Resisting Key-Extraction and Code-Compression: a Secure Implementation of the HFE Signature Scheme in the White-Box Model

Pierre Galissant and Louis Goubin

Université Paris-Saclay, UVSQ, CNRS, Laboratoire de mathématiques de Versailles, 78000, Versailles, France
{pierre.galissant,louis.goubin}@uvsq.fr

Abstract. Cryptography is increasingly deployed in applications running on open devices in which the software is extremely vulnerable to attacks, since the attacker has complete control over the execution platform and the software implementation itself. This creates a challenge for cryptography: design implementations of cryptographic algorithms that are secure, not only in the black-box model, but also in this attack context that is referred to as the white-box adversary model. Moreover, emerging applications such as mobile payment, mobile contract signing or blockchain-based technologies have created a need for white-box implementations of public-key cryptography, and especially of signature algorithms.
However, while many attempts were made to construct white-box implementations of block-ciphers, almost no white-box implementations have been published for what concerns asymmetric schemes. We present here a concrete white-box implementation of the well-known HFE signature algorithm for a specific set of internal polynomials. For a security level 2^{80}, the public key size is approximately 62.5 MB and the white-box implementation of the signature algorithm has a size approximately 256 GB.

Keywords: White-box Cryptography · Public-Key Cryptography · Multivariate Cryptography

1 Introduction

The security of cryptographic primitives is usually studied when an adversary is only given black-box access to it. This means that the secrets involved in the primitive can only be compromised through the input-output behavior of the primitive. The security notions derived in that model assume that the algorithms are known and that the devices used to compute the primitives can be trusted to protect secret information.

In practice, however, the black-box attack model fails to capture the inherent leakage of information made during any computations on a hardware. These side-channel data such as power consumption, execution time or electromagnetic emanations, can indeed be used to recover secret information stored on a hardware device. This led cryptographers to study the security of programs computing primitives on an untrusted hardware platform, resulting in a new model of attack: the grey-box model.

A natural extension of the grey-box model is to consider the worst case adversarial model, where an attacker has full access to the software and the hardware used to compute a primitive: the white-box model. In this model, the attacker has full access to the code computing the algorithm and can know the state of the memory and alter it at any time during the execution of the algorithm. It is assumed that the exact functioning of the

hardware is known to the attacker and no part of the memory is protected from this attacker. In this model, the knowledge of the software and the initial state of the memory entirely determinate the successive states of the memory. To protect an existing primitive, a cryptographer has to design a 'compiler' that transforms the usual implementation of the primitive into an implementation that resist white-box attackers.

Since the seminal paper of Chow *et al* in 2002, the research has mainly been focused on standard symmetric block-ciphers such as DES and AES. Many candidate implementations have been proposed [CEJvO02, CEJv03, LN04, BCD06, XL09, Kar11] but all have later been broken due to very powerful generic attacks or clever and specific attacks.

Regarding asymmetric candidates, very few solutions have been proposed: [Bar20b] does not stand against generic attacks and [FHW+20] and [ZHH+20] focus only on key-extraction, ignoring the incompressibility, while modifying the verifying algorithm in the white-box version of the scheme. After almost two decades of research, it is safe to say that getting a secure white-box implementation of an existing primitive is a hard open problem. In this paper, we show that the richness of multivariate cryptography allows new possibilities for asymmetric white-box cryptography, and we design and analyze an explicit construction based on these new ideas.

Our Contribution In the present paper, we exhibit the first white-box implementation of an asymmetric signature algorithm, in the following sense:

- The instances of the HFE scheme we use have security level 2^{80} against all known (black-box) attacks in the *chosen-message* model;
- The *unbreakability* property of our implementation of the signature algorithm in the white-box model is proven under a reasonable assumption about the *Isomorphism of Polynomials* (IP) problem: a key-recovery attack has complexity at least 2^{80} (and thus is not easier than in the black-box model);
- We also revisit the notion of *incompressibility* in the white-box model, giving a more precise definition, and state a precise conjecture in the case of our implementation of the signature algorithm: under this assumption, an attack aiming at obtaining a smaller implementation has complexity at least 2^{80} (and thus is not easier than in the black-box model).

The parameters of our proposal are the following:

- The secret key size is approximately 2^{20} bits $\simeq$ 125 kB (alternatively, the secret key can be obtained from a pseudo-random generator and a 80-bit secret seed);
- The public key size is approximately $2^{30}/2$ bits $\simeq$ 62.5 MB;
- The white-box implementation of the signature algorithm is approximately 2^{41} bits $\simeq$ 256 GB.

Remark 1: The size of our white-box implementation is huge and obviously makes it difficult to be used in devices such as smart phones for instance. However, it should be noted that our construction provides the first white-box implementation of a public key algorithm together with a extensive security analysis. Moreover realistic applications can be envisaged in the case of cloud-based applications. As an example, MasterCard Cloud-Based Payments (MCBP) [Masa] is a secure and scalable software-based solution developed by to digitize card credentials and enable both contactless and remote payment transactions. In this context, MasterCard has specifically recommended the use of white-box implementation for the secure storage of payment tokens [Masb].

Remark 2: Our proposal can also easily be given in the form of a public-key encryption algorithm, with a white-box implementation of the decryption algorithm and the same security levels. In this paper we prefer to describe the signature scheme, since it matches the real-world applications we describe below (see Section 2.1).

Technical Overview Technically, our proposal is heavily based on the use of multivariate cryptography. The main idea is as follows. We consider the HFE (Hidden Field Equation) cryptosystem; its design involves an internal polynomial transformation F defined on the field extension $\mathbb{K} = \mathbb{F}_{2^n}$, such that all the monomials have an exponent of Hamming weight ≤ 2. For our purpose, we intentionally choose F such that it has an *affine multiple* $A \in \mathbb{K}[X, Y]$ of low degree in Y. This means that:

- All the exponents of X involved in $A(X, Y)$ have a Hamming weight ≤ 1;
- Any solution of the equation $F(a) = b$ is also a solution of $A(a, b) = 0$

In a nutshell, the white-box implementation of the signature algorithm is derived from this affine multiple $A(X, Y)$ and the secret key, and its size can be kept moderate since the degree of A in Y is low. Moreover, for an attacker who only knows the public key, recovering the white-box implementation is not easier than breaking the scheme in the black-box model. Lastly, from the white-box implementation, it is computationally difficult for the legitimate user to recover the secret key, or even to compress the white-box implementation.

Related Work In the present paper, we consider and target the traditional (white-box) notions of *unbreakability* and *incompressibility*. In this context, we describe a white-box implementation of an HFE-based primitive, which belongs to a family of algorithms that has been extensively studied in the black-box model and belongs to state-of-the-art cryptography.

The notion of unbreakability is a very intuitive security notion for white-box cryptography and has been studied since the seminal paper of Chow *et al* in 2002 [CEJvO02]. Ever since, cryptographers have tried to propose white-box implementations of usual cryptographic algorithms, with mitigated success. Block-ciphers have been the most studied and we refer to section 2.3 for some examples.

The notion of incompressibility is a stronger security notion, first formally defined in [DLPR14] where the study of this notion is motivated as a software countermeasure against code-lifting attacks. In the same paper, the authors also propose an incompressible implementation of a special private-key RSA in the Ideal Group Model. Since then, works like [BBK14] [BP15] or [FKKM16] have considered the incompressibility of specially designed symmetric algorithms.

Organization of the paper
Section 2 provides all the necessary background on the motivations, the security notions for white-box cryptography and multivariate cryptography, in particular for what concerns the HFE family.
Section 3 describes the complete and detailed construction of our scheme, including the choice of the parameters with respect to all the known attacks in the state of the art of multivariate cryptography. We also assess the black-box security of our instance.
Section 4 analyses the security of our construction in the white-box model.

