White-Box Implementation Techniques for the HFE family

Pierre Galissant and Louis Goubin

Laboratoire de Mathématiques de Versailles, UVSQ, CNRS, Université Paris-Saclay, 78035 Versailles, France
pierre.galissant@gmail.com, 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: designing 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 the first white-box implementation technique of HFE signature primitives, for a specific set of internal polynomials. For instance our implementations of the signature primitive range from about 94MB for security level $\lambda = 80$ to 752MB for $\lambda = 128$ for variations of the $C^{*\hat{+}-}$ primitive, and similar ranges for pC^{*-}. To motivate the study of the security of the techniques we use, we also propose a challenge implementation and the white-box compiler used to produce this implementation.

Keywords: White-box cryptography, Public-key cryptography, Multivariate cryptography

1 Introduction

Since the seminal paper of Chow *et al.* in 2002, the research in the white-box model has mainly been focused on standard symmetric block-ciphers such as DES and AES. Many candidate implementations have been proposed [13, 12, 10, 43, 27, 28] but all have later been broken due to very powerful generic attacks (such as [8, 7, 25, 24]) or specific structural attacks (for instance [4, 23]). The goal of these implementations is to provide security, with respect to stronger attack models, but weaker cryptographic notions than in the black box model: unbreakability (being unable to extract the key from the code), incompressibility (being unable to compress the code) or one-wayness (being unable to invert the implementation).

Regarding asymmetric candidates, very few solutions have been proposed: the implementation of Barthelemy [3] does not stand against generic attacks and the implementations of Shamir's signature [40] in [19] and of an IEEE P1363 signature in [44] only focus on key-extraction, ignoring the incompressibility, while modifying the verifying algorithm in the white-box version of the scheme. Recently unbreakable implementations of ECDSA have been investigated in the WhibOx21 and WhibOx24 contests [41, 42]. However, among the hundreds of implementations, all were broken in few days (see [1] and the websites of the contests [41, 42]). After almost two decades of research, it is safe to say that getting a secure white-box implementation for an existing algorithm is a hard open problem.

In this paper, we propose the first investigation of white-box implementations of the HFE signature primitive (Hidden Field Equation) [35], that we quickly recall in Section 2.2. Our study shows 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.

The techniques we use are radically different from the usual techniques used in constructions for white-box symmetric cryptography. Moreover, they do not rely on security through obscurity, *i.e.* our compiler is described according to Kerckhoffs' principle and does not use any heuristic code obfuscation techniques.

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

– We propose a new white-box implementation technique for nude HFE instances and incorporate the p, $-$ and $\hat{+}$ perturbations into our implementations. This design technique follows Kerchkoffs' principle and does not rely on any hidden information or procedure.
– The *unbreakability* property of our implementation in the white-box model is proven under a reasonable assumption about the *Isomorphism of Polynomials* (IP) problem: a key-recovery attack on the white-box implementation 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 security conjecture in the case of our implementation : under this assumption, an attack aiming at obtaining a smaller implementation is not easier than all the standard attacks in the black-box model.
– We propose different parameters and choices of internal polynomials to derive concrete instantiations for different security parameters : from about 94MB for security level $\lambda = 80$ to 752MB for $\lambda = 128$ for variations of the $C^{*\hat{+}-}$ primitive, and similar ranges for pC^{*-}.
– We propose a challenge implementation as well as the white-box compiler used to produce it to motivate the study of the techniques used in the paper : `https://github.com/p-galissant/WBHFE`

Note that more material can be found in the extended version of the present paper, with additional details about multivariate cryptography, and other variants of HFE-based signature implementations in the white-box model.

Remark: From a practical point of view, the white-box implementations presented here are quite heavy, which is somewhat inherent to the white-box model (and is especially related to the concept of code incompressiblity). Nevertheless, the obtained sizes (from dozens to hundreds of MB) leave the possibility of using them in realistic contexts, typically smart-phone applications, or cloud-based applications for the strongest security levels. As an example, MasterCard Cloud-Based Payments (MCBP) [30] is a secure and scalable software-based solution developed 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 [31]. In addition, from a more theoretical point of view, it should be noted that our construction provides the first white-box implementation of a public key algorithm together with an extensive security analysis.

