Sorbonne Université
Université de Picardie Jules Verne

Master of Computer Science
Cryptology, High-performance Computing, and Algorithms

Cryptanalysis of SBC-based Post-Quantum Signature Scheme

Graduation Internship Report
2025

Author:
Paul Mekhail
Sorbonne Université, Université de Picardie Jules Verne
`paul.mekhail4@pm.me`

Supervisor:
Sorina Ionica
Laboratoire MIS, Université de Picardie Jules Verne
`sorina.ionica@u-picardie.fr`

Academic Supervisor:
Mohab Safey El Din
LIP6, Sorbonne Université
`Mohab.Safey@lip6.fr`

Contents

1 Introduction

Since the emergence of quantum computing in the late 1990s, several breakthroughs have been achieved in the field of quantum algorithms. Some of these advancements have found applications in areas such as medicine and physics, while others have demonstrated the ability to break cryptographic problems that are currently assumed to be hard for classical computers, such as RSA [RSA78] and ECDH.

In particular, quantum algorithms like Grover's [Gro96] and Shor's [Sho97] have shown that existing public-key cryptosystems are vulnerable in a post-quantum setting. This has led to an urgent need for new hard problems on which to base quantum-resistant public-key cryptography.

Post-quantum cryptographic schemes are generally built upon four main problem classes: isogeny-based, multivariate, code-based, and lattice-based cryptography. In recent years, new paradigms have also been explored to improve the efficiency and security of schemes within these classes. One such paradigm is the *MPC-in-the-Head* approach.

Although these underlying problems are believed to be hard even for quantum computers, some have been shown to be vulnerable to classical attacks due to hidden structures. A notable example is the NIST finalist Rainbow [DS05], a multivariate signature scheme that was broken in 2022 by Beullens [Beu22] through exploitation of structural weaknesses.

One might argue that post-quantum cryptography is not yet necessary, given that large-scale quantum computers capable of breaking classical cryptography do not currently exist. However, adversaries could perform so-called *harvest-and-decrypt-later* attacks, in which encrypted data is collected today and decrypted in the future when quantum capabilities become available.

In response to these concerns, the National Institute of Standards and Technology (NIST) launched an international competition in 2016 to standardize post-quantum cryptographic primitives.

The *MPC-in-the-Head* (MPCitH) paradigm, first introduced in [Ish+07], has recently gained significant attention—particularly in the second round of NIST's post-quantum signature standardization process, where 5 out of 14 candidate schemes were based on this approach.

The core idea of this paradigm is to leverage an NP-relation $\mathcal{R}(x, w)$ to construct a zero-knowledge (ZK) protocol. In such a protocol, a prover $\mathcal{P}$ convinces a verifier $\mathcal{V}$ that they possess a valid witness w for a given public input x, without revealing any additional information about w.

The main appeal of the MPCitH paradigm lies in the fact that it enables the construction of efficient, non-interactive, and quantum-resistant signature schemes with relatively small signature sizes and low verification cost and extremely small keys size.

In [HJ23], the authors introduced the *Subfield Bilinear Collision* (SBC) problem, which is rooted in the hardness of the discrete logarithm problem [Jou13]. Based on this assumption, they proposed a post-quantum signature scheme constructed using the MPCitH framework.

The objective of this internship is to conduct a cryptanalysis of this scheme, assess its security and efficiency, and explore potential improvements. This includes evaluating its concrete security parameters and identifying potential structural weaknesses.

In Section 2, we introduce the MPCitH paradigm and provide an overview of the SBC problem, including its variants as well as the identification and signature schemes built upon it.

In Section 3, we present key results in polynomial system solving and describe the main algorithms and results used in our cryptanalysis.

Finally, in Section 4, we detail our contributions, including the cryptanalysis and formal security evaluation of the SBC-based signature scheme, and summarize the most efficient known attacks against it.

2 MPC-in-the-Head and SBC signature scheme

2.1 MPCitH

The *MPC-in-the-Head* (MPCitH) framework, introduced by [Ish+07], is a technique for constructing zero-knowledge proofs. It enables a prover $\mathcal{P}$ to convince a verifier that they possess a secret witness w for a given public input x, without revealing any information about the witness itself.