2 Background

2.1 Motivations for *public-key* white-box constructions

From a technical point of view, the current lack of white-box implementations for asymmetric cryptosystems comes from the difficulty of the problem, especially if we want to achieve a provable construction, as is usually the case in the public-key model. Despite this state of affairs, it appears that more and more real-life applications would clearly benefit from such constructions and provide a strong motivation for white-box PK implementations. We advocate for the interest of *public-key* cryptographic algorithms that are resistant in the white-box model through three real-life applications illustrating their usefulness.

Mobile payment. In the recent years, the payment industry has vested great interests in the extension of the EMV specifications [EMV08] 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 however, 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 plans to publish 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 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 become 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 in the cloud 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.

Cryptocurrencies and blockchain technologies. Most solutions to store cryptocurrencies and perform transactions on the blockchain are today 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 mobile OS versions on the field, strictly relying on the operating system to protect critical assets (such as money) is very hazardous and should 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.

2.2 Security notions for white-box cryptography

Security Notions. The basic security requirement for a white-box implementation is to resist key extraction. But one should expect more from white-box cryptography and consider various security properties for the white-box implementations. Some attempts have been made to provide formal definitions and security notions for white-box cryptography. In particular some concrete white-box security notions for symmetric encryption schemes were proposed in [DLPR14]: the proposed notions –unbreakability, one-wayness, incompressibility, traceability– are derived from folklore intuitions behind white-box cryptography. Note that one-wayness is not relevant in the asymmetric setting, the decryption being the only primitive that needs to be protected. We precisely define unbreakability and incompressibility in section 4.

Primitives designed in the white-box model. Another line of works has focused on the issue of designing new cryptographic primitives with security properties inspired by the white-box context. A traceable block cipher relying on multivariate cryptography has been proposed in [BG03] but it was soon shown that the traceability could actually be bypassed [FP06]. Further "white-box" ciphers have been designed from multivariate cryptography [BBK14] and have also later been broken [MDFK15]. One should note that unbreakability alone is often trivial to achieve for a designer: one could get a twisted white-box AES by just hashing the master key, and pretending the new design is composed of the hash and a regular AES implementation. Recently, incompressible encryption primitives have been particularly investigated [BI15, BIT16, FKKM16, BKR16]. Although these works put forward interesting approaches to design new cryptographic primitives (with specific security properties), they do not address the prime goal of white-box cryptography which is to provide secure implementations for a given (standard) cryptographic algorithm such as AES.

Links between iO and white-box cryptography. Recently, interesting links have being established between iO and white-box cryptography using hardware modules ([ABF+20]). However this requires, by construction, the presence of such a hardware component. Moreover, the resulting constructions are really far from being implementable. Even estimating the size of an iO construction is kwown to be a challenge, and there is no generic proof that iO would satistfy properties such as incompressibility. That is why these hardware-and-iO based constructions cannot reasonably be used today to build purely software white-box solutions. It is important to note that the security model used in [ABF+20] is *weaker* than the full white-box model we use. In the present paper, the behavior of the whole algorithm is fully determined by the code and the initial state of the memory (this is not true any longer when one allows the use of a hardware that depends on secret data).

2.3 White-Box for Private-Key Cryptography

Many attempts have been made to construct white-box implementations for standard block-ciphers, but all published constructions are currently broken. The first white-box implementations proposed in 2002 by Chow *et al.* for the DES and AES ciphers [CEJvO02, CEJv03] were broken a couple of years after their publication [JBF02, BGEC04, GMQ07, WMGP07]. Further construction attempts followed [LN04, BCD06, XL09, Kar11] but they were also shown insecure sooner or later [DWP10, DRP13a, LRD+14, DRP13b, LR13].

Note that [DLPR14] explains how an RSA-like algorithm could have a white-box implementation that is incompressible. The fundamental idea is the following: if the encryption function is built upon $x \mapsto y = x^e \bmod n$, the decryption function is based on $y \mapsto x = y^d \bmod n$ (where d is the inverse of e modulo $\varphi(n)$) and can be implemented as $y \mapsto x = y^{d'} \bmod n$, where d' is an arbitrary large integer such that $d' \equiv d \bmod \varphi(n)$. However, for this construction to make sense, we have to assume that e is also a secret exponent (if not, a well-known argument can be used to deduce the factorization of n from d'), so that we in fact have a symmetric algorithm, and not a public-key scheme.

2.4 White-Box for Public-Key Cryptography

While many candidates have been publicly proposed to construct white-box implementations of block-ciphers, almost no white-box implementations for public-key algorithms have been published up to now, in spite of numerous research efforts.

The work of [FHW+20] and [ZHH+20] achieves unbreakability but modifies the verification algorithm and doesn't address incompressibility. As discussed in section 2.2, designing primitives for white-box cryptography for unbreakability is easier.

To the best of our knowledge, the only candidate that doesn't change the underlying scheme is due to Lucas Barthélémy [Bar20b], who recently proposed a white-box implementation of a scheme suggested by Aguilar-Melchior, Barrier, Guelton, Guinet, Kollijian and Lepoint [ABG+16], whose (black-box) security is based on the computational difficulty of the RLWE (Ring Learning with Errors) problem over the cyclotomic ring $R_q = \mathbb{Z}/q\mathbb{Z}[X]/(X^n + 1)$. Lucas Barthélémy's implementation of the decryption algorithm makes use of the NTT transformation and RNS representations to reduce the computation to small look-up tables, which can in turn be transformed using ideas dating back to [CEJv03], together with additive of multiplicative masking based on homomorphic properties of the cryptographic scheme. However, a fatal flaw was found (and acknowledged by Lucas Barthélémy in [Bar20a]): the core part of the white-box implementation consists in trying to prevent the key extraction for a function of the form $\alpha_2 - \alpha_1 \cdot sk$, where (α_1, α_2) is the ciphertext we want to decrypt, and sk is the secret key. This function is linear in α_1 and α_2, and this linear dependence on the elements coming from the ciphertext remains true, even after applying the NTT transformations and using the representations RNS. This can be clearly observed on the equations of section 4.2 in [Bar20b]. It is therefore very easy for an attacker to find the coefficients of these linear transformations (which depend on sk), then sk itself.

One could explain the lack of solution to the asymmetric problem by looking at the methods used in the symmetric case: small look-up tables or masking for instance. These methods rely on circuit representation or on the possibility of breaking up the implementation into smaller pieces. Algorithms like RSA or ECDSA appear to be not suited to this strategy if we want to get the desired levels of security. This state of affairs illustrates how challenging the task is to find a public-key scheme allowing a white-box implementation. The goal of this paper is to provide such a construction for a signature algorithm, for which all the parameters are carefully chosen to provide security (both in the black-box and the white-box models) against the best known attacks. Our proposal is

heavily based on the use of multivariate cryptography, which allows us to introduce new ideas – and new problems – in white-box cryptography.

2.5 Multivariate Cryptography

Public-Key Multivariate Cryptography is a part of public-key cryptography in which the public key is given as a set of m polynomials $(P_1, \ldots, P_m)$ in n variables, of small degree d over a small finite field $\mathbb{F}$.

Most of the time, d equals 2 and $20 \leq n \leq 300$.

The user must solve (in encryption) or find at least one solution (in signature or authentication) in $(x_1, \ldots, x_n)$ the system

$$(\mathcal{A}) \begin{cases} P_1(x_1, \ldots, x_n) = y_1 \\ \cdots \\ P_m(x_1, \ldots, x_n) = y_m \end{cases}$$

for a given m-tuple $(y_1, \ldots, y_m)$.

For a signature scheme, $(y_1, \ldots, y_m)$ can generally be viewed as the Hash of the message to be signed, and $(x_1, \ldots, x_n)$ as the signature of this message.

For a challenge-response authentication scheme, $(y_1, \ldots, y_m)$ can be viewed as the challenge, and $(x_1, \ldots, x_n)$ as the response.

