is a white-box compiler for the inversion of this function. We define

unbreakability and incompressibility, similarly as we did for signatures before. Let us describe the game for unbreakability for the compiler $\mathcal{C}_{f^{-1}}$:

- Draw at random a key k in private keyspace $\mathcal{K}$
- The adversary $\mathcal{A}$ gets the program $\mathcal{C}_{f^{-1}}(k)$ from the compiler
- The adversary $\mathcal{A}$ returns a key guess $\hat{k}$ in time τ knowing $\mathcal{C}_{f^{-1}}(k)$
- The adversary $\mathcal{A}$ succeeds if $k = \hat{k}$

Definition 4. *Let f be a secure trapdoor one-way function, $\mathcal{C}_{f^{-1}}$ a white-box compiler of its inversion and let $\mathcal{A}$ be any adversary. We define the probability of the adversary $\mathcal{A}$ to succeed in the unbreakability game by:*

$$Succ_{\mathcal{A},\mathcal{C}_{f^{-1}}} := \mathbb{P}[k \leftarrow K; \mathcal{P} = \mathcal{C}_{f^{-1}}(k), \mathcal{A}(\mathcal{P}) = \hat{k}; k = \hat{k}]$$

We say that $\mathcal{C}_{f^{-1}}$ is (τ,ϵ)-unbreakable if for any adversary $\mathcal{A}$ running in time τ, $Succ_{\mathcal{A},\mathcal{C}_{f^{-1}}} \leq \epsilon$.

We will then study signature schemes that are built 'on top' of the inversion of a trapdoor OWF, which a general term to qualify a lot of already existing techniques such as FDH or PSS.

Definition 5. *We say that a signature algorithm S is built "on top" of a trapdoor one-way function f if it can be decomposed into two algorithms:*

- *Algorithm A : On the input of an element in $Im(f)$, outputs one of its premimages*
- *Algorithm B : On the input of a message, outputs a signature $S(m)$ of a message m. The algorithm B can only perform computations using the message m, public data, data drawn at random and calls to A.*

With such algorithms we can reduce the unbreakability of the signature algorithm to the unbreakability of the trapdoor OWF.

Proposition 1. *Let f be a secure trapdoor one-way function. Let $\mathbf{WBf^{-1}}$ be any (τ, ϵ)-unbreakable implementation of f^{-1}. If S is a signature algorithm built "on top" of f (Definition 4) then there exist a (τ, ϵ)-unbreakable implementation of S.*

Proof: If $\mathbf{WBf^{-1}}$ is such implementation (τ, ϵ)-unbreakable, build the implementation of S by replacing any call to f^{-1} by a call to $\mathbf{WBf^{-1}}$ and note this implementation $\mathbf{WBS}$. Now, if any attacker $\mathcal{A}$ breaks the (τ, ϵ)-unbreakability , one can build an attacker $\mathcal{A}'$ breaking $\mathbf{WBf^{-1}}$ by simply building $\mathbf{WBS}$ and running $\mathcal{A}$. This is absurd by unbreakability of $\mathbf{WBf^{-1}}$.

For this category of algorithm, producing a white-box implementation a signature algorithm is very similar to introducing a second trapdoor to the function f. Indeed, if the first trapdoor is compute f^{-1} wit the secret-key, the implementation can be seen as the second trapdoor intermediate trapdoor that allows to compute f^{-1} without revealing the secret-key. This draws similarities with identity-based encryption we describe later on.

Remark : Note that this property is not trivially generalized to incompressibility. Indeed, composition of incompressible function do not always behave nicely.

2.2.2 Incompressibility

When a white-box implementation is unbreakable, one can try to retrieve the complete implementation to circumvent the fact that the key can't be extracted. While implementations can be hardware bound, it is not possible in the plain white-box model. To mitigate this, the implementation can be made big enough and incompressible to dissuade attackers to attempt a code-lifting attack depending on the context. This goal is similar to big-key ciphers of Bellare et al. [10] but on the implementing side rather than the design side.

If the unbreakability notion is in practice the most studied one through challenge implementations, the incompressibility notion is the one that has been the most studied generically. For instance, Bodganov et al. [23] study the 'space-hardness' of ciphers, which is essentially imcompressibility, itself based on the weak white-box security of [18]. Fouques et al. The paper of [57] study weak and strong incompressibility and design a dedicated encryption algorithm satisfying these properties. While some of these definitions are interesting on their own, we believe that the notion of incompressibility we present here, adapted from [43], is a good synthesis of them and is grounded in implementation attempts.

In the Private-Key Setting

Similarly to unbreakability, we describe the notion of incompressibility in the private-key setting for encryption algorithms as it is the most common use case. Still, the definition can be adapted to any private-key algorithm.

We now describe, for any $\sigma > 0$ the game of incompressibility for a compiler $\mathcal{C}_E$:

- Draw at random a key k in keyspace K
- The adversary $\mathcal{A}$ gets the program $\mathcal{C}_E(k)$ from the compiler
- The adversary $\mathcal{A}$ returns a program $\mathcal{P}$ knowing $\mathcal{C}_E(k)$
- The adversary $\mathcal{A}$ succeeds if $\mathcal{P} \approx \mathcal{C}_E(k)$ and $size(\mathcal{P}) \leq \sigma$

Definition 6. *Let E be an symmetric encryption algorithm, $\mathcal{C}_E$ a white-box compiler and let $\mathcal{A}$ be any adversary. We define the probability of the adversary $\mathcal{A}$ to succeed in the σ-incompressibility game by:*

$$Succ_{\mathcal{A},\mathcal{C}_E} := \mathbb{P}[k \leftarrow K; \mathcal{P} = \mathcal{A}(\mathcal{C}_E(k)) ; \mathcal{P} \approx \mathcal{C}_{ENC}(k);\ (size(\mathcal{P}) \leq \sigma)]$$

Moreover, we say that $\mathcal{C}_\mathcal{S}$ is (σ,τ,ϵ)-incompressible if for any adversary $\mathcal{A}$, Time($\mathcal{A}$)+Time($\mathcal{P}$) $< \tau$ implies $Succ_{\mathcal{A},\mathcal{C}_E} \leq \epsilon$.

When we want to precise that the compiler is (σ,τ,ϵ)-incompressible against a certain class of attacks noted ATK, we note that the compiler is (σ,τ,ϵ)-INC-ATK.

Remark: This definition can usually be found with a parameter δ that allows the program $\mathcal{P}$ to agree with the targeted function with probability δ. As no known attack exploits this fact yet, we did not include it for sake of clarity and simplicity.

The definition of incompressibility we propose here is a slightly corrected version from the usual one used in [43]. Indeed, this one is flawed as it does not constrain the running time of the program $\mathcal{P}$. If the running time is not bounded we can propose a compression of any white-box algorithm by using brute force: an attacker can compute few pairs plaintext-ciphertext, and code the brute force attack on the primitives that is white-boxed, and then code the computation of the primitive with the key found. This program can be made with few lines of code, is identically functional to the white-box code, but has an unreasonable running time. That is why we add a new time constraint: we want that the sum of the running time of the attacker and the program produced is less than a constant τ representing the whole computation time allowed.

Remark: Note that if a compiler is not unbreakable, then it cannot be incompressible for reasonable security levels: the key-recovery is indeed an extreme compression of a white-box implementation.

In the Public-Key Setting

Similarly to unbreakability, we describe the notion of incompressibility in the public setting for signature algorithms as we will need it for part III later on. This can easily be adapted to public-key encryption algorithms.

We now describe, for any $\sigma > 0$ the game of incompressibility for a compiler $\mathcal{C}_\mathcal{S}$:

- Draw at random a key k in private keyspace $K_{\mathcal{S}}$
- The adversary $\mathcal{A}$ gets the program $\mathcal{C}_{\mathcal{S}}(k)$ from the compiler
- The adversary $\mathcal{A}$ returns a program $\mathcal{P}$ knowing $\mathcal{C}_{\mathcal{S}}(k)$
- The adversary $\mathcal{A}$ succeeds if $\mathcal{P} \approx \mathcal{C}_{\mathcal{S}}(k)$ and $size(\mathcal{P}) \leq \sigma$

Definition 7. *Let $\mathcal{S}$ be an asymmetric signature algorithm, $\mathcal{C}_{\mathcal{S}}$ a white-box compiler and let $\mathcal{A}$ be any adversary. We define the probability of the adversary $\mathcal{A}$ to succeed in the σ-incompressibility game by:*

$$Succ_{\mathcal{A},\mathcal{C}_{\mathcal{S}}} := \mathbb{P}[k \leftarrow K; \mathcal{P} = \mathcal{A}(\mathcal{C}_{\mathcal{S}}(k)) ; \mathcal{P} \approx \mathcal{C}_{\mathcal{S}}(k);\ (size(\mathcal{P}) \leq \sigma)]$$

Moreover, we say that $\mathcal{C}_{\mathcal{S}}$ is (σ,τ,ϵ)-incompressible if for any adversary $\mathcal{A}$, Time($\mathcal{A}$)+Time($\mathcal{P}$) $< \tau$ implies $Succ_{\mathcal{A},\mathcal{C}_{\mathcal{S}}} \leq \epsilon$.

Metrics for Incompressibility

While the notion of incompressibility is well discussed in the state of the art, the field lacks good metrics to measure if the size σ of an implementation is satisfactory enough. Is it the ratio of the size to the key to the implementation ? Is it is the size of the implementation only ? The ratio size to key is probably a bad one as there exist algorithms with bigger keys than other for the same security level, and the absolute size of the implementation is completely depending on the usage of said implementation. If we can provide a functional white-box implementation with $\sigma \approx 2^{40}$, few GB for instance, retrieving it could probably be detected, but the running time of the implementation would be considerable. In that sense, smaller implementations might be more useful. However, if the reduction is too efficient with $\sigma \approx 2^{20}$, the incompressibility criteria might not be useful anymore: an attacker extracting the code from a device could remain unnoticed.

Overall, more grounded metrics for concrete use cases should be discussed to attempt to bridge the gap between the new incompressible algorithms the state-of-the-art proposes, and the implementations of standards that are motivated by concrete use in the industry.

The 'one-way function trick' meets composition

If the unbreakability notion suffered from a definitional problem caused by the possibility of using the 'one-way function trick', the incompressibility property suffers from a similar default. Indeed, this time, one can attempt to consider an encryption algorithm E that has been modified into E' following the one-way function trick, expect this time,

any input x is replaced by $f(x)$ where f is any bijection that admits an incompressible compiler.

If it is tempting to assess that the new E' admits a trivial incompressible compiler, some counterexamples show that one has to be careful. For instance, if $E = f^{-1}$, the incompressibility property completely vanishes. This shows that a composition of incompressible functions is not always incompressible and that incompressibility of complex algorithms can only be stated through precise analyses.

However, similarly to proofs of security in the random oracle model for hash and sign algorithms, if f is 'random' enough in regard to E, the composition has great chances to be incompressible.

2.2.3 One-Wayness

In the private-key setting, a natural improvement of the unbreakability notion is to require that the implementation cannot be inverted. Indeed, having both the functionality of encrypting and decrypting is equivalent to having the complete functionality of an encrpytion algorithm and its secret key, up to the size of these programs and the time computing these functionality takes. The idea is to extend the property of one-wayness of functions to the context a white-box program

Let us describe the game for one-wayness of a compiler $\mathcal{C}_E$:

- Draw at random a key k in key-space K
- The adversary $\mathcal{A}$ gets the program $\mathcal{C}_E(k)$ from the compiler
- The adversary gets an encryption c of a random message m
- The adversary $\mathcal{A}$ returns a message guess $\hat{m}$ in time τ knowing $\mathcal{C}_E(k)$ and c
- The adversary $\mathcal{A}$ succeeds if $m = \hat{m}$

Definition 8. *Let E be an symmetric encryption algorithm, $\mathcal{C}_E$ a white-box compiler and let $\mathcal{A}$ be any adversary. We define the probability of the adversary $\mathcal{A}$ to succeed in the one-wayness game by:*

$$Succ_{\mathcal{A},\mathcal{C}_E} := \mathbb{P}[k \leftarrow K; \mathcal{P} = \mathcal{C}_E(k), \mathcal{A}(\mathcal{P}, c) = \hat{m}; m = \hat{m}]$$

We say that $\mathcal{C}_E$ is (τ,ϵ)-one-way if for any adversary $\mathcal{A}$ running in time τ, $Succ_{\mathcal{A},\mathcal{C}_E} \leq \epsilon$.

When we want to precise that the compiler is one-way against a certain class of attacks noted ATK, we note that the compiler is (τ,ϵ)-OW-ATK.

Remark : The notion of one-wayness is only interesting in the symmetric setting, as in the asymmetric setting, an inversion of the algorithm is possible via the public-key.

Remark : There is a trivial reduction from one-wayness to unbreakability as recovering the key allows to invert an encryption algorithm.

For a private-key encryption algorithm, having a one-way implementation is similar to transforming it into a public-key algorithm. The implementation can indeed be seen as a public-key as it is sufficient to encrypt messages - but cannot be used to decrypt as it is one-way - and the secret-key still allows for decryption. This idea was already developed by Diffie and Hellman in their famous [44], where they advocate for the existence of public-key encryption, when no construction was known, through 'obfuscating' the program of a private-key algorithm.

2.3 What do we know about these notions

The goal of this section is to give insights on links between white-box cryptography and other state-of-the-art notions of cryptography or mathematics. We first discuss incompressible structures at large and then draw similarities between public-key white box and identity-based cryptography. We finish this section by discussing the links between cryptographic obfuscation and white-box cryptography.

2.3.1 Incompressible Structures

For mathematicians, the first intuition of incompressibility comes from Kolmogorov's complexity. In his axiomatic approach to the theory of probability in 1965, Kolmogorov wants to graps the notion that what is random is what is the most difficult to describe. The Kolmogorov's complexity of x is defined by the minimum amount of information that is needed to represent x. A representation of an element would be incompressible if its size is equal to its Kolmogorov's complexity. The notion of Kolmogorov' complexity is however a theoretical notion and cannot be computed. The link between incompressibility and randomness in the look of an adversary is however the track designers follow in the white-box model.

In practice, the representation of computations by randomized look-up tables is the most used technique. Since the seminal construction of Chow et al. most of the implementations advances in the state of the art have been incremental over the table network they proposed and have hence kept the same structure. This can be seen in implementations that we describe in the state of the art of AES implementations in part 2. The look-up table structure can also be seen in new algorithms that achieve incompressibility such as [23, 57].

The lack of diversity of incompressible structures for new implementations brings the following question: what other structures can be used to offer flexibility of designs ? An interesting track to follow is to use low-degree polynomials transformations instead of the full degree transformations that are represented by a look-up table. Indeed, the look-up table is often a representation of a composition of function that reaches full degree over its input, hence, the truth table is a compact way of representing it. The underlying 'hard' problem is the decomposition of functions. However, a composition of function does not always reach full degree and can be represented by polynomials, and the problem of decomposing them can still be difficult. This is the case for HFE public-keys for instance (see part 3 for more details) and it has seen a few attempts in the white box model in [31, 95, 96] without too much success. We propose in part 2 a white-box implementation of AES and in part 3 of HFE with these ideas.

2.3.2 Links with Identity-Based Cryptography

Identity-Based Cryptography was conceptualized to answer the question: "is it possible to use human readable common data, such as e-mail addresses, to derive safe cryptographic key ?". This concept was first introduced by Shamir in 1984 with an instantiation of a signature scheme [102]. The problem of building an identity-based encryption scheme was open until 2001 with the pairing-based of Boneh and Franklin [25] for instance.

The goal of Identity-Based Cryptography is to replace the random public key - that is generated in the context of an asymmetric algorithm - by publicly available data linked to a user. As a consequence, the public key of such scheme is usually known is advance for it is an e-mail adress, a name, or any semantic data derived from the user, i.e. its identity for the scheme. The main problem is to derive the secret-key from this public semantic data. Contrary to usual public-key cryptography, the secret-key is derived by a trusted central authority.

In this paragraph, we focus on identity-based encryption (IBS) as we believe it has the most pertinence, but the following remark can be applied to identity-based encryption (IBE). Formally an IBS is a signature scheme described by 4 algorithms:

- $\mathbf{IBS - SETUP}$:The trusted identity generates a master secret key MSK and a master public-key MPK

$$\mathbf{IBS - SETUP}(1^k) = (MSK, MPK)$$

- $\mathbf{IBS - EXTRACT}$: For any semantic data ID of a user, the trusted authority computes from MSK and MPK, the secret-key of the user SK_{ID}.

$$\mathbf{IBS-EXTRACT}(ID, MSK, MPK) = SK_{ID}$$

- $\mathbf{IBS-SIGN}$: Given MPK and SK_{ID}, one can compute the signature of M :

$$S = \mathbf{IBS-SIGN}_{MPK,SK_{ID}}(M)$$

- $\mathbf{IBS-VERIF}$: Given a signature (S,M) , and the public-key MPK anyone knowing ID can compute:

$$\mathbf{IBS-VERIF}_{MPK,ID}(S, M) = 1 \; iif \; S = \mathbf{IBS-SIGN}_{MPK,SK_{ID}}(M)$$

Remark: It is important to remark that the trusted authority does not take part in the communication after key-generation. Conversely, the key SK_{ID} is generated without the intervention of the signer.

For an IBS scheme to be of any use, the information of ID or MPK must not leak information about the secret key SK_{ID} in addition to being secure in the classical black-box sense.

We now make the remark that an IBS scheme can be seen as an unbreakable signature scheme and its white-box implementation. Indeed, let us assume that $(\mathbf{IBS-SETUP}, \mathbf{IBS-EXTRACT}, \mathbf{IBS-SIGN}, \mathbf{IBS-VERIF})$ is a secure IBS scheme in the black-box model. We define the public-key signature algorithm Sw/oIB by:

- $\mathbf{Sw/oIB-SETUP}$: For a fixed value ID, the algorithm first computes $\mathbf{IBS-SETUP}(1^k) = (MSK, MPK)$. The algorithm outputs the secret-key SK and the public-key PK

$$\mathbf{Sw/oIB-SETUP}(1^k) = (SK, PK) = (MSK, (MPK, ID))$$

- $\mathbf{Sw/oIB-SIGN}$: Given a secret key SK from $Sw/oIB - SETUP$, the algorithm computes $IBS-EXTRACT(ID, MSK, MPK) = (MPK, SK_{ID})$ one can then compute the IBS signature of M :

$$S = \mathbf{Sw/oIB-SIGN}_{SK}(M) = \mathbf{IBS-SIGN}_{MPK,SK_{ID}}(M)$$

- $\mathbf{Sw/oIB-VERIF}$: Given a signature (S,M) , and the public-key $PK = (MPK, ID)$ anyone knowing ID can compute :

$$\mathbf{Sw/oIB-VERIF}_{PK}(S, M) = 1 \; iif \; S = \mathbf{IBS-SIGN}_{MPK,SK_{ID}}(M)$$

The scheme Sw/oIB is just a rewriting of the IBS scheme where the generation of the ID related key is relegated to the signer. The algorithm we obtain is just a public-key signature scheme, without any identity based property. In this scheme, the master secret key MSK can be seen as the only secret element, with SK_{ID} only being an information computed at the time of signature. To get an unbreakable white-box implementation of Sw/oIB, one would need to 'hide' the key MSK, but the IBS scheme is already tailored for that as the secret key SK_{ID} (output of the extraction $\mathbf{IBS-EXTRACT}$) can be used for signing and the security of IBS is based on the fact that recovering MSK from SK_{ID} is hard. We hence get then the trivial property:

Proposition 1. *Let IBS be a secure identity-based signature scheme and Sw/oIB its associated public-key signature scheme. Let WB-Sw/oIB be an implementation of Sw/oIB where the key SK_{ID} from the extraction $\mathbf{IBS-EXTRACT}(ID, MSK, MPK)$ is hardwired into the implementation instead of the master-key MSK and $\mathbf{Sw/oIB-SIGN}_{SK}$ is replaced by $\mathbf{IBS-SIGN}_{MPK,SK_{ID}}$. The implementation WB-Sw/oIB is unbreakable for the same level of security as the black-box security of IBS.*

If this remark highlights the similarities between Identity-Based Signatures and what we require from white-box implementations of traditional pubic-key signatures, it fails to deliver solutions for concrete implementations techniques.

First, this remark only applies to the unbreakability property. Indeed, if we want to extend this result to incompressibility, we have to then consider the sizes of the keys MSK and SK_{ID}, which are the only different elements from one implementation to the other. Usually, the sizes of these keys are quite similar and having big keys SK_{ID} is not a goal of this area of research. In that sense, this remark is quite close to the "one-way function trick" (section 2.2) to get trivial unbreakable implementation of symmetric algorithms.

Secondly, while designing unbreakable or incompressible public-key signature algorithms from scratch, i.e. without implementing a standard or already deployed algorithm, can be an interesting area of research as it was done for block-ciphers [23, 57], this link with identity-based cryptography does not seem to help design implementations of standards such as RSA or ECDSA. Indeed, designing IBS schemes is already a hard problem, but transforming a traditional PKC signature algorithm into an IBS one seems out of reach for now.

To us, the important conclusion of this analogy is that the kind of 'double trapdoor' structure that white-box cryptography needs in the public-key setting exist in the field of cryptography, which opens the room for more experimentation and designs and possibly for new PKC signature algorithms designed with their white-box implementation in mind.

2.3.3 Links with Cryptographic Obfuscation

The field of cryptographic obfuscation has a lot of similarities with the goal of white-box cryptography. Especially, obfuscation techniques have been envisioned as possibilities to get white-box implementations. To precisely compare them and to see what can be learned from it, let us quickly introduce it.

The goal of cryptographic obfuscation is to construct a compiler $\mathcal{O}$ that transforms a program P into a functionally equivalent program $\mathcal{O}(P)$ that is "unintelligible". First formally introduced by Hada in [66], defining properly what "unintelligible" means for such generic obfuscator - that is, that works for any class of programs - to exist has been the first difficulty. The work of Barak et al.[5] formalizes the notion Virtual Black-Box Obfuscation which is pretty intuitive: the obfuscated program should not give more information than a black-box access to its functionality. Formally:

Definition 9 (VBB Obfuscation). *An algorithm $\mathcal{O}$, which when given a circuit in C outputs a new circuit, is said to be a black-box obfuscator for the family $\mathcal{C}$, if it has the following properties:*

- ***Preserving Functionality**: There exists a negligible function neg(n) such that for any input length n, for any $\forall C \in \mathcal{C}_n$:*

$$\mathbb{P}[\exists x \in \{0,1\}^n : \mathcal{O}(C)(x) \neq C(x)] \leq neg(n)$$

  The probability is over the random oracle and $\mathcal{O}$'s coins.

- ***Polynomial Slowdown**: There exists a polynomial p(n) such that for all but finitely many input lengths, for any $C \in \mathcal{C}_n$, the obfuscator $\mathcal{O}$ only enlarges C by a factor of p:*

$$|\mathcal{O}(C)| \leq p(|C|).$$

***Virtual Black-box**: For any polynomial-sized circuit adversary $\mathcal{A}$, there exists a polynomial-sized simulator circuit S and a negligible function neg(n) such that, for every input length n and every $C \in \mathcal{C}_n$:*

$$|\mathbb{P}[A(O(C)) = 1] \; - \; \mathbb{P}[|S^C(1^n) = 1]| \leq neg(n)$$

Where the probability is over the coins of the adversary, the simulator and the obfuscator. In the presence of a random oracle, the probability is also taken over the random oracle.

However, in the same paper, Barak et al. [5] prove that generic VBB obfuscators do not exist. To prove their result, they exhibit a special family of functions that have the property of leaking a secret on the input of a program that computes its own functionality. While this result proves a generic impossibility, it does not rule out the possibility of building VBB obfuscation for commonly used cryptographic primitives such as the AES. Having a VBB implementation of the AES would be ideal since it would trivially give a unbreakable and one-way implementation of the AES.

Since there are impossibility results for VBB in the standard model, the cryptographic community has shifted from the study of VBB obfuscation to a weaker notion of obfuscation, Indistinguishability Obfuscation (iO), that was first introduced in the same article by Barak et al. [5]. Roughly, the goal of iO is to transform a program into a pseudo-canonical form of it. This way, the obtained program should not leak more information than any program computing the same function. We recall the formal definition of iO:

Definition 10 (iO). *A probabilistic polynomial algorithm $\mathcal{O}$, which takes as input a circuit in C and outputs a new circuit, is said to be an indistinguishability obfuscator for the family $\mathcal{C}$, if it has the following properties:*

- ***Preserving Functionality**: There exists a negligible function neg(n) such that for any input length n, for any $\forall C \in \mathcal{C}_n$:*

$$\mathbb{P}[\exists x \in \{0,1\}^n : \mathcal{O}(C)(x) \neq C(x)] \leq neg(n)$$

  The probability is over the random oracle and $\mathcal{O}$'s coins.

- ***Polynomial Slowdown**: There exists a polynomial p(n) such that for all but finitely many input lengths, for any $C \in \mathcal{C}_n$, the obfuscator $\mathcal{O}$ only enlarges C by a factor of p:*

$$|\mathcal{O}(C)| \leq p(|C|).$$

- ***Indistinguishable Obfuscation**: For all large enough input lengths, for any circuit $C_1 \in \mathcal{C}_n$ and for any circuit $C_2 \in \mathcal{C}_n$ which computes the same function as C_1 with $|C_1| = |C_2|$, the two distributions $\mathcal{O}(C_1, r)$ and $\mathcal{O}(C_2, r)$ over the coins of the obfuscator are computationally indistinguishable.*

After the first plausible construction of iO by [59], research bloomed on the subject and numerous applications of iO were found, including new cryptographic primitives. Here is a short list of the notable primitives that iO could achieve:

- Functional Encryption[59] (open problem since 2005)
- Deniable Encryption [98] (open problem since 2006)
- Multiparty Key Exchange [26]
- Universal Witness Signature [94] (open problem since 2010)

As it was forecast, iO is indeed be a cryptographic master tool. The celebrated line of work leading to the result of Jain et al. [69] lately proved that a polynomials iO compiler exists under standard assumptions, while a realistic implementation is still out of reach for usual algorithms.

Link with white-box cryptography While white-box focuses on cryptographic algorithms such as encryption or signature algorithms, cryptographic obfuscation aims to be applicable to any program. Cryptographic obfuscation also radically differs from practical code obfuscation techniques, for it tries to proves its security based on standard assumptions while the other do not. The existence results for iO ensure that polynomial constructions exist, but the solutions tailored for a cryptographic algorithms will surely be more efficient.

We mention an interesting result of construction of white-box implementations provided that the targeted white-box properties exist. If a white-box compiler can do it, then an iO obfuscator can too. We state the result for unbreakability but it can also be extended to one-wayness and incompressibility. Let us first assume that we extend the notion of white-box compiler to match an obfuscator, that is, our compiler is now defined for any class of circuit $\mathcal{C}_n$ and satisfies 'Preserving Functionality' and 'Polynomial Slowdown'. In that case:

Proposition 2. *Let $\mathcal{O}$ a generic iO obfuscator and $\mathcal{P}$ a family of keyed-algorithms. If there exist an (τ, ϵ)-unbreakable compiler WB with polynomial slowdown for the family of $\mathcal{P}$, then $\mathcal{O}$ is a (τ, ϵ)-unbreakable compiler for $\mathcal{P}$.*

Proof. Let $n \in \mathbb{N}^*$ and $P \in \mathcal{P}_n$. For a fixed key k, the implementation $WB(P_k)$ is (τ, ϵ)-unbreakable by hypothesis and its size λ is polynomial in n. Now, consider an implementation of $\tilde{P}_k$ pf P_k that is obtained by padding P_k to the size λ. The programs $WB(P_k)$ and $\tilde{P}_k$ are of the same size and functionally equivalent. As WB is unbreakable for exponential security parameters, the implementation $\mathcal{O}(WB(P_k))$ is also unbreakable as $\mathcal{O}$ is polynomial. By the indistinguishably of $\mathcal{O}$, the output programs distribution $\mathcal{O}(WB(P_k))$ and $\mathcal{O}(P_k)$ are computationally indistinguishable. If the program $\mathcal{O}(P_k)$ is not unbreakable, this means that the key-recovery is a distinguisher between the two distributions, which is absurd. The compiler $\mathcal{O}$ is thus unbreakable. □

Remark: It is important to note that this result in asymptotic and hence might not be useful at all in the concrete context of white-box cryptography applied to the AES-128 primitive. It is however helpful to understand the similarities between the notions.

Lastly, a line of work initiated by Alpirez-Bock et al.[22] where they use obfuscation techniques in addition to a specific hardware to prove strong security properties. We explain this line of work in more details in chapter 3.

2.4 Discussing an Impossibility Result

In 2020, Alpirez Block et al. published, in [21], a result concerning the impossibility of a secure generic compiler in the white-box model. Their result uses the same construction as the generic VBB impossibility result of Barak et al. of 2001 [5] and gives the example of a secure encryption scheme that cannot be white-boxed that we briefly recall.

Starting from a secure encryption scheme with encryption algorithm E, they design a new encryption algorithm E' that, on the input of a message that is the code of an encryption scheme, outputs the key of E, and on the input of a regular message, outputs the corresponding output ciphertext of E. The technical detail of checking the input is dealt with a simple pseudo-random functions, which makes the construction feasible for any encryption scheme. Such algorithm E' can still be proven secure, and no secure white-box implementation can be made: if one gets an implementation of E', on the input of the same implementation, the key is output to an attacker.

While this result once again shows the similarities between obfuscation and white-box, it does not motivate a complete change of paradigm nor does it give insight to the hardness of the problem for real cryptographic algorithms. Indeed, the VBB impossibility result forced to change notions of obfuscations because the nature of cryptographic obfuscation is to be generic. The goal of white-box cryptography however, is to implement concrete functions that represent a tiny small amount of programs. For instance, a VBB implementation of the AES might still exist as the impossibility result of VBB does not rule it out and the same is true for an AES white-box secure implementation.

Chapter 3

Variations of the Plain White-Box Model: The Remote Access White-Box Model and Hardware-Module White-Box Model

The study of the plain white-box model is now about two decades old. In these decades, the cryptographers grown unsatisfied with some aspects of this model. First, the hardness of providing any secure implementations of standards in this model made deployment of peer reviewed implementations in the industry impossible. This established a gap between the concrete products that contain white-box implementations and the academic attempts towards secure implementations. Secondly, the security properties that can be achieved in the plain white-box model are sometimes too weak to ensure any security depending on the context. For instance, white-box implementations of encryption algorithms in the plain white-box model can be used to encrypt messages, which is a no go for certain applications.

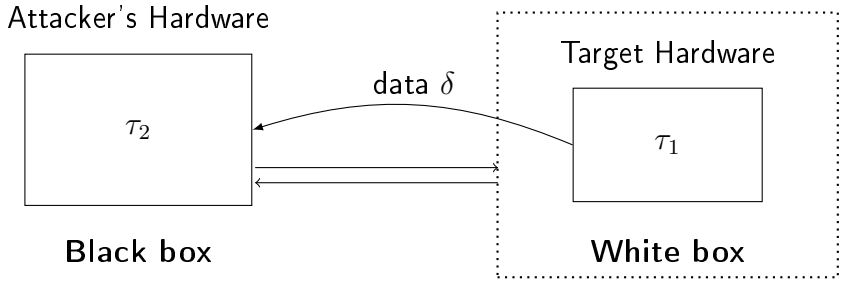
In the recent years, two other main models, have emerged. The first model is the Remote Access White-Box model, which attempts to capture the attack model of malware that are planted on a target hardware and remotely communicate with an attacker. The second one is the Hardware Module White-Box model, which studies secure implementations based on small secure hardware components.

If these two models were originally presented as security notions in the context pf the the white-box model, we here want to set them apart from the plain white-box model for sake of clarity and for a fair comparison with the security implementations can achieve in these model compared to the plain white-box model.

3.1 The Remote Access White-Box Model

To the best of our knowledge, the first evocation of the Remote-Access White-Box model was made by Todo et al. [104] to study the space-hardness[23] of two block-cipher built for the white-box model, SPNbox [23] and Yoroi [76]. The authors describe the 'hybrid' setting of their attack in the following way. If a malware is planted on a device to target an implementation, it can compute specific information about the implementation on site before sending it to a black-box adversary that has more computation power than the device allows. This setting seems interesting to us as it highlights the asymmetry of capabilities between some devices where white-box implementations are deployed (smartwatches, TVs, ...), that have limited RAM, hard-memory or computational power, and the usual cryptographic adversary: data needs to be extracted from the device to perform more sophisticated attacks, which can be hard to do remotely. While this does not guarantee security against an adversary that can extract unlimited amount of data, this encapsulates the weaker and common adversaries such as malware.

We now formally describe the Remote Access model. In this model, we consider that an adversary only has remote access to the device where the implementation is deployed. We suppose that this device has memory μ_1 and can run for a time τ_1. To attack the implementation, the adversary has two possibilities: he can mount the attack on this device or extract a certain quantity of information δ to mount an attack with its own memory μ_2 and running time τ_2 (Figure ??). Note that we add the memory constraint to the usual model that cannot be found in [67, 104] to grasp the real behavior of devices. To distinguish the two phases of the attacks, we divide the attacker into two subattackers, $\mathcal{A}_{MW}$ that builds and plant the malware and $\mathcal{A}_{remote}$ that carries on the black-box attack in the end.



This model is in essence very similar to the setting for strong incompressibility of [57] except the leakage is here computationally bounded but is only restricted by min-entropy in [57]. Another main difference is that strong incompressibility was introduced to get provable security while security in the remote-access model is similar to the

notions we described in the plain white-box model and security is based on the analysis of the best known attacks against the construction.

One of the motivations of studying this model is that by relaxing the plain white-box model, designers might achieve stronger securities properties that are useful in concrete cases. The asymmetric computational restriction (τ_1, μ_1) from the target hardware greatly empowers designers, at least at first glance.

3.1.1 Security Notions

As the remote-access model is a restriction of the plain white-box model, notions such as unbreakablility, incompressiblity and onewayness can still be studied. However, has the model restricts the plain white-box attacker, one can hope for stronger security notions, especially with the offline phase the model introduces. Similarly to chapter 2, we adapt security notions in the framework of [43].

Notions from the Plain White-Box Model

If notions such as incompressibility are well understood for the design of algorithm point of view, there are still too few implementation techniques for usual algorithms that achieve any security in the plain white-box model. Studying the security in the Remote-Access model might allow for new techniques to sprout, using the relaxation introduced by the model. We believe that studying notions such as unbreakability or one-wayness in this model for implementation of standards with relevant parameters is still a first milestone to reach.

We now describe the game of remote access incompressibility for any $\sigma, \delta > 0$ for a compiler $\mathcal{C}_\mathcal{E}$. This is a modified version of the game of unbreakability where we allow black-box access to $\mathcal{E}_k$ and a certain quantity δ of data on $\mathcal{C}_\mathcal{E}(k)$ gathered in the white-box model on the device:

- Draw at random a key k in private keyspace $K_\mathcal{E}$
- The adversary $\mathcal{A}$ request any set of data D_{WB} of size at most δ on the implementation $\mathcal{C}_\mathcal{E}(k)$ in the white-box model in time τ_1 and memory μ_1.
- The adversary $\mathcal{A}$ returns a program $\mathcal{P}$ in time τ_2 and memory μ_2 having access to D_{WB} and a black-box access to $\mathcal{E}_k$
- The adversary $\mathcal{A}$ succeeds if $P \approx E_K$ and $size(\mathcal{P}) < \sigma$

Definition 11. *Remote Access Incompressibility*

Let E be a symmetric encryption algorithm, $\mathcal{C}_E$ a white-box compiler, let $\mu_1, \tau_1, \mu_2, \tau_2 \in \mathbb{N}^*$, $\mathcal{A}$ be any adversary and D_{WB} any data set of size δ that can be

extracted from $\mathcal{C}_E$ in time τ_1 and memory μ_1 in the white-box model. We define the probability of the adversary $\mathcal{A}$ to succeed in the remote access unbreakability game by:

$$Succ_{\mathcal{A},\mathcal{C}_\mathcal{S},D_{WB}} := \mathbb{P}[k \leftarrow K; \mathcal{P} = \mathcal{C}_\mathcal{S}(k), \mathcal{A}(\mathcal{O}_{\mathcal{S}(k)}, D_{WB}) = \mathcal{P}; P \approx \mathcal{C}_E(k);\ size(\mathcal{P}) < \sigma]$$

We say that $\mathcal{C}_\mathcal{E}$ is $(\delta, (\mu_1, \tau_1), (\mu_2, \tau_2)), \sigma, \epsilon)$-remote access incompressible if $\mathcal{C}_\mathcal{E}$ is $(\mu_1, \tau_1, \epsilon)$-unbreakable and if for any adversary $\mathcal{A}$ and data set D_{WB} running in time τ_2 and memory μ_2, $Succ_{\mathcal{A},\mathcal{C}_\mathcal{S},D_{WB}} \leq \epsilon$.

Closing the Gap with the Black-Box Model

The work of Hosoyamada et al. pushes the usual security notions of the plain white-box model for notions that are closer to the ones studied in the black-box model. They extend PRI, PRP, PRF and IND-CPA in the Remote-Access White-box model. If these notions were not achievable in the plain white-box model, the two-time phase of the remote-access model might allow implementation that satisfy these properties. We only adapt the PRP definition in our setting and refer to Hosoyamada et al. for the others. We describe the security property for encryption algorithms.

The PRP security is defined by evaluating the distinguishability of two games: a real one based on the implementation and an ideal one based on oracle calls. To adapt their definition to the Remote-Access model, Hosoyamada et al. modify the usual PRP games.