Technical Overview Technically, our proposal is heavily based on the structure and flexibility 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. It is then composed with affine secrets S and T to get the public key P.

The usual way to invert P is to use the structure of F in the field extension and the trapdoor consisting in the knowledge of S and T. However, we introduce a second trapdoor with the concept of affine multiple that allows to invert P while not revealing S and T. Namely, 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, that is:

- 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. We then adapt this technique to also include perturbations of the public key P.

As a consequence, we obtain the following results: for an attacker who only knows the public key of this HFE signature scheme, recovering the white-box implementation is not easier than breaking the scheme in the black-box model; and 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*. We adapt the usual definition (see [14] for instance) to the public-key setting in section 4. In this context, we describe a white-box implementation of an HFE signature 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 [13]. Ever since, cryptographers have tried to propose white-box implementations of usual cryptographic algorithms, with mitigated success. Block-ciphers have been the most studied as seen above. Recently, the WhibOx 2021 contest [41] showed the interest of the community to produce an unbreakable implementation of the ECDSA signature algorithm and the hundreds of implementations have all been broken. For the 2024 edition, the contest is still focused on ECDSA [42].

The notion of incompressibility is a stronger security notion, first formally defined in [14] 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 [5], [6], [20] or [26] have worked on defining incompressibility and studying designed symmetric algorithms to achieve their definitions. However, they focus only on symmetric algorithms.

We also refer to [21, 22] for a recent survey on all these white-box security notions.

Acknowledgments

This research was partly supported by ANR grant SWITECH (ANR-19-CE39-0014-01) and PEPR PQ-TLS (the France 2030 program under grant agreement ANR-22-PETQ-0008 PQ-TLS).

2 Background and Motivations

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

$(M, S, K_{\mathcal{V}} \times K_{\mathcal{S}}, \mathcal{V}, \mathcal{S})$ 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, $\mathcal{S}$ is the signature algorithm, and $\mathcal{V}$ is the verification algorithm.

For any keyed cryptographic function f with key k we note C_f the compiler specified to f and $C_f(k)$ the white box implementation of f with the key k.

For two programs $\mathcal{A}$ and $\mathcal{B}$ we note $\mathcal{A} \approx \mathcal{B}$ when $\mathcal{A}$ and $\mathcal{B}$ are functionally equivalent; *i.e.* they agree on all inputs with probability 1.

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.

2.1 Public-Key White-Box Implementations

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 works of [19] and [44] achieve unbreakability but modify the verification algorithm and do not address incompressibility. It is well known that white-box security is easier to obtain if we allow ad-hoc designs. The present paper deals with a well-known and general-purpose family of signature algorithms, that was not specifically designed for the white-box scenario.

The WhibOx 2021 and WhibOx 2024 contests [41, 42] showed that for the ECDSA algorithm, even with hidden design, getting an unbreakable implementation of small size is out of reach. The implementations, bound by the challenges benchmarks - 100MB source code, 50MB executable size and RAM usage and 10sec running time for Whibox24 for instance - were all broken few days after their publications. The report of Barbu et al. [1] goes through the generic attacks and problems the implementations of the Whibox21 contest suffered. Especially, they show that due to the fact that implementing a robust randomness generator is hard in the white-box model, all the implementations can be broken using classic techniques such as fault attacks or lattice attacks.

Despite the hardness of the problem, there is a growing need for such implementations, motivated by mobile payment ([17]), mobile contract signing (following the eIDAS regulation (EU Reg. N°910/2014) for instance) or blockchain technologies.

2.2 Multivariate Cryptography

We will assume that the reader is familiar with multivariate cryptography, especially with “Big Field” trapdoors.