For an encryption scheme, $(y_1, \ldots, y_m)$ can be viewed as the ciphertext, and $(x_1, \ldots, x_n)$ as the corresponding cleartext.

Any user can easily check whether given $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_m)$ satisfy or not the system $(\mathcal{A})$ and can easily compute $(y_1, \ldots, y_m)$ from any given $(x_1, \ldots, x_n)$.

However, without the knowledge of a secret key, it should be infeasible to decrypt a message or to forge a signature or to successfully pass an authentication. In other terms, finding at least one solution $(x_1, \ldots, x_n)$ of the system $\mathcal{A}$ should be difficult for most $(y_1, \ldots, y_m)$ without the knowledge of a secret key, and become easy with the knowledge of a secret key.

Most of the time, a multivariate scheme is built from an easy-to-solve system $\mathcal{B}$, which is then "hidden" by two secret random linear (or affine) transformations s and t to obtain a new system $(\mathcal{A})$, by composing them (on the left and on the right) with $\mathcal{B}$.

More precisely, if $(\mathcal{B})$ is given by m polynomials $(Q_1, \ldots, Q_m)$ in $(a_1, \ldots, a_n)$, the new system $(\mathcal{A})$ is given by m polynomials $(P_1, \ldots, P_m)$ in $(x_1, \ldots, x_n)$ given by:

$$(P_1, \ldots, P_m)(x_1, \ldots, x_n) = t(Q_1(s(x_1, \ldots, x_n)), \ldots, Q_m(s(x_1, \ldots, x_n)))$$

where s and t are secret random linear (or affine) bijective changes of variables.

The obtained system $(\mathcal{A})$ is also quadratic and finding a solution of

$$(\mathcal{A}) \begin{cases} P_1(x_1, \ldots, x_n) = y_1 \\ \cdots \\ P_m(x_1, \ldots, x_n) = y_m \end{cases}$$

for a given m-uple $(y_1, \ldots, y_m)$ is expected to be difficult without the knowledge of s and t. Finding the secrets s and t given $(\mathcal{A})$ and $(\mathcal{B})$ has been abstracted as the *Isomorphism of Polynomial* (IP) problem in [Pat96]. It started a line of work such as [PGC98, Per05, BFFP11, MPG13] to better understand the security of general multivariate schemes.

There exist several families of multivariate schemes, corresponding to several choices for the system $\mathcal{B}$, like C^*, HFE, Rainbow or UOV. Among them, some schemes proposed in the past have been broken. For others, no efficient attacks are known. In general,

solving a set of quadratic equations over a finite field (this is called the MQ problem) is an NP-hard problem for any finite field. However, it is difficult to obtain a proof of security for multivariate schemes: since the system $(\mathcal{B})$ has to be easy to solve, it is never random, and neither is $(\mathcal{A})$ (since $(\mathcal{A})$ is obtained by linear changes of variables from $(\mathcal{B})$).

2.6 HFE

We now describe HFE in more details and review the most powerful attacks known against it: the message recovery attack and the Kipnis-Shamir attacks especially are the attacks that are considered in the black-box model to test whether an HFE instance can be used to build secure primitives. We also describe the affine multiple attack as it is the starting point of our construction.

2.6.1 Description of HFE and its variants

First described by Patarin in [Pat96], the HFE scheme is a direct descendant of C^* [MIHM83, MI88]. This time, instead of an internal monomial, the internal polynomial can be any polynomial that is of degree 2 over $\mathbb{F}_q$. We only describe it over $\mathbb{F}_2$, as it is the setting of this paper. Formally, for any prime q and any positive integers n and $D \in \mathbb{N}$ let $F \in \mathbb{F}_{2^n}[X]$ be defined by:

$$F(X) = \sum_{\substack{0 \leq i < j < n \\ 2^i + 2^j \leq D}} a_{i,j} X^{2^i + 2^j} + \sum_{\substack{0 \leq i < n \\ q^i \leq D}} b_i X^{2^i} + c$$

where the $a_{i,j}$, the b_i and c are elements of F_{2^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 $\lceil log_2(D) \rceil$ will be important in the description of attacks, we set $d = \lceil log_2(D) \rceil$

The secret key is the list of such polynomial F and a couple of affine transformations $(S, T) \in Aff_n(\mathbb{F}_2)$. 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 2^n is not too big.

To compute the public key, 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 $\pi(x_1, ..., x_n) = \sum_{i=1}^{n} x_i e_i$. The public key P is the map from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$ is then 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 $P_i \in \mathbb{F}_2[x_1, ..., x_n]$.

2.6.2 Affine Multiple Attacks

One of the first attacks considered in the seminal paper [Pat96] of Patarin is a generalization of the attack he made on C^*, called the affine multiple attack. 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 $A(x, y) = 0$. The goal of this attack is to recover A.

Definition 1. Let $F \in \mathbb{F}_{2^n}[X]$. The polynomial $A(x, y) \in \mathbb{F}_{2^n}[X, Y]$ is said to be a affine multiple of F if $A(x, y) = 0 \ mod \ F(x) + y$ and A is $\mathbb{F}_2$-linear in x.

Let us define d_{aff} to be the maximum Hamming weight of the monomials in y in the polynomial $A(x, y)$. We then remark that the composition by that affine transformation S and T leads to a new affine multiple that has the same d_{aff} value. Now, to start the attack, just notice that $A(x, y)$ is composed of about $n \times \sigma(n, d_{aff})$ unknown coefficients over $\mathbb{F}_2$, 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 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, 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.

2.6.3 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 [Fau02].

To solve a polynomial system of n equations with n unknowns with F5, the complexity is proved to be [BFS15]:

$$\mathcal{O}\left(\binom{n + d_{reg}}{d_{reg}}^\omega\right)$$

where $2 \leq \omega \leq 3$ is the linear algebra constant, and d_{reg} is the degree of regularity of the polynomial system. Roughly, 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 $\mathbb{F}_{2^n}$, it has been shown in [FJ03] 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 ([BFS15]).

2.6.4 Kipnis-Shamir Key Recovery attack

To perform a key-recovery on HFE and its variants, the most efficient attacks to date are the Kipnis-Shamir attacks. In their paper [KS99], 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 [BFS97].

Since then, variations of the attack have been thoroughly studied. Especially, variations with minor modeling and support minor have been the most successful. A recent paper of Ding *et al* [Din20] provides a key-recovery attack that completely, breaking any small HFE instances such as RedGeMSS128 [CFM+20]. In their paper, the authors show that a key-recovery for HFE can be made with minor modeling in:

$$\mathcal{O}\left(\binom{n + d + 1}{d + 1}^\omega\right)$$

using Gröbner bases algorithms like F4, and experimentally with support minor modeling and Gröbner bases in:

$$\mathcal{O}\left(\left(n^2\binom{2d + 2}{d} + n\binom{2d + 2}{d}^2\right)^\omega\right).$$

This means that key-recovery is exponential in $d = \lceil log_2(D) \rceil$, but polynomial in n.

2.6.5 Differential Attacks

The differential attacks were introduced in [DFSS06] 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 [CGSV16, DS14] showed that HFE polynomials usually used do not have any exploitable differential structure.

3 A Public-Key Scheme with WB Implementation

In this section, we go through the rationale of our construction and detail the instantiation of our white-box compiler for the desired level of security.

3.1 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 an HFE instance 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 computing the affine multiple.

The most important point at this stage is that it is way simpler to find an affine multiple with the knowledge of the secret key (S, F, T) than it is with the knowledge of the public key. Indeed, we have a gap ω between the size of the affine multiple relation over $\mathbb{F}_2$ and the computation time that is needed to recover it knowing the public key performing an affine multiple attack. As computing the affine multiple is not always possible for each instances, we then have to tweak the parameters to get an HFE instance that achieves black-box security, while its affine multiple is not too big.