- A key k is chosen at random and a program $\mathcal{C}_E(k)$ is computed by the compiler
- The adversary $\mathcal{A}$ creates a malware - or lifter. To create this malware M, the adversary only has remote-access to the devices, which is modeled as a black-box access to $\mathcal{C}_E(k)$.
- Once the malware is built, it is planted and can perform any white-box attack in time τ_1 and memory μ_1. It then leaks a data set D of size δ
- Finally, the adversary $\mathcal{A}$ can attack $\mathcal{C}_E(k)$ in the black-box model aided by the leakage D.

Remark : The choice of allowing $\mathcal{A}$ to query $\mathcal{C}_E(k)$ before building the malware can be seen as an empowerment of the usual malware adversary. Conversely, the fact that the malware cannot be modified adaptatively in regard to the white-box information extracted fails to grasp the most powerful adversaries. These variations can be discussed, but this framework allows for simplicity of description.

In PRP security, the real game needs it ideal alternative. To do so, the authors use a simulator S that simulates the behavior of the malware M on the implementation. Formally, S is an algorithm that on the input of M and the functionalities E and E^{-1}, outputs a data set D_{IDEAL} of size δ that is indistinguishable from the leakage D made by M. Hosoyamada et al. detail how such simulator should function, but we only assume here that it can be made in reasonable time.

We now describe these two games. The real experiment is as follow:

- A key k is drawn at random and a program $\mathcal{C}_E(k)$ is computed by the compiler
- $\mathcal{A}_{MW}$ builds a malware for $\mathcal{C}_E(k)$.
- The malware leaks $D = MW(\mathcal{C}_E(k))$ to $\mathcal{A}$
- $\mathcal{A}$ tries to distinguish E_k from an random permutation by using D and black-box access to $\mathcal{C}_E(k)$. It outputs $\mathbf{Exp}^{PRP-real}_{E,\mathcal{C}_E(k),\mathcal{A}} = 1$ if it thinks it is random, 0 otherwise

The ideal game goes as follows.

- A random permutation $\sigma \in \mathcal{S}_n$ is drawn at random
- $\mathcal{A}_{MW}$ builds a malware MW for σ.
- The simulator S simulate the leakage that would produce MW on the input of a programm computing σ, leaking a data set D_{IDEAL}
- $\mathcal{A}$ tries to distinguish σ from a random permutation by using D_{IDEAL} and black-box access to σ. It ouputs $\mathbf{Exp}^{PRP-ideal}_{S,\mathcal{A}} = 1$ if it thinks it is random, 0 otherwise

Definition 12. *Let $\mathcal{E}$ be a symmetric encryption algorithm, $\mathcal{C}_\mathcal{E}$ a white-box compiler, let $\mu_1, \tau_1, \mu_2, \tau_2 \in \mathbb{N}^*$, $\mathcal{A} = (\mathcal{A}_{remote}, \mathcal{A}_{MW})$ be any adversary. We define the probability of the adversary $\mathcal{A}$ to succeed in the remote access unbreakability game by:*

$$Succ_{\mathcal{A},\mathcal{C}_\mathcal{S},D_{WB}} := |Pr[\mathbf{Exp}^{PRP-real}_{E,\mathcal{C}_E(k),\mathcal{A}} = 1] - Pr[\mathbf{Exp}^{PRP-ideal}_{S,\mathcal{A}} = 1]|$$

We say that the compiler $\mathcal{C}_E$ is $(\delta, \mu_1, \tau_1, \mu_2, \tau_2, q, q_{sim}, \epsilon)$-remote-access PRP secure if for any adversary $\mathcal{A}_{MW}$ running in time and memory τ_1, μ_1 leaking data sets δ in the white-box model and $\mathcal{A}_{remote}$ running in time and memory τ_2, μ_2 with at most q queries to $\mathcal{C}_E(k)$ in the black-box model, there exists a simulator S that makes at most q_{sim} queries such that $Succ_{\mathcal{A},\mathcal{C}_\mathcal{S},D_{WB}} \leq \epsilon$.

As the authors of [67], the goal of this security notion is not to be proved, but to be tested against the best attack found in the literature, like it is common for the PRP property of usual algorithms. The authors also give $SPACE - n_\alpha$ [23], $\alpha \in \{8, 16, 24, 32\}$ as a candidate for remote-access PRP. They conjecture that it is the case for $\tau_2 = 2^{128}$,$q = 2^{128}$, $\delta = (128 - n_\alpha) \times 2^{n_{\alpha-2}}$ and $\epsilon << 1$ as long as $\tau_1 << q_{sim} < 2^{128}$. This means that the property is essentially broken once few tables have been leaked.

3.1.2 A Lack of Benchmarks

The model is young, with very few publications on the topic. The first conjectures of security about algorithms from the state of the art were made by Todo et al. [67]. These conjecture are essentially parameterized by δ and (τ_1, μ_1). To know if these security conjectures are of any use, one needs to evaluate them in the context of concrete use cases.

A lack of benchmark for the hardware parameters It is natural in white-box works to ignore hardware specification, as it is the essence of the model. However, in the remote-access model, the parameters τ_1 and μ_1 are derived from concrete hardware. Their specifications vary : an hardware on a smartwatch, a TV or a smartphone might be really different. The power consumption on devices on batteries also intervene in the estimation of the time that the device can run the malware. As the model matures, these estimations have to come into play to ground the solutions proposed and evaluate the concrete utility of the construction.

A lack of study of the control of the extraction of data The argument of the hardness of huge data extraction on devices is a long time participant in white-box discussions to bind implementations to hardware. Since the introduction of the notion of incompressibility in the plain white-box model and works on 'big-key' ciphers [10], the argument that sensible data has to be made as big as possible to make the task of code lifting difficult is central to usefulness of the solutions. However, the actual means of assuring it or the amount of data extraction that can be detected is never detailed.

3.2 The Hardware Module White-Box Model

Applications using secret data on devices such as smartphones are already deployed nowadays and widely used, notably for mobile payment. If the security of payment cards is usually a problem studied in the grey-box model of attacks as the smart card

provides hardware security, the same cannot be said for mobile payment. In this setting, applications run on smartphones can be targeted by malware or a white-box adversary.

The industrial solutions to this problem often come with the usage of a trusted component in the device that will handle sensible computations. However, the solutions often vary due to the the available hardware. For instance, Apple smartphones all have their dedicated secure component, while only the top-end Android phones have them, and they differ from one supplier to the other.

In this context, the authors of [21] advocate the usage of hardware to provide hardware-binding properties, a property that cannot be achieved in the plain white-box model. The rationale is the following. If one can have a small secure hardware with a set functionality, one can attempt to bootstrap the security property from the hardware to multiple software implementation of cryptographic algorithms. The hardware will then be necessary to the implemented algorithm, and the functionality of said algorithm cannot be replicated by the an attacker that would retrieve the implementation, leaving the hardware on site. While this idea is not novel per se as we will see in section 3.2.2, their paper is the first to label model and security notions as white-box cryptography.

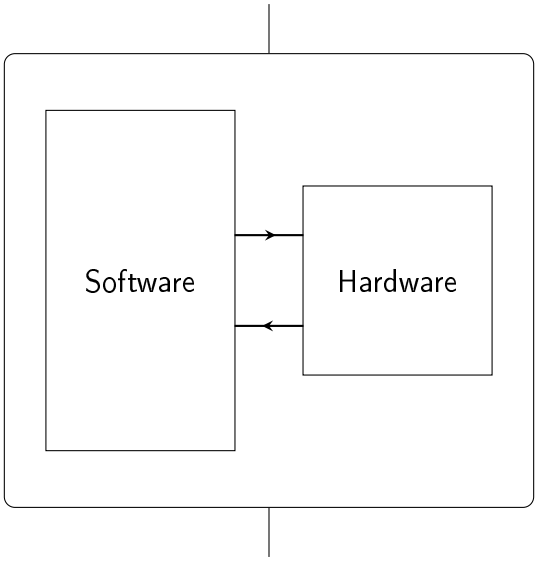


Figure 3.1: Framework for the Hardware-Module White-box model

The usage of hardware in the white-box model might feel such as going backwards from the initial goals of the white-box model. While it is partly the case, it also allows for white-box to bridge the gap between the construction in the plain white-box model

that do not offer enough security to be deployed as is and the solutions deployed in the industry that are sometimes already based on hardware functionalities. These solutions are greatly needed as the soaring of mobile payment shows. Studying the hardware-module model in academic literature can only improve the overall robustness of industrial solutions and help ordinary citizens to have access to solutions that are free of use.

For the rest of the section we will assume that we can have access to hardware in the form of a hardware module. The hardware-module HMW is assumed to be tamper-proof, to be computationally bounded and to be equivalent to having a black-box access to its functionality.

3.2.1 Security Notions

This model offers more possibilities than the plain white-box model or the remote access white-box model. It is indeed closer to the grey-box model of attack, with the security of the hardware helping to establish the security of the overall implementation. As this model is a restriction of the plain white-box model, as any implementation can be made an 'empty' hardware module, the security notions of the previous sections, such as unbreakability and one-wayness can also be considered in this model. However, for the mitigation of code-lifting attack, the correct notion to use in this context is hardware-binding. As the model is equivalent to the black-box model if we do not fix any limit on the hardware, it is also reasonable to study notions such as IND-CCA for encryption algorithms and Existential Forgery for signatures adapted to the white-box model.

For the rest of this section we consider the case of a symmetric encryption algorithm. The notions can easily be adapted to the case public-key encryption or signature when the notion is meaningful.

Notions from the Plain White-Box model

As this model is a strong relaxation of the plain white-box model, an implementation in the hardware-module white-box model should at minimum satisfy notions such as unbreakability or one-wayness. However, the main motivation of this model is to exceed plain white-box security properties.

Hardware Binding

The adaptation of incompressibility as a code-lifting counter measure in the context of hardware-module is hardware-binding. The goal is to ensure that the functionality of the program cannot be used without the hardware on the device. This is the most basic property an implementation should achieve in this model.

We now describe, for any $\tau > 0$ the game for hardware-binding security for a compiler $\mathcal{C}_\mathcal{E}$:

- Draw at random a key k in private key-space $K_\mathcal{E}$
- The adversary $\mathcal{A}$ gets the program $\mathcal{C}_\mathcal{E}(k)$ from the compiler that can make calls to a hardware module HWM.
- The adversary $\mathcal{A}$ returns a program $\mathcal{P}$ knowing $\mathcal{C}_\mathcal{E}(k)$ and making calls to HMW.
- The adversary $\mathcal{A}$ succeeds if $\mathcal{P} \approx \mathcal{C}_\mathcal{E}(k)$ and $\mathcal{P}$ does not make calls to HMW.

Definition 13. *Let $\mathcal{E}$ be an encryption algorithm, $\mathcal{C}_\mathcal{E}$ a white-box compiler and let $\mathcal{A}$ be any adversary. We define the probability of the adversary $\mathcal{A}$ to succeed game by:*

$$Succ_{\mathcal{A},\mathcal{C}_\mathcal{E}} := \mathbb{P}[k \leftarrow K; \mathcal{P} = \mathcal{A}(\mathcal{C}_\mathcal{E}(k), HWM)\,; \mathcal{P} \approx \mathcal{C}_\mathcal{E}(k)]$$

Moreover, we say that $\mathcal{C}_\mathcal{S}$ is (τ,ϵ)-hardware bound if for any adversary $\mathcal{A}$, Time($\mathcal{A}$)+Time($\mathcal{P}$) $< \tau$ implies $Succ_{\mathcal{A},\mathcal{C}_E} \leq \epsilon$.

However, as the model offers more powerful methods to protect implementation by bootstrapping the security from a secure element to the implementation, one can aim for stronger security notions. Indeed, an implementation that is bound to a hardware, there is still the possibility to inverse it on site. The ideal security notion would be to match the properties such as IND-CPA in the white-box model.

Closing the Gap with the Black-Box model

Similarly to the remote-access model, the natural idea of a stronger security model than the plain white-box model is to achieve equivalent notions to black-box notions such as IND-CCA for encryption algorithms or Existential Forgery for signature.

The approach taken by the literature is to extend indistinguishability notions to the hardware-module context. The papers of [3, 21, 22], respectively define hardware-binding security for encryption schemes, hardware-binding security for payment functionalities and security of 'global white-boxes'. Global white-boxes are out of the scope of this thesis.

These security properties aim to distinguish the behavior of the implementation on the device compared to a simulated one, similarly to the remote-access case or to cryptographic obfuscation security notions. We refer to the individual papers to the exact security properties.

3.2.2 Examples of Constructions

The goal of this section is to explore the solutions that are present in the state of the art. We review two constructions that are presented as white-box solutions and two solutions from the obfuscation state of the art, that have surprisingly not being considered in white-box publications.

Constructions under the White-Box label

The paper of Alpirez-Bock et al. [22] introduces the idea of using hardware modules to add security to software implementations. This started a line of work studying security properties of implementations using said methods. These works however do not detail hardware specifications, nor software overhead - when it is concretely computable. This bias is interesting as it is the opposite goal of the model.

White-Box Key Derivation Function Following security benchmarks of Mastercard and EMVCo, Alpirez-Bock et al. [22] build a white-box key derivation function (WDKF) that uses a hardware functionnality and define precise security properties to achieve. Their construction is based on pseudo-random function and iO, with an hardware that computes PRF functionalities. Roughly, iO helps to hide the key in the implementation and security is proven through the well-known puncturable function techniques of [98].

If their construction provides interesting proven security properties in the hardware model, the use of iO in their construction reduces the hopes of having a concrete implementation of their solution. Using watered down versions of iO would indeed void the security of the construction.

With Token Based Obfuscation The construction of Agrawal et al. [3] uses Token Based obfuscation (TBO). Token Based Obfuscation is a form of obfuscation that allow to run the obfuscated program on an input only if a token related to said input is generated. In that sense, TBO produces reusable garbled circuits [60]. A TBO obfuscator is then divided into two algorithms, the compiler $\mathcal{O}$ and the token generator $Token$:

- Like obfuscators such as iO, on the input of a circuit C the obfuscator $\mathcal{O}$ output a circuit $\mathcal{O}(C)$ but also outputs a secret key k
- The $Token$ generator outputs a token on the input of the secret key k and x an input of C such that:

$$(T_x = Token(k, x)) \implies (\mathcal{O}(C)(T_x) = C(x))$$

Their construction essentially adapts this kind of obfuscator in the white-box model by storing the token generator in the hardware, while the obfuscated circuit is freely executed on software on the input of a token given by the hardware.

If the theoretical security of this construction is well-grounded in obfuscation works, the authors however do not elaborate on the concrete complexity of their construction, which is usual for applicable white-box designs. The construction of [60]- that is a ground work for this construction - use universal circuits as a basis of their construction, which begs the question of efficiency if it is used with multiple garbled circuits, especially if we consider similar constructions already exist since a decade in the obfuscation literature (see next section).

Construction from Hardware-Based Obfucation

The idea of using hardware as a crutch to achieve notions that we do not know how to achieve in the standard model is not new. Here, we study the example of obfuscators build with hardware functionalities. As we have seen in section 2.3.3, using obfuscation to achieve white-box properties is a natural idea. The example we give here can be concretely implemented and are very flexible in their usage. As many construction of cryptographic obfuscators, the programs here are represented by circuits.

It is interesting to note that these kinds of constructions are not mentioned in the state of the art of hardware module white-box papers. We find this rather surprising considering the similarities between the goal they achieve, and the techniques they use.

The TCC2010 Construction In the context of building an obfuscator, a line of work attempted to circumvent the impossibility result of Barak et al. by using hardware - tamper-proof hardware in the obfuscation literature. The work of [64] is the first to achieve it with stateless hardware, that is, hardware that does not have to kept track of each computation made chronologically.

To build the obfuscator, the authors first consider a universal circuit, that is, a circuit U such that for any circuit C of reasonable size, and x its input, we have :

$$U(C, x) = C(x)$$

Using such circuit allows to reduce the problem of obfuscating the circuit to hiding its input through computations made by the universal circuit. Then, each circuit bit is encrypted with a non-malleable encryption scheme E with secret key k.

The software part of the obfuscator is then composed of the encrypted circuit and the universal circuit and the hardware contains the secret key k and the encyption and decryption algorithms E and E^{-1}. To evaluate the circuit C on input x, one first needs to encrypt each bit of x using the hardware. Then, for each gate of U, the user

computes the ouput bit in the following way : the user sends the two encrypted bits to the hardware that decrypts them, computes the gate and encrypts the output, to send it back to the user.

This simplistic description however have some problems. For instance, a malicious user can change values on wires by using the encryption algorithm of the hardware - as the order and values of inputs are not fixed. To circumvent these kinds of problems with stateless hardware the authors force the execution of the program to be made in the correct order, for correct inputs. To do so, the author achieve to assign to each computation what they call an 'execution identity' for the entirety of the execution of the program. This execution identity ensures that there is only a low probability that different executions of the circuit have the same execution identity. For an even simpler construction, the authors propose a stateful construction for which the hardware insert a nonce in all encrypted values. The hardware will then only compute a gate if the wire have the right nonce. For a complete description, refer to [64].

This construction is rather simple in essence as it only uses a hardware with encryption and decryption functionality - and a small check on execution identities. However, there are as many calls to the hardware than there are gates in the universal circuit, which is problematic if a user wants to use a large class of circuits C as input.

The FHE based Construction of Dötling et al. The construction of [49] is based on fully homomorphic encryption and the use of a hardware for decryption.

Similarly to the previous construction, the obfuscator uses a universal circuit to consider the circuit C as an input. The main difference here is that each bit of the circuit is encrypted using the public key of a FHE scheme. The obfuscation software contains a description of U and the encryption of C, while the hardware part is composed of the decryption algorithm of the FHE scheme with the secret key.

The circuit can then be evaluated on input x by encrypting each bit of x and evaluating homomorphically $U(C, x)$. The result then has to be decrypted by the hardware with the secret key. However, this leads to malleability problems similar to the previous construction. Here, to decrypt conditionality to a correct execution, user has to commit to a circuit C by signing its encryption and sending it to the hardware.

This idea is similar to the first GGH13 obfuscator. In their construction, the functionality of the hardware is obfuscated and allow a bootstrapping from low classes of circuits (NC^1) to any polynomial circuit. If the GGH13 construction had implementation issues due to the size the obfuscator on NC^1 circuits, this construction only relies on the security of the hardware and FHE. Using FHE might be more technical than a regular encryption scheme, this construction only makes one call the the hardware for the decryption of the output of the circuit. The circuit of decryption can also be small as there exists FHE schemes with decryption in NC^1.

3.2.3 A Lack of Metrics, Again

The Hardware Module White-Box model has been motivated by concrete use cases found in the industry. However, there is still a huge gap between the literature and the concrete use of white-box cryptography in this context.

A first problem is the lack of a good target problem. What are the most useful hardware to consider as a basis ? What are the main goals of implementations depending on the device ? If the problem of payment application has been studied in theory by Alpirez-Bock et al. [22], there are no concerns for the concrete use of such methods. The TBO based method of [3] also lacks a concrete study of the cost in term of software overhead of this bootstrapping.

Conversely, for designers, what is the metric to be optimizing if we aim at designing future hardware adapted to the white-box model? To what point is it important to minimize hardware functionalities? What are the hardware solution to minimize the implementation overhead? These questions are not discussed and no directions are taken to explore the model on concrete examples. We recall that minimizing hardware functionalities for bootstrapping security properties to software is the sine qua none condition for the usefulness of the model. Without this restraint, we are back at studying hardware security.

Chapter 4

Generic Attacks in the White-Box Model

In the previous chapters, we investigated different white-box security models. In these models, there is one constant: the attacker is a allowed to have full access to a certain amount of the software computing the target cryptographic primitive. The goal of this chapter is to describe the automated attacks that are available to every white-box attacker and that are generic, i.e can be applied to any implementation without any real understanding of the implementation.

From Grey to White As the white-box model is a generalization of the grey-box model of attack, it is natural to consider grey-box attacks and their efficiency against white-box implementations. The two main attack techniques that have been used in the literature are Differential Power Analysis (DPA) [75] and Differential Fault Analysis (DFA) [51].

If the use of these attacks has not been immediate, their use in the white-box model was quickly shown to be powerful. Adaptations of DPA and DFA in the white-box model have broken most of the state-of the-art solutions [20, 28, 68, 99] and greatly questioned the effectiveness of the state-of-the-art implementation techniques. The grey-box attacker is a real threat in the white-box model as it does not rely on a deep understanding of the implementation or an expert reverse engineering.

While being a weaker opponent than the white-box attacker, the grey-box adversary is a first opponent to beat for a designer and resisting its attacks should be a prime goal for the designer.

Original White-Box Attacks The white-box attacker can observe exact computational traces which means that attack techniques that do not have an equivalent in the

grey-box context can be envisioned. The efficiency of these attacks often surpasses the efficiency of grey-box attacks in the adapted situation.

The Linear Decoding Attack (LDA) is one of the prime examples of such attacks. This attack exploit the fact that, due to the techniques used in white-box implementation, there exists linear or low-degree relations between elements of computations traces and the target function. Finding the key can then be reduced to simply solving a linear system. This attack as first been reported to break implementations of the WhiBox19 [107] contest [62] and is particularly adapted to implementations using masking techniques.

One other notable category of attack is the collision-based attacks. Reported in papers like [97] and [111], these attacks abuse the fact that collisions in computation traces are usually equivalent to collisions on target functions. The attack of [97] exploit collisions that are inherent to the round funtion of the AES while [111] exploits collisions that are present in traditional AES implementations.

4.1 Formalizing the framework of automated attacks

The automated attacks we described previously all share the same framework that we describe now. Let us assume that an attacker targets a boolean function ϕ_k with $k \in K$. The attack is divided into tree steps.

First, the attacker collects computation traces of the execution of the program. For any number N of inputs $(x^{(i)})_{i\in\{1,...,N\}}$, the attacker collects any value read or computed by the implementation. This can take the form of values written in any memory location or the values of look-up tables. The values of the traces are acquired and ordered chronologically from the first to the last T-th value. As the trace depend on the input, we note the trace $(v^i)_{i\in\{1,...,N\}}$ where :

$$(v^i) = (v^i_1, ..., v^i_T)$$

The traces can be obtained straight from the attacked implementation or some reverse engineering can be performed to reduce the size of traces. The traces can also be analyzed to remove constant or redundant elements.

After the acquisition of traces, the attacker tries to build a distinguisher D from the traces. The distinguisher gives a score for each possible key :

$$(\gamma_k)_{k\in K} = D((x^{(1)}, ..., x^{(N)}), (v^{(1)}, ..., v^{(N)}))$$

Lastly, the attacker chooses a candidate among the ones with the best score γ_k. The attacker succeeds if the distinguisher correctly matches the best score with the correct key. The probability p is then defined by :

$$p = Pr(k^* = argmax_{k \in K} \gamma_k)$$

If the distinguisher matches not one but few keys to the best score, the attack can still succeed if the obtained key-space is small enough so that a brute-force search can eliminate the false positives.

4.2 DCA

The Differential Computation Analysis is an attack introduced by Bos et al. at CHES 2016 and Sanfelix et al. at Black Hat Europe 2015 to break many of the state-of-the-art white-box implementations of the AES. The DCA is an adaptation of the usual grey-box DPA and exploit the fact that the values of traces are correlated to the values of s-boxes or round functions.

To launch a DCA, the attacker first collects traces of executions of the program. She then chooses a target keyed function and makes a key guess. Then the correlation between the evaluations of the target function and the corresponding traces is computed for each key guess. The key with highest correlation score is chosen as the key candidate, which is then confirmed or denied with a reference implementation.

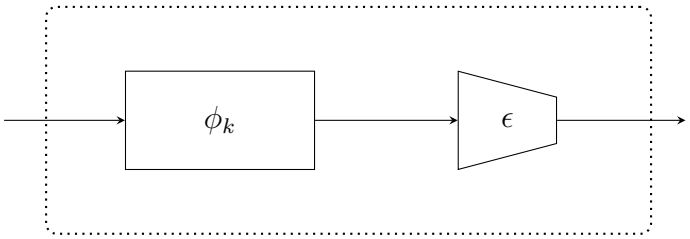
Since its seminal use, this attack has proven to break most AES implementations, in works such as [28, 97]. The most important quality of DCA is that it only requires a partial knowledge of the implementation and can be launched generically and efficiently. The attacker does not need to reverse engineer the implementation, isolate components or modify the code, she only needs to read the values at execution. By its simplicity and effectiveness, the DCA is a very potent tool for the white-box attacker, and providing proven countermeasures against it is still a major open problem in the plain white-box model.

Although it was quickly proven to be effective against the state-of-the-art implementations of AES, the reasons of why it was so efficient were not well understood. Further works with finer analysis lead to a better understanding such as [19, 20]. For this thesis, we follow the precise analysis of [97] that estimates the probability of success of the attack when the target is 'encoded' . While this might seems like a restriction, all the known techniques of open design implementations are using encoding techniques. As a consequence, their work helps to understand the effectiveness of DCA against all known open design candidates.

4.2.1 Establishing Probability of Success

The DCA attack is very generic but can only be used with certain assumptions. Let us assume that the program P computes the function f. Two requirement to the success of DCA is that the input of P can be chosen and that the program P, on the input of x output $f(x)$ were x is the chosen input. The requirement ensures that the program does not hide the input-output behavior of the program through external use like external encodings (extensive description of the encoding strategy of the AES on part II).

For the rest of this analysis, we follow [97]. While their analysis is not generic, the work of [97] gives us a good understanding of the efficiency of DCA against internal encoding implementations. Initiated by Chow et al. this technique is the most common one to build AES implementation. Formally, the target function $\phi_k : \mathbb{F}_2^n :\rightarrow \mathbb{F}_2^m$ is encoded by a bijection $\epsilon : \mathbb{F}_2^m \rightarrow \mathbb{F}_2^m$. The composition $\bar{\phi_k} = \epsilon \circ \phi_k$ can be observed in the evaluation traces.



$\overline{\phi_k}$

Definition 14 (Correlation Coefficient). *Let f and g be two balanced boolean functions from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. We define the correlation coefficient between f and g by :*

$$Cor(f,g) = \frac{1}{2^n}\sum_{x\in\mathbb{F}_2^n}(-1)^{f(x)+g(x)}$$

Definition 15 (Correlation Distinguisher). *Let $v = (v_1, ..., v_T)$ be a computation trace for an input x. For $1 \leq j \leq T$, let V_j be the random variable that returns v_j on input x. For any key guess k^*, we define the DCA distinguisher by :*

$$\gamma_{k^*} = \max_{1\leq j\leq T} |Cor((\phi_{k^*})_i, V_j)|$$

If for coordinates j where $V_j = \bar{f}_j$ we naturally have:

$$\max_{1\leq j\leq m} |Cor((\phi_{k^*})_i, V_j)| = \max_{1\leq j\leq m} |Cor((\phi_{k^*})_i, (\bar{\phi_k})_j)|$$

Proposition 3 (Adaptation from RW19). *Let k^* be the correct key guess and $k^\times$ a wrong one. The probability of success of DCA is :*

$$p = Pr[\max_{1\leq j\leq m} |Cor((\phi_{k^*})_i, (\bar{\phi_k})_j)| > \max_{1\leq j\leq m} |Cor((\phi_{k^\times})_i, (\bar{\phi_k})_i)|]$$

The probability p only depends on the encoding ϵ. If $n \geq 2m+2$ the probability p can be estimated by :

$$p \approx 1 - \frac{\binom{2^{m-1}}{2^{m-2}}^{2m}}{\binom{2^{m-1}}{2^m}^m}$$

The result of [97] is more complete shows how to evaluate the probability of success for other parameters regimes. The regime of parameter $n \geq 2m+2$ is however very commonly found in the state if the art of AES implementation. For instance, the seminal implementation of Chow et al. uses 4-bit encodings for 8-bit S-boxes, which brings the probability of success of the attack at more than 0.92. However, when m is equal to n, the probability quickly approaches 0.

The DCA attack can be extended to 'high-order' DCA by using correlation between linear combinations or products of elements of the traces instead of individual elements. We refer to [105] for more details.

Complexity Once the traces have been acquired, a DCA is made by computing the correlation scores of all the traces for each potential target. The authors of RW19 then evaluate the complexity of DCA at :

$$\mathcal{O}(T \times N \times 2^k)$$

where T is the size of each trace, and N the number of traces. In the case of $n \geq 2n+2$, the number of traces is estimated by [97] to be $N = \mathcal{O}(2^{2m})$.

4.2.2 Countermeasures

As protecting implementations with open designs against DCA is still an open problem, providing countermeasures against DCA is not entirely solved. There are however many techniques, mostly inspired by the grey-box state of the art that can be proposed to diminish its efficiency.

Linear Masking The first idea is to adapt linear masking techniques in the white-box context. Introduced to protect hardware implementations against DPA [36, 61], the goal of this technique is to share secret variables into n parts:

$$x = x_1 + ... + x_n$$

where the x_i are random. The computations are made on the share instead of the variable x and recombined at the end for correctness. In essence, if only $n-1$ shares are known, no information is leaked about x. However, this technique assume that a generator of fresh randomness can be accessed by the implementation. Otherwise the bias in the distribution of shares is detrimental to the implementation - see [81] for context in the grey-box model. As it is impossible to reveal the design of a PRNG in the plain white-box model and expect the pseudorandomness of its output, this forces designers using masking to obfuscate random number generators of unknown design into their implementations. In practice, implementations using these techniques have be defeated, as shown by the WhiBox contest.

Different Encoding Techniques In the context of encoding techniques, the efficiency of DCA heavily depends on the size of the encoding. One can then hope to use bigger encodings to thwart the attack. This rationale has been used in implementations such as [77], with only partial success. The main problem comes from the existence of function with large input size that force large encodings to be used in the implementation. With such large encodings, the size of the implementation quickly explodes, making it a debatable countermeasure. This explosion is however somewhat inherent to the SPN structure of the AES and the implementation techniques used. More on this topic in part 2 chapter 2.

Remark: Shuffling Methods and Dummy Computations are methods used in hidden design implementations to limit the effectiveness of DCA. The goal of these methods is to disrupt the flow of execution of the program by shuffling the computation in memory and time and introducing random computation to augment the overall noise of the implementation. We do not detail them as we want to focus on open design countermeasures. For the curious reader, we refer to [105] for a detailed study.

4.3 Collision Attack

Introduced by [97], the collision attacks is a variant of the DCA where the distinguisher is based on collisions happening on round function instead of a classic correlation score. The main idea is the following : collisions on an encoded function $\epsilon \circ \phi_k$ are equivalent to collisions on ϕ_k. This remarks stands for most of the implementations in the state

of the art. For this section we assume that the target function ϕ_k is from n bits to m bits and that $n > m$.

The attack is launched as follows:

- For each key guess, group inputs if they collide after ϕ_{k^*} :

$$I_{v,k^*} = \{x | \phi_{k^*}(x) = v\}$$

- Define the correlation function - that is equal to 1 if the target function collides on x and x', 0 otherwise :

$$\Phi_k(x, x') = NOT(\phi_{k^*}(x) \times \phi_{k^*}(x'))$$

- For any two inputs x, x' in I_{v,k^*}, compute correlation traces :

$$w_i^{x,x'} = NOT(v_i^x \times v_i^{x'})$$

- If $W_j^{(x,x')}$ is the function that outputs the j-th element of the correlation trace $w_i^{x,x'}$ the score for a key guess k^* is defined by:

$$\gamma_{k^*} = \max_{1 \leq j \leq T} |Cor(\Phi^k *_i, {W_j}^{(x,x')})|$$

Remark : The Mutual Information Attack is a variant of collision attacks where the distinguisher is based on mutual information. As the authors of [97] note, it can be seen as a regular MIA in the grey-box model where the encodings can be seen as complex leakages. The complexity of MIA attacks in the white-box model is similar to the collision one. As they share many similarities, we do not detail it here. We refer to [97] for any precision.

4.3.1 Probability of Success of the Attack

Similarly to the DCA, the analysis of [97] estimates the probability of success of the attack for random boolean function encoded by n-to-m bits bijections.

Proposition 2. *Let N be the number of acquired traces. If for all key guesses, the functions ϕ_k are mutually independent and balanced the probability of the collision attack is lower-bounded by :*

$$p \geq 1 - \#K \times exp(-\frac{(N-1)(N-2)}{2^{m+1}})$$

Complexity To perform this attack, one only needs to compare collisions observed on $\phi_{k^\star}$ and the observed encoded value $\epsilon \circ \phi_k$. In their paper, the authors of [97] note that when used on state-of-the-art implementations, the trace complexity of collision attacks is way smaller than the one of regular DCA. They theoretically estimate the complexity of the attack to be:

$$\mathcal{O}(T \times N^2 \times 2^k)$$

4.3.2 Countermeasures

Countermeasures to collison attacks have not been discussed in the literature. A remark that is inherent to the attack is that it assumes the use of non-injections un the implementation. This is however very common against SPN due to their structure. A simple idea to thwart collision attacks is to transforms the implementation so that it contains bijective transformation. In the case of encoding techniques for instance, this could lead to the absence of observations of collisions and any attempt at collision attacks would fail.

4.4 LDA

The Linear Decoding Attack (LDA) exploit the fact that, due to the techniques used in white-box implementation, there often exists linear or low-degree relations between elements of computations traces and the target function. Finding the key can then be reduced to simply solving a linear system. This attack has first been reported to break implementations of the WhiBox19 contest [63] and is particularly adapted to implementations using linear masking techniques.

4.4.1 Rationale of the Attack

The first goal of LDA is to tackle linear masking of sensible variable and to propose a more effective attack than DCA in this context. If a sensible variable ϕ_k is masked in an implementation, there exists a linear relation between some of the values of the computation traces and the target ϕ_k. For the rest of the section, we assume that there exists a linear relation between elements of computation traces such that:

$$\phi_k(x) = \lambda_0 + \sum_{i=1}^{t} \lambda_i v_i$$

An attacker with many execution traces on input x_i can then build a matrix on computation traces and try to solve the following system for a key guess k^*:

$$\begin{bmatrix} 1 & v_1^{(1)} & \dots & v_T^{(1)} \\ 1 & v_1^{(2)} & \dots & v_T^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & v_1^{(N)} & \dots & v_T^{(N)} \end{bmatrix} \times \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_N \end{bmatrix} = \begin{bmatrix} \phi_{k^*}(x_0) \\ \phi_{k^*}(x_1) \\ \vdots \\ \phi_{k^*}(x_N) \end{bmatrix}$$

Hopefully, the system only solves for the right key guess $k^* = k$ or a small subspace of the key-space and the correct key can be recovered.

While this attack works against linear masks or encodings, it can also be applied to higher degree masks or to certain forms of encodings. Indeed, if there exists a polynomial relation of degree d between values of a trace and the target function, one can relinearize the problem by considering a linear system depending on the monomials of the values of the traces. For a trace of execution $(v_j^{(i)})_{j\in[\![1,T]\!]}$ on input $(x^{(i)})$, we compute the list of monomials for any order $Mon_d((v^{(i)}))$ of degree d, where d is the supposed degree of the masking/encoding. Then the linear system:

$$(Mon_d((v^{(i)})))_{i\in[\![1,\sigma(T,d)]\!]} \times \lambda = (\phi_k(x^{(i)})_{i\in[\![1,\sigma(T,d)]\!]}$$

admits a solution $\Lambda \in \mathbb{F}_2^{\sigma(T,d)}$ if a relation exists for the guessed value k^*. To formalize a distinguisher according to the attack framework of 4.1, the score γ_{k^*} is equal to 1 if the vector $(\phi_{k^*}(x^{(i)})_{i\in[\![1,\sigma(T,d)]\!]}$ is in the image of the matrix we build for the attack, 0 otherwise. We note d-LDA a LDA of order d.

4.4.2 Complexity and Probability of success

We follow the analysis of [62] for the estimation of the probability of success of the attack.

This estimation is made under 3 assumptions :

- There exist a linear relation between elements of the trace and the target function ϕ_k.
- The plaintext x input of the traces is uniformly distributed.
- The traces are uniformly distributed among the t-uples that verify:

$$\phi_k(x) = \lambda_0 + \sum_{i=1}^{t} \lambda_i v_i$$

While the first hypothesis is necessary for LDA to be of any use and the second is natural as it is usual in the white-box context, the third hypothesis is an ideal assumption to help the formal analysis. This assumption can be challenged to include dependency between variables, but it would harden any estimation of the success probability. The proof of this property can be found in [62]

Proposition 4. *Let $T \geq N$ and $v_i^{(j)}$ $1 \leq i \leq T$,$1 \leq j \leq N$ satisfying the 3 above assumptions. The probability that the system is solvable for an incorrect key guess $k^\times \neq k^*$ is lower than q^{N-T-1} where q is defined by :*

$$q := \mathbf{max}(\{Pr(\Phi_{k^\times}(X) = L(\Phi_{k^*}(X)))|L : \mathbb{F}_2^n \to \mathbb{F}_2^n, L\ linear, (k^*, k^\times) \in K^2\})$$

with uniform X.

As a consequence, the probability to have false positives that are not linearly dependent to the target is exponential in q. It means that for enough traces, there will be one only positive with overwhelming probability. For instance, if we set the target ϕ to be a bit of the S-box of the AES, $q = \frac{9}{16}$ which makes the probability of failure already small with very few traces.