Description of HFE First described by Patarin in [35], the HFE scheme is a direct descendant of C^* [33, 32]. For any positive integers n and $D \in \mathbb{N}$, the central map $F \in \mathbb{F}_{q^n}[X]$ is defined by:

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

where the $a_{i,j}$, the b_i and c are elements of $\mathbb{F}_{2^n}$. As the integer D bounds the actual degree of any such F, we call D the degree of the HFE instance. As the quantity $\lceil log_q(D) \rceil$ 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 bijective 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 $\mathbb{F}_2^n$ 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 $0 \le i < n$ we note its i-th coordinate $P_i \in \mathbb{F}_q[x_1, ..., x_n]$.

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 assume that the reader is familiar with the perturbations $-$, p ([15, 34]) and $\hat{+}$ ([18]).

Security of HFE instances As we will precisely tune our HFE instances to optimize the trade-off between security and implementation size, we recall briefly the best attacks found in the state-of-the-art against various HFE instances using different perturbations. These attacks can mostly be split in two categories : message-recovery attacks and key-recovery rank attacks. For the complexity of message-recovery attacks, we follow [2]. As this complexity depends on the degree of regularity of the public-key, we use the estimation of this degree following [39, 11, 37, 18]. For the complexity of rank attacks, we follow the works of [34, 18].

3 The Implementation Technique

For the rest of the paper, unless specified otherwise, we consider HFE instances over $\mathbb{F}_2$.

3.1 Affine Multiple Attacks

The starting point of our construction is the concept of affine multiple. It was introduced by Patarin in [35] 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 $F(x) + y$. This means that if $y = F(x)$, then $A(x, y) = 0$.

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 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 2. *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 $a, a_o, ..., a_{D-1} \in \mathbb{F}_{2^n}[y]$, if $Mon(a_k)$ is the set of the monomials of a_k we define d_{aff} the affine degree of A by:*

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

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 $n \times \sigma(n, d_{\mathit{aff}})$ unknown coefficients over $\mathbb{F}_2^n$, where $\sigma(n, d_{\mathit{aff}})$ 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 $n \times \sigma(n, d_{\mathit{aff}})$ 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 $n^2 \times \sigma(n, d_{\mathit{aff}})$ and in time $(n \times \sigma(n, d_{\mathit{aff}}))^{\omega}$.

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 [35]. 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.

3.2 Rationale of the construction

The main 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 in x. If the affine multiple is of a reasonable size, computing x is as easy as evaluating the affine multiple in y.

The feasability of the construction is due to the fact that computing an affine multiple from the secrets is easier than from the public key. As we will see later, the security is expected from the hardness to extract the secret key from the affine multiple and the hardness to compress it.

3.3 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 $\mathbb{F}_{2^n}(y)[x]/_{(P(x)+y)}$ 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:

$$\exists\, a, a_o, ..., a_{D-1} \in \mathbb{F}_2(y)\,, a + \sum_{i=0}^{D-1} a_i x^{2^i} \;= 0 \bmod (F(x) + y)$$

To compute the coefficients a_k, we can use the reductions of the monomials x^{2^i} modulo $F(x)+y$:

$$\exists\, b_{i,0}, ...,, b_{i,D-1} \in \mathbb{F}_2(y)\,, x^{2^i} = \sum_{j=0}^{D-1} b_{i,j} x^j \bmod (F(x) + y)$$

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 secret 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 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 S and 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 3.3
- 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 $WBHFE(S, F, T)$.