To get a concrete white-box HFE signature instantiation, we fix $\lambda = 80$ to be our expected security level against all the usual HFE attacks. We then fix parameters F and n to get the smallest affine multiple possible and argue is section 4 that the construction is secure in the white-box model against known attacks. To do so, we propose a short study of the degree of affine multiple depending on F, and choose a polynomial that produces a white-box implementation of 256GB while having a level of security up to 2^{80} against black-box attacks and attacks against the public-key.

We do not claim the asymptotic scalability of our scheme, as it is hard to claim it for HFE in general regarding the efficient key-recovery of [Din20]. However, even if we focus on $\lambda = 80$, it is possible to change some parameters to get a higher level of security, but with a bigger implementation.

Regarding the white-box security, we rely on the fact that to the best of our knowledge, giving access to the multiple affine relation does not give more information than a black-box access to F^{-1}. This claim is defended in the white-box security part.

3.2 Construction

3.2.1 Notations

In the rest of the paper, following [DLPR14], we consider a *program* in the language-theoretic sense, interpreted in the explicit context of a programming model and an execution

model, the details of which will be kept as abstracted away as possible. Computation times are also to be considered in this execution model. Programs differ from remote oracles in the sense that their code can be executed locally, read, copied and modified at will. It is consistent with the white-box paradigm: execution can be interrupted at any moment and all the internal variables identified by the instructions of the program can be read and modified arbitrarily by the party that executes the program.

For any program or mathematical object P, we define $Size(P)$ to be the size in bits of its representation.

(M,C,$K_{\mathcal{V}} \times K_{\mathcal{S}}$,$\mathcal{V}$,$\mathcal{S}$) is an asymmetric signature scheme where:

- M is the message space
- S is the signature space
- $K_{\mathcal{V}} \times K_{\mathcal{S}}$ is the key space
- $\mathcal{V} : K_{\mathcal{V}} \times M \to S$ is the signature algorithm
- $\mathcal{S} : K_{\mathcal{S}} \times S \to M$ is the verification algorithm
- For all $m \in M$ and $(k_1, k_2) \in K_{\mathcal{V}} \times K_{\mathcal{S}}$, $\mathcal{S}(k_2, \mathcal{V}(k_1, m)) = m$

For any keyed signature algorithm $\mathcal{S}$ with key k we note $C_{\mathcal{S}}$ the compiler specified to $\mathcal{S}$ and $C_{\mathcal{S}}(k)$ the white box implementation of $\mathcal{S}$ with the private-key k.

Regarding polynomials, we note $\sigma(n, d)$ to be the number of monomials in at most n variables of at most degree d and $M(n, d)$ is the cost of multiplying polynomials of degree d with coefficients of size n. The rest of the notations are usual.

For HFE, we will note the public key $P = T \circ \pi^{-1} \circ F \circ \pi \circ S$ with respect to all the notations from the introduction part: S and T the affine secrets and F the internal HFE map.

3.2.2 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 GCD 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 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.

3.2.3 The white-box construction

We now detail the construction of the signature scheme, our white-box compiler and how to use it in practice.

To sign with HFE, we will use the simplest scheme possible, already explained in the seminal paper on HFE [Pat96]: hash and sign. If we set $h : \{0,1\}^* \to \{0,1\}^n$ to be a cryptographic hash function, one can simply sign a message m by computing $s = P^{-1}(h(m))$. Then, the verification is made by computing $P(s) = h(m)$. We denote by $\mathcal{S}_{HFE}$ this signature scheme.

Algorithm 1: HFE Signature algorithm $\mathcal{S}_{HFE}$

input : A message $M \in \{0,1\}^*$
output : The signature $\sigma = (\mathcal{S}_{HFE}(M), r)$

$r \leftarrow 0$;
$y \leftarrow \phi^{-1}(T^{-1}(h(M||r)))$;
while $y \notin Im(F)$ **do**
  $r \leftarrow r + 1$;
  $y \leftarrow \phi^{-1}(T^{-1}(h(M||r)))$;
return $s = (S^{-1}(\phi(y)), r)$;

Remark: Usually, a Feistel structure is used with P^{-1} as round function, in order to reduce the probability of collisions. The simple scheme (Algorithm 1) has obvious attacks with birthday paradox attacks in $2^{n/2}$, but we will see later that this is not a problem for us due to the chosen value n.

Note that the scheme can be easily turned into a provably secure scheme against chosen-message attacks, as shown in [SSH11], by changing the way the salt is generated. Moreover, as we will see, the security of the scheme in the white-box model remains equivalent to the unbreakability and incompressibility properties of the implementation of P^{-1}. That is why we focus on a white-box implementation of P^{-1}.

Now, to start the white-box transformation from an HFE secret-key (S, F, T) over n bits - with affine transformation S and T 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 maps S and 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, the code for inversion and the hash function h to be the white-box implementation of the computation of P^{-1}. We note this compiler *WBHFE*. This leads to the following compiling algorithm:

Algorithm 2: White-box compiler *WBHFE*

input : A HFE secret key (S, F, T) with affine S and T

output : $WBHFE_{\mathcal{S}_{HFE}}(S, F, T)$ the white-box implementation of $\mathcal{S}_{HFE}$ 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 hashes a message of size n.
- Produce a code that partially evaluate the $\tilde{A}_i$ over the $y_1, ..., y_n$
- Produce a code of the function *INV* that compute a preimage of the vector 0 through linear application given by the partial evaluations of the $\tilde{A}_i$, and output "NONE" if not solution can be found.
- Produce a code that check if the message to be signed is in the image of P and increments the salts otherwise.
- Concatenate the produced codes to get the whole white-box implementation $WBHFE_{\mathcal{S}_{HFE}}(S, F, T)$.

The produced implementation $WBHFE_{\mathcal{S}_{HFE}}(S, F, T)$ can be described by Algorithm 3.

Algorithm 3: WBHFE Signature algorithm $WBHFE_{\mathcal{S}_{HFE}}(S, F, T)$

input : A message $M \in \{0,1\}^*$
output: The signature $\sigma = (\mathcal{S}_{HFE}(M), r)$

$r \leftarrow 0$;
$y \leftarrow h(M||r)$;
Substitute the values $y = (y_1, ..., y_n)$ in $\tilde{A}_i(x_1, ..., x_n, y_1, ..., y_n)$;
$x \leftarrow INV(\tilde{A}_i(x_1, ..., x_n, y_1, ..., y_n))$;
while $x =$ *"NONE"* ***OR*** $x \notin Im(P)$ **do**
  $r \leftarrow r + 1$;
  $y \leftarrow h(M||r)$;
  Substitute the values $y = (y_1, ..., y_n)$ in $\tilde{A}_i(x_1, ..., x_n, y_1, ..., y_n)$;
  $x \leftarrow INV(\tilde{A}_i(x_1, ..., x_n, y_1, ..., y_n))$;
return $s = (x, r)$;

Remark : Note that the verifying algorithm for signature in algorithm 1 is the same as algorithm 3. A description of it can be found in section 2.5 or in [Pat96] and [SSH11].

3.3 Size analysis 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.

3.3.1 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 $0 \leq i < D$. We can deduce that the size of the polynomials $b_{i,j}$ depends on the degree D. Due to the reduction modulo $F(x) + y$, the degree of $b_{i,j}$ can be as big as 2^D. Then 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.* $D = O(1)$.

3.3.2 Size 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 2. 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 $\mathrm{Mon}(a_k)$ is the set of the monomials of a_k we define d_{aff} the affine degree of A by :

$$d_{aff} := \max_k \Big(\max_{m \in \mathrm{Mon}(a_k)} HW(deg_y(m)) \Big).$$

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 $\mathbb{F}_2$ 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.