Complexity If we assume ϕ, N and T are such that there is a unique key with overwhelming probability, the complexity of the attack is dominated by checking if the vector $(\phi_k(x^{(i)})_{i \in [\![1,\sigma(T,d)]\!]}$ is in the image of the matrix $(Mon_d((v^{(i)})))_{i \in [\![1,\sigma(T,d)]\!]}$. To do so, one can use the LU decomposition of the matrix which can be computed in $\mathcal{O}(n^\omega)$. This means that the attack can be made in:

$$\mathcal{O}(\sigma(T,d)^\omega \times 2^k)$$

This makes LDA a very efficient attack against linear masks or encodings even if the design is not known. When the attacker has a better idea of the attack window, it is as efficient against higher order countermeasures.

4.4.3 Countermeasures

As the LDA is essentially DCA with an algebraic distinguisher, the countermeasures are similar to those of DCA, with a focus on the algebraic degree of the function involved. Techniques to increase traces are still efficient to a certain degree.

While the state of the art has not been bubbling with papers on the topic, one can note the non-linear masking of [16] and its extension by [101]. Implementations uisng this countermeasure force attackers to perform higher degree LDA which pushes

the complexity of attacks on unknown designs to their limits against obfuscated implementations. However, these techniques still need PRNGs to provide any security.

There is undoubtedly more exploration to be done against LDA. For instance, one could try to minimize the probability q for all degrees up to a certain threshold. While this would be the case for high degree masking techniques, we believe that this strategy could be used for encoding based implementations such as it is common for open-design implementations. More on this topic will be seen in part II.

4.5 BCA

The Bucket Computational Attack is a variation of the Statistical Bucket Attack that was introduced by Chow et al. to break their first white-box DES implementation. Revisited by Zeyad et al. [111] to be adapted to the AES, this attack is based on the existence of collisions in existing AES implementations. In that sense, it is similar to the collision attacks of [97]. However, it targets non-injection that are caused by the implementation, not due to the structural collision of the underlying algorithm. This attack has only been described for DES and AES implementations, so we attempt here to abstract it in our usual framework.

The seminal setting for this attack is the special encodings of Chow et al. implementation. In that setting, the keyed S-box of the AES is encoded by a 4-to-4 bit bijection, building a 8-to-4 bit non-injection. As the encoding is a bijection on 4 bits, having collision on the four leftmost bit of the S-box will be equivalent to have collision on the encoding.

In the framework of this chapter, we assume that the target function is of the form $\phi_k(x) = \phi(x + k)$ for a bijection ϕ. Then, the m leftmost bits of target function $\phi_k : \mathbb{F}_2^n \to \mathbb{F}_2^n$ are encoded by a bijection $\epsilon : \mathbb{F}_2^m \to \mathbb{F}_2^m$ so that the encoding $\epsilon \circ \phi_k[1 : m]$ can be observed in the implementation. To build a distinguisher, we now build a distinguisher in the following way:

- Compute, from the computation traces, the values $\epsilon \circ \phi_k[1 : m](x)$ for each $x \in \mathbb{F}_2^n$
- Choose two m-bit values v_0 and v_1
- Form two sets for each key guess k^* I_{0,k^*} and I_{1,k^*}, such that:

$$I_{b,k^*} := \{x | \phi[1 : m](x + k^*) = v_b\}$$

- Then, define for each key guess the subsets of traces - computed from the implementation:

$$V_{b,k^*} := \{\epsilon \circ \phi_k[1:m](x) | x \in I_{b,k^*}\}$$

- For each key guess k^* compute the score:

$$\gamma_{k^*} = \#\{i | V_{0,k^*}[i] \neq V_{1,k^*}[i]\}$$

The correct key guess is attributed to the biggest score as usual.

4.5.1 Estimating the Success of the Attack

This attack has been successfully used to break the usual white-box implementations of the AES such as the CHES 2016 challenge or Lee et al. CASE 1 algorithm. The author explain that their attack achieves lower traces size compared to DCA is most cases and similar time for complete key recovery. However, no extrapolation to other settings has been given by the author. An extension to the second order has been given in [80] with similar experimental arguments.

Complexity As the attack needs to compare the collisions of the target function and the elements of the trace on each input, for each key guess, the complexity is similar to a DCA:

$$\mathcal{O}(T \times N \times \#K)$$

However, the experimental analysis of [111] on candidate white-box implementations with nibble encodings shows that the size T of traces is overall smaller than DCA for complete key-recovery.

4.5.2 Countermeasures

Similarly to Collision Attacks, BCA can be thwarted if non-injections are difficult to analyse in the program. One can for instance use bigger encodings to force the attacker to look for collisions in other potential target functions, with an hopefully bigger key-space.

4.6 DFA

The Differential Fault Analysis (DFA) is an active attack that was originally applied in the grey-box context. The goal of the attack is to disrupt sub-computations of a key-dependent cryptographic algorithm to collect faulted executions that can help to

recover the secret key. This attack was pioneered by Boneh et al. in 1996 for public-key cryptosystems[27] and then extended to DES by Biham and Shamir [14], and to AES by Dusart et al. [51]

The difficulty of using fault attacks in the grey-box model of attack is to precisely fault the device to recover useful information. If the hardness of this practical aspect has been considerably diminished, exactly faulting an implementation is trivial in the white-box context. As it was shown by Sanfelix et al. [99], DFA is a potent method to break white-box implementations.

While the method is well-known to be an important toolbox of the white-box attacker, there are to very few analyses of the efficiency of the attack against various targets and fault models. This fact can partly be attributed to the fact the automated attacks are the main focus of the literature for they are agnostic to the implementation. Implementations are often obfuscated through obscurity and fault attacks often need reverse engineering to be applied.

In view of the state of the art, it would be presumptuous to draw generic conclusions on the probabilty of success of fault attacks. They indeed differ greatly depending on the algorithm and the implementation technique used. For most of the state of the art, DFA on white-box implementation can be reduced to DFA against implementations in the grey-box model.

In regards to these similarities, the countermeasures to grey-box DFA are similar to white-box DFA. They often come in the form of redundancy added to the implementation or error correcting codes embedded in the implementation to cancel certain types of faults. One can note parity-checks [12, 70] for block-ciphers, invariance-based codes [65] for the AES. If these techniques might be useful in the white-box context, it is only conditional to the fact that this redundancy cannot be separated from the rest of the implementation which is harder to guarantee.

Part II

Implementation of AES in the White-Box Model

Chapter 1

State-of-the-art : Implementing the AES in the White-Box Model

In this chapter, we propose a state-of-the-art synthesis of open white-box implementations techniques of the AES-128 primitive in the plain white-box model. We choose to take a chronological approach to this synthesis. Indeed, we think that explaining constructions and attacks in reaction to these constructions leads to a better understanding of design development. Without the historical context, some of the constructions might feel too optimist in their security claims.

This chapter starts with a short summary of the research timeline on white-box implementations. We focus only on published open designs. Then, we detail the most notable constructions in the seminal table-based rationale of Chow *et al.*. We also detail the new attacks that were used to break these constructions at the time. We then continue this state-of-the-art synthesis by detailing two polynomial based solutions and their cryptanalysis. At the end of the chapter, we propose an overall security analysis of all these constructions against state-of-the-art white-box attacks.

1.1 Description of the AES-128 Primitive

The standard known as Advanced Encryption Standard (AES) is a symmetric encryption algorithm that won the 'AES competition' launched by the American National Institute of Standards and Technology (NIST) in 1997. The goal of this competition was to define a new standard to replace the 3DES. The RIJNDAEL candidate, designed by Rijmen and Daemen [41], was chosen among the candidates to be the winner of the contest and is now called AES.

The AES algorithm is a block-cipher with a 128 bit block size, and with three possible key sizes - 128, 192 or 256 - for different security levels. By design, it is a

Substitution Permutation Network (SPN) that is composed of 10, 12 or 14 rounds for the respective key sizes 128, 192 or 256. We focus here on the 128 bit version, as it is the most commonly found in the state-of-the-art implementations.

In the AES specification, the operations are described at the byte level. Bytes are represented as elements of $\mathbb{F}_{2^8}$ with the fixed representation $\mathbb{F}_{2^8} = \mathbb{F}_2[X]/(X^8+X^4+X^3+X+1)$. Elements in the field are represented by two words in hexadecimal such that, for any element $b_0+b_1X+b_2X^2+b_3X^3+b_4X^4+b_5X^5+b_6X^6+b_7X^7$ is mapped to 0x$(b_0b_1b_2b_3)_2(b_4b_5b_6b_7)_2$. In the AES specification, the prefix "0x" is removed when it is obvious due to the context.

The AES algorithm is composed of 10 rounds. These rounds are described on a 4 by 4 state of bytes $s = (s_{i,j})_{i,j\in[\![1,4]\!]}$. From the master key K of 128 bits, eleven round key K^t of 128 bits each are derived by the key schedule of the algorithm, stored in a 4 by 4 array $(K^t_{i,j})_{i,j\in[\![1,4]\!]}$). One round of AES is the composition of four algorithms: AddRoundKey, SubBytes, ShiftRows and MixColumns. We now describe these algorithms.

- On the input of $t \in [\![0, 10]\!]$ and a state s, **AddRoundKey** that $XORs$ the i-th round key to the state:

$$\mathbf{AddRoundKey}(s, t)_{i,j} = s_{i,j} \oplus K^t_{i,j}$$

- On the input of a state s, the algorithm **SubBytes** applies the sbox SB to each byte of the state:

$$\mathbf{SubBytes}(s)_{i,j} = SB(s_{i,j})$$

  where SB is the composition of the multiplicative inverse on $\mathbb{F}_{2^8}$, extended to 0 by 0 and the affine map defined by:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

- On the input of a state s, **ShiftRows** permutes bytes of the state:

$$\mathbf{ShiftRows}(s)_{i,j} = s_{i,(j+i)\,mod\,4}$$

- On the input of a state s, **MixColumn** performs a linear transformation on each 'column' of the state:

$$\mathbf{MixColumn}(s_{1,j}, s_{2,j}, s_{3,j}, s_{4,j}) = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} s_{1,j} \\ s_{2,j} \\ s_{3,j} \\ s_{4,j} \end{bmatrix}$$

  For future use, we note MC the MixColumn matrix and MC_i its columns.

The AES algorithm starts by AddRoundKey(x,0). Then, it performs 9 rounds composed of SubBytes, ShiftRows, MixColumns and AddRoundKey. The last round is only composed of SubBytes, ShiftRows and AddRoundKey.

Definition 16. *For $1 \leq t \leq 9$ and $1 \leq i, j \leq 4$ and $(K^0, ..., K^{10})$ the round keys after key-schedule. The four 32-to-32 bit round functions $R^t_j(x_1, x_2, x_3, x_4)$ of the AES-128 are defined by:*

$$R^t_j(x_1, x_2, x_3, x_4) := MC_1 \times SB(x_1 + K^{t-1}_{1,j}) + MC_2 \times SB(x_2 + K^{t-1}_{2,j})$$
$$+MC_3 \times SB(x_3 + K^{t-1}_{3,j}) + MC_4 \times SB(x_4 + K^{t-1}_{4,j})$$

The output state of a round is the collections of $R^t_{i,j}(x_1, x_2, x_3, x_4)$, where $R^t_{i,j}$ is the i-th byte of the function R^t_j.

For $t = 10$

$$R^{10}_{i,j}(x) = SB(x + K^9_{i,j}) + K^{10}_{i,j}$$

Remark: We do not describe the key-schedule as it is not useful to the understanding of the rest of this chapter. For sake of completeness, it can be found in [41]

The Standard Implementation The AES submission document [41] contains a reference implementation. As it is the starting point for many implementation techniques including the seminal Chow *et al.* technique, we include it for reference. The goal of this implementation is to be optimized on 32-bit processors. To do so, the authors reduce the implementation to 8-to-32-bit look up tables and XORs between the output of these tables.

To build the implementation, the authors use the commutation between some of the operations of the round function. First, ShiftRows and AddroundKey commute as they are linear on bytes, hence:

$$ShiftRows(AddRoundKey(S, K^t)) =$$
$$AddRoundKey(ShiftRows(s), ShiftRows(K^t))$$

Then, the MixColumn transformation can be reduced to the sum of 8-to-32-bit transformations by the following equation:

$$\mathbf{MixColumn}(s_{1,j}, s_{2,j}, s_{3,j}, s_{4,j}) = \begin{bmatrix}02\\01\\01\\03\end{bmatrix} \times s_{1,j} + \begin{bmatrix}03\\02\\01\\01\end{bmatrix} \times s_{2,j} + \begin{bmatrix}01\\03\\02\\01\end{bmatrix} \times s_{3,j} + \begin{bmatrix}01\\01\\03\\02\end{bmatrix} \times s_{4,j}$$

If we note:

$$SB'_1(x) = \begin{bmatrix}02\\01\\01\\03\end{bmatrix} \times x\,,\ SB'_2(x) = \begin{bmatrix}03\\02\\01\\01\end{bmatrix} \times x\,,\ SB'_3(x) = \begin{bmatrix}01\\03\\02\\01\end{bmatrix} \times x\,,\ SB'_4(x) = \begin{bmatrix}01\\01\\03\\02\end{bmatrix} \times x$$

The (i, j)-th block of 8 bits of the round function is then the XOR of the key and the functions SB' on permuted inputs :

$$SB'_1(s_{0,jmod4}) + SB'_2(s_{2,j+1mod4}) + SB'_3(s_{3,j+2mod4}) + SB'_4(s_{4,j+3mod4}) + K^t_{i,j}$$

The authors conclude that the AES round function can then be made made with 4 look-up tables SB' and 4 XORs per column and per round. This implementation weights approximately 4kB and can use a lot of parallelism, for fast performances. While it does not provide any security in the grey or white-box models, it is often used as a comparison point for performances uses.

1.2 State-of-the-Art of the Table Based Rationale

The first part of the state of the art contains table-based solution. Started by the seminal implementation of Chow *et al.*, this technique of implementation represents sub-computations of the AES round functions by look-up tables. The goal of this strategy is to use look-up tables as incompressible elements in which the randomness introduced by the compiler hides secret information related to the key.

1.2.1 The Seminal Chow *et al.* Construction

The 2002 paper of Chow et al. [37] introduces the concept of white-box cryptography and proposes an implementation of the AES that is designed to stand against key-extraction in the white-box model. While their implementation has been broken by structural and automated attacks, their work laid foundation for several other constructions using the same rationale of design.

The first design techniques used in their implementation is to use the 'internal encoding' technique so that the key does not leak from sub computations of a round function. Roughly, one can represent the AES algorithm by the composition of N 'small' algorithms $(E_i)_{i\in[\![1,N]\!]}$ and to be conjugated by bijections that are cancelled from one small algorithms to the other.

Definition 17. *We say that an implementation of an algorithm $\mathcal{A} = \bigcirc_{i=1}^{N} E_i$ is using the 'internal encoding' rationale if it has been decomposed in the form:*

$$\mathcal{A} = \bigcirc_{i=1}^{N} E'_i$$

where

$$E'_1 = g_1 \circ E_1,\ E'_i = g_i \circ E_i \circ g_{i-1}^{-1},\ E'_N = E_N \circ g_{N-1}^{-1}$$

and g_i are bijections. For E'_i, g_i is called the input encoding and g_{i-1}^{-1} the output encoding.

In their paper, Chow *et al.* also propose to use external encodings, that is, to transform E'_1 into $E'_1 \circ f_{in}$ and E'_N into $f_{out} \circ E'_N$ to encode inputs and outputs as well. However, these transformations have to be hidden in the implementation as well. It is hence not a good design method in the plain white-box model. However, since it is the first design ever proposed, we include it with external encodings in the state-of-the-art

Definition 18. *We say that an implementation of an algorithm $\mathcal{A} = \bigcirc_{i=1}^{N} E_i$ is using the 'external encoding' rationale if it has been decomposed in the form:*

$$\mathcal{A}' = \bigcirc_{i=1}^{N} E'_i$$

where

$$E'_1 = E_1 \circ g_{in},\ E'_i = E_i,\ E'_N = g_{out} \circ E_0$$

and g_{in} and g_{out} are bijections. The description of g_{in}^{-1} and g_{out}^{-1} are necessary to correctly compute $\mathcal{A}$.

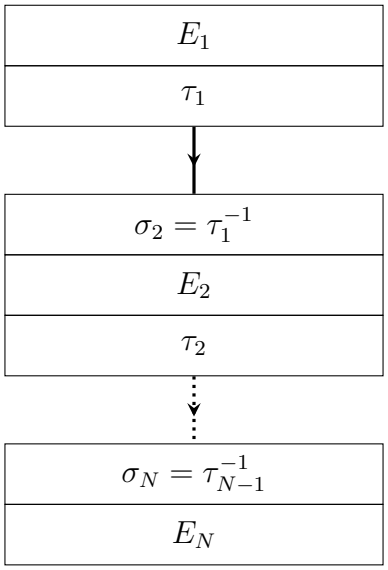


Figure 1.1: Internal Encodings

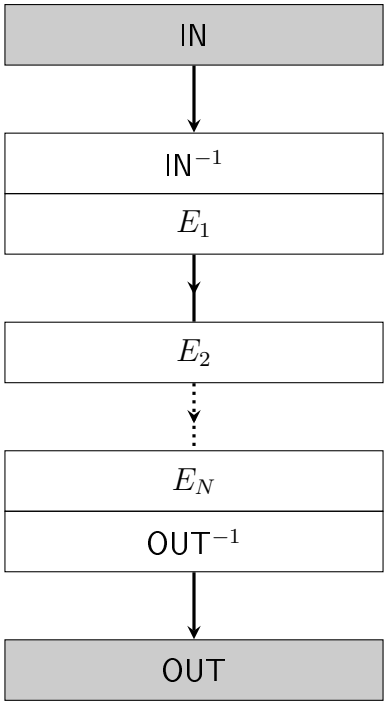


Figure 1.2: External Encodings

For this rationale to work, the transformations E'_i need to hide the bijections g_i. Indeed, if g_i can be recovered, an attacker can then trivially recover E_i to break the implementation. To circumvent this problem, the authors break the round function of the AES into small key-related transformation that can be encoded and represented by their look up tables. The main idea is that not leaking intermediate values can help to make the look up tables look random to an attacker, which is of course related to the unbreakability security notion we presented in part I.

Definition 19. *We say that an implementation is a look-up table network if it can be written by composition of 'small' look-up tables that are accessed to in a specific order.*

This rationale of design has given birth to variations that we will discuss later in this section.

1.2.2 Chow et al. AES implementation

In the formalism of part 1, Chow et al. propose a white-box compiler for AES, that is, their propose an algorithm that, on the input of a key, outputs a white-box implementation, with randomness drawn at random during the compilation when needed.

Decomposing the AES round function

T-boxes and MixColumn Similarly to the reference implementation, the method of Chow et al. starts by defining look up tables that allow to compute the round function, that is, the composition of AddRoundKey and SubBytes :

$$T^t_{i,j}(x) = SB(x + K^{t-1}_{i,j})$$

$$T^{10}_{i,j}(x) = SB(x + K^9_{i,j}) + K^{10}_{i,j}$$

They then compose these function with MixColumn for rounds 1 to 9, like in the reference implementation to obtain:

$$TMC^t_{1,j}(x) = \begin{bmatrix}02\\01\\01\\03\end{bmatrix} \times T^t_{1,j}\,,\ TMC^t_{2,j}(x) = \begin{bmatrix}03\\02\\01\\01\end{bmatrix} \times T^t_{2,j}\,,$$

$$TMC^t_{3,j}(x) = \begin{bmatrix}01\\03\\02\\01\end{bmatrix} \times T^t_{3,j}\,,\ TMC^t_{4,j}(x) = \begin{bmatrix}01\\01\\03\\02\end{bmatrix} \times T^t_{4,j}$$

The idea is to represent the key dependent computations succinctly by 16 8-to-32-bit look-up tables.

XOR tables Then, like it was noted for the reference implementation, the round function can be computed by XORing the output of the tables TMC. However, a round still needs four 32 bit values to be XORed. To be efficient with encodings later, the four 32-to-32 XOR table are each split into 32 tables T_{xor}, XORing two 4 bits values, 8 per quarter of state.

Encoding Phase The compiler can now randomize the XOR tables and the maps $TMC^t_{i,j}$ following the internal and external encoding rationals. We now describe the look up tables that compose the implementation after encoding :

- $IN_{i,j}$ encodes the 8-to-128 bits mapping composed of the inverse of the external g^{-1}_{in} input encoding and a linear bijection. The input encoding of the lookup table is a 8-bit non-linear bijection and the output encoding is a concatenation of 32 4-bit encodings.
- $\tilde{TMC}^t_{i,j}$ encodes the tables $TMC^t_{i,j}$. The input encoding is a concatenation of two 4-to-4-bit random bijections composed by a linear 8-to-8-bit bijection. The output encoding is in the inverse sense, that is a linear 32-to-32-bit bijection followed by the concatenation of 8 4-to-4-bit bijections.
- $\tilde{T}^{10}_{i,j}$ encodes the last round map $x \rightarrow SB(x + K^9_{i,j}) + K^{10}_{i,j}$. The input encoding are of the same form as the other T-box input encodings with the addition of the output encoding g_{out} that is composed of 32 4-bit bijections.
- $\tilde{T}_{xor}$ encodes the XOR of two 4 bits words. As the input encodings have to be compatible with the the other tables, they are of the same form, that is, concatenation of 4-bit bijections.
- $\tilde{Tlin}^t_{i,j}$ encodes the composition of the round function and a linear map. The encodings are consistent with tables $\tilde{TMC}^r_{i,j}$ by removing the previous output encoding and applying the inverse input encoding.

1.2.3 Cryptanalysis

The early cryptanalysis of Chow *et al.* implementation was made difficult by the use of external encodings. Indeed, the astute reader would have noticed that all the attacks of part 1 chapter 4 break this implementation without external encodings. We detail the BGE attack, the first attack to break Chow et al. implementation with external

encodings. Historically, the collision attack of Lepoint and Rivain [78] was also used to great success. As it is similar to collisions and bucket attacks, detailed in Part 1, due to the flaw it uses, we do not describe it here.

The BGE Attack

The BGE attack [15], named after its authors, Billet, Gilbert and Ech Chatbi, is an elegant attack that was developed to break the external encoding implementation of Chow et al. in about 2^{30} operations. Hence, it is also a potent attack without any external encodings, and is an important tool to cryptanalyse white-box implementations of AES and other SPNs. As we will see, the attack fully rely on the SPN structure of AES and the internal encoding paradigm to succeed, which means it is not a priori a generic attack against all white-box designs.

We present here the version of the attack that was used to break Chow et al. implementation, but it is important to note that it was later generalized to more complex implementations later by Michiels *et al.* [84] to break implementations with the same rationale of design. More on this attack will be given below.

The attack is based on the fact that a composition of the right tables in the implementation leads to having access to an encoding of the round function. This remark is obvious since the composition of $\tilde{TMC}$ and $\tilde{T_{xor}}$ is by definition an encoding of the round function, simply by correctness of the algorithm. What is not obvious however, is how to go around the external encoding strategy, which does not allow to control the input. We focus here on the recovery of the non-linear part of the encodings, as it is the most important part of the cryptanalysis and clearly shows the structural weakness of the encoding method.

Here is the trick the authors use to go around this problem. If $\psi_{i,j}(x_1, x_2, x_3, x_4)$ denotes the 32-to-8-bit function that represents encoded byte of the round function $R^1_{i,j}$, $Q^1_{i,j}$ the output encoding of the last XOR table and $P^1_{i,j}$ the input encoding of the tables TMC, we have the following equality:

$$\psi_{i,j}(x_1, x_2, x_3, x_4) = Q^1_{i,j}(R^1_{i,j}(P^1_{1,j}(x_1), P^2_{2,j}(x_2), P^1_{3,j}(x_3), P^4_{4,j}(x_4))$$

Then, if we note c_1, c_2, c_3, c_4 constants of $\mathbb{F}_2^8$, we can fix 3 out of the 4 bytes of the encoded round function to get:

$$\phi_{i,j}(x_1, c_2, c_3, c_4) = Q^1_{i,j}(02 \times SB(x_1 + K^1_{1,j}) + \alpha_1(c_2, c_3, c_4)).$$

These parametrized 8-to-8-bit functions are bijections since the constants α_i take all possible values in $\mathbb{F}_2^8$. The authors then make the key remark that for any fixed c_2, c_3, c_4 and free x_2 in $\mathbb{F}_2^8$:

$$\psi_{(c_2,c_3,c_4)} \circ \psi^{-1}_{(x_2,c_3,c_4)}(x_1) = Q^1_{i,j}(02 \times SB(y_1 + K^1_{1,j}) + \alpha_1(c_2, c_3, c_4))$$

with:

$$y_1 = (02 \times SB)^{-1}((Q^1_{i,j})^{-1}(x_1) + \alpha_1(x_2, xc_3, c_4)) + K^1_{1,j}$$

Hence, after simplifying the expression:

$$\psi_{(c_2,c_3,c_4)} \circ \psi^{-1}_{(x_2,c_3,c_4)}(x_1) = Q^1_{i,j}((Q^1_{i,j})^{-1}(x_1) + \alpha_1(x_2, c_3, c_4) + \alpha_1(c_2, c_3, c_4))$$

As $\alpha_1(x_2, c_3, c_4) + \alpha_1(c_2, c_3, c_4)$ also describes $\mathbb{F}_2^8$ as x_2 goes through $\mathbb{F}_2^8$ with fixed c_1, c_2 and c_3, we get 2^8 distinct bijections, and if we note $\oplus_\delta$ the function such that $\oplus_\delta(x) = x + \delta$ we have that :

$$\{\psi_{(c_2,c_3,c_4)} \circ \psi^{-1}_{(x_2,c_3,c_4)} |\, x_2 \in \mathbb{F}_2^8\} = \{Q^1_{i,j} \circ \oplus_\delta \circ (Q^1_{i,j})^{-1} |\, \delta \in \mathbb{F}_2^8\}$$

At this point, we have a set that depends only on the internal encodings $Q^1_{i,j}$ which has the excellent property to be a vector space for composition. The following property shows that we can recover $Q^1_{i,j}$ up to an affine transformation.

Proposition 5. *Given the set $V_\delta = \{\epsilon \circ \oplus_\delta \circ \epsilon^{-1} | \delta \in \mathbb{F}_2^8\}$, where $\oplus_\delta(x) = x + \delta$. A map ϵ' can be computed where $\epsilon \circ \epsilon'^{-1}$ is affine.*

Proof. The map $\Phi : V_\delta \to \mathbb{F}_2^8$ defined by $\Phi(\epsilon \circ \oplus_\delta \circ \epsilon^{-1}) = \delta$ is an injection as ϵ is a bjection and a surjection by cardinality. It is then an isomorphism. The knowledge of Φ is enough to recover ϵ, as if $\Phi(g) = \delta$ we have:

$$g(0) = \epsilon \circ \oplus_\delta \circ \epsilon^{-1}(0) = \epsilon(\delta + \epsilon^{-1}(0))$$

which helps us determine ϵ as g spans V_δ.

However, it is not directly possible to get Φ as we cannot get access to δ with our encodings. It is still possible to get Φ up to an affine map.

By computing elements of V_δ, it is possible to form a basis of V_δ by completing the basis starting from a non-zero element g_1. We go through V_δ by 'brute-force' to check independent elements. We then get access to a basis $(g_1, ..., g_8)$ of V_δ. We can then define an isomorphism Ψ defined over the basis b_i by $\Psi(g_i) = b_i$, which is Φ written in the base b_i instead of the canonical base. If l is the basis change map from the canonical basis to b_i, we have $\Psi(g) = l^{-1}(\Phi(g))$, hence

$$g_i(0) = l(b_i + (\epsilon \circ l)^{-1}(0))) = l(b_i) + \epsilon^{-1}(0).$$

If we note A the affine map $A(x) = l(x) + \epsilon^{-1}(0)$, we have for any g:

$$A \circ \oplus_{\Psi(g)} \circ A^{-1} = \oplus_{\Phi(g)}$$

Hence,

$$g = \epsilon \circ A \circ \oplus_{\Psi(g)} \circ A^{-1} \circ \epsilon^{-1}$$

and we can recover $\epsilon \circ A$ over a basis since :

$$g_i(0) = \epsilon \circ A \circ \oplus_{\Psi(g)} \circ A^{-1} \circ \epsilon^{-1}(0) = (\epsilon \circ A)(b_i).$$

□

The affine part is then recovered by solving an overdefined system of equations and the input encodings are easily peeled out, to complete the key-recovery. We refer to [15] for the complete attack.

1.2.4 Upgrade and Attacks

In this section, we briefly detail the improvement and variations to the state-of-the-art candidate constructions. As constructions are often reactions to new attacks or optimizations of attacks, we also include the chronological cryptanalysis of the constructions.

The Xiao Lai Implementation

The first idea to resist the BGE attack is to consider larger encodings and to encode more than one byte at a time. The implementation of Xiao and Lai [110] uses this idea to modify the original implementation of Chow *et al.*. In this construction, the authors use linear encodings to widen the size of the encoding, while not having an exponential explosion in degree due to the composition of random non-linear encodings. The authors claimed that their construction was standing against the BGE attack. This implementation still has external encodings.

Historic Cryptanalysis The cryptanalysis of this implementation was made by De Mulder et al. [86] by using an adaptation of Biryukov et al. [17] algorithm to find linear equivalence. Indeed, while the implementation of Chow et al. had non-linear encodings and needed the strucural step of the BGE attack, the implementation of Xiao and Lai only has linear encodings and can be attacked using only linear techniques. The attack of De Mulder *et al.* is really efficient, with a work factor 2^{32}.

For the sake of completeness, we also discuss the security against the BGE attack. The claim of the author was proven to be wrong, as a generalization of the attack on all traditional encoding strategies was made by Michiels *et al.* [84]. More on this attack will be given below.

Karoumi Dual Cipher Implementation

The implementation idea of Karoumi [71] is to use the underlying algebraic structure of the AES to propose a family of distinct implementations. The remark made by the

author is that the implementation is made using a fixed representation of the field, that is, $\mathbb{F}_{2^8} = \mathbb{F}_2[X]/(X^8 + X^4 + X^3 + X + 1)$. However, for any unitary irreducible polynomial P of degree 8, there is a representation of the field $\mathbb{F}_2[X]/(P)$. As there are 30 distinct such polynomials, there is a degree of liberty to build a round function using multiplication and addition in these special representations.

Definition 20. *Let E and E' be two encryption algorithm with space K. E and E' are said to be dual ciphers if there exist bijections f, g and h such that :*

$$\forall m \in M,\, k \in K,\ f(E_k(m)) = E'_{g(k)}(h(m)).$$

Finding dual-ciphers of the AES as been a subject of research in cryptanalysis [9], and Karoumi exploits the fact that there exist more than 9120 different dual ciphers of the AES. More precisely, there exist 240 linear maps Δ such that:

$$\Delta(AES_k(m)) = AES_{\Delta(k)}(\Delta(m)).$$

Once the rounds are transformed into their dual form, Karoumi uses the implementation technique of Chow *et al.* on his round function. We follow the precise description of De Mulder [85]. We fix a representation of $\mathbb{F}_{2^8}$ by fixing an irreducible polynomial P_l of degree 8, there are 30 such polynomials. We note Ψ_l an isomorphism from the AES representation to this new representation. We note respectively the addition $\oplus_l$ and the multiplication $\otimes_l$. For any element $\alpha \in \mathbb{F}_{2^8}^{\times}$, let $m_\alpha(x) = \alpha \times x$ and F^t the t-th iteration of the Frobenius, that is $F^t(x) = x^{2^t}$. We now define an AES dual round:

Definition 21 (Dual AES Round). *Let D be the set defined by*

$$D := \{\Psi_l \circ m_\alpha \circ F^t \,|\, l \in [\![1, 30]\!]\,,\ \alpha \in \mathbb{F}_{2^8}\,,\ t \in [\![0, 7]\!]\ \}$$

Let for any $1 \leq r \leq 10$, $1 \leq j \leq 4$, let $\Delta_{r,j} \in D\,, \exists (l, \alpha, t)\,, \Delta_{r,j} = \Psi_l \circ m_\alpha \circ F^t$ and define $\delta_{r,j} = \Psi_l \circ F^t$. The AES Dual Round for a representation Ψ_l, noted $AES^{r,\Delta_{r,j}}(x_1, x_2, x_3, x_4)$ is the 4-uple (w_1, w_2, w_3, w_4), where w_i are defined by:

$$\delta_{r,j}(MC_1) \otimes_l SB_{r,j}(x_1 \oplus_l \Delta_{r,j}(K^{t-1}_{1,j})) \oplus_l \delta_{r,j}(MC_2) \otimes_l SB_{r,j}(x_2 \oplus_l \Delta_{r,j} K^{t-1}_{2,j}))$$

$$\oplus_l \delta_{r,j}(MC_3) \otimes_l SB_{r,j}(x_3 \oplus_l \Delta_{r,j} K^{t-1}_{3,j})) \oplus_l \delta_{r,j}(MC_4) \otimes_l SB_{r,j}(x_4 \oplus_l \Delta_{r,j} K^{t-1}_{4,j}))$$

where $SB_{r,j} = \Delta_{r,j} \circ SB \circ \Delta^{-1}_{r,j}$.

We then have:

$$AES^{r,\Delta_{r,j}}_j \circ (\Delta_{r,j}, \Delta_{r,j}, \Delta_{r,j}, \Delta_{r,j}) = (\Delta^{-1}_{r,j}, \Delta^{-1}_{r,j}, \Delta^{-1}_{r,j}, \Delta^{-1}_{r,j}) \circ AES^r_j$$

The AES can then be computed by using these new round functions. The obtained result is then conjugated to the real result which allows to compute the correct value with the maps $\Delta_{r,j}$. The implementation of Karoumi uses these new round functions to build his implementation, to which he applies the implementation technique of Chow et al. As the transformation he uses are just different representations of the same computations, the implementation size is the same as Chow *et al.*.

Cryptanalysis The implementation technique of Karoumi looked interesting because it transformed the AES round functions into new "randomized" ones before applying the encoding strategy. While Karroumi estimated the complexity of a BGE attack on his implementation to be 2^{93}, De Mulder et al. showed that the dual round functions of Karroumi were in fact equivalent to regular AES rounds with the same key. The implementation would then be broken with a simple adaptation of the BGE attack.

Luo *et al.* Large Non-Linear Encodings

The implementation of Luo *et al.* tries to thwart the BGE attack by augmenting the size of the encodings, while not falling into the trap induced by the linear encodings of the Xiao and Lai implementation. Indeed, if the linear encodings can be peeled off by linear equivalence techniques, larger non-linear encodings might be enough to provide security to against the BGE attack.

For their implementation technique, Luo *et al.* merge the technique of Chow *et al.* with the diffusion technique of Xiao and Lai, to get larger encodings that do not blow up exponentially due to their size.

Historic Cryptanalysis While the technique of Xiao and Lai incorporates many of the improvements discussed before and the BGE attack cannot be applied as is, the work of Michiels et al. [84] shows that the BGE attack can be generalized to break a larger class of implementation, to which the implementation of Xiao and Lai belongs.

Michiels *et al.* adapt the BGE attack on any implementation of any SPN. Formally, the attack requires that:

- The round function is bijective and composed of a round key addition, a layer of S-boxes and a linear layer.
- The linear layer must be Maximum Distance Separable (MDS)
- In the implementation the attacker can read encoded outputs of round functions.

These conditions are pretty easy to satisfy and are trivially satisfied by any previously discussed implementations. If these conditions are satisfied, one can state the following theorem:

Theorem 1.2.1 (Michiels *et al.*). *Given traces of the function* $\epsilon \circ R_1[0 : m-1]$, *the vector space* $V_\delta = \{\epsilon \circ \oplus_\delta \circ \epsilon^{-1} | \delta \in \mathbb{F}_2^m\}$ *can be computed in* $\mathcal{O}(m \times 2^m)$ *where* $\oplus_\delta(x) = x + \delta$. *If the vector space* V_δ *can be computed, a map* ϵ' *can be computed in* $\mathcal{O}(2^{3m})$ *where* $\epsilon \circ \epsilon'$ *is affine.*

This result shows that in essence, the implementations that we considered up to this point are too simple in design and are structurally flawed. Hopefully, there is still a lot of design space to be explored.

Lee *et al.* Implementation

The implementation technique of Lee *et al.* [77] aims to merge the implementation techniques of Chow *et al.* with the masking techniques inherited from the grey-box hardware techniques. Instead of encoding the keyed tables T as usual, Lee *et al.* first draw at random a function R and consider the transformation $T + R$. This transformation is then encoded using the Chow *et al.* rationale (see Figure 1.3).

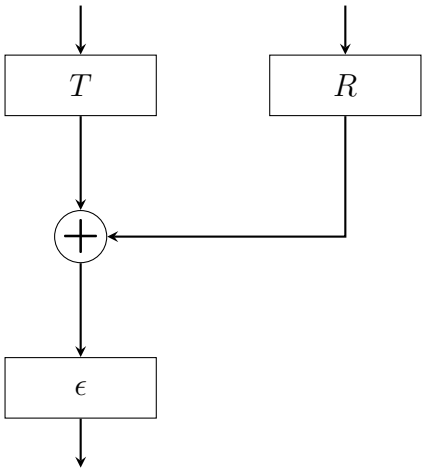


Figure 1.3: Lee *et al.*'s basic tranformation

For correctness, the transformation R is carried on through the computations and encoded as well. The rest of the implementation roughly follows the same design as the previous implementations in the Chow *et al.* rationale. The authors proposed three variations, CASE 1, 2 and 3, with only the CASE 1 implementation published as a challenge. These three implementations only differ in the size of the encodings, CASE 2 and 3 having bigger encodings than CASE 1.