Let $\mathcal{L}$ be a language in NP (i.e. $x \in \mathcal{L} \iff \exists w, \mathcal{R}(x, w) = 1$). $\mathcal{R}$ is a NP-relation. Given a public x, $\mathcal{P}$ wants to prove that he knows w such that: $\mathcal{R}(x, w)$.

The main idea is to simulate a secure multiparty computation (MPC) protocol internally *in the head* of the prover $\mathcal{P}$. This simulation involves a set of fictitious parties who jointly evaluate a circuit that depends on the witness w. The verifier $\mathcal{V}$ is then allowed to inspect a subset of the simulated views to ensure the prover is behaving honestly, while maintaining the zero-knowledge property.

We introduce some standard notations in MPCitH protocols.

- $[w]$, w is a shared value.
- $w^{[\![i]\!]}$ is the i-th share of w.
- $w^{[e]}$ is a value used in the e-th round of the protocol.
- δ_w is the offset for the additive sharing of w s.t. $w = \delta_w + \sum_i w^{[\![i]\!]}$

To represent the secret witness w, a linear secret sharing scheme is used among N simulated parties. Each party i receives a random share denoted by $w^{[\![i]\!]}$, and the prover computes an additional correction term δ_w.

This ensures that the full witness can be correctly reconstructed from the shares and the correction term during the simulated execution.

Next, the prover $\mathcal{P}$ commits to the views of the N fictitious parties, denoted as $View_i$, and sends these commitments to the verifier $\mathcal{V}$. Each view contains all messages sent and received by the corresponding party during the simulated execution of the MPC protocol.

The verifier $\mathcal{V}$ then selects a random index $i \in [1, N]$ and sends it to $\mathcal{P}$. In response, $\mathcal{P}$ opens the views of all parties except the i-th one, i.e., for all $i' \in [1, N]$ such that $i' \neq i$.

In order for this protocol to satisfy the zero-knowledge property, the underlying MPC protocol must be $(N-1)$-private, meaning that the joint view of any $N-1$ parties reveals no information about the secret witness w.

Finally, we discuss several optimizations that have been proposed in the literature since and are now considered standard and widely used.

2.1.1 GGM tree

Since most of the values used in the MPC protocol are randomly generated, a technique known as the GGM tree can be employed to transmit only a seed, rather than all the individual random values. This technique has been introduced in [GGM86].

A GGM tree is essentially a binary tree rooted at a seed, on which a pseudorandom function (PRF) is applied. The PRF must produce an output of twice the bit-length of its input. The first half of the output serves as the root of the left subtree, and the second half as the root of the right subtree. This process is applied recursively until enough leaves are generated, which correspond to the required random values.

In our case, we do not transmit the seed due to the use of linear secret sharing. Since we typically employ an $(N-1)$-private MPC protocol, we only share the co-path of the i-th leaf, which remains hidden from $\mathcal{V}$. As a result, instead of sharing N values, we only need to share $log(N)$ values.

Usually, the linear private share of a value x is done this way:

1. we create n random shares $x^{[\![i]\!]} \xleftarrow{\$} \mathbb{F}_q, \forall i \in [1, N]$.
2. we compute the offset $\delta_x = x - \sum_{i=1}^{N} x^{[\![i]\!]} \bmod \mathbb{F}_q$.