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 appears 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 them 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 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:

$$\tilde{A}_i(x_1, ..., x_n, y_1, ..., y_n) = \sum_{j=0}^{n} \tilde{A}_{i,j}(y_1, ..., y_n) \times x_j$$

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 compared to a GB - that is few kB. Our construction is composed of :

- A code that hashes a message $m = (m_1, ..., m_n)$. Since any secure hash function can be used, this code size is negligible.
- A code that evaluates, on an input m 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^2 files $File_{i,j}$ $i,j < n$ for which the *k-th* bit of the file $File_{i,j}$ is 1 if the *k-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 $\sigma(n, d_{aff})$. 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.
- 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 compute a solution to 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 if 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^2. The rest of the code is negligible.
- A code that salts m if the previous check failed. This code is obviously negligible.

This means that the code is composed of the n^2 $File_{i,j}$ for $n^2 \times \sigma(n, d_{\text{aff}})$ bits, the matrix $Mat_{\tilde{A}}$ of n^2 bits, the public key of n^2 bits, and some neglibible code. The full size is then :

$$n^2 \times \sigma(n, d_{\text{aff}}) + 2n^2 + negl \approx n^2 \times \sigma(n, d_{\text{aff}})$$

3.3.3 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.

To get better estimations, we performed experiments with Sage for $D < 17$ for quadratic forms over $\mathbb{F}_2$. The affine degree observed are below the bound D and seem to depend heavily on the rank. For instance, for skew-symmetric forms of at most degree 12 and rank 4, the affine degree varies between 4 and 6. However for skew-symmetric forms of at most degree 12 and rank 3, the affine degree varies between 1 and 5. This encourages us to consider the affine degree on a case by case basis depending on the rank.

Experiments were also made over bigger field $\mathbb{F}_{2^n}$, and it seems that considering the same rank and degree for our polynomials with coefficients over $\mathbb{F}_{2^n}$ leads to a small increase of the affine degree. To our understanding, it is just a consequence of the algebraic simplifications that used to happen over $\mathbb{F}_2$ and are not present over $\mathbb{F}_{2^n}$.

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 [Dob99]) has a multiple affine of affine degree 3 over $\mathbb{F}_{2^{2m+1}}$ for every value m ([Pat96]), which is really different from the observed values.

3.4 Choice of parameters: Size and Black-Box security

We now detail the instantiation chosen to reach $\lambda = 80$ as a security level. The parameters to be chosen for the HFE instance are n and F. We focus here on the security regarding the public-key and discuss the white-box security in the following section.

First, let us recall the complexity of the best known attacks against HFE: key-recovery and inversion. If we set d_{reg} to be the degree of a fixed HFE instance, D its degree and $d = \lceil log_2(D) \rceil$, the best inversion attack is made in $\mathcal{O}\left(\binom{n+d_{reg}}{d_{reg}}^{\omega}\right)$ and the best key-recovery in $\mathcal{O}\left(\left(n^2\binom{2d+2}{d} + n\binom{2d+2}{d}^2\right)^{\omega}\right)$. As recalled in the introduction on HFE, resisting these attacks is enough to get a secure black box HFE instance.

3.4.1 Finding Ideal Parameters

We need to choose F with the smallest d_{aff} possible, while the degree d_{reg} and the rank of the associated HFE instance are maximal. However, there are obstacles to the growth of these parameters. Following the study we made on the affine degree, polynomials F with high degree D and low d_{aff} are rare – or inexistant – and the known examples are of low rank (which makes the MinRank problem easier to solve), and with coefficients over $\mathbb{F}_2$ (which leads to efficient key-recoveries [BFJT09]). This is why our strategy here is to get d_{aff}, d_{reg} and d as close as possible. It also allows us to stay close to our rationale, that is, to keep a gap ω between the size of the construction and the best known attacks.

Following the study from [FJ03] on the degree of regularity of HFE instances, we know that for any integer n, an HFE instance with a polynomial F of degree $4 \leq D < 17$ has a degree of regularity equal to 3. This means that if we fix such a polynomial, the inversion attack has complexity $\mathcal{O}\left(\binom{n+3}{3}^{\omega}\right)$ and is polynomial in n of degree 3ω. For the same F, the maximum d is 4, so we want the rank of F to be as close as possible to 4 while having the smallest possible affine degree: the MinRank key-recovery attack is the most restraining, and the affine multiple attack is one of the weakest.

Exhaustive search of quadratic forms of rank 4 with $D = 16$ and coefficients over $\mathbb{F}_2$ shows that all the affine multiples have at least $d_{aff} = 4$. As our discussion on the affine degree shows, this degree seems to be a generic lower bound if we take coefficients in $\mathbb{F}_{2^n}$. This means that to reach $\lambda = 80$ against key-recovery, we need $log(n) \approx 9.5$. The size of our white-box construction with $d_{aff} = 4$ is then 2^{57}, which is a bit too prohibitive.

Among the quadratic forms of lower rank, we use the fact that for any $a, b \in \mathbb{F}_{2^n}$, the polynomial

$$F(x) = x^6 + ax^5 + bx^3$$

has an affine multiple of the form:

$$\begin{aligned}
&(y^3(a^{13} + a^{10}b + a^7b^2 + a^4b^3) + y^2(a^{10}b^3 + a^4b^5 + ab^6) + a^4b^7y) \times x^{2^0} \\
&+ ((a^{12} + a^9b) \times y^3 + (a^{12}b^2 + b^6) \times y^2) \times x^{2^1} \\
&+ (a^4y^4 + y^3(a^4b^2 + ab^3) + y^2(a^{13}b + a^4b^4 + ab^5) + y(a^{10}b^4 + ab^7) + a^4b^8) \times x^{2^2} \\
&+ (y^4 + y^2(a^9b + a^6b^2 + a^3b^3 + b^4) + y(a^{12}b^2 + a^9b^3 + a^6b^4) + a^{12}b^4 + b^8) \times x^{2^3} \\
&+ ((a^4b^2 + ab^3) \times y + a^{16} + a^4b^4) \times x^{2^4} \\
&+ x^{2^5} = 0
\end{aligned} \tag{1}$$

The polynomial F is of rank 3 as a quadratic form for any $a, b \neq 0$ and has an affine multiple of degree 2 if and only if all the coefficients in $y^3x^{2^k}$ are not 0. As the coefficient of $y^3x^{2^1}$ is $a^{12} + a^9b$, we see that if $b \neq a^3$, our affine multiple is of affine degree 2. This choice of the coefficients gives us a polynomial F with $d_{aff} = 2$ and $d = 3$.

3.4.2 Size and Black-Box Security

To reach $\lambda = 80$ for $F = x^6 + ax^5 + bx^3$ against key-recovery, following from the study of [Din20]: we need $log(n) \approx 10.5$. Then the HFE instance is secure as well against the best known attacks and our security becnhmark is met.

We can now proceed to size analysis with $log(n) \approx 10.5$ and the polynomial $F(x) = x^6 + ax^5 + bx^3$, plugging the values into the formula of the previous part:

$$Size(WBHFE_{\mathcal{S}_{HFE}}(S, F, T))) = 2^{41} \text{ bits}$$

This means that we achieve the desired level of security for about 256GB of memory. Regarding greater values of λ, one can increase the value of n to increase security, but the implementation would of course be bigger: this is a common feature of white-box implementations. However, it is clear that the gap between the implementation size and the security level will shrink if F is fixed and n increases.

Regarding time, as the Gaussian inversion is made over $\mathbb{F}_2$, it is negligible, the main part is to perform the polynomials evaluation which can be made in 2^{41} as well. This gives an estimation of signature time of several minutes on a modern computer.

4 Security analysis in the white-box model

4.1 White-Box Security Notions

The usual software white-box security notions that are discussed in the literature are unbreakability, incompressibility, one-wayness and traceability. While we prove some strong point for the first two, we do not consider traceability, and one-wayness does not make sense for decryption or signature in asymmetric cryptography: the one-wayness of decryption/signature would mean that encryption/signature is impossible, which is a contradiction by definition. The goal of this section is to explain why the expected security level is 2^{80} in the white-box model.