Historic Cryptanalysis As Lee *et al.* implementation has been published after first reports of DCA and the soaring of white-box automated attacks, it is natural to try

these attacks against Lee CASE 1 implementation. Although it was claimed to stand these attacks, at least two reports break Lee CASE 1 implementation.

The work of Rivain and Wang [97] shows that due to a lack of mask in the round function, which is simply encoded, DCA, Collision and MIA attack all break CASE 1 implementation in few thousand traces, for few minutes of attacks. The work of Zeyad *et al.* [111] also breaks the CASE 1 implementation using BCA for the same reasons. The attack is on the same range of parameters as RW19. For these two attacks, breaking CASE 2 and 3 should not be more difficult as the encoding size does not change the structural weakness of Lee *et al.* design. Even if there are no publications clearly addressing it, the CASE 1 implementation is also be vulnerable to LDA as the round function is only encoded by a small encoding.

This realization that automated attacks can break the most up to date state-of-the-art implementations shows that implementations have to be more carefully designed and that new design spaces have to be explored.

1.3 A State of the art of Polynomial-Based Implementations

We now present a line of implementation designs that differ from the Chow *et al.* table-based encoding rationale by the representation of computation used. In the following work, some or all the sub-computations that allow to evaluate the AES implementations are represented by polynomials.

From a high level point of view, the technique used in these implementations is similar to a regular encoding strategy, that is, secret transformations are composed with functions that are necessary to the computation of the whole AES. The security of the implementation is then conditional to the fact that the encodings cannot be split to trivialize key-recovery.

Contrary to look-up tables, whose size is always exponential in their input, the size of a polynomial is, aptly, polynomial in its number of inputs. If its degree is not too high, one can hope to represent computation over a large number of variables by a composition of polynomials that will keep the overall size of the implementation reasonable. The natural goal using this technique is to allow more flexibility for the design.

1.3.1 Bringer et al. Construction

The idea comes the Billet and Gilbert block cipher in which they introduce perturbations to enhance the security against algebraic attacks. Roughly speaking, the block

cipher from Billet and Gilbert is composed of several rounds of IP-instances. However, Faugère and Perret showed how to break it and Ding explained how to repair it with perturbations.

The implementation of Bringer et al. [31]bounces back from this idea and represent the AES round functions as polynomials over $\mathbb{F}_2^8$ and tries to hide the SPN round structure by adding perturbation polynomials. These polynomials are spread from one round to another, and the implementation still follows the 'internal encoding' paradigm by mixing perturbation with the round function with linear encodings. However, the correctness of the implementation cannot be guaranteed off of one representation of the ten perturbed AES rounds. To circumvent that problem, the implementation contains four instances with perturbations and the correct result is computed by a majority vote over the four instances.

If this implementation technique was envisioned for AES, the cryptanalysis of Coron in the the Bringer et al. paper shows the weakness of linear encodings in the white-box context without external encodings. To circumvent this problem, the authors propose an AES with random Sboxes.

Contrary to the usual description at the byte level, the implementation of Bringer et al. uses a description of the round function over variables over $\mathbb{F}_{2^8}$. This means that the round function is represented as a polynomial of $\mathbb{F}_{2^8}[X_1, X_2, X_3, X_4]$. For instance, the function $SubByte(X_i)$ is a composition of X_i^{254} and a linear map which translates to a sum of powers of two in $\mathbb{F}_{2^8}$. The complete round function is a linear transformation of the S-boxes, so the round function has 4 variables and is of degree at most 254. The AES algorithm can then be described by the 10 family of these ANFs, 16 for each round. We note, for $1 \leq t \leq 10$ and $1 \leq i \leq 16$, S_i^t the ANFs of the round function.

The main idea of the construction is to add the 'controlled faults' with 4 perturbations polynomials and 23 random polynomials for the first 9 rounds, and 16 perturbation cancellation polynomials at the last round. The last round XORs the cancellation polynomials to the round functions S^{10} to cancel the perturbations. By this method, the system S_i^t is transformed into its perturbed version $S_j'^t$ with $1 \leq i \leq 43$. Finally, 43-to-43-bit linear internal encodings are applied to system $S_j''^t$.

Remark: Note that the perturbation of the system implies an explosion of the number of variables as rounds functions from round 2 to 9 are now in 43 variables. This means that the implementation weights 142MB per AES instance, which is a far more than the usual table-based implementations.

Tools for the implementation

The key components of the implementation are the perturbation polynomials. The goal is to build a system of polynomials $\tilde{\Phi}$ that will take the values $(\phi_1, ..., \phi_4)$ most of the time. To do so, the author of [30] propose to translate the ϕ_i by a polynomial evaluating to 0 most of the time, for which they propose a construction (we rerefer to [30]).

Definition 22. *(Perturbation Polynomials) Let $\phi_1, ..., \phi_4 \in \mathbb{F}_{2^8}$ and $\tilde{0} \in \mathbb{F}_{2^8}[X_1, ..., X_{16}]$ be a polynomial from the construction [30] that ouputs 0 for any input x in a message subset $U_1 \subset (\mathbb{F}_{2^8})^{16}$. The system of perturbation polynomials $\Phi = (\Phi_1, ..., \Phi_4)$ is defined by:*

$$\Phi_i = \tilde{0} \ + \ \phi_i$$

For any system of perturbation polynomials Φ, we define the perturbation cancelling polynomials 0^{Φ} to be a system of 16 polynomials in $\mathbb{F}_{2^8}[X_1, ..., X_4]$ such that:

$$0_i^{\Phi}(\phi_1, .., \phi_4) = 0$$

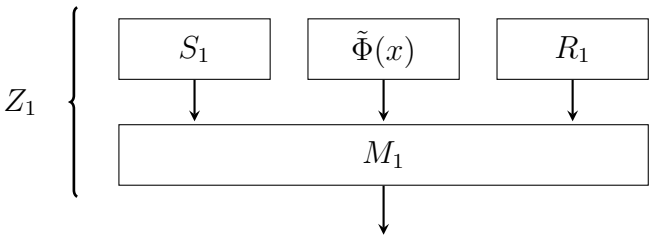
The implementation uses particular linear encodings for diffusion reasons. We refer to the original paper for their generation.

Description of the Implementation

We first recall that the implementation is composed of four instances with different perturbations polynomials. We describe one instance for a set of perturbation polynomials and then explain how the authors generate these pertubations polynomials for correctness.

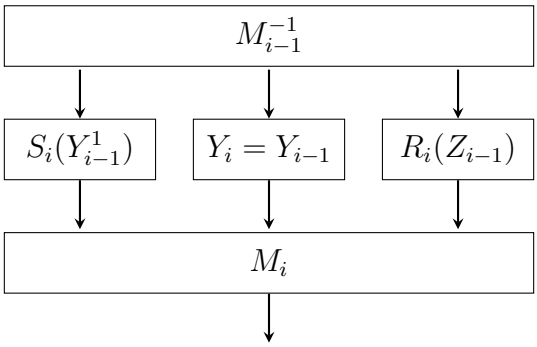
The compiler encodes the first round function with perturbation polynomials and random polynomials. To do so, it draws 19 random polynomials RP^1 in $\mathbb{F}_{2^8}[X_1, ..., X_{16}]$, and a 43 by 43 linear encoding M^1. It then builds the perturbation polynomials $\Phi = (\Phi_1, ..., \Phi_4)$ and computes the encoding of the round system S_1:

$$Z_1 = M_1 \circ (S_1 || \Phi || RP_1)$$



For any round $2 \leq t \leq 9$, the perturbations polynomials are carried on by the identity function, new random polynomials RP^t and linear encoding M_t are drawn and the previous linear encodings are cancelled so that the round function Z_t is defined by:

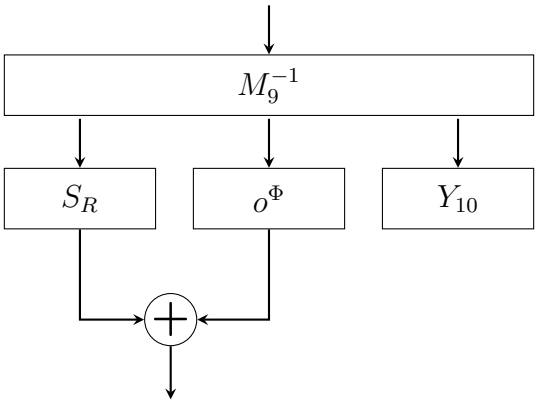
$$Z_t = M_t \circ (S_t||ID||RP_t) \circ M_{t-1}^{-1}$$



Finally, the last round function Z_{10} cancels the perturbations by adding O^ϕ into the circuit:

$$Z_{10}(x) = S_{10}(\tilde{M_9^{-1}}(x)) + O^\Phi(\tilde{M_9^{-1}}(x))$$

with $\tilde{M_9^{-1}}$ being the first 20 coordinates of the inverse of M_9.



This last operation effectively eliminates the random polynomials, and if the perturbation polynomials evaluate to $(\phi_1, ..., \phi_4)$, they are cancelled by O^Φ.

However, as Φ is not always $(\phi_1, ..., \phi_4)$, the authors introduce 3 other instances to ensure the correctness of the construction with probability 1. To do so, they generate 4 perturbation polynomials on complementary sets. Let us split the message space

in two sets $(\mathbb{F}_{2^8})^{16} = U_1 \cup U_1^c = U_2 \cup U_2^c$. We then define a set of perturbation polynomials Φ_1,Φ_1^c,Φ_2 and Φ_2^c such that :

$$\forall x \in U_i \, , \, \Phi_i(x) = (\phi_1, ..., \phi_4)$$

$$\forall x \in U_i^c \, , \, \Phi_i^c(x) = (\phi_1, ..., \phi_4)$$

Then:

- if $x \in U_1 \cup U_2$, $\Phi_1(x) = \Phi_2(x) = (\phi_1, ..., \phi_4)$ and $\Phi_1^c(x)$ and $\Phi_2^c(x)$ are random.
- if $x \in U_1 \cup U_2^c$, $\Phi_1(x) = \Phi_2^c(x) = (\phi_1, ..., \phi_4)$ and $\Phi_1^c(x)$ and $\Phi_2(x)$ are random.
- if $x \in U_1^c \cup U_2$, $\Phi_1^c(x) = \Phi_2(x) = (\phi_1, ..., \phi_4)$ and $\Phi_1(x)$ and $\Phi_2^c(x)$ are random.
- if $x \in U_1^c \cup U_2^c$, $\Phi_1^c(x) = \Phi_2^c(x) = (\phi_1, ..., \phi_4)$ and $\Phi_1(x)$ and $\Phi_2(x)$ are random.

It means that there are always at least two perturbation polynomials that agree for any input, and that the correct value $(\phi_1, ..., \phi_4)$ can always be found by taking the majority vote out of all the 4 possible values.

The final implementation is composed of four instances of the construction described before, one for each of the perturbation polynomials we just described.

Cryptanalysis

A first cryptanalysis of the construction is the report by Bringer *et al.* themselves of an attack by Coron. The main flaw is due to the fact that the encodings are linear. Targeting a S-box, one can attempt to solve a linear system by fixing input 3 input bytes out of the 4 on the first systems of polynomials Z_1. This is essentially a LDA.

To circumvent this problem, the authors propose to use random S-boxes instead of the s-box of the AES, turning them into a new block-cipher AEw/oS. This variation seems to lead to a secure implementation even if this changes the original problem while proposing new designs to explore.

However, the cryptanalysis of this variation is broken by De Mulder *et al.*. The authors propose an inversion attack on the implementation, breaking its one-way property. To do so, De Mulder *et al.* recover equivalent S-boxes that are easier to invert than the heavy polynomials of the implementation. Essentially, the attack abuses the weak linear encodings of the implementation on round 10 and finds equivalent S-boxes by peeling out linear encodings from the bottom-up. At the end 160 equivalent 8-bit S-boxes are retrieved, which trivializes the inversion.

1.3.2 Rasoamiaramanana et al. Construction

The implementation of Rasoamiaramanana et al. [96] was first published as a candidate to the WhiBox 19 contest [107] and was broken among the last candidates. Rasoamiaramanana published the design of the implementation with as security analysis in her PhD thesis which we describe now. The implementation uses hybrid methods with look-up tables and polynomial representation, which is the first published using this design rationale. Roughly speaking, the implementation technique follows the internal encoding rationale but uses larger encoding. These encodings are structured to allow to represent the encoded round functions by look-up tables and degree 2 polynomials.

Tools for the Implementation

The implementation is based on two particular encodings. The first encoding type called type 1 encoding, aims to protect the keyed Sbox of the round function, while including redundancy to lower the degree of its inverse encoding. The type 2 encoding is a degree 2 bijections that encodes the round function, while cancelling the redundancy of the previous encoding.

The type 1 encoding is based on a degree 2 bijection. In their description, they use Dobbertin's polynomials [48].

Proposition 6. *Let $m = 2n + 1$ be an integer. The Dobbertin Polynomial*

$$X^{2^{m+1}+1} + X^3 + X$$

is a bijection of $\mathbb{F}_{2^n}$. Its coordinate are degree 2 polynomials over $\mathbb{F}_2^n$

Remark: The choice of Dobbertin polynomials for this construction seems to be entirely practical. The author do not give any advantage of using Dobbertin's polynomials over any bijection of degree 2 over $\mathbb{F}_2^n$.

Definition 23. *Let $m > 8$ be an integer and let $f : \mathbb{F}_2^8 \to \mathbb{F}_2^8$. For any m-to-m bits linear map A and n-to-n bit bjection σ, we say that $\epsilon(y)$ is a type 1 encoding of $y = f(x)$ if it is of the form :*

$$\epsilon(y) = A(y + q(\sigma(x))||\sigma(x))$$

With the notation $\bar{y} = \epsilon(y)$, the inverse type 1 encoding ϵ^{-1} is:

$$\epsilon^{-1}(\bar{y}) = A^{-1}(\bar{y})[0:7] + q(A^{-1}(\bar{y})[8:m])$$

Definition 24. *Let $m' > 8$ and $j > 1$ be an integer, $f : \mathbb{F}_2^{8j} \to \mathbb{F}_2^8$ and y a vector of $\mathbb{F}_2^{m'j}$. For any m'-to-m' bits linear map A, a $m'j$-to-$(m' - 8)$-bit linear map l, a*

$(m'-8)$-to-8-bit quadratic function q and a Dobbertin polynomial on $\mathbb{F}_{2^{m'-8}}$, we say that $\xi(y)$ is a type 1 encoding of $y = f(x)$ if it is of the form:

$$\xi(z) = A(z + q(l(\bar{y}))||\sigma(l(\bar{y}))$$

With the notation $\bar{y} = \epsilon(y)$, the inverse type 1 encoding ϵ^{-1} is:

$$\xi^{-1}(\bar{z}) = A^{-1}(\bar{z})[0:7] + q(\sigma^{-1}(A^{-1}(\bar{z})[8:m']))).$$

Description of the Implementation

The implementation follows the decomposition of Chow et al., that is, the round key is composed with the Sbox and the linear part is set aside to carry the encodings onto the next round. We now describe the compiler.

The first part of the construction encodes the T-boxes, that is, the functions:

$$T^t_{i,j}(x) = SB(x + K^{t-1}_{i,j})$$

$$T^{10}_{i,j}(x) = SB(x + K^9_{i,j}) + K^{10}_{i,j}$$

The compiler then draws type 1 encodings ϵ^r_i for $0 \leq i \leq 15$ and $0 \leq r \leq 9$ and type 2 encodings ξ^r_i for $0 \leq i \leq 15$ and $0 \leq r \leq 9$.

$$\bar{T}^1_i(x) = \epsilon^1_i(T^1_i(x))$$

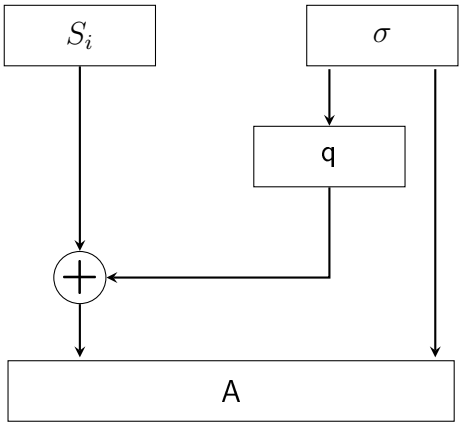
$$\bar{T}^t_i(x) = \epsilon^t_i(T^1_i((\xi^{t-1}_i)^{-1}(x))$$

$$\bar{T}^{10}_i(x) = T^1_i((\xi^9_i)^{-1}(x)$$

The sizes of the encodings depend of the round, according to the following table:

Round	m	m'
1	24	17
2	24	13
3 to 7	13	13
8	4 × 13	13
9	24	17
10	-	-

Table 1.1: Size of each round encodings



The tables are encoded so that encodings are cancelled after compositions of the encoded functions. For the correctness of the implementation, the encoding of the tables are carried onto the linear map. If we denote by L to be the 32-to-32-bit linear map that is the composition of ShiftRows and Mixcolumns, the compiler computes:

$$\bar{L}^r = (\xi_0^r||...||\xi_{15}^r) \circ (L||L||L||L) \circ ((\epsilon_0^{r-1})^{-1}||...||(\epsilon_{15}^{r-1})^{-1})$$

Round 8 has been modified to include a redundancy countermeasure against DFA, hence, the linear map L^8 has been tweaked to add redundancy to thwart fault attacks.

It is important to note that the form of the type 2 encodings allows for the polynomials $\bar{L}^r$ to be only of degree 2. Indeed, the implementation is then composed of:

- The truth tables $\bar{T}_i^1(x)$, $\bar{T}_i^t(x)$ and $\bar{T}_i^{10}(x)$. These truth tables are T-box encoded with type 1 input encodings and type 2 output encoding.
- The collection of polynomials $\bar{L}^r$. These are encodings of the round function with type 2 input encodings and type 1 output encodings.

For the sake of clarity, we detail a short correctness and size analysis of the implementation.

First, it is important to note that the polynomials $\bar{L}^r$ are of degree 2 over its inputs. Indeed, as the functions $R^t(L||L||L||L) \circ ((\epsilon_0^{r-1})^{-1}||...||(\epsilon_{15}^{r-1})^{-1})(\bar{y})$ compute the round function by correctness of the inverse encodings ϵ_i^t, the encoded function $\bar{L}^r$ is such that:

$$\xi_i^t(\bar{y}) = A(R^t[8i:8i+7](x) + q(l(\bar{y})||\sigma(l(\bar{y}))))$$

As R^t is of degree 2 in $\bar{y}$, the encoding $\xi_i^t(\bar{y})$ is a degree 2 function.

Cryptanalysis

As a contestant of the WhiBox19 constest, this implementation has been broken, like of the other participants. In Rasoamiaramanana PhD Thesis, the security analysis is focused on DCA and Collision Attacks. If DCA against S-boxes are ruled out due to the encoding size and the analysis of [97], the authors attribute the break of their construction to collision attacks. Indeed, the collision attack strategy is adapted to internal encoding implementations (part I chapter 4). The authors also claim the security of their implementation against DFA, which we do not discuss here.

We want to underline another weakness of this implementation, which lies in the small degree of the inverse of the encodings. To start, let us recall that the map $\bar{L}^r$ is an encoding of the round function of the form:

$$\xi_i^t(\bar{y}) = A(R^t[8i:8i+7](x) + q(l(\bar{y})||\sigma(l(\bar{y})))$$

As a consequence, the inverse encoding $(\xi_i^t)^{-1}$ is also of degree 2 in the output $\bar{y}$ of the encoded tables and the output $\bar{z} = \xi_i^t(\bar{y})$ of the polynomials $\bar{L}^r$. Indeed, if we note $\bar{z} = \xi_i^t(\bar{y})$, one can remark that:

$$R^t[8i:8i+7](x) = A^{-1}(\bar{z})[0:8] + q(l(\bar{y}))$$

This means that even if the inverse encoding ξ^{-1} is not of degree 2 in $\bar{z}$, there exist a degree 2 relation in $\bar{z}$ and $\bar{y}$ that allows to recover $R^t[8i:8i+7](x)$. An attacker can then launch a degree 2 LDA on $R^t[8i:8i+7](x)$ by fixing 3 out of the 4 bytes. The complete key recovery can be made in complexity $16\times(\sigma(17\times4,2)+\sigma(17\times16,2))\times2^8$.

1.4 Synthesis of Security of White-Box implementation of the AES

We now make a complete summary of the the security of the state-of-the-art implementations we described earlier in the chapter. We evaluate the security of the implementation against key-extraction in the plain white-box model, that is, external encodings, if any, are included in the clear in the implementation, and can then be peeled off. This might seems like a disadvantage for earlier designs, but keep in mind that early designs were defeated by variants of the BGE attack, even with external encodings, and that more modern designs avoided the use of external encodings. Also, we include the implementation of Bringer *et al.* as an AES implementation without the keyed S-boxes for a fair comparison.

To argument the results of the summary table 1.2, we first detail the efficiency of known attacks based on the state of the art of Part 1 and Chapter 2 Part 2. We assume

that the implementations are taken without external encodings for a fair comparison in the plain white-box model. We also assume that the Bringer *et al.* implementation is made with the AES S-boxes.

Against Linear Methods (LM) We group in linear methods the attacks that exploit the linear structure of the encodings that are not LDA. Historically, these are the Linear Equivalence methods and the cryptanalysis of Bringer *et al.* implementation. We set them aside from linear techniques such as LDA to emphasize their non-automatic behavior. All the implementations with non linear encodings are not vulnerable to these attacks.

Against BGE In this category, we group the BGE attack and its extension by Michiels *et al.*. All the implementations using the technique of Chow *et al.* are structurally broken by it. The only constructions that stand against it are the ones that mask the round function to avoid to have a simple bijective encoding of the round function, namely, Rasoamiaramanana *et al.* and Bringer *et al.* constructions.

Against DCA DCA has been showed to be really efficient to break all the state-of-the-art implementations as we have seen in part I. When it is not possible to break the implementation targeting the S-box, it is possible to target the encoding of the round function.

Against CCA and MIA Reported to be efficient against many state-of-the-art implementations with internal encodings in [97], the CCA breaks all the implementations using the Chow *et al.* design. It also breaks the Lee *et al.* implementation and the Rasoamiaramanana *et al.* construction (see the previous security analysis). The only implementation for which it is not sure if the collision strategy can be exploited is the Bringer *et al.* construction which uses many random polynomials.

Against LDA If LDA was originally designed to detect linear combination in masked implementations, it also can work in the encoding paradigm. Indeed, if the inverse encodings are of a too low degree, high degree LDA can detect this weakness. This is the case for all the implementations using the Chow *et al.* technique, were encodings are of size 4 and at most degree 3.

Against BCA The Bucketing Computational Attack is not a historic threat but is proven to be really efficient against state of the art constructions. The paper of Zeyad *et al.* [111] shows that most of the implementations with small encodings are broken by it in few minutes. This is the case of all the papers using Chow et al.

techniques. For Lee et al., the flaw resides in encodings at the round level that are not randomized like the s-boxes. Due to the heavy use of random polynomials in Bringer *et al.* implementation, it is not clear whether collisions can be exploited in this setting. The implementation of Rasoamiaramanana *et al.* should not be susceptible to BCA at the S-box levels as the encodings are larger than the S-box size.

Table 1.2: Comparative Security Analysis against Key-Extraction

Implementation	Size	LM	BGE	DCA	CCA	LDA	BCA
Chow et al.	752kB		*	*	*	*	*
Xiao and Lai	20MB	*	*		*	*	*
Karroumi	752kB		*	*	*	*	*
Luo et al.	25.8MB		*	*	*	*	*
Lee et al.	796 kB		*	*	*	*	*
Bringer et al.	568 MB	*		?	?	*	?
Rasoamiara--manana et al.	44 MB				*	*	*

Note : We mark by $*$ if an attack is efficient against an implementation and by ? if it is unknown. We refer to the previous paragraph for the justifications. We do not put the exact complexity of the attack as for many of these implementations, some improvements can be done on the complexity of attacks with adaptation of state-of-the-art techniques.

Chapter 2

A SPN implementation Technique: Randomized Circuit Knitting

In this chapter we describe a rationale of implementation we call Randomized Circuit Knitting that is aimed to resist algebraic attacks such as high-order LDA or the BGE attack. We first describe it through an implementation of AES, a challenge, and a security analysis. Then, we show that this rationale can be applied to other SPNs to provide similar security properties in the plain white-box model.

2.1 The AES-RCK Implementation

In this section, we detail our implementation of the AES. We first explain the rationale of the construction from a high level point of view. We then detail the ten rounds of our implementation. We show its correctness and briefly study its size.

2.1.1 Links with previous works

The implementation we propose is made according to the internal encoding rationale. However, similarly to the implementation of Rasoamiaramanana *et al.*, it uses the polynomial representation of computations to achieve security. Especially, it uses particular encodings that are adapted to this structure.

2.1.2 Rationale of the Construction

The starting point of our construction is the weakness of usual design against automated attacks, especially LDA. The main problem with the usual internal encoding

strategy is that it does not make an optimal use of the available parameters – the degree of the encoding and its support – to increase the complexity of the attack. For seminal implementations, encodings where made on 4-bit nibbles, later augmented to 8-bit or more, but without a diffusion of said encodings on the round function. It means that attacks can easily be made on 8- or 16-bit keyspace, with a small computation trace when the support is small.

To circumvent this problem, we divide our construction into two parts. First, we "expand" the usual AES by adding random transformations to round functions. These functions are incorporated to the round function with a 1-round Feistel transformation and for correctness, the 1-round Feistel is composed onto the next round. This transformation allows us to locally augment the degree and to increase the support of the substitution part. The counterpart for the augmentation of the degree is an exponential augmentation of the substitution part size. This expansion phase is however designed to keep the size of the substitution part reasonable.

After we have expanded our round functions, we perform an encoding step on the expanded rounds to "knit" rounds and secret transformations together. To avoid linear encodings while keeping the implementation size reasonable, we use encodings in that have inverses of degree 2. Using these bijections will allow us to augment the degree of the involved transformations once again. Their composition will lead to a degree 4 transformations over a relatively large support, but once again, this step is crafted to keep the size reasonable.

The complete implementation is then the composition of encoded substitution parts Σ and the degree 4 encoding compositions Λ. As Σ are transformations of full degree over their inputs, we represent them as truth tables for a tight representation. The polynomial coordinates of Λ are however only of degree 4, so we represent them by their algebraic normal forms.

2.1.3 Preliminaries

We here recall or define few notations that we will need for the description of our implementation. To avoid using too many multi-indexes, for any function f from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$ we note – when it is well defined – $f[a:b] = (f_a, ..., f_b)$.

For any set element $(x_1, .., x_n) \in \mathbb{F}_2^n$ and integer d, we note $Mon_d(x_1, ..., x_n)$ the set of all the monomials in the x_i of at most degree d. We note its cardinal $\sigma(n, d) = \sum_{i=0}^{d} \binom{n}{i}$.

For any bijection $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$, we note f^{-1} its inverse. For any injection $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$ with $m \geq n$ we also note f^{-1} its left inverse, *i.e.* the function $\mathbb{F}_2^m \to \mathbb{F}_2^n$ such that $f^{-1} \circ f = id_{\mathbb{F}_2^n}$.

We now describe families of functions we will need for our description:

Definition 25. *Let d_1, d_2, $n, m \in \mathbb{N}^*$. We define the set $\mathfrak{F}_{n,m}^{d_1}$ of all functions $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ such that $\forall i \in [\![1, m]\!]\ deg(f_i) = d_1$ We define the set $\mathfrak{S}_n^{d_1,d_2}$ to be the set of all bijections $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ such that $\forall i \in [\![1, m]\!]\ deg(f_i) = d_1$ and $deg(f_i^{-1}) = d_2$. We note $\mathfrak{S}_n^{d,*} = \cup_{i=1}^n \mathfrak{S}_n^{d,i}$ and $\mathfrak{S}_n^{*,d} = \cup_{i=1}^n \mathfrak{S}_n^{i,d}$*

To the best of our knowledge, to provide a generic efficient sampler over $\mathfrak{S}_n^{2,d}$ for any $n, d \in \mathbb{N}^*$ is not an easy task. We do not provide a solution to this problem, but we provide a solution to sample on a subset of $\mathfrak{S}_n^{2,*}$ that will be large enough as n increases.

Sampling in $\mathfrak{S}_n^{2,*}$

For the need of our implementation technique, we will need a sampler in $\mathfrak{S}_n^{2,*}$. However, to the best of our knowledge, sampling uniformly in this set is not an easy task, and there is no published sampling algorithm available. To circumvent that problem we will sample in a well-known subset of $\mathfrak{S}_n^{2,*}$ that stems from multivariate techniques – it is almost exactly the C^* trapdoor function.

For any integer n, one can study monomials in $\mathbb{F}_{2^n}$ that are of degree two when projected onto $\mathbb{F}_2^n$, that is, monomials of the form X^D with $HW(D) = 2$. Now, consider the special case where we take n such that: $\exists D \in [\![2,^n -1]\!]\ gcd(D, 2^n-1) = 1$ and $HW(D) = 2$. Then $F = X^D$ is a bijection of inverse $F^{-1} = X^{D'}$ where $DD' = 1$ mod $2^n - 1$. The degree of F^{-1} over $\mathbb{F}_2$ is $HW(D')$.

It is important to note that this construction will not work certain values of n. For instance, there are such no bijective monomials for $n = 32$ due to the smoothness of $2^{32} - 1$. To be sure it exists, we will sample in $\mathfrak{S}_n^{2,*}$ when n is odd only.

Now that we have a candidate bijection in $\mathfrak{S}_n^{2,*}$ we use the well-known multivariate to get a family of bijections: we compose x^D with linear operations S and T. Formally, we project x^D over $\mathbb{F}_2^n$ bijection over $(\mathbb{F}_2)^n$. To do so let us fix a basis $(e_1, ..., e_n) \in (\mathbb{F}_{2^n})^n$ of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$. It induces an isomorphism π from $(\mathbb{F}_2)^n$ to $\mathbb{F}_{2^n}$ such that $\pi(x_1, ..., x_n) = \sum_{i=1}^n x_i e_i$. Then we define the bijection $\sigma_{S,F,T}$ as the map from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$ is defined by:

$$\sigma_{S,F,T} = T \circ \pi^{-1} \circ F \circ \pi \circ S$$

Regarding our sampling problem, for each linear transformations S and T, we get a bijection $\sigma_{S,x^D,T}$. However, as shown in [109], there are some equivalent choices of S and T that get to the same P – essentially, maps that commute to a certain extend with the central map. The number of maps we get is the cardinal of $GL_n(\mathbb{F}_2) \times GL_n(\mathbb{F}_2)$ divided by the cardinal of the equivalence classes:

$$\frac{(\prod_{i=1}^{n-1}(2^n-2^i))^2}{n(2^n-1)} = \mathcal{O}(\frac{2^{(n-1)^2}}{n})$$

Hence the following property:

Proposition 7. *For any integer n and any $D \in [\![2,^n -1]\!]$ $gcd(D, 2^n-1) = 1$ and $HW(D) = 2$. Let π be the canonical isomorphism from $(\mathbb{F}_2)^n$ to $\mathbb{F}_{2^n}$. We have :*

$$\#\{T \circ \pi^{-1} \circ x^D \circ \pi \circ S | (S,T) \in GL_n(\mathbb{F}_2)^2\} = \frac{(\prod_{i=1}^{n-1}(2^n-2^i))^2}{n(2^n-1)}$$

and the cardinal of the set of bijections $\mathfrak{S}_n^{2,*}$ is at least $\frac{(\prod_{i=1}^{n-1}(2^n-2^i))^2}{n(2^n-1)}$.

This means that for n reasonably small, we can sample in a subset of $\mathfrak{S}_n^{2,*}$ that has more than 2^λ elements for a brute force search.

2.1.4 Our Implementation

We now describe the compiler. It is composed of 10 rounds transforming the 10 AES rounds (S^i, L^i) into 10 new round functions $(\Sigma^i; \Lambda^i)$. The functions Σ^i are stored as truth tables and the functions Λ^i are stored as their ANF.

For sake of simplicity, we introduce notations, grouping transformations by their functionality:

- First, we relabel bytes of the state in the following way :

$$\begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} \rightarrow \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix}$$

- We define S^i, and S^i_j its components byte by byte, to be the composition of AddRoundKey and SubBytes, that is we consider the keyed S-boxes:

$$S^i(x) = (S^i_0(x_0), ..., S^i_{15}(x_{15})) = (SB(x_0 + K^i_0), ..., SB(x_{15} + K^i_{15}))$$

- We define L^i, and L^i_j its component 32 bits per 32 bits, to be the composition of ShiftRows and MixColumn for $i \leq 9$ and the composition of ShiftRows and AddRoundKey for $i = 10$.

With these notations the rounds of the AES are just the composition of S^i and L^i. We note R^i the round functions.

Round 1

We describe the algorithm for the first quarter of the state, that is, for the round functions R_0^1 to R_{31}^1. The algorithm applies the same way to other quarter of states, with the corresponding change of indexes.

The Substitution Σ The compiler first draws random functions $r_{1,i}^1 \in \mathfrak{F}_{16,5}$ for each $1 \leq i \leq 2$, and transformations $f_{i,j}^1 \in \mathfrak{F}_{5,8}$ of degree 5.

The compiler then computes two first masks for pairs of keyed S-boxes by computing:

$$mask_{1,1}^1 = S_0^1(z_0) + f_{1,1}^1(r_{1,1}^1(z_0, z_5))$$

$$mask_{1,2}^1 = S_5^1(z_5) + f_{1,2}^1(r_{1,1}^1(z_0, z_5))$$

$$\tilde{\Sigma}_{1,1}^1 = mask_{1,1}^1 || mask_{1,2}^1 || r_{1,1}^1(z_0, z_5)$$

The same is done with other half of the state with s_{10} and s_{15} to compute:

$$\tilde{\Sigma}_{1,2}^1 = mask_{2,1}^1 || mask_{2,2}^1 || r_{1,2}^1(z_{10}, z_{15})$$

The compiler finally computes the complete substitution part Σ by drawing two bijections $\sigma_1, \sigma_2 \in \mathfrak{S}_{21}^{*,2}$ and composing them respectively with $\tilde{\Sigma}_1$ and $\tilde{\Sigma}_2$:

$$\Sigma_{1,i}^1 = \sigma_{1,i} \circ \tilde{\Sigma}_{1,i}$$

The Diffusion Λ The linear layer is then modified accordingly for correctness. If we note y_i the output of $\Sigma_{1,i}$:

$$\tilde{y}_i = {\sigma_{1,i}^1}^{-1}(y_i),$$

the inverse of the encodings are composed with the linear part of the round L_1, for $k \in [\![1, 4]\!]$:

$$\tilde{L}_{1,k}^1 := L_{1,1}^1(\tilde{y}_1[0:7], \tilde{y}_1[8:15], \tilde{y}_2[0:7], \tilde{y}_2[8:15])[8k:8k{+}7] \,||\, \tilde{y}_1[16:20] \,||\, \tilde{y}_2[16:20]$$

The functions Λ are extended to functions $\mathfrak{F}_{42,21}$ by linearly sharing the coordinates into 21 functions, that is, the compiler computes an application $ext_{1,k}^1 \in \mathfrak{F}_{18,21}$ such that there exist $ext\tilde{L}_{1,k}^1 \in \mathfrak{F}_{21,18}$:

$$\tilde{L}^1_{1,k} = ext^1_{1,k} \circ ext\tilde{L}^1_{1,k}$$

The compiler then encodes bits corresponding to a S-box input of the next round by drawing, for $k \in [\![1,4]\!]$, $\tau_{1,k} \in \mathfrak{S}^{2,*}_{21}$ at random computing:

$$\Lambda^1_{1,k} = \tau_{1,k} \circ ext\tilde{L}^1_{1,k}$$

The same is done for all bytes of the round function.

Output: The output of the first round is the collection of the applications $\Sigma^1 = (\Sigma^1_{1,1}, \Sigma^1_{1,2}, ..., \Sigma^1_{4,1}, \Sigma^1_{4,2})$ and the collection of the applications $\Lambda^1 = (\Lambda^1_{1,1}, ..., \Lambda^1_{1,4}, ..., \Lambda^1_{4,1}, ..., \Lambda^1_{4,4})$. The applications $1 \leq i \leq 4{,}1 \leq j \leq 4$ $\tau^1_{i,j}$ and $ext^1_{i,j}$ and the functions $1 \leq i \leq 4{,}1 \leq j \leq 2$, $\sigma^1_{i,j}$ and $f^1_{i,j}$ are carried as inputs for the next round.

Round 2

We describe the algorithm for the first quarter of the state, that is, for the round functions R^2_0 to R^2_{31}. The algorithm applies the same way to other quarter of states, with the corresponding change of indexes. We suppose that we have the output functions $1 \leq i \leq 4{,}1 \leq j \leq 4$ $\tau^1_{i,j}$ and $ext^1_{i,j}$ and the functions $1 \leq i \leq 4{,}1 \leq j \leq 2$, $\sigma^1_{i,j}$ and $f^1_{i,j}$ of the previous round. We note z to be the output of Λ^1, and we define z_i to be the coordinates of z by blocs of 21 bits, that is $z = (z_0, ...z_{20})$.

The Substitution Σ The compiler first composes the previous round transformation with the S-boxes of the current round and the 1-Round Feistel structure for correctness:

$$z'_k = {\tau^1_{i,j}}^{-1}(z_k)$$

$$S'^2_k = S^2_k(z'_k[0:7] + L^1_1(f^1_{1,1}(z'_k[8:12]), f^1_{1,2}(z'_k[8:12]), f^1_{2,1}(z'_k[13:17]), f^1_{2,2}(z'_k[13:17])))$$

The compiler draws random functions for each $r^2_{1,i} \in \mathfrak{F}_{21,3}$ $(1 \leq i \leq 2)$ transformations of degree $f^2_{i,j} \in \mathfrak{F}_{3,3}$.