3.4 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 smallest 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. The central part of this analysis is the affine degree d_{aff} (definition 2), the greatest Hamming weight that appears in the description of the affine multiple A over $\mathbb{F}_2$. Over $\mathbb{F}_2$, $d_{\mathrm{aff}}+1$ 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 $\sigma(n, d_{\mathrm{aff}})$ to be the number of monomials in at most n variables of degree at most d_{aff}, this means that we have about $n \times \sigma(n, d_{\mathrm{aff}})$ coefficients in $\mathbb{F}_2^n$ 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 $n \times \sigma(n, d_{\mathrm{aff}})$ 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 sizes are "negligible" if they are small - that is a 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 $\sigma(n, d_{\mathrm{aff}})$ 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 $\sigma(n, d_{\mathrm{aff}})$.
- The $n \times n$ 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 $\sigma(n, d_{\mathrm{aff}})$ bits. As we need each of the coordinates, the whole size is $n^2 \times \sigma(n, d_{\mathrm{aff}})$. 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 $n^2 \times \sigma(n, d_{\text{aff}})$ 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:

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

For the rest of the paper we will define:

$$\sigma_{WB} := n^2 \times \sigma(n, d_{\text{aff}})$$

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:

$$\tau_{WB} := n^{\omega} + \frac{n^2}{n_p} \times \sigma(n, d_{\text{aff}})$$

Discussion on the affine degree. As seen in the previous section, the size of 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 examples 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 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 [16]) has a multiple affine of affine degree 3 over $\mathbb{F}_{2^{2m+1}}$ for every value m ([35]), which is – in general – really different from the observed values.

3.5 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. Note that these methods to incorporate perturbations into affine multiples do not change state-of-the-art perturbations on the public-key.

The *p* Perturbation To transform a nude public key to 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 on 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, through the isomorphism π, 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 :

$$G_i(x, y') = \prod_{j=1}^{\epsilon_i} (y' + u_{i,j} - F(x))$$

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 3. *Let $\delta \in \mathbb{N}$, $\forall i$, $F_i \in \mathbb{F}_{2^n}[x]$ and $G(x, y) = \prod_{i=1}^{\delta} (F_i(x) - y)$. The polynomial $A(x, y) \in \mathbb{F}_{2^n}[x, y]$ is said to be a composite affine multiple of G if $A(x, y) = 0 \mod G(x, y)$ 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:

$$H_i(x, y) = \prod_{j=1}^{\delta_i} (y - F(x) - v_{i,j})$$

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.

4 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 construction 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.

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 study 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 in a public-key setting would mean that public-key encryption/verification is impossible, which is a contradiction by definition.

We follow [14] to extend the security notions of (τ, ϵ)-unbreakability and (σ, τ, ϵ)-incompressibility and adapt it to in the public-key setting. Furthermore, we shortly show that the unbreakability and incompressibility of the HFE one-way function is the key element to study.

4.2 Unbreakability

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 $\mathcal{A}$ 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 $\mathcal{C}_\mathcal{S}(k)$
- The adversary $\mathcal{A}$ succeeds if $k = \hat{k}$

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 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 1: 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.

Remark 2: 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, we did not include it for sake of clarity.

4.3 Incompressibility

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 $\mathcal{A}$ 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 5. *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 $Succ_{\mathcal{A},\mathcal{C}_{\mathcal{S}}} \leq \epsilon$.

The definition of incompressibility we propose here is a slightly corrected version from the usual one used in [14]. 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.

4.4 White-Boxing the Inversion of a Secure Trapdoor One-way Function is Enough

The goal of this section is to show that for notions like unbreakability and incompressibility, it is sometime enough to look at the white-box implementation of a core component of the signature algorithm, and not the entire algorithm itself. Note that all the following remarks also work for encryption for instance.

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 also the case for HFE with Feistel-Patarin (used in GeMSS [11]). 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, $f : \mathcal{X} \times \mathcal{K} \to \mathcal{Y}$ is a secure trapdoor one-way function 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. The game for unbreakability for the compiler $\mathcal{C}_{f^{-1}}$ is as follows:

- 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 6. *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, $Succ_{\mathcal{A},\mathcal{C}_{f^{-1}}} \leq \epsilon$.

Remark 1: This definition can also be extended to incompressibility as in section 4.1.