4.1.1 Definitions

As we focus here on the white-box security of an asymmetric signing primitive based on a decryption primitive, we describe unbreakability and incompressibility in the case of asymmetric signature. These definitions are obviously really close to the symmetric ones, with the main difference that one-wayness for an asymmetric signature scheme is not relevant. The main difference with definitions found in [DLPR14] is that we do not specialize the adversary to certain classes of attacks: it is the most generic possible.

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

- Draw at random a key k in private keyspace $K_{\mathcal{S}}$
- The adversary $\mathcal{A}$ gives a key k to get the program $\mathcal{C}_{\mathcal{S}}(k)$ from the compiler
- The adversary $\mathcal{A}$ returns a key guess $\hat{k}$ in time τ knowing $\mathcal{C}_{\mathcal{S}}(k)$.
- 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, $Succ_{\mathcal{A},\mathcal{C}_{\mathcal{S}}} \leq \epsilon$.

Remark: Here, 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.

The definition of incompressibility we propose here is a slightly corrected version from the usual one used in [DLPR14]. Indeed, this one is flawed and almost empty 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 couples 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 functionnal 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 the previous modification does not invalidate the proofs found in the literature for incompressibility. Indeed, these proofs are made in the special models

where the programs produced by the attacker are restrained by a model (Ideal Group Model for instance), whereas the program we proposed does not fit into these models. However, it does not change the fact that the general definition in [DLPR14] is incomplete.

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

- Draw at random a key k in private keyspace $K_\mathcal{S}$
- The adversary $\mathcal{A}$ gives a key k to get 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 $\mathcal{P}$ and $\mathcal{C}_\mathcal{S}(k)$ and $size(\mathcal{P}) \leq \sigma$.

Definition 4. 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:

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

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 $Succ_{\mathcal{A},\mathcal{C}_\mathcal{S}} \leq \epsilon$.

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 exploit this fact, we did not include it for sake of clarity.

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.

4.2 First Links with the Cubic IP1S Problem

To analyse unbreakability and incompressibility, we rely on two properties of the polynomials describing our white-box implementation. The first will be the key-recovery of the underlying cubic IP1S (Isomorphism of polynomials with one secret) instance, a problem that has been studied in papers such as [PGC98, Per05, BFFP11, MPG13] 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.

4.2.1 Secret recovery on Cubic 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 cubic 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 do not know any other attack against it (as an IP problem) than the best generic ones. We now state our first assumption:

Assumption 1: Security of Cubic IP1S The cubic IP1S instance defined the familly of polynomial A_i is secure to the level of security 2^{80} against any secret-recovery attack.

Although our instance is not a generic one, we can rely on the state of the art against cubic IP1S problem to back this assumption up. To the best of our knowledge, the best generic attack on cubic IP1S with affine secrets is in complexity $\mathcal{O}(n^6 q^n)$. This means that the best attack known against this instance is exponential in $n \approx 2^{10.5}$, which allows us to be far above 2^{80}.

Remark: Even if it does not change the security for our set of parameters, we would like to point out that the attack described in [BFFP11] is in $\mathcal{O}(n^{12}q^n)$ and not $\mathcal{O}(n^6 q^n)$ as it was claimed. The computation of the Gröbner basis used in the attack is made over n^2 variables (coordinates of a formal matrix), and the degree of regularity is reported to be 2 for the system studied. This means that the overall complexity is $\mathcal{O}((n^2)^{2\times\omega})q^n)$, which cannot be $\mathcal{O}(n^6 q^n)$.

4.2.2 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 transformation 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 5. 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: 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, 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 breakability problem, for which the best known attacks require a computational effort way larger than 2^{80} (see paragraph about Secret recovery on Cubic IP1S). In the intermediate cases $size(S) < \sigma \leq size(P_i)$, to the best of our knowledge, no attack has been found in the literature, which leads us to the following assumption:

Assumption 2: Incompressibility of IP1S The cubic IP1S instance defined the familly of polynomial A_i is $(2^{40},2^{80})$-incompressible.

Remark: As it is hard to give precise values for the sizes and times involved (due to programming language and model of computation), we always underestimate our parameters. For instance, the size of our construction is estimated to be 2^{41} bits, but we try to prove its $(2^{40}, \tau)$-incompressibility, giving us a noticeable margin of error. The practical security might be even closer to the implementation size.

Remark: Assumption 2 is two-edged. If we can provide for a functional white-box implementation with $\sigma < 2^{40}$, few GB for instance, deploying our solution in a regular devices would be easier. In that sense, some compression might be interesting. 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.

4.3 Reduction and Discussion on General Attacks

The goal of this section is to show the strength of multivariate cryptography methods for white-box cryptography. Roughly, the idea is to show that the implementation we have is has strong as the underlying IP instance, because the rest of the code is just made of generic evaluations. The rest of the section proves the unbreakability using Assumption 1 and argues for the incompressibility under Assumption 2.

4.3.1 Important remark on the IP problem and Generic White-Box Attacks

Before we start to discuss the unbreakability and the incompressibility of our construction, let us discuss the efficiency of generic white-box attacks against IP instances.

No mention of Fault Attacks (DFA), DCA or LDA can be found in the state-of-the-art attacks against IP instances. This means that these methods are not known to be efficient to solve the IP problem. One could also try to explain directly why these attacks do not work. For instance, for DCA, if we cannot identify a computation that depend on few key bits, the complexity grows exponentially fast. For IP problems, the only subcomputations we can identify are evaluations or coefficients of the polynomials. However, it can be seen that these results already depend on at least n key bits, so any attempt of DCA is not realistic. For DFA, faults can be made to solve a regular IP instance, but any evaluation of the polynomials is already allowed to solve IP. This is a huge difference compared to the usual targets of these attacks: in the state-of-the-art it is usually a Sbox of an SPN, and most often of the AES. This means that the generic white-box attacks are not a threat to IP instances.

4.3.2 Unbreakability

Theorem 1. *Let $(\tilde{A}_i)_{i\in[\![1,n]\!]}$ be the IP instance of with known polynomials $(A)_{i\in[\![1,n]\!]}$ and secrets S and T. Let $\mathcal{S}_{HFE}$ be our HFE signature scheme. If $(\tilde{A}_i)_{i\in[\![1,n]\!]}$ is secure for secret recovery for any adversary $\mathcal{A}$ running in at least time τ, the white-box implementation $\boldsymbol{WBHFE}_{\mathcal{S}_{HFE}}$ is $(\tau, 1)$-unbreakable in the white box model.*

Proof. Let us prove this result *ad absurdum*. To do so, we suppose that there exist an attacker $\mathcal{A}$ running in time τ that can recover the secret key S and T from our white-box implementation. Then we remark that the white-box code computing D_{HFE} only performs generic partial evaluation of the polynomials $(\tilde{A}_i)_{i\in[\![1,m]\!]}$, and then performs Gaussian invertion on the linear system of equation left after the values y_i are plugged in. We

can now build an attacker $\mathcal{B}$ that uses $\mathcal{A}$ to break the overall IP problem: $\mathcal{B}$ asks for an IP1S instance of the $(\tilde{A}_i)_{i\in[\![1,m]\!]}$ form. She then programs the generic evaluation of these polynomials, the linear inversion after evaluation to get a program P. She then gives P to $\mathcal{A}$ who find S and T in time τ. As the programming part done by $\mathcal{B}$ is trivial, this leads to an overall attack in time τ.

□

This proof means that there is no gap between the security the kind of IP problem we are studying and the white-box security as long as the operations we perform are generic. This also means that if usually powerful attacks like DCA or DFA could break our construction, they would also be a powerful cryptanalysis for IP problems in general.