The compiler then computes a mask for each keyed S-boxes by computing:

$$mask^2_{1,i} := S'^2_{5k}(z'_{5i}) + f^2_{1,i}(r^2_{1,i}(z'_{5i}))$$

$$\tilde{\Sigma}^2_{1,i} := mask^2_{1,i} \,||\, r^2_{1,i}(z_{5i})$$

The compiler finally computes the complete substitution part Σ by drawing a bijection for each S-box $\sigma^2_{1,i}, \in \mathfrak{S}^{*,2}_{11}$ and composing them with $\tilde{\Sigma}^2_{1,i}$:

$$\Sigma^2_{1,i} = \sigma^2_{1,i} \circ \tilde{\Sigma}^2_{1,i}$$

The Diffusion Λ The linear layer is then modified accordingly for correctness. If we note y_i the output of $\Sigma_{1,i}$:

$$\tilde{y_i} = {\sigma^2_{1,i}}^{-1}(y_i),$$

the inverse of the encodings are composed with the linear part of the round L_1, for $k \in [\![1, 4]\!]$:

$$\tilde{L}^2_{1,k} := \alpha \,||\, \tilde{y}_1[8:12] \,||\, \tilde{y}_2[8:12] \,||\, \tilde{y}_3[8:12] \,||\, \tilde{y}_4[[8:12]$$

with

$$\alpha = L^1_{1,1}(\tilde{y}_1[0:7], \tilde{y}_2[0:7], \tilde{y}_3[0:7], \tilde{y}_4[0:7])[8k:8k+7]$$

The functions Λ are extended to functions $\mathfrak{F}_{42,21}$ by linearly sharing the coordinates into 21 functions, that is, the compiler computes an application $ext^1_{1,k} \in \mathfrak{F}_{20,21}$ such that there exist $ext\tilde{L}^1_{1,k} \in \mathfrak{F}_{21,20}$:

$$\tilde{L}^2_{1,k} = ext^2_{1,k} \ \circ \ ext\tilde{L}^2_{1,k}$$

The compiler then encodes bits corresponding to a S-box input of the next round by drawing, for $k \in [\![1, 4]\!]$, $\tau_{1,4} \in \mathfrak{S}^{2,*}_{21}$ at random computing:

$$\Lambda^2_{1,k} = \tau_{1,k} \circ ext\tilde{L}^2_{1,k}$$

The same is done for all bytes of the round function.

Output: The output of the second round is the collection of the applications $\Sigma^2 = (\Sigma^2_{1,1}, ..., \Sigma^2_{1,4}, ..., \Sigma^2_{4,1}, ..., \Sigma^2_{4,4})$ and the collection of the applications $\Lambda^2 = (\Lambda^2_{1,1}, ..., \Lambda^2_{1,4}, ..., \Lambda^2_{4,1}, ..., \Lambda^2_{4,4})$. The applications $1 \leq i \leq 4$,$1 \leq j \leq 4$ $\tau^2_{i,j}$ and $ext^2_{i,j}$, $\sigma^2_{i,j}$ and $f^2_{i,j}$ are carried as inputs for the next round.

Round 3 to 9

We describe the algorithm for the first quarter of the state, that is, for $3 \leq t \leq 9$ the round functions R^t_0 to R^t_{31}. The algorithm applies the same way to other quarter of states, with the corresponding change of indexes. We suppose that we have the functions $1 \leq i \leq 4$,$1 \leq j \leq 4$ $\tau^{t-1}_{i,j}$ and $ext^{t-1}_{i,j}$ and the functions $1 \leq i \leq 4$,$1 \leq j \leq 2$, $\sigma^{t-1}_{i,j}$ and $f^{t-1}_{i,j}$ of the previous round. We note z to be the output of Λ^{t-1}, and we define z_i to be the coordinates of z by blocs of 21 bits, that is $z = (z_0, ...z_{20})$.

The Substitution Σ The compiler first composes the previous round transformation with the S-boxes of the current round and the 1-Round Feistel structure for correctness:

$$z'_k = \tau_{i,j}^{t-1^{-1}}(z_k)$$

$$S''^t_k = S^t_k(z'_k[0:7] + L^t_1(f^{t-1}_{1,1}(z'_k[8:10]), f^{t-1}_{1,2}(z'_k[11:13]), f^{t-1}_{1,3}(z'_k[14:16]), f^{t-1}_{1,4}(z'_k[17:19]))$$

The compiler draws random functions for each $r^2_{1,i} \in \mathfrak{F}_{21,3}$ $(1 \leq i \leq 2)$ transformations of degree $f^t_{i,j} \in \mathfrak{F}_{3,3}$.

The compiler then computes a mask for each keyed S-boxes by computing:

$$mask^t_{1,i} := S''^t_{5k}(z'_{5i}) + f^t_{1,i}(r^t_{1,i}(z'_{5i}))$$

$$\tilde{\Sigma}^2_{1,i} := mask^t_{1,i} \,||\, r^t_{1,i}(z_{5i})$$

The compiler finally computes the complete substitution part Σ by drawing a bijection for each S-box $\sigma^t_{1,i}, \in \mathfrak{S}^{*,2}_{11}$ and composing them respectively with $\tilde{\Sigma}^t_1$ and $\tilde{\Sigma}^t_2$:

$$\Sigma^t_{1,i} = \sigma^t_{1,i} \circ \tilde{\Sigma}^t_{1,i}$$

The Diffusion Λ The linear layer is then modified accordingly for correctness. If we note y_i the output of $\Sigma_{1,i}$:

$$\tilde{y}_i = {\sigma^t_{1,i}}^{-1}(y_i),$$

the inverse of the encodings are composed with the linear part of the round L_1, for $k \in [\![1,4]\!]$:

$$\tilde{L}^t_{1,k} := \alpha \,||\, \tilde{y}_1[8:10] \,||\, \tilde{y}_2[8:10] \,||\, \tilde{y}_3[8:10] \,||\, \tilde{y}_4[[8:10]$$

with

$$\alpha = L^{t-1}_{1,1}(\tilde{y}_1[0:7], \tilde{y}_2[0:7], \tilde{y}_3[0:7], \tilde{y}_4[0:7])[8k:8k+7]$$

The functions Λ are extended to functions $\mathfrak{F}_{44,21}$ by linearly sharing the coordinates into 21 functions, that is, the compiler computes an application $ext^1_{1,k} \in \mathfrak{F}_{20,21}$ such that there exist $ext\tilde{L}^t_{1,k} \in \mathfrak{F}_{21,20}$:

$$\tilde{L}^t_{1,k} = ext^t_{1,k} \;\circ\; ext\tilde{L}^t_{1,k}$$

The compiler then encodes bits corresponding to a S-box input of the next round by drawing, for $k \in [\![1,4]\!]$, $\tau_{1,4} \in \mathfrak{S}^{2,*}_{21}$ at random computing:

$$\Lambda^t_{1,k} = \tau^t_{1,k} \circ ext\tilde{L}^t_{1,k}$$

The same is done for all bytes of the round function.

Output: The output of the second round is the collection of the applications $\Sigma^t = (\Sigma^t_{1,1}, ..., \Sigma^t_{1,4}, ..., \Sigma^t_{4,1}, ..., \Sigma^t_{4,4})$ and the collection of the applications $\Lambda^t = (\Lambda^t_{1,1}, ..., \Lambda^t_{1,4}, ..., \Lambda^t_{4,1}, ..., \Lambda^t_{4,4})$. The applications $1 \leq i \leq 4$,$1 \leq j \leq 4$ $\tau^t_{i,j}$ and $ext^t_{i,j}$, $\sigma^t_{i,j}$ and $f^t_{i,j}$ are carried as inputs for the next round.

Round 10

We describe the algorithm for the first quarter of state, that is, for the round functions R^{10}_0 to R^{10}_{31}. The algorithm applies to the same way to other quarter of states, with the corresponding change of indexes. We suppose that we have the functions $f^9_{i,1}$ and $f^9_{i,2}$ $(1 \leq i \leq 4)$ of the previous round. We note z the output of Λ^9, and we define z_i to be the coordinates of z by blocs of 21 bits, that is $z = (z_0, ...z_{21})$.

The Substitution Σ The compiler first composes the previous round transformation with the S-boxes of the current round:

$$z'_k = \tau_k^{-1}(l_k^{-1}(z_k)$$

$$S'^t_k = S^t_k(z'_k[0:7] + L^t_1(f^{t-1}_{1,1}(z'_k[8:10]), f^{t-1}_{1,2}(z'_k[11:13]), f^{t-1}_{1,3}(z'_k[14:16]), f^{t-1}_{1,4}(z'_k[17:19]))$$

As the linear layer is a permutation for last round, the compiler just composes the two layers:

$$\Sigma_k = L(S'_0, ...S'_{15})[8k : 8k+7]$$

Output: The output of the last round is the collection of the applications $\Sigma^{10} = (\Sigma_0, ..., \Sigma_{15})$.

2.1.5 Correctness and Size of the Implementation

We now show the correctness of our algorithm and detail its size.

Correctness

Round 1 For round 1, we just show that the composition of Σ and Λ is a special encoding of the round function. We only show this for the first quarter of state and the proposition follows for the others.

Proposition 8. *The function* $\Lambda^1_{1,1}(\Sigma^1_{1,1}, \Sigma^1_{1,2}, \Sigma^1_{1,3}, \Sigma^1_{1,4})$ *is an encoding of 8 bits of the round function* $R^1_{1,1}$*. More precisely, with the notations of section 3.2 of round 1,if we note* y *the output of* $\Lambda^1_{1,1}(\Sigma^1_{1,1}, \Sigma^1_{1,2}, \Sigma^1_{1,3}, \Sigma^1_{1,4})$ *for an input* x *and* $y' := ext^1_{1,1}({\tau^1_{1,1}}^{-1}(y)$*, the following relation stands :*

$$R^1_{1,1}(x) = y[0:7] + L^1_1(f^1_{1,1}(y[8:12]), f^1_{1,2}(y[8:12]), f^1_{2,1}(y[13:17]), f^1_{2,2}(y[13:17]))[0:7]$$

Proof. By construction, the mappings ${\sigma^1_{1,j}}^{-1}$ and $\sigma^1_{1,j}$ cancel out when composed. Then, as L is a linear map, it commutes with the addition of the masks $mask_{i,j}$, hence, $\Lambda^1_{1,1}(\Sigma^1_{1,1}, \Sigma^1_{1,2}, \Sigma^1_{1,3}, \Sigma^1_{1,4})$ is an encoding of :

$$R^1_{1,1} + L^1_1(mask^1_{1,1}, mask^1_{1,2}, mask^1_{2,1}, mask_{2,2})[0:7] \,||\, r^1_{1,1} \,||\, r^1_{1,2}$$

The rest follows from the inverse of the extension and of $\tau^1_{1,1}$ to encode the following round.

□

Round 2 to 9 The construction is very similar for rounds 2 to 9 with the exception of round two having specific parameters. We show the result for round 2 and the proof for round 3 to 9 follows. We state that after $t-1$ rounds of our construction, the t-th round is an encoding of the t-th round of the AES. We only show this for the first quarter of state and the proposition follows for the others.

Proposition 9. *Let* z *be the output of the* $(t-1)$*-th round of the implementation, that is, the output of* Λ^{t-1} *after the composition of the round functions on the input of* x*. The function* $\Lambda^t_1(\Sigma^t_{1,1}, ..., \Sigma^t_{1,4}, ..., \Sigma^t_{4,1}, ..., \Sigma^t_{4,4})[0:20](z)$ *is an encoding of the first 8 bits of the composition of* t *rounds of the AES. More precisely, with the notations of section 3.2 of round 1,if we note* y *the output of this function for an input* x*, and* $y' := ext^t_{1,1}({\tau^t_{1,1}}^{-1}(y)$ *the following relation stands*

$$R^t_{1,1}(x) = y[0:7] + L^1_1(f^1_{1,1}(y[8:10]), f^1_{1,2}(y[11:13]), f^1_{1,3}(y[14:16]), f^1_{1,4}(y[17:19]))[0:7]$$

Proof. To prove it by induction, consider the basic case $t = 2$. By the correctness property of round 1, the first round of the implementation is an encoding of the first round of the AES of the form, with the notations for x, y and $y' := ext^1_{1,1}(\tau 1_{1,1}{}^{-1}(y)$ defined in the proposition, the following relation stands :

$$R^1_{1,1}(x) = y[0:7] + L^1_1(f^1_{1,1}(y[8:12]), f^1_{1,2}(y[8:12]), f^1_{2,1}(y[13:17]), f^1_{2,2}(y[13:17]))[0:7]$$

Then, by construction, the mapping Σ_i^2 starts by cancelling out $\tau_{1,1}^1$. Then, it computes masks $f_{i,j}(r^{k,l})$ and adds $L_1^1(f_{1,1}^1(r_{1,1}^1), f_{1,2}^1(r_{1,1}^1), f_{2,1}^1(r_{1,2}^1), f_{2,2}^1(r_{1,2}^1))$ to the masked round. As L^1 is linear, it commutes with the additions of the masks and correctly unmasks the round. The rest of the round is similar to round 1, except the input is already large, so we choose smaller masks.

For heredity, the proof stems from the same arguments as the basic case, except the encodings σ and τ are applied to all the input/output of their respective mappings. □

Round 10 Round 10 is just composed of the inverse encoding of the previous round and a substitution part. The correctness stems from the same arguments as for Σ^t's correctness.

Size of the Implementation

We now give a short synthesis of the sizes of the implementation. All the sizes of each component for each is given in bits. The size a n-to-m bits truth table is $m \times 2^n$ and the size of a polynomial of degree 4 in a boolean polynomial ring is given by its monomials, that is at most $\sigma(n, 4)$.

Round 1 The substitution part Σ^1 is composed of 16-bit truth tables. As we have 21 outputs for each of them per pair of S-boxes, the substitution part of round 1 is composed of $4 \times 2 \times 21 \times 2^{16} \approx 2^{23.4}$ bits.

The encoding part Λ^1 is composed of 17 polynomials over 21×2 variables per S-box of the following round. Hence, the encoding part for round 1 has its representation composed of $17 \times 16 \times \sigma(21 \times 2, 4) \approx 2^{25.5}$ bits.

Round 2 to 9 The size of round 2 to 9 is the same as for round 1 except the substitution part Σ has a 21 bit input. Hence the size of the substitution part is $16 \times \times 11 \times 2^{21} \approx 2^{28.45}$ bits and the size of the polynomials of Λ is $16 \times 21 \times \sigma(11 \times 4, 4) \approx 2^{25.5}$ bits.

Round 10 The last round is only composed of the substitution part Σ. Its size is then $16 \times 8 \times 21 \times 2^{21} \approx 2^{25}$ bits.

Synthesis and Remarks Adding up all rounds the implementation weighs about $2^{31.77}$ bits ≈ 450MB. While it is non-negligible, it is still deployable in most devices used nowadays.

The fast growth of the size of the polynomials is a problem this technique cannot avoid which limits the array of 'reasonable' parameters. However, if we want bigger implementations, we can use bigger tables for Σ by adding variables in the first round substitution part or using a larger encoding in round 2 to 10. The number of coordinates of Λ will then grow linearly, and the size of the truth table will rapidly catch up to dominate the implementation size. This regime of parameters is surely interesting for better security claims if bigger implementations are not a problem. For the challenge construction however, we tried to minimize the space requirements. More on this topic in the security conjecture section.

Concerning running time, similarly to the size analysis, the polynomials of Λ take the most time to be computed. As we use a simple polynomial evaluation technique in our implementation, that is, computing the monomials in a given input and then adding the corresponding monomials, the time used is then proportional to the size of our polynomials.

2.2 Security Analysis

We now perform the analysis of our construction. We recall that the study of fault attacks are out of the scope of this thesis. The goal of this analysis is to estimate the unbreakability of our compiler against state-of-the-art attacks, especially the algebraic ones.

When implementations are only composed of truth tables, no intermediate values appear during the computation of these tables, and the security analysis can be performed with only the input-ouput behavior of these atomic elements. However, when we add polynomials to an implementation however we add intermediate values such as monomials and their sums to computation traces. To perform a security analysis, one has to either consider each of these monomials and their combinations, or try to reduce the analysis to the input-output behavior of the polynomials - under right hypotheses. As the first option makes a precise analysis impossible due to the combinatorial complexity of the problem, we will formalize hypotheses and rely on the second strategy.

It is important to note that this strategy is not new by any means. Indeed, it is common in multivariate cryptography to make these kind of assumptions when known structural attacks are not efficient. For instance, for cryptosystems based on the IP problem such as HFE or UOV (see also Part III), if the state-of-the-art attacks are not efficient enough, we consider that the polynomials are similar to a 'black-box', that is, the monomials they are composed of and their partial sums do not give more information than the complete input-ouput behavior of these polynomials.

For the rest of the analysis, we make the following reasonable hypothesis: having the polynomials of the application Λ on our algorithm is equivalent to having their truth table for the cryptanalysis. This hypothesis can obviously be challenged. A more detailed discussion about these hypotheses will be given in part III.

Furthermore, automated attacks are usually launched with a set of target functions to test. For this analysis, we will consider that targets are the most common ones, that is, outputs of the S-boxes and outputs of round functions.

2.2.1 Against LDA

To study the complexity of LDA or high order LDA against our construction, we detail the mappings that allow to recover the target function. We then discuss their degree.

Targeting a S-box We focus our analysis on round 1 as it is the case where the key-space is the smallest. We also detail the analysis only for Σ_1^1 as it is easy to generalize to the complete mapping Σ^1. The analysis for S-boxes stems from the following property:

Proposition 10. *Let $y = \Sigma_{1,1}^1(x_0, x_{15})$. We have:*

$$S_0(x_0) = {\sigma_{1,1}^1}^{-1}(y)[0:7] + f_{1,1}^1({\sigma_{1,1}^1}^{-1}(y)[16:19])$$
$$S_5(x_5) = {\sigma_{1,1}^1}^{-1}(y)[8:15] + f_{1,2}^1({\sigma_{1,1}^1}^{-1}(y)[16:19])$$

As we note $S_0(x_0) = \psi_0(y)$ and $S_5(x_5) = \psi_5(y)$, for any other keyed S-box S_0' and S_5' we have:

$$S_0'(x_0) = S_0'(S_0^{-1}(\psi_0(y))$$
$$S_5'(x_5) = S_5'(S_5^{-1}(\psi_5(y))$$

Proof. The proof is trivial from the construction of Σ^1. □

According to this result, a d-LDA of order $d = deg(\psi_0)$ will have a solution for S_0. If d is smaller, the LDA will not succeed as ψ_0 is by definition the inverse of the encoding transformation. Furthermore, the second part of the property shows that a LDA of order $deg(S_0'(S_0^{-1}(\psi_0)))$ will also have a solution for any other keyed sbox S_0'.

To continue our analysis, we have to estimate the degree of $deg(\psi_0)$ and $deg(S_0'(S_0^{-1}(\psi_0)))$ when S_0, S_0', $f_{i,j}$ and σ_1 vary through their respective spaces. We propose this conjecture, supported by numerical tests:

Conjecture: If $S_0 \neq S'_0$ are keyed s-boxes drawn at random and $f_i \in \mathfrak{F}_{5,5}$ and $\sigma_1 \in \mathfrak{S}_{21}^{*,2}$ drawn at random, then the degree of coordinates of ψ_0 and and $deg(S'_0(S_0^{-1}(\psi_0))$ are greater or equal to 10 independently of the key with overwhelming probablility.

If this conjecture stands, an attacker with LDA has to use $\sigma(21, 10) > 2^{16}$ monomials for the 2^{16} possible inputs and the system admits solutions for more than a single key : it contains about 2.5 more variables than equations. The LDA then fails targeting the mappings Σ^1.

Remark: We stated a strong version of the conjecture we really need, because it is a more succinct property. Indeed, it is enough for us that this result stands for enough S'_0 so that an adversary cannot reduce the key-space enough to brute force the leftover keys afterwards.

Targeting Round Functions We also focus on round 1 for key-space reasons and on the first byte of the round function. By property 2, the composition of Σ^1 and Λ^1 is an special encoding of the AES first round function. Let us note the degree of its decoding relation d.

Contrary to the attack on S-boxes, as the encoding is only of size 21, we have a solution for degree d for the right key-guess and no a priori solutions for other key-guesses. Indeed, a byte of an AES round function with a fixed key k is not a basis for round functions with other keys $k' \neq k$. We can then evaluate the complexity of the attack according to section 2:

$$\sigma(21, d)^2 \times 2^{32}$$

Again, we have to study the degree d depending of its composing elements:

Conjecture : For $f_{i,j} \in \mathfrak{F}_{5,5}$ and $\tau_1 \in \mathfrak{S}_{21}^{*,2}$ drawn at random, with high probability, $d \geq 11$.

If this conjecture stands, the attack can be made in 2^{74} operations.

Remark: Note that to conduct this attack, one has to evaluate Λ over many inputs and the polynomials of Λ are composed of $\sigma(42, 4) \approx 2^{24.1}$ monomials, The evaluation of the polynomials Λ can be a lower bound for the attack.

2.2.2 Against DCA

For this section, we base our analysis on the analysis of DCA made on random encodings by Rivain and Wang in [97].

Targeting Sboxes In [97], the authors prove that for random functions, the probability of success of a DCA against encoded S-box is small when the size of the encoding is bigger than the S-box support. In our case, S-boxes σ_i are "encoded" into Σ_i that is a mapping from $\mathbb{F}_2^{16}$ to $\mathbb{F}_2^{21}$. In an ideal setting, this means that a DCA is not successful against with S-box as targets.

Targeting Round Functions In this setting, the size of the encoding of the round function is 21. This means that DCA can be used with a reasonable probability of success. However there are no results for this kind of encodings. If we apply the strategy of [97] for 16-to-8-bit encodings, as our encodings are packed by pairs of bytes, we can fix any bytes to simplify the attacks an the complexity is, with the notations of their paper:

$$T \times N \times \#K = 21 \times 2^{21 \times 2} \times 2^{32}$$

2.2.3 Against MIA and Collision Attacks

To perform MIA and collision attacks, one has to target a non-injective function of the algorithm. In [97], the authors directly target the round function and exploit the equivalence between a collision on the round function and a collision on the encoded round function, as the encoding is a bijection. However, in our setting, the encodings of rounds are not local bijections. We still focus our study on the first byte of round 1, which can be generalized to the rest of the round.

Proposition 11. *Let $x = (x_0, x_5, x_{10}, x_{15})$ and $x' = (x'_0, x'_5, x'_{10}, x'_{15})$ be two distinct vectors of 4 bytes. If there is a collision $\Lambda_1[0:16](x) = \Lambda_1[0:16](x')$ then :*

$$\begin{cases} R_1[0;7](x) = R_1[0;7](x') \\ r^1_{1,1}(x_0, x_5) = r^1_{1,1}(x'_0, x'_5) \\ r^1_{1,2}(x'_{10}, x'_{15}) = r^1_{1,2}(x'_{10}, x'_{15}) \end{cases}$$

Proof. By property 2, a collision implies a collision for a function of the form:

$$\tau_1(R_1[0:7] + L^1_1(f^1_{1,1}(r^1_{1,1}), f^1_{1,2}(r^1_{1,1}), f^1_{2,1}(r^1_{1,2}), f^1_{2,2}(r^1_{1,2})||r^1_{1,1}||r^1_{1,2})$$

We can then remove the bijection τ_1, but as we carry the random terms from the construction of Σ, a collision of the complete state implies a collision on all the coordinates and as the masks are computed from the rightmost part of the state, we have the desired property. □

This property means that collisions are observed on the encoded state when the random polynomials $r^1_{1,1}(x_0, x_5)$ and $r^1_{1,2}(x_{10}, x_{15})$ and the round function simultaneously collide. In other words, the set of collisions is the intersection of the set of collision points of these 3 functions. If we consider the polynomials $r^1_{1,1}(x_0, x_5)$ and $r^1_{1,2}(x_{10}, x_{15})$ as random functions from $\mathbb{F}_2^{16}$ to $\mathbb{F}_2^5$, it means that they have 2^{11} collisions on average. The cardinal of the intersection of the sets of collision points is now bounded by 2^{11} and a subset of the collision of the round function, which reduces the set of collisions which was of about 2^{24} collisions for a round function. To know whether this set is small enough to protect against collision attacks is a challenge for future works.

2.2.4 Against BCA

In the state of the art, BCA has been mostly used against non-injective encodings of S-boxes. In our case, the encodings of the S-boxes are wider than their input, hence the collision property that is needed by BCA to function is not present. We conjecture our construction to be secure against this form of BCA.

2.2.5 Against the BGE Attack

Let us first recall the main theorem used for the BGE attack. In the BGE attack setting, an m bits encoding of a round function can be found in execution traces. Let assume that $R^1[0:m-1]$ is encoded by a m-to-m-bit bijection ϵ so that $\epsilon \circ R^1[0:m-1]$ can be observed in execution traces. In the following we use the generalized attack of Michiels *et al.* [84].

In our setting, the encodings are not local bijections and the set V_δ cannot be computed. Indeed, the proof of the theorem stands and the set V_δ can be computed if there exist two sets of inputs $\tilde{x}_1$,$\tilde{x}_2$ and functions ψ_1 and ψ_2 such that $h(\tilde{x}_1, \tilde{x}_2) = \epsilon(\psi_1(\tilde{x}_1) + \psi_2(\tilde{x}_2)) = \epsilon \circ R^1[0:m-1](x)$ and for any constant c_1 and c_2, $h(\tilde{x}_1, c_2)$ and $h(c_1, \tilde{x}_2)$ are bijections (see [84] for more details).

As we add random 16-to-5 noise polynomials, that are not bjiections, there is no such decomposition of the encoding of our round function and the BGE attack cannot be applied against our design.

2.2.6 Variations of the Challenge Implementation

To improve the security of the construction we can adjust the parameters of the construction. The first parameter to adjust is the size of the Σ tables. The input size of these tables is dependent on the number of random polynomials added. If n_r is the number of random polynomials we introduced, their input size is at least $8+2\times n_r+1$

for the first round and $8 + 4 \times n_r + 1$ for the other rounds, since we need an odd number of inputs for our encodings. For the challenge, $n_r = 5$ on the first round and $n_r = 3$ for the rest of the implementation. Another possibility is to increase the size of the table outputs, hence the number of variables of the polynomials. It is one of the less costly adjustment since the size only increases polynomially by a degree 4 when we add variables. With these remarks in mind we propose 2 other implementations that are variations of the challenge.

The first one is LIGHT-RCK-AES. For this implementation we choose $n_r = 5$ for the first round and $n_r = 2$ for the rest, and the output size of the tables to be the same as the original challenge. In this setting, the encodings of functions and the input of truth tables are at most of size 17 and polynomials have at most 44 variables. Extrapolating the analysis on the challenge, the security level of the implementation is conjectured to be 60. The implementation would weight 2^{29} bits ≈ 70 MB.

The second one is HEAVY-RCK-AES. For this implementation we choose $n_r = 5$ for the first round and $n_r = 4$ for the rest. In this setting, the encodings of round functions and the inputs of truth tables are of size 25 and polynomials have 100 variables. Extrapolating the analysis once again the security conjectured would be at a security level of 80. The implementation would weight 2^{36} bits ≈ 8.5 GB .

2.3 The Randomized Circuit Knitting Rationale of Design

The goal of this section is to show that the technique we used for the implementation is rather generic and can be used for SPNs in general provided the S-boxes are not too big. Let us assume that we want to implement a SPN with a keyed S-box S^i and linear layer L^i so that $L^i(S^i)$ is the round function. We now describe the three parts of this method: a pre-processing step, a randomized expansion step and a knitting phase.

2.3.1 A Pre-processing Step

As the method of implementation we propose is somewhat agnostic to the mathematical structure the SPN is based on, we can represent S^i and L^i as their algebraic normal form over $\mathbb{F}_2$. Then the round function can be tweaked at will into any S'^i and L'^i, such that $L^i(S^i) = L'^i(S'^i)$. This includes adding redundancy for DFA countermeasures or artificially splitting the S-box layer to increase the support of S'^i. This phase can be seen as an 'unknitting' of the circuit corresponding to the round function. If the original round function of the SPN has really small S-boxes, depending on the linear layer, one can attempt to use the composition of two rounds instead of one - more on this strategy with PRESENT will be given in the following sections.

2.3.2 Randomized Expansion

The first phase, the randomized expansion phase, allows to locally increase the degree of the transformations involved, while only paying for a small exponential overhead in the substitution part of the SPN and a linear increase in the permutation part. This phase follows ideas closer to [31] as it allows to introduce "random" polynomials that are chosen during the compilation. The goal of these random polynomials is to mask the round functions monomials by monomials and to increase the degree of algebraic relations leading to sensible variables. The masking can be done at the level of the layer S, or directly on L. The masking at the level of S is usually easier as the support of L is usually big enough so that polynomials of these degree are not adapted to 'reasonable' sizes of implementations. The random polynomials are then carried on to the next round to ensure the correctness of the computations. To control the size of our construction, we force these polynomials to be of a certain form. At the end of this procedure, we get new ANFs that allow to evaluate the SPN. At this point the representation of our SPN is still 'unknitted' but more complex from an algebraic standpoint.

2.3.3 Knitting Phase

The goal of the knitting phase is to factorize the ANFs we built in the previous step to hide the randomness we introduced. It allows to increase the support and the degree of the algebraic relation between sensible variables involved in the construction. The input of this transformation is the output of the Randomized Expansion algorithm. First, we share the ANFs of the round function so that the shares are polynomials with specific support, leading to a representation of the round function $L"(S")$ where the support of each coordinate of $S"$ is 'small' enough. This step is needed to avoid an exponential blow-up of encoded shares. We then group these shares by support and encode them with bijections, whose inverse are of degree 2 . This leads to the functions Σ on the AES implementation. To protect the round function, we encode these degree 2 transformations whose inverse are of degree as high as possible to get a degree 4 part. This leads to the polynomials Λ in the AES implementation. The encodings are then carried on to the ANFs of the next round. If the randomized expansion phase as been correctly parameterized, the composition of randomized expansion and knitting onto the complete RCK algorithm does lead to a local exponential growth in size in the substitution part, and a polynomial one in the 'permutation part'.

2.3.4 Using RCK on PRESENT

PRESENT is an example of an SPN by Bogdanov et al. [24]. It is composed of 31 rounds, the block length is 64 bits and two key lengths of 80 and 128 bits are supported.

It is a lightweight algorithm that has been designed to be fast in restrained environments. According to the designers, PRESENT has been meant with the following target goals, among others:

- To be implemented in hardware
- To be deployed in situations where moderate security levels are needed.
- To be used in challenge-response authentication protocols

PRESENT has not been designed with the white-box model in mind. It is almost the opposite: it has been designed for restrained hardware environments. However, in the context of authentication, asymmetries might arise that would lead to consider implementing PRESENT in the white-box model. For instance, if PRESENT is used in the hardware of an IoT object for authentication, one can have an implementation of the encryption or decryption algorithm on a smartphone, that is software based. In this context, the hardware efficiency of PRESENT is still relevant, but one needs to protect the other party with the white-box model in mind. That is why we propose to see how to use the RCK framework to implement PRESENT in the white-box model.

Description of PRESENT

The round function of PRESENT is composed of AddRoundKey, an S-box layer and a permutation layer. We do not cover the key schedule and refer to its specification.

PRESENT S-box The S-box of PRESENT is a 4-to-4-bit bijection S defined by the following table in hexadecimal, where the 64-bit state is divided into consecutive 4-bit nibbles:

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S(x)	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

PRESENT Permutation Layer The permutation layer of PRESENT is a simple switch of wires defined by the following table:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P(i)	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	6	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
P(i)	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
P(i)	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
P(i)	2	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

An implementation of PRESENT with RCK

We now explain how we would use the RCK framework to implement PRESENT with similar security properties to the AES.

Pre-processing To use the RCK framework on PRESENT, we start by a reprocessing step that starting from round 1, consist of composing the round function of the t-th round with the S-box of the following round. For any $t \leq 30$, $t = 1 mod 3$ we define $S^{t,t+1}$ by:

$$R^{t+1} \circ R^t(x) = P(S(P(S(x + k^t)) + k^{t+1})) =: P \circ S^{t,t+1}(x)$$

Due to the structure of PRESENT, the coordinates of the map $S^{t,t+1}$ each have 16 inputs. We let round $t+2$ as is. We now can represent PRESENT with an alternation of 2 round functions defined by $P \circ S^{t,t+1}$ and $P \circ S$, for 21 total round functions. The goal of this step is to adapt the representation of the round function to expansion, and then apply the knitting phase.

Randomized expansion We are now ready to expand round functions. For the rounds of the form $P \circ S^{t,t+1}$ we expand the "S-boxes" $S^{t,t+1}$ in the same way we did for the AES, with n_r random polynomials for the S-boxes sharing a 16-bit support. As the layer P involves 4 S-boxes, the number of inputs needed for evaluating the S-box of the following round is $n_r \times 4 + 4$. The inverse of the expansion is carried onto the following S-box round, which limits the input of the next round of S-boxes to $n_r \times 4 + 4$. A randomized expansion step can also be taken with round of the form $P \circ S$, but it will lead to S-boxes with more than 20 inputs, which is not ideal for minimizing the size of the implementation.

Knitting Phase For the knitting phase, as the layer P mixes 4 nibbles of 4 bits together, the randomized S-boxes functions can be grouped by 4 to encode at least 16 bits together. To further augment the size of the encodings, one can share the output of the S-boxes to artificially introduce a linear layer that is not only a bit permutation. Once the size of the encodings of the S-boxes are decided, the output encoding is fixed by the size of the S-boxes of the next round.

Synthesis Following this method, the implementation is composed of tables Σ that have at least 16 bits inputs and depend on more key-bits than the original round function and the secret polynomials and encodings introduced. The composed degree-2 encodings with the sharing of the S-box output leads to a map represented by degree-4 polynomials that can be made similar in size to the maps Λ of the AES implementation.

With these similarities in mind, the security of such an implementation depends on the same algebraic conjectures.

Conclusion of Part II

Implementing the AES is one of the first goals established by the white-box community and is still far from being closed.

As we have seen from the short state-of-the-art study, attackers are stronger than designers by a huge margin. This is partly due to the impressive efficiency of the automated attacks, but also due to the structural vulnerabilities of the methods used, namely the internal encoding rationale of design.

In the new design we propose, we try to push this rationale of design by using a polynomial representation to produce an implementation that is resilient to algebraic attacks such as LDA or the BGE attack. The conjectured security of our implementation is based on conjectures on the algebraic degree of the relation between the usual target values of attackers and the values computed by our implementation. We also motivate the study of our technique by publishing a challenge implementation.

We also abstract the design of our implementation as the RCK framework for SPNs and describe how to implement PRESENT to support the versatility of this framework.

As following works, it would be interesting to investigate whether the algebraic countermeasures we propose can be mixed with countermeasures specific to other attacks, such as DCA, in the same spirit as the masking techniques of [16] can be mixed with linear masking for resilience against DCA and LDA.

Part III

Multivariate Cryptography in the White-Box Model

Chapter 1

Introduction to Multivariate Cryptography and HFE

In this chapter, we propose a short introduction to multivariate cryptography, its usage, the problems it is based on and the usual attack techniques. We then shortly explain the "big field" trapdoor rationale and extensively describe one of its well known element, the Hidden Field Equation (HFE) one-way trapdoor function. We then precise the state of the art of attacks against HFE specifically

1.1 Introduction to Multivariate Cryptography

Multivariate cryptography is a part of public-key cryptography that relies on the hardness of solving polynomial equations. This rationale of design has been initiated by algorithms such as C^* in the 80's and then followed by a lineage of algorithms such as HFE [88] or UOV [73] and their descendants.

Among the original appeal of this rationale, one can note its efficiency for short signatures and its flexibility of design as reflected by the high number of techniques used to repair broken candidates or design new ones. However, multivariate cryptography is also one of the rare promising candidates for post-quantum cryptography, together with hash, code and lattice-based cryptography. Indeed, while polynomial time quantum algorithms such as Shor's algorithm threaten public-key cryptography based on factorisation or discrete logarithm, multivariate cryptography is conjectured to stand against quantum opponents. This is why multivariate schemes have been proposed to the post-quantum NIST competition, for instance GeMSS (an HFE variant).

Even if cryptanalysis showed that the parameters for these candidates were too optimist, the latest round of selection was gifted with new submitted multivariate algorithms such as VOX and PROV, showing the everlasting flexibility of multivariate cryptography.

1.1.1 Multivariate Cryptography Rationale

In Public-Key Multivariate Cryptography, the public key is given as a set of m polynomials $(\mathcal{P}) = (P_1, \ldots, P_m)$ in n variables, of small degree d over a small finite field $\mathbb{F}$. Most of the time, d equals 2 and $20 \leq n \leq 300$:

$$(\mathcal{P}) \begin{cases} P_1(x_1, \ldots, x_n) \\ \cdots \\ P_m(x_1, \ldots, x_n) \end{cases}$$

In practice, this system is structured: a central system $\mathcal{P}' : \mathbb{F}^n \to \mathbb{F}^m$ called the central map is hidden by two affine secrets $T : \mathbb{F}^m \to \mathbb{F}^m$ and $S : \mathbb{F}^n \to \mathbb{F}^n$ to form a system $\mathcal{P}$ defined by:

$$\mathcal{P} = \mathcal{T} \circ \mathcal{P}' \circ \mathcal{S}$$

The private key is then defined by the set $(S, \mathcal{P}', \mathcal{T})$. The system $\mathcal{P}'$ is built to be easy to solve, and the system $\mathcal{P}$ hard to solve. We now explain how to use this public-key to build encryption and signature algorithms.