We now study signature schemes that are built "on top" of the inversion of a trapdoor OWF, which a general term to qualify many existing techniques like FDH, PSS or Feistel-Patarin cited before.

Definition 7. *We say that a signature algorithm S is built "on top" of a trapdoor one-way function f if it can decomposed in two algorithms:*

- *Algorithm A : On the input of an element in $Im(f)$, outputs one of its pre-image*
- *Algorithm B : On the input of a message, outputs a signature $S(m)$ of m. The 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 8) 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 WBf^{-1} by simply building WBS and running $\mathcal{A}$. This is absurd by unbreakability of WBf^{-1}.

This result does not seems to be easily extendable to incompressibility. Indeed, it is not trivial to obtain a compression of f^{-1} from a compression of S. However, for the usual "reasonable" constructions (FDH, PSS, Feistel-Patarin,...) we still conjecture that the security of the one way-function carries onto the complete signature algorithm.

Remark: There exist algorithms S that are easily compressible even if f has an incompressible implementation. For instance, if we use full domain hash with a hash function of the form $h \circ f$, trivial compression arises if h has a short description. This encourages to further the study of the incompressibility of $f \circ g$ when f or g are incompressible.

4.5 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. As we will focus the study the unbreakability and incompressiblity of our challenge construction, we will focus on perturbations $\hat{+}$ and (-,1). A reduction for (-,2) can be made similarly to $\hat{+}$.

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 $(\hat{+})$ For the $(\hat{+})$ perturbation, with the notations of section 3.5, notice that 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. In our setting, the exact values $v_{i,j}$ are unknown. To the best of our knowledge, recovering $Q(x)$ can only be made by factoring $\prod_{j=1}^{\epsilon_i} (Q(x) + v_{i,j}) = 0$, which is hard for any instance we will use later.

Remark 2: 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 [18] which confirms our security analysis.

4.6 The Implementation as a $(d_{\text{aff}} + 1)$-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 $(d_{\text{aff}}+1)$-IP1S instance. The IP (Isomorphism of Polynomials) problem – studied in papers such as [36, 38, 9, 29] – has been introduced to explore the security of multivariate cryptography in general. We quickly recall that for an integer d, an d-IP instance is defined by a system of polynomials $g = b \circ f \circ a$ where F is a known system of n polynomials in n variables, and hidden a and b affine bijective n-by-n transformations. To solve the IP instance, one needs to find a, b. When there is only one secret b, the instance is usually called a IP instance with one secret abbreviated d-IP1S.

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 $(d_{\text{aff}} + 1)$-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 $(d_{\mathit{aff}}+1)$-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 $\mathcal{O}(n^6q^n)$ ([9]). This means that the best known attack against $3-$IP1S instances is exponential in n . For our instances, $(d_{\mathit{aff}}+1) \geq 3$, 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, since 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 8. *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 $\mathcal{A}$ be an adversary. We say that $(\tilde{P}_i)_{i\in[\![1,m]\!]}$ is (σ,τ,ϵ)-incompressible if there is no adversary $\mathcal{A}$ that wins the σ-incompressibility game with probability ϵ 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 large computational efforts - see paragraph above, about Secret recovery on $(d_{\mathit{aff}} + 1)$-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.

4.7 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 1: We can think of the following factorization-flavoured analogy. Assume that 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 gets so high that this kind of DCA (resp. DFA) strategy is not relevant here.

4.8 Conclusion of the Analysis of Security

In the sections above, we considered all the known attacks in the white-box context. The conclusion of the analysis is that, with the best of our knowledge, none of these attacks is able to retrieve the secret key (unbreakability) nor to compress the implementation (incompressibility). Nevertheless, we are able to formulate the security of our white-box implementation as a set of two conjectures, explicitly stating the links between the (white-box) security of our implementation and natural hypotheses about well-known objects of multivariate cryptography (HFE, modifiers, IP1S problem, affine multiple).