Remark 1: This theorem formalizes the remark made in the previous paragraph regarding general white-box attacks (DCA, DFA or LDA). It means that these attacks are not successful against our construction. Otherwise, it would be a huge breakthrough in the analysis of the IP problem.

Remark 2: This proof could be adapted to any primitive using multivariate cryptography: as long as generic operation are made by the white box implementation, there is no gap between the security of the overall instance and the white-box implementation.

With the help of Assumption 1, this easily leads to the following security result:

Corollary 1. *If Assumption 1 holds, then for any secret key (S, F, T), the white-box compiler $\boldsymbol{WBHFE}_{\mathcal{S}_{HFE}}$ is $(2^{80}, 1)$-unbreakable.*

4.3.3 Incompressibility

The incompressibility of $\mathbf{WBHFE}_{\mathcal{S}_{HFE}}$ can easily be deduced from Assumption 2 if the following conjecture is true:

Conjecture: Any attacker that breaks the $(2^{40}, 2^{80}, 1)$-incompressibility of *WBHFE* can break the $(2^{40}, 2^{80})$-incompressibility of the IP instance defined by the polynomials $(A_i)_{i\in[\![1,n]\!]}$

The main difficulty to prove such a statement lies in the fact that the polynomials $\tilde{A}_i$ stored in the implementation allow not only the computation of $x = P^{-1}(y)$ (by plugging an explicit value of y and then solving $A(x, y) = 0$ in x by Gaussian reduction), but also the computation of $\tilde{A}_i(x, y)$ even if $y \neq P(x)$. However we conjecture that, from a "compressed" implementation for P^{-1}, we could derive a "compressed" implementation allowing to compute $(A(x, y)$ for any pair (x, y), where A is an affine multiple.

The incompressibility of the polynomials A_i (Assumption 2) thus appears to be crucial, because – under the conjecture above – it implies the incompressibility of our white-box implementation. As already mentioned, since the polynomials $\tilde{A}_i$ are given in their normal form, it is very likely that they are not compressible and that the generic operations needed to compute their output value do not allow the attacker to compress the whole computation of P^{-1} (Note that if the $\tilde{A}_i$ where truely random, they would be impossible to compress).

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 [DLPR14], using an Ideal Group Model).

5 Conclusion and opening remarks

While many attempts have been made to construct white-box implementations of block-ciphers, almost no white-box implementations have been published for what concerns asymmetric schemes. We have presented here a concrete construction of such a scheme, together with a white-box implementation of the signature algorithm. For a security level 2^{80}, the public key size is approximately 62.5 MB and the white-box implementation of the signature algorithm has a size approximately 256 GB. Even if we focus on a precise security level, the overall security can be increased at the cost of a bigger implementation.

We propose a new approach to unbreakability and incompressibility in the general white-box model through the use of multivariate polynomials in normal form and more precise definitions for these concept. For unbreakability, we reduce it to breaking a specific cubic IP instance. For incompressibility, we break the security argument into an incompressibility problem on IP instances and a conjecture on the attacker's ability. This breaks from the usual techniques in literature that were relying on special models or the use of truth table for instance, and motivates the usage of multivariate polynomials in normal form in general white-box constructions.

A challenge for further research would be to find a way of evaluate the value of the affine degree d_{aff} for all HFE instances. It has indeed being shown that the cost of computing such a relation for high degree polynomials is prohibitive. For instance, one can wonder if the value d_{aff} can be computed without its affine relation or if less costly methods can be developped.

Also, examples like the Dobbertin polynomials show that there might exist polynomials with high degree but low affine degree. Finding such familly of polynomials with associated HFE instances of high rank and reasonable degree of regularity could lead to improvements regarding our scheme and would probably help to reach higher levels of security with a smaller implementation.

One could remark that $A(x, y)$ is not strictly equivalent to F^{-1}. As it was used to provide a solution here, it seems interesting to consider such structures for other white-box implementations. Having access to a structure that is not equivalent to the primitive that is to be white-boxed might give more freedom to designers.

References

[ABF+20] Estuardo Alpirez Bock, Chris Brzuska, Marc Fischlin, Christian Janson, and Wil Michiels. Security reductions for white-box key-storage in mobile payments. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 221–252, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany.

[ABG+16] Carlos Aguilar Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, and Tancrède Lepoint. NFLlib: NTT-based fast lattice library. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 341–356, San Francisco, CA, USA, February 29 – March 4, 2016. Springer, Heidelberg, Germany.

[Bar20a] Lucas Barthelemy. *A First Approach To Asymmetric White-Box Cryptography and a Study of Permutation Polynomials Modulo* 2^n *in Obfuscation*. Ph.d thesis, Sorbonne Université, Paris, December 2020.

[Bar20b] Lucas Barthelemy. Toward an asymmetric white-box proposal. Cryptology ePrint Archive, Report 2020/893, 2020. https://eprint.iacr.org/2020/893.

[BBK14] Alex Biryukov, Charles Bouillaguet, and Dmitry Khovratovich. Cryptographic schemes based on the ASASA structure: Black-box, white-box, and public-key (extended abstract). In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 63–84, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany.

[BCD06] Julien Bringer, Herve Chabanne, and Emmanuelle Dottax. White box cryptography: Another attempt. Cryptology ePrint Archive, Report 2006/468, 2006. http://eprint.iacr.org/2006/468.

[BFFP11] Charles Bouillaguet, Jean-Charles Faugère, Pierre-Alain Fouque, and Ludovic Perret. Practical cryptanalysis of the identification scheme based on the isomorphism of polynomial with one secret problem. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 473–493, Taormina, Italy, March 6–9, 2011. Springer, Heidelberg, Germany.

[BFJT09] Charles Bouillaguet, Pierre-Alain Fouque, Antoine Joux, and Joana Treger. A family of weak keys in HFE (and the corresponding practical key-recovery). Cryptology ePrint Archive, Report 2009/619, 2009. http://eprint.iacr.org/2009/619.

[BFS97] Jonathan F. Buss, Gudmund S. Frandsen, and Jeffrey O. Shallit. The computational complexity of some problems of linear algebra, 1997.

[BFS15] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of the F5 gröbner basis algorithm. *J. Symb. Comput.*, 70:49–70, 2015.

[BG03] Olivier Billet and Henri Gilbert. A traceable block cipher. In Chi-Sung Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 331–346, Taipei, Taiwan, November 30 – December 4, 2003. Springer, Heidelberg, Germany.

[BGEC04] Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi. Cryptanalysis of a white box AES implementation. In Helena Handschuh and Anwar Hasan, editors, *SAC 2004*, volume 3357 of *LNCS*, pages 227–240, Waterloo, Ontario, Canada, August 9–10, 2004. Springer, Heidelberg, Germany.

[BI15] Andrey Bogdanov and Takanori Isobe. White-box cryptography revisited: Space-hard ciphers. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 1058–1069, Denver, CO, USA, October 12–16, 2015. ACM Press.

[BIT16] Andrey Bogdanov, Takanori Isobe, and Elmar Tischhauser. Towards practical whitebox cryptography: Optimizing efficiency and space hardness. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 126–158, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.

[BKR16] Mihir Bellare, Daniel Kane, and Phillip Rogaway. Big-key symmetric encryption: Resisting key exfiltration. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 373–402, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[BP15] Alex Biryukov and Léo Perrin. On reverse-engineering S-boxes with hidden design criteria or structure. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 116–140, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[CEJv03] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. White-box cryptography and an AES implementation. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 250–270, St. John's, Newfoundland, Canada, August 15–16, 2003. Springer, Heidelberg, Germany.

[CEJvO02] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. A white-box DES implementation for DRM applications. In Joan 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.

[CFM+20] 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.

[CGSV16] Ryann Cartor, Ryan Gipson, Daniel Smith-Tone, and Jeremy Vates. On the differential security of the HFEv- signature primitive. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016*, pages 162–181, Fukuoka, Japan, February 24–26 2016. Springer, Heidelberg, Germany.