Encryption and Decryption For encryption, plaintexts are elements of $\mathbb{F}^n$ and ciphertexts are elements of $\mathbb{F}^m$. Using the public key, a message $(x_1, ..., x_n) \in \mathbb{F}^n$ is encrypted by :

$$c = (c_1, ..., c_m) = \mathcal{P}(x_1, ..., x_n)$$

In this context, the public key is the encryption algorithm. This encryption process is quite fast as evaluating such system of polynomials can be made in $\mathcal{O}(mn^2)$.

For decryption, one need to use the trapdoor, i.e to use the secret-key $(S, \mathcal{P}', T)$. For any $c \in \mathbb{F}^m \cap Im(\mathcal{P})$, one can inverse map by map the public key:

$$x = S^{-1} \circ \mathcal{P}'^{-1} \circ T^{-1}(c)$$

Note that c has to be in the image of $\mathcal{P}$ to have a solution and that we must have $m \geq n$ to hope for a unique solution.

Signature and Verification For signature, as usual, we use the direct sense of the trapdoor to verify and the inverse to sign. Contrary to encryption, m can be smaller than n without any impact on the ability to sign. We give the example of a hash-and-sign algorithm. Let $h : \{0,1\}^* \to \mathbb{F}^m$ be a hash function. To sign a message $x \in \{0,1\}^*$, one needs to compute the preimage of its hash through $\mathcal{P}$:

$$s = S^{-1} \circ \mathcal{P}'^{-1} \circ T^{-1}(h(x))$$

This can only be done with the knowledge of the trapdoor. If $h(x)$ is not in the image of $\mathcal{P}$, one can pad x with a counter until its hash is in the image of $\mathcal{P}$. To verify the signature, we simply evaluate $\mathcal{P}$ on s and compare it to $h(x)$:

$$\mathcal{P}(s) \stackrel{?}{=} h(x)$$

1.1.2 The PoSSo Problem

For this rationale to work, the public key P needs at least to be one-way. Indeed, if one can solve $y = P(x)$, it means for instance that an attacker can forge signatures or decrypt. The hardness of solving polynomial equations is then a requirement for multivariate cryptography.

Definition 26. *PoSSo and MQ Problems*

Let $n, m, d \in \mathbb{N}^*$ with $d \geq 2$ and $\mathbb{F}$ a finite field. The Polynomial System Solving problem is defined as follows.

Given a system of polynomials $(\mathcal{P})(P_1(x_1, \ldots, x_n)P_m(x_1, \ldots, x_n)$ in $\mathbb{F}$, find an n-tuple $(x_1, ..., x_n) \in \mathbb{F}^n$ such that:

$$(\mathcal{P}) \begin{cases} P_1(x_1, \ldots, x_n) = 0 \\ \cdots \\ P_m(x_1, \ldots, x_n) = 0 \end{cases}$$

When $d = 2$ we call this problem Multivariate Quadratic (MQ).

In general, this problem is known to be NP-Hard [58], which explains the initial interest for this rationale. However, with a complete random system, operations such that signature or decryption are impossible - that is, finding preimages of y through P - especially because of the hardness of PoSSo.

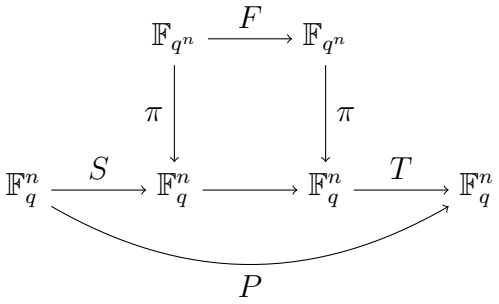
The Isomorphism of Polynomials Problem In the context of multivariate cryptography, finding the secrets S and T given $(\mathcal{P})$ and $(\mathcal{P}')$ has been abstracted as the *Isomorphism of Polynomial* (IP) problem in [88]. It started a line of work such as [29, 79, 90, 92] to better understand the security of general multivariate schemes.

There exist several families of multivariate schemes, corresponding to several choices for the system $\mathcal{P}'$, such as C^* [82], HFE [88], Rainbow [45] or UOV [73]. Their security depend on the internal polynomial and "perturbations" added to the system. If in

general, solving a set of quadratic equations over a finite field is an NP-hard problem for any finite field it is however difficult to obtain a proof of security for multivariate schemes: since the system $(\mathcal{P}')$ has to be easy to solve, it is never random, and neither is $(\mathcal{P})$ (since $(\mathcal{P})$ is obtained by linear changes of variables from $(\mathcal{P}')$).

1.1.3 Trapdoor Techniques

The Big Field Trapdoor For this trapdoor technique, let us assume that we have F a map built on an extension $\mathbb{F}_{q^n}$ of a finite field $\mathbb{F}_q$. Let us further assume that the map F can be easily inverted over $\mathbb{F}_{q^n}$, think of a bijective monomial for instance. If π denotes an isomorphism from $\mathbb{F}_{q^n}$ to $\mathbb{F}_q^n$, we then have the following commutative diagram :



The public key is defined by :

$$P = T \circ \pi^{-1} \circ F \circ \pi \circ S$$

and the trapdoor is computed with the knowledge of (S, T) and the big field representation of F. This trapdoor is the basis of schemes such as C^* [82] and HFE [88].

The Unbalanced Oil and Vinegar Trapdoor The Unbalanced Oil and Vinegar (UOV) trapdoor is the basis of the UOV [73] scheme and is used in other designs such as Rainbow [45]. This trapdoor needs two distinct sets of variables of size o and v such that $n = o + v$. We note the set of oil variables O and the set of vinegar variables V. Consider now a central map whose coordinates are of the form:

$$\sum_{(x,y)\in O\times V} a_{x,y}xy + \sum_{(x,y)\in V\times V} b_{x,y}xy + \sum_{x\in O\cup V} c_x x + d$$

The idea is that, using map such as these, if we fix vinegar variables, we get linear equations in oil variables. The system of equation can then be solved.

The security of such trapdoor is based on the fact that after a composition with the secrets S and T, distinguishing oil and vinegar variables is difficult.

1.2 A Short Plea for White-Box Multivariate Cryptography

As we have seen from the theoretical study of part I and the state of the art of white-box techniques for SPNs in part II, the array of techniques and mathematical objects used for white-box cryptography is quite limited. This is probably because the AES is the most studied algorithm in the white-box model. Indeed, its structure - i.e. the SPN structure with slow diffusion of the key through ten rounds - does not help designers to decompose the algorithm into secure blocks in the white-box model. Essentially, with current state-of-the-art techniques the 8-to-8-bit layer of S-boxes followed by a 32-to-32 linear layer does not translate into a succinct representation of the round that can be composed with the next round to ensure a combinatorial hardness for adversaries. This is why rounds are usually broken down into tables with internal encodings, trying to decompose the round function - or the round function composed with a secret transformation.

From these remarks, it seems that studying cryptosystems based on other rationals and mathematical structures can offer new design spaces. Even if some of them, such as lattice-based algorithms that need randomness to be secure, might be burdensome, others may offer new design avenues with new techniques due to the mathematical structures they are based on. This is why this part will be dealing with white-box techniques for multivariate cryptography.

A first selling point is that polynomial representation helps to diffuse key bits into the implementation and the decomposition of the public key is already taken into account by the black-box security study. In a sense, multivariate cryptography is one of the closest designs in the black-box model that takes white-box aspects in consideration, due to the nature of its public key. Also by using structured trapdoors, especially for big field cryptosystems, a designer can take advantage of the vector space and the field structure to use new algebraic techniques for implementations.

1.3 Usual Attack Techniques Against Multivariate Cryptography

In this section, we recall the main attack strategies that are specific to multivariate cryptography. This section is just a brief overview and we will detail them against the algorithms we study later.

1.3.1 Direct Inversion Attack

In a direct inversion attack, the goal of the attacker is to break the trapdoor one-way property of the public-key, hence breaking the PoSSo problem. The most common and efficient method is to use Gröbner bases. We do not detail the theory of how to compute them, and refer to [40] for the curious reader. The algorithms to compute these bases have been improved over the years, the most powerful ones being Faugère algorithms F4 [56] and F5 [55]. and the hybrid approach of [13].

1.3.2 Rank Attacks

The goal of Rank Attacks is to abuse the structure of the central map to recover the secret-key. More precisely, it ties the key recovery to an instance of the MinRank problem [39]. A MinRank instance is defined by a set of matrices. To solve a MinRank problem, one needs to find a linear combination of the matrices that minimize the rank. In the context of multivariate cryptography, it has been first used in the Kipnis-Shamir [72] attack: this attack is based on the fact that the initial system $\mathcal{P}'$ is often of low rank as a quadratic form. Solving a MinRank instance based on the equation of $\mathcal{P}$ allows to recover the map T, and the map S is recovered by solving an overdefined system of linear equations.

1.4 The Hidden Field Equation Familly

We now describe HFE in more details and review the most powerful attacks known against it: the Gröbner bases message recovery attack and the key-recovery rank attacks are especially the attacks that are considered in the black-box model to test whether an HFE instance can be used to build secure primitives.

1.4.1 Description of HFE

First described by Patarin in [88], the HFE scheme is a direct descendant of C^* [82, 83]. This is one of the 'big field' algorithms, using polynomials that are of degree 2 over $\mathbb{F}_q$. For any positive integers n and $D \in \mathbb{N}$ let $F \in \mathbb{F}_{q^n}[X]$ be defined by:

$$F(X) = \sum_{\substack{0 \leq i < j < n \\ q^i + q^j \leq D}} a_{i,j} X^{q^i + q^j} + \sum_{\substack{0 \leq i < n \\ q^i \leq D}} b_i X^{q^i} + c$$

where the $a_{i,j}$, the b_i and c are elements of $\mathbb{F}_{q^n}$. As the integer D bounds the actual degree of any such F, we call D the degree of the HFE instance. As the quantity $\lceil log_q(D) \rceil$ will be important in the description of attacks, we set $d = \lceil log_q(D) \rceil$

The secret key is the list of such polynomial F and a couple of affine transformations $(S,T) \in AFF_n(\mathbb{F}_q)$. We note it (S,F,T). We remark that F is efficiently invertible on its image due to the Berlekamp algorithm if D is not too big and that S and T are trivially invertible as long as q^n is not too big.

To compute the public key, let us fix a basis $(e_1,...,e_n) \in (\mathbb{F}_{q^n})^n$ of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$. It induces an isomorphism π from $(\mathbb{F}_q)^n$ to $\mathbb{F}_{q^n}$ such that $\pi(x_1,...,x_n) = \sum_{i=1}^{n} x_i e_i$. The public key P is the map from $\mathbb{F}_q^n$ to $\mathbb{F}_2^n$ defined by:

$$P = T \circ \pi^{-1} \circ F \circ \pi \circ S$$

The public key is represented by the n coordinates of P: for each $0 \leq i < n$ we note its i-th coordinate $P_i \in \mathbb{F}_q[x_1,...,x_n]$.

1.4.2 Perturbations

Usually, a "nude" instance (that is, an instance without perturbations) is not enough to resist the state-of-the-art attacks. That is why perturbations were introduced to reinforce these nude instances, while keeping the trapdoor property. We describe only the ones that are useful to us in this thesis.

The Minus (-) Perturbation Let a be an integer and $a < n$. The minus perturbation, denoted by "$-$", simply consists in keeping secret some of the n polynomials $(P_1,\ldots,P_n)$. Only $n-a$ are made public and will be used to check the validity of the equations. We formalize it by transforming the full affine transformation T into $T^- : \mathbb{F}_q^n \to \mathbb{F}_q^{n-a}$ which is the $(n-a)$ first coordinates of the full transformation T.

Then, messages of size $n-a$ can be signed by selecting the a last elements in $\mathbb{F}_q$ at random and signing with the n elements obtained. Note that these last a elements must be random; otherwise the last a polynomials can be interpolated.

A HFE scheme with the perturbation $-$ is called HFE$^-$ with public key

$$P = T^- \circ \pi^{-1} \circ F \circ \pi \circ S$$

The Projection (p) Perturbation The projection variant p of HFE has been reinvestigated recently to enhance the resilience of the public key against support minor modeling attacks ([87]). In this variant, the map S is not a bijective affine map over $\mathbb{F}_q^n$, but an affine map of full rank pS from $\mathbb{F}_q^{n-p}$ to $\mathbb{F}_q^n$ where the integer p is the projection parameter. With such map S, there is a probability $\frac{1}{q^p}$ that an element $y \in \mathbb{F}_q^n$ has no preimage through the internal map. This means that there is a q^p

overhead to find an element in the image of P, hence a q^p overhead in signature time. More details on this variant are given in [46, 87].

A HFE scheme with the modifiers p is called pHFE with public key:

$$P = T \circ \pi^{-1} \circ F \circ \pi \circ pS$$

The Plus Hat ($\hat{+}$) Perturbation Introduced in [54], the Plus Hat $\hat{+}$ perturbation modifies the central polynomial F. To do so, let t be an integer, $(\beta_1, ..., \beta_t) \in (\mathbb{F}_q)^t$ and $(p_1, ..., p_t) \in \mathbb{F}_q[x_1, ..., x_n]^t$ define a transformation Q such that:

$$Q(X) = \sum_{0 \leq i \leq t} \beta_i \times p_i(x_1, ..., x_n)$$

where we identify X to its representation $(x_1, ..., x_n)$ in $(\mathbb{F}_q)^n$. The central polynomial is then modified to be $H = F + Q$. A HFE scheme with the modifiers $\hat{+}$ is called HFE$^{\hat{+}}$ with public key:

$$P = T \circ \pi^{-1} \circ H \circ \pi \circ S$$

To sign with such instance, we can remark that the image of Q is a small subvector space of $\mathbb{F}_q^n$ of dimension at most t. Then the q^t values of Q can be guessed, and the regular inversion of the instance can be made for each of these guess. The signature is the only x corresponding to the correct value of Q. More details on the Plus Hat variant can be found in [54].

1.4.3 The C^* and D^* schemes

We give a brief description of C^* and D^*, ancestors of HFE, as we will use them for particular instantiations in section 5.

Historically, HFE is a descendant of the scheme C^* [82, 83]. This scheme is identical to HFE except this time, the internal polynomial is only a monomial $F = x^{1+q^\theta}$ for any integer θ. The cryptanalysis of C^* by Patarin [88] caused the disuse of this simple central polynomials for nude instances and the complete key recovery of [50] broke the C^{*-} perturbation. However, the state of the art for pertubations still enables the design of unbroken multivariate schemes such as $C^{*\hat{+}-}$.

The D^* scheme is more contemporary of C^* and was proposed in [89]. The idea is to use $F = x^2$ as a central polynomial over $\mathbb{F}_q^n$ where $q \neq 2$. Its security is discussed in the seminal paper [89] but without the considering perturbations like p or $\hat{+}$. More on this topic will be given in section 5.

1.5 Attacks on HFE variants

In this section, we recall the best attacks on HFE instances, that is, the best known message-recovery attacks and key-recovery attacks. The complexity of these attacks will help us benchmark our instances for the white-box implementations later.

1.5.1 Message Recovery Attacks

The idea of a direct message recovery attack is to invert the polynomial system defined by the public key for a fixed value y. To that extent Gröbner bases are the best tools available to date. The computation of these bases have greatly been improved since Buchberger original discovery, with new algorithms to compute them such as J.-C. Faugère F4 and F5 algorithms [55].

To solve a polynomial system of n equations with n unknowns with F5, the complexity is proved to be [8]:

$$\mathcal{O}\left(\binom{n+d_{reg}}{d_{reg}}^{\omega}\right)$$

where $2 \leq \omega \leq 3$ is the linear algebra constant, and d_{reg} is the degree of regularity of the polynomial system. Roughly speaking, the degree of regularity is the maximum degree that is reached during the Gröbner basis computation. As the complexity is exponential in d_{reg}, it is really important to know how to compute it for practical HFE instances.

For HFE instances, the degree of regularity is not well understood theoretically, but it has been well studied experimentally. Due to the structure of the equations over $\mathbb{F}_{q^n}$, it has been shown in [53] that d_{reg} behaves as $log_q(D)$, whatever value is chosen for n. This is highly different from what has been theoretically investigated for random systems, where d_{reg} is supposed to be close to n ([7]).

Degree of Regularity for Perturbed HFE For the pHFE$^-$ variant, this degree of regularity is conjectured to behave as

$$\left\lceil \frac{d+a-p+7}{3} \right\rceil$$

Experimentally, this linear dependence on $d+a$ has been observed in works such as [93] or the GeMSS specification [35]. The linear dependence in $-p$ has been conjectured by Patarin et al. in [91].

For the HFE$^{\hat{+}-}$ perturbation, the public-key can be reduced to a system of equations with degree of regularity conjectured in [54] :

$$\left\lceil \frac{(q-1)(d+a+t-\epsilon)}{2} \right\rceil$$

where $\epsilon = 1$ if q is even, $\epsilon = 0$ otherwise.

1.5.2 Key Recovery Rank-Attacks

To perform a key-recovery on HFE and its variants, the most efficient attacks to date are the Kipnis-Shamir attacks. In their paper [72], A. Kipnis and A. Shamir show that finding equivalent keys to an HFE instance can be done by solving a particular instance of the MinRank problem [33]. We note $C_R(n, d)$ the complexity of this attack.

Since then, variations of the attack have been thoroughly studied. Especially, variations with minor modeling and support minor have been the most successful. Recently, Ding *et al.* [103] proposed an efficient use of support minor modeling. This line of work was continued by [87] and [4] to provide a key-recovery attack that breaks any small HFE instances such as RedGeMSS128 [35]. Especially, the authors of [87] show that a key-recovery for HFE can be made with minor modeling in:

$$\mathcal{O}\left(\left[n^2\binom{2d+1}{d} + n\binom{2d+1}{d}^2\right]^\omega\right)$$

An improvement can be made when there exists $2d+1 \leq n' \leq n-a$ such that $\binom{n'}{d} \geq n$, showed by the author of [4] and the complexity is then:

$$\mathcal{O}\left(n^\omega \binom{n'}{d}^\omega\right)$$

This means that key recovery is exponential in $d = \lceil log_2(D) \rceil$, but polynomial in n. More details on the links between the parameters regime and the complexity of the attack can be found in [87].

The Projection Perturbation and Support Minor Modeling Attacks In [87], the authors analyse the complexity of rank attacks on pHFE variants. They prove that a pHFE instance with internal polynomial of degree D and p projected variables is equivalent to an HFE instance with internal polynomial of degree $2^p \times D$. Experimentally, the support minor modeling attacks on pHFE instances can be made with $d' = d + p$.

The Plus Hat Perturbations and Support Minor Modeling Attacks The authors of [54] analyse rank attacks against $HFE^{\hat{+}}$. They conjecture the best attack to be in:

$$\mathcal{O}\left(q^{t(2d+1)} \times C_R(n,d)\right)$$

where $C_R(n,d)$ is the complexity of the support minor modeling attack on the underlying HFE instance.

1.5.3 Differential Attacks

The differential attacks were introduced in [50] to attack the scheme of Matsumoto and Imai. They exploit the simple structure of the differential of the monomial used in the scheme, as well as a commutation with some multiplication operation.

Even if they were really efficient against C^*, no adaptation to HFE was ever found and the authors of [34, 42] showed that the usually chosen HFE polynomials do not have any exploitable differential structure.

Chapter 2

Affine Multiple Implementation of HFE

In this chapter, we detail our implementation technique of the HFE trapdoor. We adapt it to some perturbations of HFE and propose a challenge implementation. Finally, we conduct a security analysis of our construction in the plain white-box model.

2.1 The Technique

2.1.1 Affine Multiple Attacks

The starting point of our construction is the concept of affine multiple. It was introduced by Patarin in [88] to generalize an inversion attack on C* to HFE. The central idea of this attack is that for any polynomial $F \in \mathbb{F}_{2^n}[X]$ there always exists a polynomial $A(x, y) \in \mathbb{F}_{2^n}[X, Y]$ that is $\mathbb{F}_2$-linear in x and a multiple of the polynomial $P(x) + y$. This means that if $y = F(x)$, then $A(x, y) = 0$.

Definition 27. *Let $F \in \mathbb{F}_{2^n}[X]$. The polynomial $A(x, y) \in \mathbb{F}_{2^n}[X, Y]$ is said to be an affine multiple of F if $A(x, y) = 0 \ mod \ F(x) + y$ and A is $\mathbb{F}_2$-linear in x.*

The goal of the original affine multiple attack is to recover A via interpolation. When A is known to an attacker, they can plug any value of y to get a linear system in x of reasonable size they can solve to efficiently sign without using the structure of the equation in $\mathbb{F}_{2^n}$.

To further detail this attack, let us define the affine degree d_{aff} of an affine multiple:

Definition 28. *Let A be an affine multiple of $F(x) + y$, that is, $A(x, y) = a + \sum_{i=0}^{D-1} a_i x^{2^i}$ with $a, a_o, ..., a_{D-1} \in \mathbb{F}_2[y]$, if Mon($a_k$) is the set of the monomials of a_k*

we define d_{aff} the affine degree of A by:

$$d_{aff} := \max_k \Big(\max_{m \in Mon(a_k)} HW(deg_y(m)) \Big).$$

i.e. the maximum Hamming weight of the monomials in y in the polynomial $A(x, y)$.

We then remark that the composition by these affine transformations S and T leads to a new affine multiple that has the same d_{aff} degree. Now, to start the attack, just notice that $A(x, y)$ is composed of about $n \times \sigma(n, d_{aff})$ unknown coefficients over $\mathbb{F}_2^n$, where $\sigma(n, d_{aff})$ is the number of monomials in at most n variables in degree d. We will go through this in more details in Section 3. This means that with enough queries $(x, P(x))$, the unknown coefficients of A are the solution of a linear system of n equations with $n \times \sigma(n, d_{aff})$ unknowns that we can obtain with a Gaussian reduction. This gives us an attack, if F is known to have such affine multiple, in space $n^2 \times \sigma(n, d_{aff})$ and in time $(n \times \sigma(n, d_{aff}))^\omega$.

This attack is quite inefficient in general as the degree d_{aff} is usually quite high. Also, it is well known that the perturbation minus $(-)$ protects from affine multiple attacks [88]. It is important to remark that the cost of computing an affine multiple of $F(x) + y$ is easier when the private key (S, F, T) is known than when only the public key P is known, especially when modifiers like minus are used. This means that an affine multiple might be accessible with the knowledge of (S, F, T) but not with the knowledge of P only.

2.1.2 Rationale of the construction

The starting point of our construction is to use the affine multiple relation over $\mathbb{F}_2$ as an alternative way to inverse a public key P: the affine multiple will be our white-box implementation. Indeed, if we take $y = F(x)$ in the image of F, computing x knowing an affine multiple $A(x, y)$ boils down to plugging y onto the expression and then solving linear system of the size of x. If the affine multiple is of a reasonable size, computing x is as easy as evaluating the affine multiple in y.

2.1.3 Construction for nude Public-Keys

Computing the affine multiple

We now detail the existence of affine multiples and expose an algorithm to efficiently compute them if possible.

To prove the existence of affine multiple, let us assume that $D = deg(F)$ is such that $D < n$ and let us consider the vector space $\mathbb{F}_2(y)[x]/_{(P(x)+y)}$ of dimension $D = deg(F)$ over $\mathbb{F}_2(y)$. Now, the $(D+1)$ $\mathbb{F}_2$-linear polynomials $(1, x^{2^0}, x^{2^1}, ..., x^{2^{D-1}})$ are linearly dependent:

$$\exists\, a, a_o, ..., a_{D-1} \in \mathbb{F}_2(y)\,, a + \sum_{i=0}^{D-1} a_i x^{2^i} \ = 0 \bmod (P(x) + y)$$

To compute the coefficients a_k, we can use the reductions of the monomials x^{2^i} modulo $F(x) + y$:

$$\exists\, b_{i,0}, ..., , b_{i,D-1} \in \mathbb{F}_2(y)\,, x^{2^i} = \sum_{j=0}^{D-1} b_{i,j} x^j \bmod (P(x) + y)$$

We then reinject into the previous equation:

$$a + \sum_{i=0}^{D-1} a_i \sum_{j=0}^{D-1} b_{i,j} x^j \ = 0$$

This clearly translates into the linear system:

$$\begin{pmatrix} 1 & b_{0,0} & \cdots & b_{D-1,0} \\ 0 & b_{0,1} & \cdots & b_{D-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & b_{0,D-1} & \cdots & b_{D-1,D-1} \end{pmatrix} \times \begin{pmatrix} a \\ a_0 \\ \vdots \\ a_{D-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

We can then solve this system with Gaussian reduction in $\mathbb{F}_2(y)$. Multiplying this relation by the LCM of the $a'_k s$ denominators leads to an affine multiple polynomial $A(x, y)$. We now note $a_k \in \mathbb{F}_2[y]$ the polynomials in y obtained that way. This proves the existence of an affine multiple, and gives us at the same time an algorithm to compute it.

Remark: We do not claim the uniqueness of the affine multiple. It is indeed clear that considering D other $\mathbb{F}_2$-linear monomials leads to a similar relation, but it is not clear how it affects the coefficients a_k obtained.

The white-box construction

We now detail the construction of our white-box compiler and how to use it in practice. As we mentioned earlier, we focus on implementing the P^{-1} functionality as it is enough to guaranty unbreakability and incompressibility (see Section 4.4).

Now, to start the white-box transformation from an HFE secret key (S, F, T) over n bits - with bijective affine transformation $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and bijective affine transformation T: $\mathbb{F}_2^n \to \mathbb{F}_2^n$ and the public transformation π - the compiler computes first the affine multiple $A(x, y)$ of F over $\mathbb{F}_{2^n}$ following the algorithm described in the previous section.

From the polynomial $A(x, y)$, the compiler can now compute the coordinates $A_i(x_1, ..., x_n, y_1, ..., y_n)$ (for $i \in [\![1, n]\!]$) of A through the isomorphism π, and its composition with the secrets map S and full map T:

$$\tilde{A}_i(x_1, ..., x_n, y_1, ..., y_n) = A_i(S(x_1, ..., x_n), T^{-1}(y_1, ..., y_n))$$

Now, to compute $P^{-1}(y)$, one can plug the coordinates $y_1, ..., y_n$ of y into the polynomials $\tilde{A}_i$ to get a linear system of n equations in $x_1, ..., x_n$ that can be solved through Gaussian inversion for instance. However, one should worry that $(F(x) = y)$ only implies $(A(x, y) = 0)$, and not the reciprocate. Indeed, when we plug y in, we might get a solution x such that $(A(x, y) = 0$ and $F(x) \neq y)$. To avoid such cases, we have to verify that for the chosen y, the signature is valid (*i.e.* $F(x) = y$). As the verification is made with the public key, the time needed to perform this check is negligible.

We define the collection of the n polynomials $\tilde{A}_i$, the code for evaluation and the code for linear inversion to be the white-box implementation of the computation of P^{-1}. We note this compiler $WBHFE$. This leads to the compiling Algorithm 1 :

Algorithm 1: White-box compiler $WBHFE$

input : A HFE secret key (S, F, T) with affine projected S and affine complete T

output: $WBHFE(S, F, T)$ the white-box implementation of P^{-1} with key (S, F, T)

- Compute the affine multiple $A(x, y)$ of F with algorithm of section 3.2
- Compute the composition with secrets S and T and projection maps to get the coordinates of $\tilde{A}_i$:

$$\tilde{A}_i(x_1, ..., x_n, y_1, ..., y_n) = A_i(S(x_1, ..., x_n), T^{-1}(y_1, ..., y_n))$$

- Produce a code that partially evaluates the $\tilde{A}_i$ over the $y_1, ..., y_n$
- Produce a code that compute a pre-image of the vector 0 through linear application given by the partial evaluations of the $\tilde{A}_i$, and output "NONE" if no solution can be found.
- Produce a code that check if the message to be signed is in the image of P.
- Concatenate the produced codes to get the whole white-box implementation $WBHFE(S, F, T)$.

2.1.4 Dimensioning of the construction

The study of size of our solution boils down to two points: the computation of the multiple A over $\mathbb{F}_{2^n}$ and the size of its coordinates $\tilde{A}_i$ over $(\mathbb{F}_2)^n$. Indeed, the code size needed to perform evaluations of polynomials and Gaussian inversion is negligible. For the representation of polynomials, we use the so-called "sparse" representation to get the smaller size possible. The size analysis is of course to be linked with the white-box incompressibility which is detailed later on.

Cost of computing an Affine Multiple.

To compute the coefficients $b_{i,j}$, we have to go through the reduction of the monomials x^{2^i} modulo $F(x)+y$, with $0 \leq i < D$. Due to the reduction modulo $F(x)+y$, the degree of $b_{i,j}$ can be as big as 2^D and the $b_{i,j}$ can have up to 2^D monomials. The last part of the algorithm only solves a $D \times D$ system, leading to a worst case complexity if the polynomials $b_{i,j}$ are dense:

$$\mathcal{O}(M(n, 2^D)D^\omega)$$

where $M(n, 2^D)$ is the cost of multiplying polynomials of degree 2^D with coefficients of size n. This means that with $D = \mathcal{O}(n)$, computing this relation is usually impossible. However, we will use it with smaller polynomials than in usual HFE, *i.e.* $D = O(1)$.

Size and Running Time of the WBHFE implementation.

For the rest of the construction, let us introduce the affine degree of F. The affine degree d_{aff} is the greatest Hamming weight that appears in the description of the affine multiple A over $\mathbb{F}_2$.

Definition 29. *Let $A(x,y) = a + \sum_{i=0}^{D-1} a_i x^{2^i}$ with $a, a_o, ..., a_{D-1} \in \mathbb{F}_2[y]$, if Mon($a_k$) is the set of the monomials of a_k we define d_{aff} the affine degree of A by:*

$$d_{aff} := \max_k \Big(\max_{m \in Mon(a_k)} HW(deg_y(m)) \Big).$$

Over $\mathbb{F}_2$, $d_{aff}+1$ is the highest degree of the monomials encountered in the expression of $A(x,y)$. Indeed, $A(x,y)$ is linear in the x_i but of degree d_{aff} in the y_i.

If we note $\sigma(n, d_{aff})$ to be the number of monomials in at most n variables of degree at most d_{aff}, this means that we have about $n \times \sigma(n, d_{aff})$ coefficients over $\mathbb{F}_2^n$ needed to compute a coordinate A_i of $A(x,y)$. When we compose by S and T to get $\tilde{A}$, the overall degree does not change since S and T are affine transformations. This

means that each coordinate $\tilde{A}_i$ is composed of at most $n \times \sigma(n, d_{aff})$ monomials since S is a map from n bits to n bits.

As we are computing n coordinates, a lot of monomials will be shared in the expressions of the $\tilde{A}_i$ and it is more efficient to compute all the monomials that appear in these expressions, and then sum them. Since we want to only evaluate these polynomials in y_i to get a linear system in x_i, to then inverse it, we will represent the n polynomials $\tilde{A}_i$ as a matrix of polynomials over the y_i. This transformed expression is of the same size but will be more suited to our goals. To do so, we define the polynomials $\tilde{A}_{i,j}(y_1, ..., y_n)$ by the expression:

$$\tilde{A}_i(x_1, ..., x_{n-p}, y_1, ..., y_n) = \sum_{j=0}^{n} \tilde{A}_{i,j}(y_1, ..., y_n) \times x_j$$

Key elements of WBHFE: We can now precisely state the size of our construction. Since we will need to use great values of n, we will say in our size study that code size are 'negligible' if they are small - that is few kB. Our construction is composed of:

- A code that evaluates, on an input $m = (m_1, ..., m_n)$ of size n, all the monomials of at most degree 2 in the m_i. As we compute all the monomials, a generic code can be made. This means that this part is negligible in code size. However, this code will produce $\sigma(n, d_{aff})$ bits during its execution. We will also suppose - without loss of generality - that these monomials are computed in an ordered way, with a label from 1 to $\sigma(n, d_{aff})$.

- The $n \times n$ files $File_{i,j}$ $i,j < n$ for which the $k\text{-}th$ bit of the file $File_{i,j}$ is 1 if the $k\text{-}th$ monomial computed by the precedent code is in the expression of the polynomial $\tilde{A}_{i,j}$. These files are the heaviest part of our implementation: their size is $\sigma(n, d_{aff})$ bits. As we need each of the coordinates, the whole size is $n^2 \times \sigma(n, d_{aff})$. We divide the n^2 polynomials into n^2 files so we can load them one at a time during evaluation, with potential for parallelization.

- A code that computes the evaluation of $\tilde{A}_{i,j}(m_1, ..., m_n)$ given the evaluations of the monomials of degree 2 in m_i and the file $File_{i,j}$. To do so, one just has to go through the file $File_{i,j}$ and sum the corresponding monomials as they go. This code is negligible and can load one file $File_{i,j}$ at a time.

- A code that computes the n by n binary matrix $Mat_{\tilde{A}}$ such that $(Mat_{\tilde{A}})_{i,j} = \tilde{A}_{i,j}(m_1, ..., m_n)$. This code is also negligible.

- A code that computes a solution for the linear system $Mat_{\tilde{A}}X = 0$, $X = (x_1, ..., x_n)^T$. This can be done by Gaussian elimination. Hence, it is negligible.

- A code that checks whether m was in the image of P, i.e. if $P(x) = m$. This can be done with the public key, whose size is $n \times \sigma(n, 2)$. The rest of the code is negligible.

Size and Time This means that the code is composed of the n^2 $File_{i,j}$ for $n^2 \times \sigma(n, d_{aff})$ bits, the matrix $Mat_{\tilde{A}}$ of n^2 bits, the public key of $n \times \sigma(n, 2)$ bits, and some negligible code. The full size is then:

$$n^2 \times \sigma(n, d_{aff}) + n^2 + n \times n^2 + negl \approx n^2 \times \sigma(n, d_{aff})$$

For the rest of the paper we will define:

$$\sigma_{WB} := n^2 \times \sigma(n, d_{aff})$$

Regarding time, it is interesting to note that the computation of P^{-1} can be parallelized. Indeed, the $n \times n$ polynomials in the files $File_{i,j}$ can be computed independently. When the polynomials are evaluated, there is only a small n by n system to solve. If the evaluation of polynomials in the files $File_{i,j}$ is parallelized $n_p < n^2$ times, the time for inverting P^{-1} is:

$$\tau_{WB} := n^\omega + \frac{n^2}{n_p} \times \sigma(n, d_{aff})$$

Discussion on the affine degree.

As seen in the previous section, the size our construction is exponential in the affine degree d_{aff}, so it is important for us to understand its variations depending on F.

As a consequence of the algorithm we used to prove the existence of a multiple affine, we know that the degrees involved in the computation of the multiple are upper bounded by 2^D where D is the degree of F. The affine degree is then bounded by D but this bound is clearly an overestimate.

Through our experimentation we found that polynomials with degree of at most degree 12 can reach any affine degree ranging from 2 to 6. These experiments show that most of small d_{aff} can be reached and that there are many polynomials reaching these values. We give example of some families of these polynomials with $d = 3$ in the following table but these are just mere examples and any polynomial with the desired affine degree can be used in our construction.

Internal Polynomial F , $\forall A, B \in \mathbb{F}_2^n$	d_{aff}
$x^{12} + Ax^{10} + Bx^6$	2
$x^{12} + Ax^4 + Bx^3$	3
$x^{10} + Ax^6 + Bx^3$	4
$x^{10} + Ax^5 + Bx^3$	4
$x^{12} + Ax^{10} + Bx^5$	5
$x^{12} + Ax^5 + Bx^3$	5
$x^{10} + Ax^6 + Bx^4$	6

Table 2.1: For the polynomials F proposed, the affine multiple is easily computable with the algorithm of section 3.2. The values d_{aff} are exact, provided that – for our choice of A and B – the terms of degree d_{aff} do not vanish (this only happens on few singular points).

Besides our experiments on low degree polynomials, there are some examples that are really far from any expected bound. For instance, the Dobbertin polynomial $x^{2^{m+1}+1} + x^3 + x$ (see [47]) has a multiple affine of affine degree 3 over $\mathbb{F}_{2^{2m+1}}$ for every value m ([88]), which is – in general – really different from the observed values.

2.1.5 Using Perturbations

Usually, nude HFE instances are not sufficient by themselves to get reasonable black-box security. The goal of this section is to explain how we can turn the implementation of a nude HFE instance into a perturbed one. We will focus on three of them: p, $-$, and $\hat{+}$. For each perturbation on the public-key, we associate a perturbation or a list of perturbations between parentheses (p corresponding to (p) for instance) that are applied on the affine multiple of the nude HFE instance to match the perturbations made to the public-key.

The p Perturbation

To transform a nude public key to a one perturbed with p, one only needs to replace the bijective affine application $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$ with an affine transformation of full rank $S_p : \mathbb{F}_2^{n-p} \to \mathbb{F}_2^n$ for a small integer p.

This replacement can be made in the affine multiple representation by also replacing S by S_p, so the perturbation is compatible with the affine multiple structure.

For a HFE instance perturbed with "p", we associate its affine multiple where S is replaced by S_p. When we use this perturbation en the affine multiple, we note that $\tilde{A}$ is perturbed by (p).

The $-$ Perturbation

To transform a nude public-key to a one perturbed with $-$, one needs to remove some coordinates from the public-key. Let a be the number of equations removed. To sign with the private key, one needs to select at random these last a coordinates, and then proceed to the normal signing procedure.