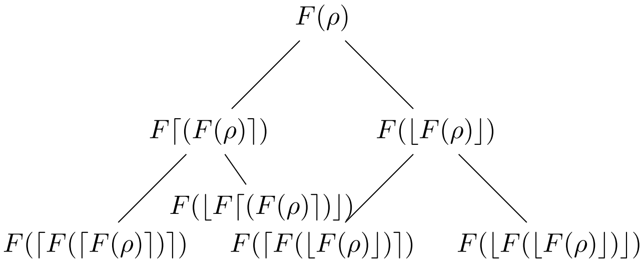
$F(\rho)$
$F\lceil(F(\rho)\rceil)$
$F(\lfloor F(\rho)\rfloor)$
$F(\lfloor F\lceil(F(\rho)\rceil)\rfloor)$
$F(\lceil F(\lceil F(\rho)\rceil)\rceil)$
$F(\lceil F(\lfloor F(\rho)\rfloor)\rceil)$
$F(\lfloor F(\lfloor F(\rho)\rfloor)\rfloor)$

Figure 1: GGM tree example to secretly share a value x using a PRF F. $\lceil F(\rho)\rceil$ (resp. $\lceil F(\rho)\rceil$) denotes the first (resp. second) half of the output of $F(\rho)$.

One challenge in this approach is the need to communicate the offset to the verifier so that they can verify the computations. However, a recent technique called *Correlated GGM* (cGGM) has been introduced in [Guo+22], which allows the prover to generate random values $x^{[\![i]\!]}$ using a modified GGM tree, such that $\sum_{i=1}^{N} x^{[\![i]\!]} = x$, with no computational overload. This is achieved by enforcing an invariant over the cGGM tree: at each level, the sum of the nodes remains constant. More precisely, given a parent node x, the left child is defined as $F(x)$ and the right child as $x - F(x)$, where F is a cryptographic PRF. As a result, only approximately half of the tree nodes require PRF evaluations, compared to all nodes in the classical GGM construction. This technique not only reduces the computational cost but also eliminates the need to transmit the offset to the verifier, thereby improving the overall communication efficiency.

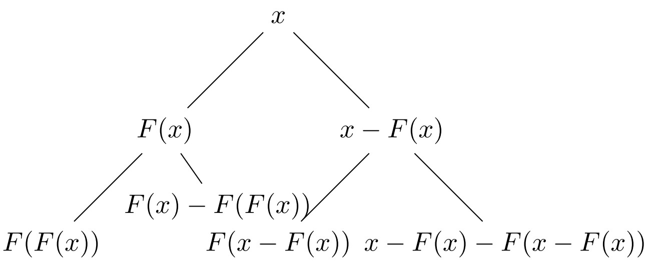

Figure 2: cGGM tree example. Here, it is easy to verify that the sum at each level is equal to x.

In practice, most MPCitH schemes use hash functions with a salt value but the size of the entry must be 2λ to achieve 2^λ security bits. A recent work by Bui et al. studied the security of using the AES as the PRF. specially because of it's hardware implementation which enables much faster tree generation than other PRF like SHA256 or other hash functions. The security of AES as a cryptographic PRF in a GGM tree generation has been studied in [Bui+24]. Thanks to AES hardware implementation like Intel's AES-NI instruction set, schemes can be up to 50x times faster than versions using hash functions like SHA256.

2.1.2 HyperCube technique

This optimization technique was introduced in [AM+22] and is commonly used in MPCitH schemes. This geometric approach models the N shares as forming a d-dimensional hypercube of side length k. The often used parameters are $k = 2$ and $N = 2^d$, so the number party counts reduces to $log(N) + 1$. This technique often leads to MPC computations less expensive, smaller proof sizes and larger party sizes. However, these benefits come at the cost of increased reliance on GGM tree generation, which can become computationally expensive.

2.2 Subfield Bilinear Collision Problem

The problem is the following: let q be a prime power and two positive integers k, n.
Given two vectors $\vec{u}, \vec{v} \in (\mathbb{F}_{q^k})^n$, which are linearly independent over $\mathbb{F}_q$, find two non-colinear vectors $\vec{x}, \vec{y} \in (\mathbb{F}_q)^n$ such that

$$(\vec{u} \cdot \vec{x})(\vec{v} \cdot \vec{y}) = (\vec{u} \cdot \vec{y})(\vec{v} \cdot \vec{x}),$$

where $(\cdot)$ is the dot product between 2 vectors.