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 $(d_{\mathit{aff}} + 1)$-IP1S instance. (Section 4.3)

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 $(d_{\mathit{aff}} + 1)$-IP1S instance. (Section 4.3)

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 [14], using an Ideal Group Model).

5 Challenge Implementation

In this section, we propose an instance close to $C^{*\hat{+}-}$ for our challenge implementation. For more details about C^*, please refer to [33, 32].

In this section, the instance includes 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:

$$H_i(x,y) = \prod_{j=1}^{\delta_i} (y + v_{i,j} - F(x))$$

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 black-box security of this unusual instance can be checked using attacks of section 2.2: our instance needs to stand against direct inversion with Gröbner bases and key-recovery rank attacks, hence the high value of t. 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 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 $d_{\mathit{aff}} = 3$. 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. 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 : 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 is available at `https://github.com/p-galissant/WBHFE`. It contains the public-key, one affine multiple and resources to manipulate them.

Bibliography

[1] Barbu, G., Beullens, W., Dottax, E., Giraud, C., Houzelot, A., Li, C., Mahzoun, M., Ranea, A., Xie, J.: Ecdsa white-box implementations: Attacks and designs from whibox 2021 contest. Cryptology ePrint Archive, Paper 2022/385 (2022), `https://eprint.iacr.org/2022/385`

[2] Bardet, M., Faugère, J., Salvy, B.: On the complexity of the F5 gröbner basis algorithm. J. Symb. Comput. **70**, 49–70 (2015), `https://doi.org/10.1016/j.jsc.2014.09.025`

[3] Barthelemy, L.: 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)

[4] Billet, O., Gilbert, H., Ech-Chatbi, C.: Cryptanalysis of a white box AES implementation. In: Handschuh, H., Hasan, M.A. (eds.) Selected Areas in Cryptography, 11th International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004, Revised Selected Papers. Lecture Notes in Computer Science, vol. 3357, pp. 227–240. Springer (2004), `https://doi.org/10.1007/978-3-540-30564-4_16`

[5] Biryukov, A., Bouillaguet, C., Khovratovich, D.: Cryptographic schemes based on the ASASA structure: Black-box, white-box, and public-key (extended abstract). In: Sarkar, P., Iwata, T. (eds.) 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. Lecture Notes in Computer Science, vol. 8873, pp. 63–84. Springer (2014), `https://doi.org/10.1007/978-3-662-45611-8_4`

[6] Biryukov, A., Perrin, L.: On reverse-engineering s-boxes with hidden design criteria or structure. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 116–140. Springer (2015), `https://doi.org/10.1007/978-3-662-47989-6_6`

[7] Bock, E.A., Bos, J.W., Brzuska, C., Hubain, C., Michiels, W., Mune, C., Gonzalez, E.S., Teuwen, P., Treff, A.: White-box cryptography: Don't forget about grey-box attacks. J. Cryptol. **32**(4), 1095–1143 (2019), `https://doi.org/10.1007/s00145-019-09315-1`

[8] Bos, J.W., Hubain, C., Michiels, W., Teuwen, P.: Differential computation analysis: Hiding your white-box designs is not enough. In: Gierlichs, B., Poschmann, A.Y. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9813, pp. 215–236. Springer (2016), `https://doi.org/10.1007/978-3-662-53140-2_11`

[9] Bouillaguet, C., Faugère, J., Fouque, P., Perret, L.: Practical cryptanalysis of the identification scheme based on the isomorphism of polynomial with one secret problem. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6571, pp. 473–493. Springer (2011), `https://doi.org/10.1007/978-3-642-19379-8_29`

[10] Bringer, J., Chabanne, H., Dottax, E.: White box cryptography: Another attempt. IACR Cryptol. ePrint Arch. p. 468 (2006), `http://eprint.iacr.org/2006/468`

[11] Casanova, A., Faugère, J.C., Macario-Rat, G., Patarin, J., Perret, L., Ryckeghem, J.: GeMSS. Tech. rep., National Institute of Standards and Technology (2020), available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`