The fact that the a last coordinates are random or at least secret is important to avoid any reduction to a nude public-key. Indeed, if the values of these coordinates are known for each signature produced, an attacker can then interpolate them and completely remove the $-$ modifier from the key: the last a coordinates cannot be freely chosen in the white-box model.

For instance, one could try to replace the last a coordinates of y with any linear combination of the first $n-a$ coordinates. Then, the attacker knows that the missing polynomials in x are equal to a linear application in y and can then interpolate them from some couples message-signature (even in the black-box model). Note that, in practice, the choices we can make here are limited. If we take polynomials of degree 2 in the first $n-a$ coordinates for instance, the degree of the affine multiple increases. If these coordinates depend on x the affine multiple is not linear in x anymore.

If the HFE instance is itself resistant enough in the white-box model, one possibility is to let the last a coordinate free to be chosen at random for signing:

For a HFE instance perturbed with "-", we can associate its affine multiple without any modification from the nude one, but the last a coordinate of the signature are taken at random. When we use this perturbation on the affine multiple, we note that $\tilde{A}$ is perturbed by $(-,1)$.

Remark : Even if the affine multiple is not modified we can remark that in the black-box model, the last a coordinate are random so the reduction to an HFE instance without minus is not possible in the black-box model. This means that the $(-,1)$ perturbation on the affine multiple does not carry the security from minus in the white-box model but keeps it in the black-box model. More on the security of $(-,1)$ in section 4.

A direct consequence of the previous remark is that the $-$ perturbation is not directly compatible with the affine multiple structure. To partially go around this problem, we propose a countermeasure. To efficiently interpolate the missing equations of the public key, the attacker needs to know that the a last coordinates depend from x and y in a simple way. We will then try to hide these coordinates in a single special affine multiple.

First, we decompose $y = y' + y_-$ from a direct sum $\mathbb{F}_2^n = \mathbb{F}_2^{n-a} \oplus \mathbb{F}_2^a$. The idea is to have an affine multiple that represents at least 2 fixed choices for the last a coordinates y_-. To do so, let us take any integer $n_s > 0$ and split $\mathbb{F}_2^a = \cup_{i=1}^{n_s} U_i$ where $\#U_i = \epsilon_i \geq 2$. We set $U_i = \{u_{i,1}, ..., u_{i,\epsilon_i}\}$. Now consider the polynomials :

$$G_i(x) = \prod_{j=1}^{\epsilon_i} (y' + u_{i,j} - F(x))$$

We now want an object that is similar to an affine multiple for the polynomial F, but for the polynomials G_i. To do so, we introduce the composite affine multiple:

Definition 30. *Let $\delta \in \mathbb{N}$ and $\forall i$, $F_i \in \mathbb{F}_{2^n}[X]$ and $G(x) = \prod_{i=1}^{\delta}(F_i(x) - y)$. The polynomial $A(x, y) \in \mathbb{F}_{2^n}[X, Y]$ is said to be a composite affine multiple of G if $A(x, y) = 0 \mod G(x)$ and A is $\mathbb{F}_2$-linear in x.*

Remark: It is obvious that the algorithm of section 3.3 can be adapted to compute a composite affine multiple. The modulus simply needs to be changed to the product G. Its functionality is then similar to a regular affine multiple, except the solution satisfies one of the equations $F_i(x) = y$.

A composite affine multiple $B_i(x, y')$ of G_i in unknowns x and y' can produce signatures for which the last a coordinates of the message can be any element of U_i (relatively to the nude public key). When signing in the white-box model, the value of the $u_{i,j}$ are then not revealed and the signature can be chosen randomly among the ones satisfying the $n - a$ first coordinates. An attacker can then, at best, guess that the missing coordinates lie in a set of ϵ_i values to interpolate.

To incorporate the "$-$" perturbation in the white-box model, we will use the polynomials G_i and apply the affine multiple construction:

For a HFE instance perturbed with "-", we can associate the collection of the affine multiples of the G_i. When we use this perturbation, we note that the implementation is perturbed by $(-, 2)$ with parameters $(\epsilon_1, ..., \epsilon_{n_s})$.

Remark : In the $(-, 2)$ perturbation, we provide a collection of n_s affine multiple to cover the whole vector space $\mathbb{F}_2^a$. Note that it is possible to do it for only few values, only one of the U_i for instance, and then get an implementation that produces a signature only when the last a coordinates are in U_i. This will lead to a smaller signature space and a smaller implementation size. More details on this technique will be given in section 5.

The $\hat{+}$ Perturbation

To transform a nude public key to a one perturbed with $\hat{+}$, one needs to change the central polynomial F into $F+Q$ where Q is quadratic over $\mathbb{F}_2$ and of high degree over $\mathbb{F}_2^n$ such that $\forall x \in \mathbb{F}_2^n,\ Q(x) \in V$, where V is a small vector space of dimension t.

Similarly to the "$-$" modifier, the value of $Q(x)$ for any message y cannot be known to an attacker, otherwise Q can be interpolated. That is why the $\hat{+}$ perturbation is not directly compatible with the affine multiple construction. Also, it is not possible to propose a perturbation close to $(-,1)$ as the values of Q cannot be left to be freely chosen.

We propose the same kind of perturbation as $(-,2)$ to not reveal the values of $Q(x)$. To do so, let us take any integer $m_s > 0$ and split $Im(Q) = \cup_{i=1}^{m_s} V_i$ where $\#V_i = \delta_i \geq 2$. We set $V_i = \{v_{i,1}, ..., v_{i,\delta_i}\}$. Now consider the polynomials:

$$H_i(x) = \prod_{j=1}^{\delta_i} (y - F(x) - v_{i,j})$$

A composite affine multiple $C_i(x,y)$ of H_i in unknowns x and y and fixed v_k can produce signatures for which the value of Q can be any $v_{i,j}$. An attacker can then at most guess that the missing coordinates lie in a set of δ_k values to interpolate.

For a HFE instance perturbed with "$\hat{+}$", we can associate the collection of the affine multiples of the H_i. When we use this perturbation, we note that the implementation is perturbed by $(\hat{+})$ with parameters $(\delta_1, ..., \delta_{m_s})$.

Remark : In the same spirit as the remark on the $(-,2)$ perturbation, it is possible to not use the whole collection of the H_i, but only few, or one of them. More details on this technique will be given in section 5.

Combining Perturbations

Usually, multiple perturbations are used on a nude-HFE to achieve satisfactory security parameters be it pHFE$^-$ or the more recent HFE$^{\hat{+}-}$.

For our white-box implementation, we proceed similarly, although the compatibility depends on the perturbation used. For instance, the (p) and $(-,1)$ perturbation are easily compatible with $(\hat{+})$ but $(-,2)$ and $(\hat{+})$ needs more a subtle adaptation to work together: instead of splits of $\mathbb{F}_2^a$ and $Im(Q)$, one needs to work with a split of $\mathbb{F}_2^a \times Im(Q)$ and work with affine multiples of products of polynomials of the form $(y + u - F(x) - v)$, with $u \in \mathbb{F}_2^a$ and $v \in Im(Q)$.

In section 5, we detail more on the compatibility of perturbations for affine multiples in a case by case basis on the implementation.

2.2 Security analysis

In this section, we define the white-box security notions of unbreakability and incompressibility for public-key signature algorithms. We then analyse the security of our implementation designs for these notions. For the rest of this section we suppose that A is an affine multiple of F or a composite affine multiple with respect to perturbations of section 3.5.

We will study the unbreakability and incompressiblity of our construction. We recall that for unbreakablility, studying the unbreakability of the one-way function is sufficient (part I, chapter 1).

2.2.1 Attack by Reduction to a Weaker HFE Instance

This paragraph only deals with HFE implementations with perturbations. If perturbations are used on an implementation (section 3.5), one idea to perform an attack is to try to remove the perturbation on the affine multiple construction, in the same way attackers try to remove perturbations on public-keys for attacks in the black-box model. We describe such reductions for some of the perturbations described in section 3.5.

Reduction for (-,1) If we have an implementation perturbed with (-,1), we can remove the perturbation - on the public-key by recovering the missing coordinates. We can fix the last a coordinates to any constant and gather enough signatures to solve a linear system with the coefficients of the monomials of our missing coordinates as unknowns. Experimentally, this allows us to recover vector space of dimension a that contains these equations, which is sufficient to recover an equivalent HFE public key. Since there are $\binom{n}{2}$ unknowns, the cost of this attack is $\mathcal{O}(n^{2\omega} + n^2 \times \tau_{WB})$.

Remark 1: The time τ_{WB} can be increased by other used perturbations, and needs to be taken into account for concrete security analysis.

Reduction for (-,2) With the (-,2) perturbation, the a last coordinates are not known: they are grouped by sets of ϵ_i unkown values. Indeed the affine multiples B_i of the polynomials G_i only provide a signature if the last coordinates are in U_i. If the same attack as for (-,1) cannot be used, we can still know that the signatures from B_i will satisfy:

$$\prod_{j=1}^{\epsilon_i}(\Pi_a(P(x)) + u_{i,j}) = 0$$

where Π_a is the projection onto $\mathbb{F}_2^a$. This product can then be interpolated by gathering signatures, using the same method as in the reduction for $(-,1)$. Then, a recovery of $\Pi_a(P(x))$ can be attempted. In our setting, the exact values $u_{i,j}$ are unknown. To the best of our knowledge, recovering $\Pi_a(P(x))$ can only be made by factoring $\prod_{j=1}^{\epsilon_i}(\Pi_a(P(x)) + u_{i,j}) = 0$, which is hard for any instance we will use later.

Remark 2: Note that the same polynomial relation can be recovered for any HFE$^-$ instance in black-box for $\epsilon_i = 2^a$. This means that the security problem we have identified is similar to recovering the missing equations in a HFE$^-$ instance for small values a.

Reduction ($\hat{+}$) For the ($\hat{+}$) perturbation, the same strategy as (-,2) can be used. Indeed for the affine multiples C_i of the polynomials H_i the only possible values of $Q(x)$ are in V_i. The signatures from B_i will satisfy:

$$\prod_{j=1}^{\delta_i}(Q(x) + v_{i,j}) = 0$$

This product can then be interpolated by gathering signatures using the same method as for the reduction for $(-,1)$. Then, a recovery of $Q(x)$ can be attempted using the same method as for $(-,2)$, except this time, the entire set $Im(Q)$ is unknown. This reduction is hence similar than the (-.2) reduction.

Remark 3: Note that these equations can be gathered for any HFE$^{\hat{+}}$ instance in black-box for $V_i = Im(Q)$. This means that any algorithm solving this problem if $\#V_i = 2^t$ can break the corresponding HFE$^{\hat{+}}$ instance. Even if this perturbation is young, no attack of this kind has been reported by the authors of [54] which confirms our security analysis.

2.2.2 The Implementation as a $(d_{aff}+1)$-IP1S Problem

To analyse unbreakability and incompressibility directly on the affine multiple, we rely on two properties of the polynomials describing our white-box implementation. The first will be the key recovery of the underlying $(d_{aff}+1)$-IP1S (Isomorphism of polynomials with one secret whose internal polynomial is of degree $(d_{aff}+1)$) instance, a problem that has been studied in papers such as [29, 79, 90, 92] to explore the security of multivariate cryptography in general. The second one is a variant of the regular IP1S problem that we call "incompressibility of IP instances". We detail these two properties,

how they are linked to our problem and how well studied they are. For this section, let A be an affine multiple of any HFE instance with perturbations (p), (-,1), (-,2) or $(\hat{+})$.

Secret recovery on $(d_{aff}+1)$-IP1S

Let us first recall that our white-box implementation is composed of the n polynomials $\tilde{A}_i$ and a deterministic generic way to evaluate them (compute all the monomials then sum them up). The polynomials $\tilde{A}_i$ are defined by a composition with two affine transformations S and T such that:

$$\tilde{A}_i(x_1,...,x_n,y_1,...,y_n) = A_i(S(x_1,...,x_n),T^{-1}(y_1,...,y_n))$$

It is then obvious that $\tilde{A}_i$ is an instance of a $(d_{aff}+1)$-IP1S problem over $2n$ variables, with A_i as the known polynomials and a block-affine transformation composed of S and T. This problem has a structured secret, so it is not generic, but we do not know any other attack against it (as an IP problem) than the generic ones. To the best of our knowledge, the best generic attack on $3-$IP1S with affine secrets has complexity $\mathcal{O}(n^6q^n)$ ([29]). This means that the best known attack against $3-$IP1S instances is exponential in n . For our instances, $(d_{aff}+1) \geq 3$, these pieces of evidence allow us to conjecture that our instances are secure for the desired security level.

Remark 1: For the perturbation (p), the secret $S : \mathbb{F}_2^{n-p} \to \mathbb{F}_2^n$ is not a bijection, this instance is different from the one studied in general. However, as the projection variant is efficient against key recovery for HFE instances, one can hope that it will help our IP instance to stand against secret-recovery for the same reasons.

Remark 2: For perturbations $(-,2)$ and $(\hat{+})$, the polynomial $\tilde{A}$ are composite affine multiples, but the same analysis can be made.

Incompressibility of IP1S instances

The main goal of this part is to highlight a specificity of multivariate cryptography in general that will help us to prove the incompressibility of our white-box construction. To do so, we formalize a new problem around IP instances, and analyse it on our instance $(\tilde{A}_i)_{i\in[\![1,n]\!]}$.

We define the (σ,τ)-incompressibility of an IP instance with known polynomials $(P_i)_{i\in[\![1,m]\!]}$:

- Draw at random two secret affine transformations S,T in $AFF_n(\mathbb{F}_2)$
- The adversary $\mathcal{A}$ is given an IP instance $(\tilde{P}_i)_{i\in[\![1,m]\!]}$ composed of $(P_i)_{i\in[\![1,m]\!]}$, S and T

- The adversary $\mathcal{A}$ returns a program $\mathcal{P}$ that allows to evaluate $(\tilde{P}_i)_{i\in[\![1,m]\!]}$ for every element $(\mathbb{F}_2)^n$
- The adversary $\mathcal{A}$ wins if $size(\mathcal{P}) \leq \sigma$

Definition 31. *Let $(\tilde{P}_i)_{i\in[\![1,m]\!]}$ be an IP instance with polynomials in n variables over $\mathbb{F}_2$, with known polynomials $(P_i)_{i\in[\![1,m]\!]}$ and secrets S, T and let $\mathcal{A}$ an adversary. We say that $(\tilde{P}_i)_{i\in[\![1,m]\!]}$ is (σ,τ)-incompressible if there is no adversary $\mathcal{A}$ that wins the σ-incompressibility game with probability 1 and Time($\mathcal{A}$)+Time($\mathcal{P}$) $< \tau$.*

Remark 1: We could also consider, similarly to the incompressibility for white-box, that $\mathcal{A}$ does not have to agree with $(P_i)_{i\in[\![1,m]\!]}$ on all inputs or that it can be probabilistic. However, known attacks do not use this flexibility.

It is well known that, for truly random polynomials, compressibility is not possible by definition in the sense of Kolmogorov, even with an unbounded computation power. In contrast, in our context, a compressed version of the $\tilde{A}_i$ polynomials is obviously given if we can recover the secrets S and T. This problem of secret recovery on an IP instance corresponds to the extreme case $\sigma = size(S) + size(T)$ in Definition 4, and boils down to the unbreakability problem, for which the best known attacks require a computational effort way larger than 2^{80} (see paragraph above, about Secret recovery on $(d_{aff}+1)$-IP1S). In the intermediate cases $size(S) + size(T) < \sigma \leq size(P_i)$, to the best of our knowledge, no attack has been found in the literature.

2.2.3 Generic White-Box Attacks on Multivariate Cryptography

In the literature, generic automated attacks are proven to be very efficient against the state-of-the-art white-box implementations of block-ciphers due to the techniques used (i.e masking or internal encodings for instance). These attacks include Differential Computation Analysis (DCA) and Differential Fault Analysis (DFA) as their most potent representatives. In this section we argue against their usefulness against our technique.

The main point of this section is to understand how IP instances are secure against these attacks, even if they are used in HFE schemes and hence have a trapdoor. A first example of this is how HFE public keys are not vulnerable to DPA attacks or their white-box DCA counterpart. Indeed, even if the inversion of the public key can be attacked because it decomposes the inversion of P into the inversion of S, T and F separately, the public key itself is not vulnerable against DCA, even if it contains the complete key (S, T). This is due to the structure of the IP problem. Indeed, all the bits of S and T are diffused by polynomial composition into the coefficients of the public key P. This means that unless a specific computation depending on few key bits

is found on a specific instance, the probability is negligible that a "nice" target exists for generic white-box attacks, as the complexity of DCA is exponential in the number of key bits on which the target depends. The state of the art HFE security ignores these attacks for such reasons. For DFA, it is easy to make faults on the evaluation of P, but any evaluation of any function depending on the polynomials' coefficients of the public is already allowed to solve IP and hence does not provide new information. This is a huge difference compared to state-of-the-art targets of these attacks, which usually are S-Boxes in SPN schemes, and most often specifically those of the AES algorithm.

For our implementation, the affine multiple structure allows to diffuse S and T into A, in the same way they are diffused into the public key. Unless a specific relation is found for this particular instance, the DCA has no more chances to succeed than the ones targeting the public key. The same argument also stands for DFA. In summary, such a DCA-like (resp. DFA-like) attack is not expected to be able to circumvent the similar hard-to-solve algebraic problem on which the algorithm's black-box security relies.

Remark: We can think of the following factorization-flavoured analogy. Assume an attacker is given an RSA public key. Since n depends on the secrets primes p and q, one could wonder if DCA (resp. DFA) could be applied to n (or to a computation making use of n) to recover the secret elements p and q. However, it is also easy to see that in this case the algebraic complexity of the bits of n (as functions of the bits of p and q) quickly get so high that this kind of DCA (resp. DFA) strategy is not relevant here.

2.2.4 Conclusion of the Analysis of Security

In this section we synthesize the analysis of security above into two conjectures of similar nature, one for unbreakability, one for incompressibility.

Unbreakability Conjecture:

Let P be a HFE public key with modifiers chosen among $-$, p and $\hat{+}$. Let **WBHFE** be the white-box implementation of P^{-1} and A an associated composite affine multiple with corresponding perturbations. Let $\epsilon < 1$ be a small probability and λ be our security level. If

1. The HFE instance associated to the public key P is secure against key recovery in the black-box model up to security level λ.
2. The knowledge of A does not help to remove modifiers from P (Section 4.2) in less than 2^λ operations.
3. The affine multiple A is $(2^\lambda,\epsilon)$-unbreakable as a $(d_{aff}+1)$-IP1S instance. (Section 4.3.1)

then the implementation **WBHFE** of the primitive P^{-1} is $(2^\lambda,\epsilon)$-unbreakable in the white-box model.

Remark 1: The first point of this conjecture is trivial if we want our HFE instance to be of any use. However, as we will see in section 5, this is an important dimensioning parameter to optimize the implementation size. This is why we make it part of the conjecture.

Incompressibility Conjecture:

Let P be a HFE public-key with modifiers chosen among $-$, p and $\hat{+}$. Let **WBHFE** be the white-box implementation of P^{-1} and A an associated composite affine multiple with corresponding perturbations. Let $\epsilon < 1$ be a small probability and λ be our security level. If :

1. The HFE instance associated to the public-key P is secure against key recovery in the black-box model up to security level λ.
2. The knowledge of A does not help to remove modifiers from P (Section 4.2) in less than 2^λ operations.
3. The affine multiple A is $(\sigma_{WB},2^\lambda,\epsilon)$-incompressible as a $(d_{aff}+1)$-IP1S instance. (Section 4.3.2)

then the implementation **WBHFE** of the primitive P^{-1} is $(\sigma_{WB},2^\lambda,\epsilon)$-incompressible in the white-box model.

Remark 2: This conjecture is very similar to the unbreakability one. This is due to the fact that the only compression we know from the public key is key recovery.

Of course, proving this conjecture still requires new insights, in particular to clarify the deep algebraic links between the polynomial systems arising from $A(x, y)$ on the one hand, and from P^{-1} on the other hand. However, we believe this paves the way for a better understanding of the incompressibility property, which up to now could be formally verified only for white-box implementations of symmetric cryptosystems in very restricted models (see the proof of incompressibility of an RSA-like symmetric encryption scheme in [43], using an Ideal Group Model).

2.3 Instantiations

In this section, we propose instances of HFE based on the construction of section 3 according to the security analysis of section 4 in the white-box model. We propose challenges for a subset of them.

2.3.1 Nude HFE

As a starting example, we propose a Nude HFE instantiation. This first example is a proof of concept and shows the need of perturbations to reach any reasonable size of implementation.

Security analysis Let us first recall the main attacks in the black-box model. Our instance needs to stand against direct inversion with Gröbner bases and key-recovery rank attacks (Section 2.10). As our instance will not have any perturbation, it also needs to be protected against the affine multiple attack.

For the white-box security, according to our conjecture, the instance needs to stand against attacks on d_{aff}-IP1S instances of section 4.4. As the HFE instance will be nude, the problem of removing modifiers is not relevant here.

Parameter Choices The goal of this section is to minimize the size of the implementation for a given security level. For the Nude HFE instance, the only parameters to be chosen are n and d_{aff}. For a better compromise between size and security, we can take polynomials for which there exist affine multiples with $d_{aff} = 2$ and d and d_{reg} is maximal. The maximum we experimentally found is $d_{reg} = 3$ for internal polynomials of the form $F = x^6 + Ax^5 + Bx^3$, $A, B \in \mathbb{F}_2^n$. Note that getting higher values of d_{reg} for the same d_{aff} would improve the security/size trade-off. The instances we consider are resistant to the attacks of section 4.4.2.

Target	$(log_2(n), d)$	$log_2(\sigma_{WB})$	C_G	C_R	C_{AFF}
Smallest for $\lambda = 80$	(11.76,3)	45.7	91.1	80	95.5
Smallest for $\lambda = 128$	(20.22,3)	79.8	137	128	167.6

Table 2.2: Set of parameters d_{aff} and n which satisfies particular security levels λ. The value $log_2(\sigma_{WB})$ is the corresponding log_2 of implementation size in bits. The values C_G, C_R and C_{AFF} are the log_2 of the complexity of respectively best Gröbner basis attack, best rank attack and best affine multiple attack.

Remark 1: It clearly appears that d being small is a problem to scale the construction. While we use perturbations to partly solve this problem, having a better understanding of the affine degrees could lead to smaller implementations (section 3.4.3). This remark carries onto the following instances.

2.3.2 Instances close to pC^{*-}

As the Nude HFE instance is too weak for concrete use, we include state-of-the-art perturbations p and $-$. To do so, we include perturbations (p) and $(-, 2)$ on the affine multiple. This means that the implementation is composed of the composite affine multiples of the polynomials G_i:

$$G_i(x) = \prod_{j=1}^{\epsilon_i} (y' + u_{i,j} - F(x))$$

for which the last p coordinates of x are projected. For this section, we will choose internal polynomials of the form $F = x^3 + Ax^2$, thus close to C^*.

Security analysis As for Nude HFE our instance needs to stand against direct inversion with Gröbner bases and key-recovery rank attacks (Section 2.10). However, the affine multiple attack is not applicable to public keys with $-$.

For white-box security, according to the conjecture, our instance needs to stand against attacks on d_{aff}-IP1S instances of section 4.4. This time, we need to ensure that perturbations cannot be removed in the white-box model. According to our conjecture, we only need to ensure that the attack of section 4.3 cannot be used. To do so, we take $\epsilon_i = 3$ so that the polynomial

$$\prod_{j=1}^{\epsilon_i} (\Pi_a(P(x)) + u_{i,j}) = 0$$

is not linear in $\Pi_a(P(x))$.

Parameter Choices The main problem to minimize the implementation size for a given security level is that the affine degree of the affine multiple is high if we take F at random as soon as $\epsilon_i \geq 2$. To reduce this degree, we choose polynomials F of the

form $x^3 + Ax^2$, $A \in \mathbb{F}_2^n$ and take $\epsilon_i = 3$. Indeed, with these parameters, we get affine multiples with $d_{aff} = 3$. We then optimize n, p and a for the desired level of security.

Target	(n, d, p, a)	$log_2(size((B_i))$	C_G	C_R
Smallest for $\lambda = 80$	(101,2,12,21)	30.5	86.4	83
Smallest for $\lambda = 90$	(116,2,14,26)	31.5	101.3	91.6
Smallest for $\lambda = 128$	(169,2,23,41)	34.3	136.5	130

Table 2.3: Set of parameters d_{aff}, a, p and n which satisfies particular security level λ. The value $log_2(size(B_i))$ is the corresponding log_2 of size of an affine multiple in bits. The values C_G and C_R are the log_2 of the complexity of respectively best Gröbner basis attacks and best rank attacks.

Remark 1: To get an implementation of the complete signature algorithm, one needs to gather all the affine multiple B_i. However, with only one of them, a signature can be computed if the last a coordinate of the signature lie in U_i. This means that a message drawn at random will be signed with roughly probability $\frac{3}{2^a}$. This probability is quite low but this remark can be used to have a smaller signature algorithm in the white-box model while not changing the public key and the verification algorithm.

2.3.3 Instances close to $C^{*\hat{+}-}$

In this section we include the perturbations $\hat{+}$ and $-$. To do so, we include perturbations $(\hat{+})$ and $(-, 1)$ in the affine multiple. This means that the implementation is composed of the composite affine multiples of the polynomials H_i:

$$H_i(x) = \prod_{j=1}^{\delta_i} (y + v_{i,j} - F(x))$$

and the last a coordinates of y are chosen at random for signing. We will choose internal polynomials of the form $F = x^3 + Ax^2$, thus close to C^*.

Security analysis The analysis is similar to the previous one, with p changed to $\hat{+}$ for better security against rank attacks: our instance needs to stand against direct inversion with Gröbner bases and key-recovery rank attacks (Section 2.10). However, once again, the affine multiple attack is not applicable to public keys with $\hat{+}$ and $-$.

For white-box security, according to the conjecture, our instance needs to stand against attacks on d_{aff}-IP1S instances of section 4.4, i.e. we only need to ensure that

the attack of section 4.3 cannot be used. To do so, we take $\delta_i = 3$ so that the polynomial

$$\prod_{j=1}^{\delta_i}(Q(x) + v_{i,j}) = 0$$

is not linear in $Q(x)$.

Parameter Choices For the same reasons as for the previous instance, to reduce the affine degree, we choose polynomials F of the form $x^3 + Ax^2$, $A \in \mathbb{F}_2^n$ and take $\epsilon_i = 3$. Indeed, with these parameters, we get affine multiples with $d_{aff} = 3$. We then optimize n, t and a for the desired level of security.

Target	(n, d, t, a)	$log_2(size((C_i))$	C_G	C_R
Smallest for $\lambda = 80$	(85,2,9,5)	29.5	82.4	80.3
Smallest for $\lambda = 90$	(96,2,11,6)	30.3	106.5	91.3
Smallest for $\lambda = 128$	(132,2,18,4)	32.6	138.3	128.9

Table 2.4: Set parameters d_{aff}, a, t and n which satisfies a particular security level λ. The value $log_2(size(C_i))$ is the corresponding log_2 of size of an affine multiple in bits. The values C_G and C_R are the log_2 of the complexity of respectively best Gröbner basis attacks and best rank attacks

Remark 3: Similarly to the previous instance with one of the affine multiples C_i, a signature can be computed if its image through Q lies in V_i. This means that a message drawn at random will be signed with roughly probability $\frac{3}{2^t}$. This probability is quite low but this remark can be used to have a smaller signature algorithm in the white-box model while not changing the public-key and the verification algorithm.

Challenge: To motivate the cryptanalysis of our technique, we propose a challenge implementation corresponding to the line $(85, 2, 9, 5)$ on the previous table. The code in Sage[1] contains the public-key, one affine multiple and resources to manipulate them.

2.3.4 Instances close to $D^{*\hat{+}-}$

As a last example, we propose to use a variation of the long standing D^* scheme. Even if the security of D^* and its variations have not been studied in the recent literature, we use this instance to motivate the understanding of affine multiple structures over

[1]available at the following URL: `https://github.com/p-galissant/WBHFE`

different field: the composite affine multiples of the $D^{*\hat{+}-}$ have the smallest d_{aff} among the ones presented here.

One of the problems of the previous instances is that the affine multiples have to be built with $\delta_i = 3$ so that attacks from section 4.3 cannot be used. This is due to the fact that squaring is linear over $\mathbb{F}_2$. The signature algorithm D^* is however designed over $\mathbb{F}_q$, $q \neq 2$, with internal polynomial $F = x^2$: we want to use this fact to find small affine multiple with $d_{aff} = 2$.

To do so, the public key is an instance with $F = x^2 + Ax$ for any $A \in \mathbb{F}_3^n$ and use the $\hat{+}$ and $-$ perturbations. We include perturbations $(\hat{+})$ and $(-, 1)$ on the affine multiple, as in section 5.3. This means that the implementation is composed of the composite affine multiples of the polynomials H_i:

$$H_i(x) = \prod_{j=1}^{2}(y - v_{i,j} - x^2 - Ax)$$

with $v_{i,j}$ in the image of Q.

Security analysis The main problem with D^* without perturbations is that polarisation attacks ([89]) easily break the public-key. However, the recent $\hat{+}$ makes the polarisation attack fail, for Q is not a square only.

Even if $D^{*\hat{+}-}$ is not a state-of-the-art algorithm, we will assume the same relevant attacks as in the case of HFE, that is to say rank attacks and direct inversion attacks. While it might need more research to confirm the soundness of this assumption, we will rely on these attacks to parameterize our proof of concept.

Parameter Choices The parameters n, t and a are the same as for the previous section. Indeed, we only we get a variation of d_{aff} from 3 to 2.

Target	(n, d, t, a)	$log_2(size((C_i))$	C_G	C_R
Smallest for $\lambda = 80$	(85,2,9,5)	25.7	82.4	80.3
Smallest for $\lambda = 90$	(96,2,11,6)	26.4	106.5	91.3
Smallest for $\lambda = 128$	(132,2,18,4)	28.2	138.3	128.9

Table 2.5: Set parameters a, t and n which satisfies a particular security level λ. The value $log_2(size(C_i)$ is the corresponding log_2 of size of an affine multiple in bits. The values C_G and C_R are the log_2 of the complexity of respectively best Gröbner basis attacks and best rank attacks

Chapter 3

Another Multivariate Scheme and Some Perspectives

The goal of this chapter is to shortly discuss how multivariate cryptography problems such as the IP problem can be used for white-box cryptography in general and why multivariate cryptography is an adapted setting for white-box cryptography. As an example, we show a construction based on QUAD that inherently has incompressible properties.

3.1 Introducing IP-like problems for White-Box Cryptography

The implementation we made is essentially based on a variation of the IP problem. This 'decomposition' problem is very interesting in the white-box model as it let diffuse all the key bits into the implementation at once, which is very unusual for white-box techniques. Usually, the key is 'glued' to a local transformation and hidden with an encoding. The key itself is not diffused into the code and the encoding only locally randomizes the implementation. This leads to very efficient automated attacks on target functions with small key-spaces.

The HFE cryptosystem itself shows that there are inherent white-box properties to be found in multivariate cryptography. If the HFE instance is secure, the public key can be seen has an incompressible implementation of the composition of S, P and T. Studying links between white-box cryptography and multivariate cryptography is surely a refreshing and fruitful approach.

For cryptosystem not related to multivariate cryptography, we believe this approach can also be interesting. First, polynomial transformations are a natural extension of the table-based methods to extend the support of the functions while keeping succinct

representations. Attempting to composing them with techniques similar to the encoding rationale would lead to IP-like problems. Our AES construction of part II is made in this spirit.

One of the main problems to this approach is to actually decompose the algorithm into polynomials of small degrees. If ad hoc techniques might be successful for specific cryptosystem, we believe that structural techniques such as the affine multiple we used for HFE need to be investigated. The ARX implementation of Ranea *et al.* [95] is an example of such attempt, but it also uses external encodings. To us, the structure of public-key cryptosystems can only offer more flexibility to white-box designers.

3.2 An example : A Stream Cipher in the white-box model

In this section, we adapt the stream cipher QUAD [11] to get incompressible properties in the white-box model. Unlike HFE, QUAD is a private-key algorithm. To do so, we replace the random polynomial system that allows to compute the stream by a structured one. This approach is similar to the design of incompressible block-cipher presented in part I.

3.2.1 Description of QUAD

We describe QUAD over $\mathbb{F}_2$, knowing that it can be done in any characteristic. In the stream cipher QUAD published by Berbain *et al.* [11], the key stream generation is made as follows. For any integers n and k, let S be a system of kn polynomials in n variables over $\mathbb{F}_2$:

$$S = (Q_1, ..., Q_{kn})$$

Then define S_{out} and S_{it} as subsets of S: $S_{it} = (Q_1, ..., Q_n)$ and $S_{out} = (Q_{n-1}, ..., Q_{kn})$. The keystream generation is composed of the iteration of S_{it} in the following way. Let x be the intermediate state:

- Compute the system $S(x)$
- Output $S_{out}(x)$ as key-stream values
- Update the internal state $x \leftarrow S_{it}(x)$

For initialization, assume that the secret key K and vector IV are defined by a n-bit vector and that we have access to other n-by-n random systems of polynomials of degree 2, namely S_0 and S_1. We initialize the internal state x to the value K. Then,

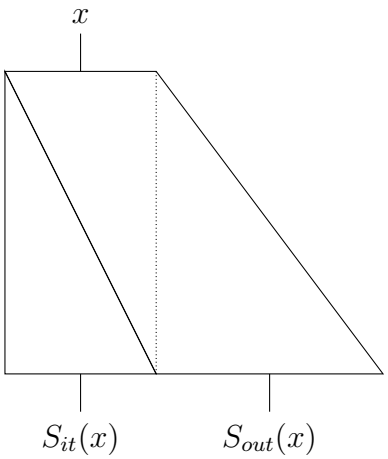


Figure 3.1: QUAD keystream generation

for each bit of IV, the state is updated as follows: starting from $i = 1$, $IV_i = b$, x is updated by $S_b(x)$. Note that in QUAD, the system S, S_0 and S_1 are made public.

The authors of QUAD prove that, provided S, S_0 and S_1 are random systems, their algorithm is secure. More precisely, they show that if the MQ problem stands for these systems of polynomials, then the stream cipher is secure.

3.2.2 A variation of QUAD with small representation

The original QUAD algorithm uses random systems S, S_0 and S_1 with n variables over $\mathbb{F}_2$. Such systems are inherently incompressible as random systems of equations. We now propose to introduce a 'trapdoor' in these systems to get a compact version of it. We can then have a short description of this algorithm, and use the whole system of polynomials as the incompressible representation.

To do so, we will generate the system S as a pHFE$^{\hat{+}-}$ public key. While this variation is not used for signature due to the large overheads induced by $\hat{+}$ and p, we only use the system in the direct sense, so that these concerns are not relevant here. We then conjecture that if the MQ problem stands on the HFE instance we propose, the tweaked QUAD stream cipher admits incompressible implementations in the white-box model, with a non negligible gap between the short and long implementations.

HFE-TRAP-QUAD We now define a new stream cipher based on QUAD. Instead of taking a random polynomial S, we use a system $S_{(S,F,T),Q}$. This system will be secret unlike regular QUAD. It is defined in the following way. Let (S, F, T) be a naked HFE instance over $\mathbb{F}_{2^{(k+1)\times n}}$. We now perturb the naked public key $P = T^- \circ \pi^{-1} \circ F \circ \pi \circ S$

with $p = kn$, $a = n$ and the $\hat{+}$ perturbation with quadratic form Q with parameter $t \in \mathbb{N}$. We get a perturbed public key denoted by $S_{(S,F,T),Q}$, omitting indexes p and a since they are fixed. Due to the chosen parameters, the public key $S_{(S,F,T),Q}$ is a system of kn polynomials over n variables. We repeat this process with $k = 1$ to define two systems $S_{(S_0,F_0,T_0)_0,Q_0}$ and $S_{(S_1,F_1,T_1),Q_1}$, with the same trapdoor structure.

We define the HFE-TRAP-QUAD stream cipher by using the exact same procedure for key-stream generation, except the private key is defined by the original private key K of QUAD, plus the secrets of the HFE instances, that is $((S, F, T), Q)$, $((S_0, F_0, T_0), Q_0)$ and $((S_1, F_1, T_1), Q_1)$. We then define the white-box implementation of WBHFE-TRAP-QUAD by the implementation of HFE-TRAP-QUAD where the systems polynomials $S_{(S,F,T),Q}$, $S_{(S_0,F_0,T_0),Q_0}$ and $S_{(S_1,F_1,T_1),Q_1}$ are computed with the knowledge of this secret key and stored that way. The rest of the implementation is identical to a naive implementation of QUAD. If the secret key cannot be recovered from the polynomials, the implementation is obviously unbreakable. For incompressibility however, we state the following conjecture.

Conjecture: Let $S_{(S,F,T),Q}$, $S_{(S_0,F_0,T_0),Q_0}$ and $S_{(S_1,F_1,T_1),Q_1}$ be the HFE instances defined above. If these instances define secure trapdoor one-way functions, the stream-cipher HFE-TRAP-QUAD is secure in the black-box model. If these instances are incompressible, the compiler WBHFE-TRAP-DOOR is incompressible. More precisely, if for any $\epsilon < 1$, $S_{(S,F,T),Q}$ is (σ, τ, ϵ)-incompressible, $S_{(S_0,F_0,T_0),Q_0}$ is $(\sigma_0, \tau_0, \epsilon)$-incompressible and $S_{(S_1,F_1,T_1),Q_1}$ is $(\sigma_1, \tau_1, \epsilon)$-incompressible, WBHFE-TRAP-QUAD is $(\sigma + \sigma_0 + \sigma_1, \tau + \tau_0 + \tau_1, \epsilon)$-incompressible in the plain white-box model.