[DFSS06] Ivan B. Damgård, Serge Fehr, Louis Salvail, and Christian Schaffner. Oblivious transfer and linear functions. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 427–444, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany.

[Din20] Chengdong Tao Albrecht Petzoldt Jintai Ding. Improved key recovery of the HFEv- signature scheme. Cryptology ePrint Archive, Report 2020/1424, 2020. https://eprint.iacr.org/2020/1424.

[DLPR14] Cécile Delerablée, Tancrède Lepoint, Pascal Paillier, and Matthieu Rivain. White-box security notions for symmetric encryption schemes. In Tanja Lange, Kristin Lauter, and Petr Lisonek, editors, *SAC 2013*, volume 8282 of *LNCS*, pages 247–264, Burnaby, BC, Canada, August 14–16, 2014. Springer, Heidelberg, Germany.

[Dob99] Hans Dobbertin. Almost perfect nonlinear power functions on gf(2n): The welch case. *IEEE Trans. Inf. Theory*, 45:1271–1275, 1999.

[DRP13a] Yoni De Mulder, Peter Roelse, and Bart Preneel. Cryptanalysis of the Xiao-Lai white-box AES implementation. In Lars R. Knudsen and Huapeng Wu, editors, *SAC 2012*, volume 7707 of *LNCS*, pages 34–49, Windsor, Ontario, Canada, August 15–16, 2013. Springer, Heidelberg, Germany.

[DRP13b] Yoni De Mulder, Peter Roelse, and Bart Preneel. Revisiting the BGE attack on a white-box AES implementation. Cryptology ePrint Archive, Report 2013/450, 2013. http://eprint.iacr.org/2013/450.

[DS14] Taylor Daniels and Daniel Smith-Tone. Differential properties of the HFE cryptosystem. Cryptology ePrint Archive, Report 2014/398, 2014. http://eprint.iacr.org/2014/398.

[DWP10] Yoni De Mulder, Brecht Wyseur, and Bart Preneel. Cryptanalysis of a perturbated white-box AES implementation. In Guang Gong and Kishan Chand Gupta, editors, *INDOCRYPT 2010*, volume 6498 of *LNCS*, pages 292–310, Hyderabad, India, December 12–15, 2010. Springer, Heidelberg, Germany.

[EMV08] EMV. Integrated circuit card specifications for payment systems. Book 2. Security and Key Management. Version 4.2. June 2008. www.emvco.com., 2008.

[Fau02] Jean-Charles 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.

[FHW+20] Qi Feng, Debiao He, Huaqun Wang, Neeraj Kumar, and Kim-Kwang Raymond Choo. White-box implementation of shamir's identity-based signature scheme. *IEEE Syst. J.*, 14(2):1820–1829, 2020.

[FJ03] Jean-Charles Faugère and Antoine Joux. Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using gröbner bases. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 44–60, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

[FKKM16] Pierre-Alain Fouque, Pierre Karpman, Paul Kirchner, and Brice Minaud. Efficient and provable white-box primitives. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 159–188, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.

[FP06] Jean-Charles Faugère and Ludovic Perret. Polynomial equivalence problems: Algorithmic and theoretical aspects. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 30–47, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.

[GMQ07] Louis Goubin, Jean-Michel Masereel, and Michaël Quisquater. Cryptanalysis of white box DES implementations. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *SAC 2007*, volume 4876 of *LNCS*, pages 278–295, Ottawa, Canada, August 16–17, 2007. Springer, Heidelberg, Germany.

[JBF02] Matthias Jacob, Dan Boneh, and Edward W. Felten. Attacking an Obfuscated Cipher by Injecting Faults. In Joan 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.

[Kar11] Mohamed Karroumi. Protecting white-box AES with dual ciphers. In Kyung Hyune Rhee and DaeHun Nyang, editors, *ICISC 10*, volume 6829 of *LNCS*, pages 278–291, Seoul, Korea, December 1–3, 2011. Springer, Heidelberg, Germany.

[KS99] Aviad Kipnis and Adi Shamir. Cryptanalysis of the HFE public key cryptosystem by relinearization. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 19–30, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany.

[LN04] Hamilton E. Link and William D. Neumann. Clarifying obfuscation: Improving the security of white-box encoding. Cryptology ePrint Archive, Report 2004/025, 2004. http://eprint.iacr.org/2004/025.

[LR13] Tancrède Lepoint and Matthieu Rivain. Another nail in the coffin of white-box AES implementations. Cryptology ePrint Archive, Report 2013/455, 2013. http://eprint.iacr.org/2013/455.

[LRD+14] Tancrède Lepoint, Matthieu Rivain, Yoni De Mulder, Peter Roelse, and Bart Preneel. Two attacks on a white-box AES implementation. In Tanja Lange, Kristin Lauter, and Petr Lisonek, editors, *SAC 2013*, volume 8282 of *LNCS*, pages 265–285, Burnaby, BC, Canada, August 14–16, 2014. Springer, Heidelberg, Germany.

[Masa] Mastercard. Mastercard cloud-based payments – mobile payment application functional description. Version 1.0, August 2014.

[Masb] Mastercard. Mastercard mobile payment sdk security guide for mp sdk v1.0.6. Version 2.0, January 2017. https://developer.mastercard.com/media/32/b3/b6a8b4134e50bfe53590c128085e/mastercard-mobile-payment-sdk-security-guide-v2.0.pdf.

[MDFK15] Brice Minaud, Patrick Derbez, Pierre-Alain Fouque, and Pierre Karpman. Key-recovery attacks on ASASA. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 3–27, Auckland, New Zealand, November 30 – December 3, 2015. Springer, Heidelberg, Germany.

[MI88] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynominal-tuples for efficient signature-verification and message-encryption. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, pages 419–453, Davos, Switzerland, May 25–27, 1988. Springer, Heidelberg, Germany.

[MIHM83] Tsutomu Matsumoto, Hideki Imai, Hiroshi Harashima, and Hiroshi Miyakawa. A class of asymmetric cryptosystems using obscure representations of enciphering functions. *1983 National Convention Record on Information Systems, IECE Japan*, 1983.

[MPG13] Gilles Macario-Rat, Jérôme Plût, and Henri Gilbert. New insight into the isomorphism of polynomial problem IP1S and its use in cryptography. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 117–133, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany.

[Pat96] Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 33–48, Saragossa, Spain, May 12–16, 1996. Springer, Heidelberg, Germany.

[Per05] Ludovic Perret. A fast cryptanalysis of the isomorphism of polynomials with one secret problem. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 354–370, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany.

[PGC98] Jacques Patarin, Louis Goubin, and Nicolas Courtois. Improved algorithms for isomorphisms of polynomials. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 184–200, Espoo, Finland, May 31 – June 4, 1998. Springer, Heidelberg, Germany.

[SSH11] Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari. On provable security of UOV and HFE signature schemes against chosen-message attack. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*, pages 68–82, Tapei, Taiwan, November 29 – December 2 2011. Springer, Heidelberg, Germany.

[WMGP07] Brecht Wyseur, Wil Michiels, Paul Gorissen, and Bart Preneel. Cryptanalysis of white-box DES implementations with arbitrary external encodings. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *SAC 2007*, volume 4876 of *LNCS*, pages 264–277, Ottawa, Canada, August 16–17, 2007. Springer, Heidelberg, Germany.

[XL09] Yaying Xiao and Xuejia Lai. A secure implementation of white-box AES. 2nd International Conference on Computer Science and its Applications (CSA 2009), 2009.

[ZHH+20] Yudi Zhang, Debiao He, Xinyi Huang, Ding Wang, Kim-Kwang Raymond Choo, and Jing Wang. White-box implementation of the identity-based signature scheme in the IEEE P1363 standard for public key cryptography. *IEICE Trans. Inf. Syst.*, 103-D(2):188–195, 2020.