[12] Chow, S., Eisen, P.A., Johnson, H., van Oorschot, P.C.: White-box cryptography and an AES implementation. In: Nyberg, K., Heys, H.M. (eds.) Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers. Lecture Notes in Computer Science, vol. 2595, pp. 250–270. Springer (2002), https://doi.org/10.1007/3-540-36492-7_17

[13] Chow, S., Eisen, P.A., Johnson, H., van Oorschot, P.C.: A white-box DES implementation for DRM applications. In: Feigenbaum, J. (ed.) Security and Privacy in Digital Rights Management, ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002, Revised Papers. Lecture Notes in Computer Science, vol. 2696, pp. 1–15. Springer (2002), https://doi.org/10.1007/978-3-540-44993-5_1

[14] Delerablée, C., Lepoint, T., Paillier, P., Rivain, M.: White-box security notions for symmetric encryption schemes. In: Lange, T., Lauter, K.E., Lisonek, P. (eds.) Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8282, pp. 247–264. Springer (2013), https://doi.org/10.1007/978-3-662-43414-7_13

[15] Ding, J., Schmidt, D., Werner, F.: Algebraic attack on hfe revisited. In: Wu, T.C., Lei, C.L., Rijmen, V., Lee, D.T. (eds.) Information Security. pp. 215–227. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)

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

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

[18] Faugère, J., Macario-Rat, G., Patarin, J., Perret, L.: A new perturbation for multivariate public key schemes such as HFE and UOV. IACR Cryptol. ePrint Arch. p. 203 (2022), https://eprint.iacr.org/2022/203

[19] Feng, Q., He, D., Wang, H., Kumar, N., Choo, K.R.: White-box implementation of shamir's identity-based signature scheme. IEEE Syst. J. **14**(2), 1820–1829 (2020), https://doi.org/10.1109/JSYST.2019.2910934

[20] Fouque, P., Karpman, P., Kirchner, P., Minaud, B.: Efficient and provable white-box primitives. In: Cheon, J.H., Takagi, T. (eds.) 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. Lecture Notes in Computer Science, vol. 10031, pp. 159–188 (2016), https://doi.org/10.1007/978-3-662-53887-6_6

[21] Galissant, P.: Contributions to white-box cryptography : models and algebraic constructions. Theses, Université Paris-Saclay (Dec 2023), https://theses.hal.science/tel-04457255

[22] Galissant, P., Goubin, L.: Introduction to White-Box Cryptography. In: Embedded Cryptography, Vol. 3. Sciences Collection, John Wiley & Sons (2025)

[23] Goubin, L., Masereel, J., Quisquater, M.: Cryptanalysis of white box DES implementations. In: Adams, C.M., Miri, A., Wiener, M.J. (eds.) Selected Areas in Cryptography, 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers. Lecture Notes in Computer Science, vol. 4876, pp. 278–295. Springer (2007), https://doi.org/10.1007/978-3-540-77360-3_18

[24] Goubin, L., Paillier, P., Rivain, M., Wang, J.: How to reveal the secrets of an obscure white-box implementation. J. Cryptogr. Eng. **10**(1), 49–66 (2020), https://doi.org/10.1007/s13389-019-00207-5

[25] Goubin, L., Rivain, M., Wang, J.: Defeating state-of-the-art white-box countermeasures with advanced gray-box attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(3), 454–482 (2020), https://doi.org/10.13154/tches.v2020.i3.454-482

[26] Hosoyamada, A., Isobe, T., Todo, Y., Yasuda, K.: A modular approach to the incompressibility of block-cipher-based aeads. In: Agrawal, S., Lin, D. (eds.) 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. Lecture Notes in Computer Science, vol. 13792, pp. 585–619. Springer (2022), https://doi.org/10.1007/978-3-031-22966-4_20