Remark: The incompressibility property of the implementation does not trivially stems for the incompressibility of the system. Indeed, similarly to the HFE implementation earlier, the stream-cipher generation functionality is not equivalent to the functionality of evaluating the polynomials but a subset of it.

We shortly discuss the size of the private key of HFE-TRAP-QUAD. For the system $S_{(S,F,T),Q}$, the size of the underlying HFE key is essentially the size of the matrices S and T and the size of the structure quadratic form Q which is composed of t random quadratic polynomials. As it is the same for $S_{(S_0,F_0,T_0),Q_0}$ and $S_{(S_1,F_1,T_1),Q_1}$ but with only n variables, the size of the private key is dominated by $2(kn)^2 + t \times (kn)^2$. Once the HFE public keys have been composed, we obtain a system of kn equations in n variables for $S_{(S,F,T),Q}$, and systems of n equations in n variables for $S_{(S_0,F_0,T_0)_0,Q_0}$ and $S_{(S_1,F_1,T_1),Q_1}$. Their total size is then dominated by $kn \times n^2$. This means there is essentially a gap of size n between the 'compact' private key and the incompressible representation. As values of k such as 2 or 3 are recommended in [11] and small t can be taken for secure HFE instances, the incompressible implementation is n times bigger than its private key.

Conclusion of part III

The white-box state of the art of public-key cryptography is really young. It is however a subject that needs more attention considering the growing needs in the industry.

We propose the first white-box implementation technique for the inversion of the HFE trapdoor one-way function based on the structure of its 'big field' representation. This technique is based on the affine multiple principle, which relies on the vector-space structure of the big field representation. After proposing a first implementation for nude HFE instances, we incorporate the perturbations $p, -$ and $\hat{+}$ into the design strategy. We then propose a security analysis that is based on a variation of the IP problem, and a challenge implementation to motivate cryptanalytic studies of this new kind of white-box implementation.

We then support the use of multivariate cryptography at large in the white-box context, and show some inherent white-box properties of multivariate schemes through an example based on QUAD.

As research on white-box cryptography has been mainly split between designing implementations of AES and attacking them, and theoretical constructions, there undoubtedly remain unturned stones for white-box implementations of public-key algorithms, and many areas are to be explored.

If one can look into new structures that support this 'double trapdoor' behavior of white-box signature implementation, one can also attempt to better understand the behavior of affine multiples. Indeed, we only explored affine multiples with a practical viewpoint, and a better understanding of their affine degree is a goal for future works, in order to generalize our implementation techniques to more HFE instances, or even to other multivariate schemes.

General Conclusion

Assessing Security Contexts and Adversaries

In this thesis, we showed that in the state of the art, depending on the modeling of the attacker and the devices, an array of different security notions can be considered and the methods to achieve them can be vastly different. For instance, the use of hardware modules can enable very strong properties, but what are the adapted uses of an hardware in the context of a white-box study?

In that sense, we think that the concrete use cases for white-box cryptography need to be clarified. A consultation between manufacturers and academics, to establish the concrete needs of the industry in terms of security goals and implementation environments, would be very useful. Some questions naturally emerge: is a secure hardware available at all? is there a control of the communications of the device? or is its computational power limited? To us, asking these question is essential, and precisely quantified answers could directly influence the production of hardware (see Apple's phones for instance) and the accessibility of such technologies to individuals without the restraints of a specific hardware.

As we relax the white-box model to consider the hardware-module white-box model, we should also be careful of introducing again the problems of mistrusted hardware. We can think of an 'hybrid' construction in the following sense. An implementation can be first constructed in the plain white-box model and then transformed into a hardware-module implementation. That way, if the hardware on which the security in the hardware module is malicious or faulty, the underlying security of the implementation in the plain white-box model mitigates the flaws.

With the ever increasing needs of white-box cryptography, we also believe that the standardization initiatives should lean toward imposing constraints of compatibility with white-box on their candidates. This is already the case for special use cases such as lightweight cryptography as shown by the recent NIST competition. If standardization does not include these constraints, the design of white-box implementation might be very arduous, as the post-quantum NIST candidates based on LWE may illustrate since they require precise noise distributions to be secure. In the plain white-box model, using randomness is surely a hard problem.

White-Box Implementation of Standards, Mostly the AES

In the last few years, few different implementation techniques appeared, especially for the AES. The implementation we propose is in a line of work that attempts to replace the countermeasures coming from the grey-box model, and based on randomness, by

countermeasures based on the algebraic or combinatorial complexity of finding relations between sensible functions and values computed by the implementation. The questions to know if such design rationale can work in open design, and if a structural improvement can be gained by hidden design, are still open.

Considering new techniques, one track would be to consider designs inspired by constructions of cryptographic obfuscators. For instance, considering universal circuits – such as the constructions of part I – would reduce the problem of white-boxing particular algorithms to the evaluation of encoded inputs. This change of viewpoint has been very fruitful for iO, and could probably help to produce new designs.

The underlying problem of implementing standards such as AES or RSA is that the 'cutting' of the evaluation circuit is not very suited to the white-box context, with few bits of the key diffused at the same time. One idea, that comes with the use of multivariate cryptography we made in part III, is that representing algorithms as a composition of multivariate polynomials systems of low degree would enable more possible techniques. This is however a very difficult problem in general and it is very likely that such 'cutting' would need to exploit the particular structure of the implemented algorithm.

More Structure for New Implementation Techniques

If the implementation of very common standards is out of reach for high security parameters for now, cryptographers can turn themselves to algorithms with more mathematical structure. Usually, if there is more mathematical structure, there are more attacks to be considered. However in our case – as shown by our white-box implementation of HFE –, it allows for techniques that are far from the usual 'internal encoding' rationale of implementation.

One could also try to implement functionalities that are extended from the original algorithm but still allow to compute the algorithm. For instance, one could remark that in our HFE implementation, $A(x, y)$ is not strictly equivalent to F^{-1}. Having access to a structure that is not equivalent to the primitive that is to be white-boxed might give more freedom to designers.

If the algorithm or its underlying structure can be chosen, new designs constraints might arise. Finding 'double trap-door' structures, to help the design of cryptosystems adapted to the plain white-box model, is an interesting problem on its own that could greatly facilitate the deployment of white-box cryptography for asymmetric contexts. One of the most interesting problems would be to implement the decryption algorithm of a fully homomorphic encryption (FHE) scheme, to allow to consider the boostrapping in relation to other functionalities, as it was done for iO in [59].

Bibliography

[1] J. Aarestad, D. Acharyya, R. M. Rad, and J. Plusquellic. Detecting trojans through leakage current analysis using multiple supply pad i_{ddq} s. *IEEE Trans. Inf. Forensics Secur.*, 5(4):893–904, 2010. doi: 10.1109/TIFS.2010.2061228. URL https://doi.org/10.1109/TIFS.2010.2061228.

[2] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar. Trojan detection using IC fingerprinting. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, pages 296–310. IEEE Computer Society, 2007. doi: 10.1109/SP.2007.36. URL https://doi.org/10.1109/SP.2007.36.

[3] S. Agrawal, E. A. Bock, Y. Chen, and G. J. Watson. White-box cryptography with device binding from token-based obfuscation and more. *IACR Cryptol. ePrint Arch.*, page 767, 2021. URL https://eprint.iacr.org/2021/767.

[4] J. Baena, P. Briaud, D. Cabarcas, R. A. Perlner, D. Smith-Tone, and J. A. Verbel. Improving support-minors rank attacks: Applications to gđisplaystyle emss and rainbow. In Y. Dodis and T. Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 376–405. Springer, 2022. doi: 10.1007/978-3-031-15982-4_13. URL https://doi.org/10.1007/978-3-031-15982-4_13.

[5] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In J. Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2001. doi: 10.1007/3-540-44647-8_1. URL https://doi.org/10.1007/3-540-44647-8_1.

[6] G. Barbu, W. Beullens, E. Dottax, C. Giraud, A. Houzelot, C. Li, M. Mahzoun, A. Ranea, and J. Xie. Ecdsa white-box implementations: Attacks and designs from whibox 2021 contest. Cryptology ePrint Archive, Paper 2022/385, 2022. URL https://eprint.iacr.org/2022/385. https://eprint.iacr.org/2022/385.

[7] M. Bardet, J. Faugère, and B. Salvy. On the complexity of the F5 gröbner basis algorithm. *J. Symb. Comput.*, 70:49–70, 2015. doi: 10.1016/j.jsc.2014.09.025. URL https://doi.org/10.1016/j.jsc.2014.09.025.

[8] M. Bardet, J. Faugère, and B. Salvy. On the complexity of the F5 gröbner basis algorithm. *J. Symb. Comput.*, 70:49–70, 2015. doi: 10.1016/j.jsc.2014.09.025. URL https://doi.org/10.1016/j.jsc.2014.09.025.

[9] E. Barkan and E. Biham. In how many ways can you write rijndael? In Y. Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 160–175. Springer, 2002. doi: 10.1007/3-540-36178-2_10. URL https://doi.org/10.1007/3-540-36178-2_10.

[10] M. Bellare, D. Kane, and P. Rogaway. Big-key symmetric encryption: Resisting key exfiltration. In M. Robshaw and J. Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 373–402. Springer, 2016. doi: 10.1007/978-3-662-53018-4_14. URL https://doi.org/10.1007/978-3-662-53018-4_14.

[11] C. Berbain, H. Gilbert, and J. Patarin. QUAD: A practical stream cipher with provable security. In S. Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2006. doi: 10.1007/11761679_8. URL https://doi.org/10.1007/11761679_8.

[12] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri. Error analysis and detection procedures for a hardware implementation of the advanced encryption standard. *IEEE Trans. Computers*, 52(4):492–505, 2003. doi: 10.1109/TC.2003.1190590. URL https://doi.org/10.1109/TC.2003.1190590.

[13] L. Bettale, J. Faugère, and L. Perret. Hybrid approach for solving multivariate systems over finite fields. *J. Math. Cryptol.*, 3(3):177–197, 2009. doi: 10.1515/JMC.2009.009. URL https://doi.org/10.1515/JMC.2009.009.

[14] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In B. S. K. Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997. doi: 10.1007/BFb0052259. URL https://doi.org/10.1007/BFb0052259.

[15] O. Billet, H. Gilbert, and C. Ech-Chatbi. Cryptanalysis of a white box AES implementation. In H. Handschuh and M. A. Hasan, editors, *Selected Areas in Cryptography, 11th International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004, Revised Selected Papers*, volume 3357 of *Lecture Notes in Computer Science*, pages 227–240. Springer, 2004. doi: 10.1007/978-3-540-30564-4_16. URL https://doi.org/10.1007/978-3-540-30564-4_16.

[16] A. Biryukov and A. Udovenko. Attacks and countermeasures for white-box designs. In T. Peyrin and S. D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 373–402. Springer, 2018. doi: 10.1007/978-3-030-03329-3_13. URL https://doi.org/10.1007/978-3-030-03329-3_13.

[17] A. Biryukov, C. D. Cannière, A. Braeken, and B. Preneel. A toolbox for cryptanalysis: Linear and affine equivalence algorithms. In E. Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 33–50. Springer, 2003. doi: 10.1007/3-540-39200-9_3. URL https://doi.org/10.1007/3-540-39200-9_3.

[18] A. Biryukov, C. Bouillaguet, and D. Khovratovich. Cryptographic schemes based on the ASASA structure: Black-box, white-box, and public-key (extended abstract). In P. Sarkar and T. Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 63–84. Springer, 2014. doi: 10.1007/978-3-662-45611-8_4. URL https://doi.org/10.1007/978-3-662-45611-8_4.

[19] E. A. Bock, C. Brzuska, W. Michiels, and A. Treff. On the ineffectiveness of internal encodings - revisiting the DCA attack on white-box cryptography. In B. Preneel and F. Vercauteren, editors, *Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings*, volume 10892 of *Lecture Notes in Computer Science*, pages 103–120. Springer, 2018. doi: 10.1007/978-3-319-93387-0_6. URL https://doi.org/10.1007/978-3-319-93387-0_6.

[20] E. A. Bock, J. W. Bos, C. Brzuska, C. Hubain, W. Michiels, C. Mune, E. S. Gonzalez, P. Teuwen, and A. Treff. White-box cryptography: Don't forget about grey-box attacks. *J. Cryptol.*, 32(4):1095–1143, 2019. doi: 10.1007/s00145-019-09315-1. URL https://doi.org/10.1007/s00145-019-09315-1.

[21] E. A. Bock, A. Amadori, C. Brzuska, and W. Michiels. On the security goals of white-box cryptography. *IACR Cryptol. ePrint Arch.*, page 104, 2020. URL https://eprint.iacr.org/2020/104.

[22] E. A. Bock, C. Brzuska, M. Fischlin, C. Janson, and W. Michiels. Security reductions for white-box key-storage in mobile payments. In S. Moriai and H. Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I*, volume 12491 of *Lecture Notes in Computer Science*, pages 221–252. Springer, 2020. doi: 10.1007/978-3-030-64837-4_8. URL https://doi.org/10.1007/978-3-030-64837-4_8.

[23] A. Bogdanov and T. Isobe. White-box cryptography revisited: Space-hard ciphers. In I. Ray, N. Li, and C. Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 1058–1069. ACM, 2015. doi: 10.1145/2810103.2813699. URL https://doi.org/10.1145/2810103.2813699.

[24] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007. doi: 10.1007/978-3-540-74735-2_31. URL https://doi.org/10.1007/978-3-540-74735-2_31.

[25] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In J. Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual*

International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001. doi: 10.1007/3-540-44647-8_13. URL https://doi.org/10.1007/3-540-44647-8_13.

[26] D. Boneh and M. Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In J. A. Garay and R. Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 480–499. Springer, 2014. doi: 10.1007/978-3-662-44371-2_27. URL https://doi.org/10.1007/978-3-662-44371-2_27.

[27] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In W. Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 1997. doi: 10.1007/3-540-69053-0_4. URL https://doi.org/10.1007/3-540-69053-0_4.

[28] J. W. Bos, C. Hubain, W. Michiels, and P. Teuwen. Differential computation analysis: Hiding your white-box designs is not enough. In B. Gierlichs and A. Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 215–236. Springer, 2016. doi: 10.1007/978-3-662-53140-2_11. URL https://doi.org/10.1007/978-3-662-53140-2_11.

[29] C. Bouillaguet, J. Faugère, P. Fouque, and L. Perret. Practical cryptanalysis of the identification scheme based on the isomorphism of polynomial with one secret problem. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*, volume 6571 of *Lecture Notes in Computer Science*, pages 473–493. Springer, 2011. doi: 10.1007/978-3-642-19379-8_29. URL https://doi.org/10.1007/978-3-642-19379-8_29.

[30] J. Bringer, H. Chabanne, and E. Dottax. Perturbing and protecting a traceable block cipher. In H. Leitold and E. P. Markatos, editors, *Communications and Multimedia Security, 10th IFIP TC-6 TC-11 International Conference, CMS 2006, Heraklion, Crete, Greece, October 19-21, 2006, Proceedings*, volume 4237

of *Lecture Notes in Computer Science*, pages 109–119. Springer, 2006. doi: 10.1007/11909033_10. URL https://doi.org/10.1007/11909033_10.

[31] J. Bringer, H. Chabanne, and E. Dottax. White box cryptography: Another attempt. *IACR Cryptol. ePrint Arch.*, page 468, 2006. URL http://eprint.iacr.org/2006/468.

[32] O. Bronchain, S. Faust, V. Lallemand, G. Leander, L. Perrin, and F. Standaert. MOE: multiplication operated encryption with trojan resilience. *IACR Trans. Symmetric Cryptol.*, 2021(1):78–129, 2021. doi: 10.46586/tosc.v2021.i1.78-129. URL https://doi.org/10.46586/tosc.v2021.i1.78-129.

[33] J. F. Buss, G. S. Frandsen, and J. O. Shallit. The computational complexity of some problems of linear algebra, 1997.

[34] R. Cartor, R. Gipson, D. Smith-Tone, and J. Vates. On the differential security of the hfev- signature primitive. In T. Takagi, editor, *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings*, volume 9606 of *Lecture Notes in Computer Science*, pages 162–181. Springer, 2016. doi: 10.1007/978-3-319-29360-8_11. URL https://doi.org/10.1007/978-3-319-29360-8_11.

[35] A. Casanova, J.-C. Faugère, G. Macario-Rat, J. Patarin, L. Perret, and J. Ryckeghem. GeMSS. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[36] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In M. J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999. doi: 10.1007/3-540-48405-1_26. URL https://doi.org/10.1007/3-540-48405-1_26.

[37] S. Chow, P. A. Eisen, H. Johnson, and P. C. van Oorschot. A white-box DES implementation for DRM applications. In J. Feigenbaum, editor, *Security and Privacy in Digital Rights Management, ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002, Revised Papers*, volume 2696 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2002. doi: 10.1007/978-3-540-44993-5_1. URL https://doi.org/10.1007/978-3-540-44993-5_1.

[38] S. Chow, P. A. Eisen, H. Johnson, and P. C. van Oorschot. White-box cryptography and an AES implementation. In K. Nyberg and H. M. Heys, editors,

Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers, volume 2595 of *Lecture Notes in Computer Science*, pages 250–270. Springer, 2002. doi: 10.1007/3-540-36492-7_17. URL https://doi.org/10.1007/3-540-36492-7_17.

[39] N. T. Courtois. Efficient zero-knowledge authentication based on a linear algebra problem minrank. In C. Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, volume 2248 of *Lecture Notes in Computer Science*, pages 402–421. Springer, 2001. doi: 10.1007/3-540-45682-1_24. URL https://doi.org/10.1007/3-540-45682-1_24.

[40] D. A. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms - an introduction to computational algebraic geometry and commutative algebra (2. ed.)*. Undergraduate texts in mathematics. Springer, 1997. ISBN 978-0-387-94680-1.

[41] J. Daemen and V. Rijmen. Rijndael for AES. In *The Third Advanced Encryption Standard Candidate Conference, April 13-14, 2000, New York, New York, USA*, pages 343–348. National Institute of Standards and Technology,, 2000.

[42] T. Daniels and D. Smith-Tone. Differential properties of the HFE cryptosystem. In M. Mosca, editor, *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*, volume 8772 of *Lecture Notes in Computer Science*, pages 59–75. Springer, 2014. doi: 10.1007/978-3-319-11659-4_4. URL https://doi.org/10.1007/978-3-319-11659-4_4.

[43] C. Delerablée, T. Lepoint, P. Paillier, and M. Rivain. White-box security notions for symmetric encryption schemes. In T. Lange, K. E. Lauter, and P. Lisonek, editors, *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, volume 8282 of *Lecture Notes in Computer Science*, pages 247–264. Springer, 2013. doi: 10.1007/978-3-662-43414-7_13. URL https://doi.org/10.1007/978-3-662-43414-7_13.

[44] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976. doi: 10.1109/TIT.1976.1055638. URL https://doi.org/10.1109/TIT.1976.1055638.

[45] J. Ding and D. Schmidt. Rainbow, a new multivariable polynomial signature scheme. In J. Ioannidis, A. D. Keromytis, and M. Yung, editors, *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*, volume 3531 of *Lecture Notes in Computer Science*, pages 164–175, 2005. doi: 10.1007/11496137_12. URL https://doi.org/10.1007/11496137_12.

[46] J. Ding, D. Schmidt, and F. Werner. Algebraic attack on hfe revisited. In T.-C. Wu, C.-L. Lei, V. Rijmen, and D.-T. Lee, editors, *Information Security*, pages 215–227, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-85886-7.

[47] H. Dobbertin. Almost perfect nonlinear power functions on gf(2n): The welch case. *IEEE Trans. Inf. Theory*, 45:1271–1275, 1999.

[48] H. Dobbertin. Uniformly representable permutation polynomials. In T. Helleseth, P. V. Kumar, and K. Yang, editors, *Sequences and their Applications - Proceedings of SETA 2001, Bergen, Norway, May 13-17, 2001*, Discrete Mathematics and Theoretical Computer Science, pages 1–22. Springer, 2001. doi: 10.1007/978-1-4471-0673-9_1. URL https://doi.org/10.1007/978-1-4471-0673-9_1.

[49] N. Döttling, T. Mie, J. Müller-Quade, and T. Nilges. Basing obfuscation on simple tamper-proof hardware assumptions. *IACR Cryptol. ePrint Arch.*, page 675, 2011. URL http://eprint.iacr.org/2011/675.

[50] V. Dubois, P. Fouque, A. Shamir, and J. Stern. Practical cryptanalysis of SFLASH. In A. Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2007. doi: 10.1007/978-3-540-74143-5_1. URL https://doi.org/10.1007/978-3-540-74143-5_1.

[51] P. Dusart, G. Letourneux, and O. Vivolo. Differential fault analysis on A.E.S. In J. Zhou, M. Yung, and Y. Han, editors, *Applied Cryptography and Network Security, First International Conference, ACNS 2003. Kunming, China, October 16-19, 2003, Proceedings*, volume 2846 of *Lecture Notes in Computer Science*, pages 293–306. Springer, 2003. doi: 10.1007/978-3-540-45203-4_23. URL https://doi.org/10.1007/978-3-540-45203-4_23.

[52] EMV. Integrated circuit card specifications for payment systems. Book 2. Security and Key Management. Version 4.2. June 2008. www.emvco.com., 2008.

[53] J. Faugère and A. Joux. Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using gröbner bases. In D. Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 44–60. Springer, 2003. doi: 10.1007/978-3-540-45146-4_3. URL https://doi.org/10.1007/978-3-540-45146-4_3.

[54] J. Faugère, G. Macario-Rat, J. Patarin, and L. Perret. A new perturbation for multivariate public key schemes such as HFE and UOV. *IACR Cryptol. ePrint Arch.*, page 203, 2022. URL https://eprint.iacr.org/2022/203.

[55] J.-C. Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero (f5). *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC*, pages 75–83, 01 2002. doi: 10.1145/780506.780516.

[56] J.-C. Faugére. A new efficient algorithm for computing gröbner bases (f4). *Journal of Pure and Applied Algebra*, 139(1):61–88, 1999. ISSN 0022-4049. doi: https://doi.org/10.1016/S0022-4049(99)00005-5. URL https://www.sciencedirect.com/science/article/pii/S0022404999000055.

[57] P. Fouque, P. Karpman, P. Kirchner, and B. Minaud. Efficient and provable white-box primitives. In J. H. Cheon and T. Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 159–188, 2016. doi: 10.1007/978-3-662-53887-6_6. URL https://doi.org/10.1007/978-3-662-53887-6_6.

[58] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. ISBN 0-7167-1044-7.

[59] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 40–49. IEEE Computer Society, 2013. doi: 10.1109/FOCS.2013.13. URL https://doi.org/10.1109/FOCS.2013.13.

[60] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In

D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 555–564. ACM, 2013. doi: 10.1145/2488608.2488678. URL https://doi.org/10.1145/2488608.2488678.

[61] L. Goubin and J. Patarin. DES and differential power analysis (the "duplication" method). In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999. doi: 10.1007/3-540-48059-5_15. URL https://doi.org/10.1007/3-540-48059-5_15.

[62] L. Goubin, P. Paillier, M. Rivain, and J. Wang. How to reveal the secrets of an obscure white-box implementation. *J. Cryptogr. Eng.*, 10(1):49–66, 2020. doi: 10.1007/s13389-019-00207-5. URL https://doi.org/10.1007/s13389-019-00207-5.

[63] L. Goubin, M. Rivain, and J. Wang. Defeating state-of-the-art white-box countermeasures with advanced gray-box attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):454–482, 2020. doi: 10.13154/tches.v2020.i3.454-482. URL https://doi.org/10.13154/tches.v2020.i3.454-482.

[64] V. Goyal, Y. Ishai, A. Sahai, R. Venkatesan, and A. Wadia. Founding cryptography on tamper-proof hardware tokens. In D. Micciancio, editor, *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, volume 5978 of *Lecture Notes in Computer Science*, pages 308–326. Springer, 2010. doi: 10.1007/978-3-642-11799-2_19. URL https://doi.org/10.1007/978-3-642-11799-2_19.

[65] X. Guo and R. Karri. Invariance-based concurrent error detection for advanced encryption standard. In P. Groeneveld, D. Sciuto, and S. Hassoun, editors, *The 49th Annual Design Automation Conference 2012, DAC '12, San Francisco, CA, USA, June 3-7, 2012*, pages 573–578. ACM, 2012. doi: 10.1145/2228360.2228463. URL https://doi.org/10.1145/2228360.2228463.

[66] S. Hada. Zero-knowledge and code obfuscation. In T. Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, volume 1976 of *Lecture Notes in Computer Science*, pages 443–457. Springer, 2000. doi: 10.1007/3-540-44448-3_34. URL https://doi.org/10.1007/3-540-44448-3_34.

[67] A. Hosoyamada, T. Isobe, Y. Todo, and K. Yasuda. A modular approach to the incompressibility of block-cipher-based aeads. In S. Agrawal and D. Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part II*, volume 13792 of *Lecture Notes in Computer Science*, pages 585–619. Springer, 2022. doi: 10.1007/978-3-031-22966-4_20. URL https://doi.org/10.1007/978-3-031-22966-4_20.

[68] M. Jacob, D. Boneh, and E. W. Felten. Attacking an obfuscated cipher by injecting faults. In J. Feigenbaum, editor, *Security and Privacy in Digital Rights Management, ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002, Revised Papers*, volume 2696 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2002. doi: 10.1007/978-3-540-44993-5_2. URL https://doi.org/10.1007/978-3-540-44993-5_2.

[69] A. Jain, H. Lin, and A. Sahai. Indistinguishability obfuscation from well-founded assumptions. In S. Khuller and V. V. Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 60–73. ACM, 2021. doi: 10.1145/3406325.3451093. URL https://doi.org/10.1145/3406325.3451093.

[70] R. Karri, G. Kuznetsov, and M. Gössel. Parity-based concurrent error detection of substitution-permutation network block ciphers. In C. D. Walter, Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, volume 2779 of *Lecture Notes in Computer Science*, pages 113–124. Springer, 2003. doi: 10.1007/978-3-540-45238-6_10. URL https://doi.org/10.1007/978-3-540-45238-6_10.

[71] M. Karroumi. Protecting white-box AES with dual ciphers. In K. H. Rhee and D. Nyang, editors, *Information Security and Cryptology - ICISC 2010 - 13th International Conference, Seoul, Korea, December 1-3, 2010, Revised Selected Papers*, volume 6829 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 2010. doi: 10.1007/978-3-642-24209-0_19. URL https://doi.org/10.1007/978-3-642-24209-0_19.

[72] A. Kipnis and A. Shamir. Cryptanalysis of the HFE public key cryptosystem by relinearization. In M. J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 19–30. Springer, 1999. doi: 10.1007/3-540-48405-1_2. URL https://doi.org/10.1007/3-540-48405-1_2.

[73] A. Kipnis, J. Patarin, and L. Goubin. Unbalanced oil and vinegar signature schemes. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 206–222. Springer, 1999. doi: 10.1007/3-540-48910-X_15. URL https://doi.org/10.1007/3-540-48910-X_15.

[74] P. C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In N. Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996. doi: 10.1007/3-540-68697-5_9. URL https://doi.org/10.1007/3-540-68697-5_9.

[75] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999. doi: 10.1007/3-540-48405-1_25. URL https://doi.org/10.1007/3-540-48405-1_25.

[76] Y. Koike and T. Isobe. Yoroi: Updatable whitebox cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):587–617, 2021. doi: 10.46586/tches.v2021.i4.587-617. URL https://doi.org/10.46586/tches.v2021.i4.587-617.

[77] S. Lee, T. Kim, and Y. Kang. A masked white-box cryptographic implementation for protecting against differential computation analysis. *IEEE Trans. Inf. Forensics Secur.*, 13(10):2602–2615, 2018. doi: 10.1109/TIFS.2018.2825939. URL https://doi.org/10.1109/TIFS.2018.2825939.

[78] T. Lepoint and M. Rivain. Another nail in the coffin of white-box AES implementations. *IACR Cryptol. ePrint Arch.*, page 455, 2013. URL http://eprint.iacr.org/2013/455.

[79] G. Macario-Rat, J. Plût, and H. Gilbert. New insight into the isomorphism of polynomial problem IP1S and its use in cryptography. In K. Sako and P. Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 117–133. Springer,

2013. doi: 10.1007/978-3-642-42033-7_7. URL https://doi.org/10.1007/978-3-642-42033-7_7.

[80] H. Maghrebi and D. Alessio. Revisiting higher-order computational attacks against white-box implementations. In S. Furnell, P. Mori, E. R. Weippl, and O. Camp, editors, *Proceedings of the 6th International Conference on Information Systems Security and Privacy, ICISSP 2020, Valletta, Malta, February 25-27, 2020*, pages 265–272. SCITEPRESS, 2020. doi: 10.5220/0008874602650272. URL https://doi.org/10.5220/0008874602650272.

[81] S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007. ISBN 978-0-387-30857-9.

[82] T. Matsumoto and H. Imai. Public quadratic polynominal-tuples for efficient signature-verification and message-encryption. In C. G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings*, volume 330 of *Lecture Notes in Computer Science*, pages 419–453. Springer, 1988. doi: 10.1007/3-540-45961-8_39. URL https://doi.org/10.1007/3-540-45961-8_39.

[83] T. Matsumoto, H. Imai, H. Harashima, and H. Miyakawa. A class of asymmetric cryptosystems using obscure representations of enciphering functions. *1983 National Convention Record on Information Systems, IECE Japan*, 1983.

[84] W. Michiels, P. Gorissen, and H. D. L. Hollmann. Cryptanalysis of a generic class of white-box implementations. In R. M. Avanzi, L. Keliher, and F. Sica, editors, *Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers*, volume 5381 of *Lecture Notes in Computer Science*, pages 414–428. Springer, 2008. doi: 10.1007/978-3-642-04159-4_27. URL https://doi.org/10.1007/978-3-642-04159-4_27.

[85] Y. D. Mulder. *White-Box Cryptography: Analysis of White-Box AES Implementations (White-Box Cryptografie: Analyse van White-Box AES implementaties)*. PhD thesis, Katholieke Universiteit Leuven, Belgium, 2014. URL https://lirias.kuleuven.be/handle/123456789/430072.

[86] Y. D. Mulder, P. Roelse, and B. Preneel. Cryptanalysis of the xiao - lai white-box AES implementation. In L. R. Knudsen and H. Wu, editors, *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *Lecture Notes in Computer Science*, pages 34–49. Springer,

2012. doi: 10.1007/978-3-642-35999-6_3. URL https://doi.org/10.1007/978-3-642-35999-6_3.

[87] M. Øygarden, D. Smith-Tone, and J. A. Verbel. On the effect of projection on rank attacks in multivariate cryptography. In J. H. Cheon and J. Tillich, editors, *Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20-22, 2021, Proceedings*, volume 12841 of *Lecture Notes in Computer Science*, pages 98–113. Springer, 2021. doi: 10.1007/978-3-030-81293-5_6. URL https://doi.org/10.1007/978-3-030-81293-5_6.

[88] J. Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In U. M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 1996. doi: 10.1007/3-540-68339-9_4. URL https://doi.org/10.1007/3-540-68339-9_4.

[89] J. Patarin and L. Goubin. Trapdoor one-way permutations and multivariate polynominals. In Y. Han, T. Okamoto, and S. Qing, editors, *Information and Communication Security, First International Conference, ICICS'97, Beijing, China, November 11-14, 1997, Proceedings*, volume 1334 of *Lecture Notes in Computer Science*, pages 356–368. Springer, 1997. doi: 10.1007/BFb0028491. URL https://doi.org/10.1007/BFb0028491.

[90] J. Patarin, L. Goubin, and N. T. Courtois. Improved algorithms for isomorphisms of polynomials. In K. Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*, pages 184–200. Springer, 1998. doi: 10.1007/BFb0054126. URL https://doi.org/10.1007/BFb0054126.

[91] J. Patarin, G. Macario-Rat, M. Bros, and E. Koussa. Ultra-short multivariate public key signatures. Cryptology ePrint Archive, Report 2020/914, 2020. https://eprint.iacr.org/2020/914.

[92] L. Perret. A fast cryptanalysis of the isomorphism of polynomials with one secret problem. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 354–370. Springer, 2005. doi: 10.1007/11426639_21. URL https://doi.org/10.1007/11426639_21.

[93] A. Petzoldt. On the complexity of the hybrid approach on hfev-. *IACR Cryptol. ePrint Arch.*, page 1135, 2017. URL http://eprint.iacr.org/2017/1135.

[94] C. Qian, M. Tibouchi, and R. Géraud. Universal witness signatures. In A. Inomata and K. Yasuda, editors, *Advances in Information and Computer Security - 13th International Workshop on Security, IWSEC 2018, Sendai, Japan, September 3-5, 2018, Proceedings*, volume 11049 of *Lecture Notes in Computer Science*, pages 313–329. Springer, 2018. doi: 10.1007/978-3-319-97916-8_20. URL https://doi.org/10.1007/978-3-319-97916-8_20.

[95] A. Ranea, J. Vandersmissen, and B. Preneel. Implicit white-box implementations: White-boxing ARX ciphers. In Y. Dodis and T. Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 33–63. Springer, 2022. doi: 10.1007/978-3-031-15802-5_2. URL https://doi.org/10.1007/978-3-031-15802-5_2.

[96] S. Rasoamiaramanana. *Design of white-box encryption schemes for mobile applications security. (Conception de schémas de chiffrement boîte blanche pour la sécurité des applications mobiles)*. PhD thesis, University of Lorraine, Nancy, France, 2020. URL https://tel.archives-ouvertes.fr/tel-02949394.

[97] M. Rivain and J. Wang. Analysis and improvement of differential computation attacks against internally-encoded white-box implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):225–255, 2019. doi: 10.13154/tches.v2019.i2.225-255. URL https://doi.org/10.13154/tches.v2019.i2.225-255.

[98] A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In D. B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484. ACM, 2014. doi: 10.1145/2591796.2591825. URL https://doi.org/10.1145/2591796.2591825.

[99] E. Sanfelix, C. Mune, and J. de Haas. Unboxing the white-box: Practical attacks against obfuscated ciphers. *Blackhat*, 2015.

[100] A. Saxena, B. Wyseur, and B. Preneel. Towards security notions for white-box cryptography. In P. Samarati, M. Yung, F. Martinelli, and C. A. Ardagna, editors, *Information Security, 12th International Conference, ISC 2009, Pisa, Italy, September 7-9, 2009. Proceedings*, volume 5735 of *Lecture Notes in Computer Science*, pages 49–58. Springer, 2009. doi: 10.1007/978-3-642-04474-8_4. URL https://doi.org/10.1007/978-3-642-04474-8_4.

[101] O. Seker, T. Eisenbarth, and M. Liskiewicz. A white-box masking scheme resisting computational and algebraic attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(2):61–105, 2021. doi: 10.46586/tches.v2021.i2.61-105. URL https://doi.org/10.46586/tches.v2021.i2.61-105.

[102] A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984. doi: 10.1007/3-540-39568-7_5. URL https://doi.org/10.1007/3-540-39568-7_5.

[103] C. Tao, A. Petzoldt, and J. Ding. Improved key recovery of the hfev- signature scheme. *IACR Cryptol. ePrint Arch.*, page 1424, 2020. URL https://eprint.iacr.org/2020/1424.

[104] Y. Todo and T. Isobe. Hybrid code lifting on space-hard block ciphers application to yoroi and spnbox. *IACR Trans. Symmetric Cryptol.*, 2022(3):368–402, 2022. doi: 10.46586/tosc.v2022.i3.368-402. URL https://doi.org/10.46586/tosc.v2022.i3.368-402.

[105] J. Wang. *On the practical security of white-box cryptography. (De la théorie à la pratique de la cryptographie en boite blanche)*. PhD thesis, University of Luxembourg, Luxembourg City, Luxembourg, 2020. URL https://tel.archives-ouvertes.fr/tel-02953586.

[106] WhibOx Organizing Committee. Ches 2017 ctf challenge – whibox contest. https://whibox.io/contests/2017/, 2017.

[107] WhibOx Organizing Committee. Ches 2019 ctf challenge – whibox contest. https://whibox.io/contests/2019/, 2019.

[108] WhibOx Organizing Committee. Ches 2021 ctf challenge – whibox contest. https://whibox.io/contests/2021/, 2021.

[109] C. Wolf and B. Preneel. Equivalent keys in hfe, c^*, and variations. In E. Dawson and S. Vaudenay, editors, *Progress in Cryptology - Mycrypt 2005, First International Conference on Cryptology in Malaysia, Kuala Lumpur, Malaysia, September 28-30, 2005, Proceedings*, volume 3715 of *Lecture Notes in Computer Science*, pages 33–49. Springer, 2005. doi: 10.1007/11554868_4. URL https://doi.org/10.1007/11554868_4.

[110] Y. Xiao and X. Lai. A secure implementation of white-box aes. In *2009 2nd International Conference on Computer Science and its Applications*, pages 1–6, 2009. doi: 10.1109/CSA.2009.5404239.

[111] M. Zeyad, H. Maghrebi, D. Alessio, and B. Batteux. Another look on bucketing attack to defeat white-box implementations. In I. Polian and M. Stöttinger, editors, *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*, volume 11421 of *Lecture Notes in Computer Science*, pages 99–117. Springer, 2019. doi: 10.1007/978-3-030-16350-1_7. URL https://doi.org/10.1007/978-3-030-16350-1_7.