[27] Karroumi, M.: Protecting white-box AES with dual ciphers. In: Rhee, K.H., Nyang, D. (eds.) Information Security and Cryptology - ICISC 2010 - 13th International Conference, Seoul, Korea, December 1-3, 2010, Revised Selected Papers. Lecture Notes in Computer Science, vol. 6829, pp. 278–291. Springer (2010), https://doi.org/10.1007/978-3-642-24209-0_19

[28] Lee, S., Kim, T., Kang, Y.: A masked white-box cryptographic implementation for protecting against differential computation analysis. IEEE Trans. Inf. Forensics Secur. **13**(10), 2602–2615 (2018), https://doi.org/10.1109/TIFS.2018.2825939

[29] Macario-Rat, G., Plût, J., Gilbert, H.: New insight into the isomorphism of polynomial problem IP1S and its use in cryptography. In: Sako, K., Sarkar, P. (eds.) 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. Lecture Notes in Computer Science, vol. 8269, pp. 117–133. Springer (2013), https://doi.org/10.1007/978-3-642-42033-7_7

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

[31] 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

[32] Matsumoto, T., Imai, H.: Public quadratic polynominal-tuples for efficient signature-verification and message-encryption. In: Günther, C.G. (ed.) Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings. Lecture Notes in Computer Science, vol. 330, pp. 419–453. Springer (1988), https://doi.org/10.1007/3-540-45961-8_39

[33] Matsumoto, T., Imai, H., Harashima, H., Miyakawa, H.: A class of asymmetric cryptosystems using obscure representations of enciphering functions. 1983 National Convention Record on Information Systems, IECE Japan (1983)

[34] Øygarden, M., Smith-Tone, D., Verbel, J.A.: On the effect of projection on rank attacks in multivariate cryptography. In: Cheon, J.H., Tillich, J. (eds.) Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20-22, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12841, pp. 98–113. Springer (2021), https://doi.org/10.1007/978-3-030-81293-5_6

[35] Patarin, J.: Hidden fields equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In: Maurer, U.M. (ed.) Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding. Lecture Notes in Computer Science, vol. 1070, pp. 33–48. Springer (1996), https://doi.org/10.1007/3-540-68339-9_4

[36] Patarin, J., Goubin, L., Courtois, N.T.: Improved algorithms for isomorphisms of polynomials. In: Nyberg, K. (ed.) Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding. Lecture Notes in Computer Science, vol. 1403, pp. 184–200. Springer (1998), https://doi.org/10.1007/BFb0054126

[37] Patarin, J., Macario-Rat, G., Bros, M., Koussa, E.: Ultra-short multivariate public key signatures. Cryptology ePrint Archive, Report 2020/914 (2020), https://eprint.iacr.org/2020/914

[38] Perret, L.: A fast cryptanalysis of the isomorphism of polynomials with one secret problem. In: Cramer, R. (ed.) Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3494, pp. 354–370. Springer (2005), https://doi.org/10.1007/11426639_21

[39] Petzoldt, A.: On the complexity of the hybrid approach on hfev-. IACR Cryptol. ePrint Arch. p. 1135 (2017), http://eprint.iacr.org/2017/1135

[40] Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings. Lecture Notes in Computer Science, vol. 196, pp. 47–53. Springer (1984), https://doi.org/10.1007/3-540-39568-7_5

[41] WhibOx Organizing Committee: Ches 2021 ctf challenge – whibox contest. https://whibox.io/contests/2021/ (2021)

[42] WhibOx Organizing Committee: Ches 2024 ctf challenge – whibox contest. https://whibox.io/contests/2024/ (2024)

[43] Xiao, Y., Lai, X.: A secure implementation of white-box aes. In: 2009 2nd International Conference on Computer Science and its Applications. pp. 1–6 (2009). https://doi.org/10.1109/CSA.2009.5404239

[44] Zhang, Y., He, D., Huang, X., Wang, D., Choo, K.R., Wang, J.: 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), http://search.ieice.org/bin/summary.php?id=e103-d_2_188