We denote an instance of the SBC problem given by two vectors $\vec{u}, \vec{v} \in (\mathbb{F}_{q^k})^n$ by SBC$[\vec{u}, \vec{v}]$. If the vectors $\vec{x}, \vec{y} \in (\mathbb{F}_q)^n$ are solution of SBC$[\vec{u}, \vec{v}]$, we use the notation $(\vec{x}, \vec{y}) \in$ SBC$[\vec{u}, \vec{v}]$.

Normalized Subfield Bilinear Collision variant. The authors also introduce a particular version of this problem called normalized SBC (NSBC) where $\vec{x}, \vec{y} \in (\mathbb{F}_q)^n$ and $\vec{x} = (\vec{x'}, 1, 0), \vec{y} = (\vec{y'}, 0, 1)$ to ensure that they are not colinear without needing to regenerate each time they are not.

We also denote $(\vec{x}, \vec{y}) \in$ NSBC$[\vec{u}, \vec{v}]$ if $(\vec{x}, \vec{y})$ are solutions to the NSBC problem on $(\vec{u}, \vec{v})$. **Key generation**. The private key $(\vec{x}, \vec{y})$ is constructed as follows: uniformly at random choose $\vec{x'}, \vec{y'} \in (\mathbb{F}_q)^{n-2}$ and set $\vec{x} = (\vec{x'}, 1, 0)$ and $\vec{y} = (\vec{y'}, 0, 1)$. For the public key, randomly choose $\vec{u} \in (\mathbb{F}_{q^k})^n$ and $v_1, \ldots, v_{n-1} \in \mathbb{F}_{q^k}$. Finally, set v_n as:

$$v_n = \frac{(\vec{u} \cdot \vec{y})(\sum_{i=1}^{n-1} v_i x_i) - (\vec{u} \cdot \vec{x})(\sum_{i=1}^{n-1} v_i y_i)}{y_n(\vec{u} \cdot \vec{x})}.$$

2.3 Keys and parameters estimations

The public key of this signature scheme is $(\vec{u}, \vec{v})$ and the private key is $(\vec{x}, \vec{y})$. The signature and verification protocol use the MPCitH paradigm, they are described in detail in [HJ23] with proof of soundness, correctness and zero-knowledge. The authors estimate that a good parameter for NSBC problem to be computationally secure are $q = 2, n = 130, k = 257$ using complexity analysis of what they think is the best known cryptanalysis of such problem on a classical computer in [FSS11] with $\mathcal{O}(2^{354})$ operations. For the quantum cryptanalysis the estimate that the best attack would be using Grover's algorithm [Gro96] with $\mathcal{O}(n^{128})$ operations.

2.4 Multiparty computation protocol for the SBC problem

A Multi-Party Computation (MPC) protocol is an interactive protocol between N parties $P_1, \ldots, P_N$, who wish to jointly compute a function f over their private inputs $x^{[\![i]\!]}$. At the end of the protocol, each party outputs its own computed value, denoted $f_i(x) = f(x^{[\![i]\!]})$.

Several adversarial models exist for MPC protocols. In the *semi-honest* (or passive) model, parties are assumed to follow the protocol honestly but may attempt to infer secret information from the messages they receive. In contrast, the *malicious* (or active) model allows adversaries to arbitrarily deviate from the protocol, including corrupting other parties, in order to compromise security.

In the context of the MPC-in-the-Head (MPCitH) paradigm, we consider only the semi-honest model. This is because the parties are simulated *in the head* of the prover, and the only external entity is the verifier, who may attempt to extract the witness from the simulated communications between the parties.

We recall here the definition of a secure MPC protocol in the semi-host threat model.

Definition 2.1 (def 2.4.5 [Fen23])**.** *Let N and t be integers such that $1 \leq t \leq N$. Let Π_f be an MPC protocol with N parties $P_1, \ldots, P_N$, computing a function f. The protocol Π_f is t-private in the semi-honest model if for all $I \subset [1, N]$ such that $|I| \leq t$, there exists a PPT algorithm $\mathcal{S}$ such that $\mathcal{S}(I, [\![x]\!]_I, f_I(x))$ is perfectly indistinguishable from the joint distribution of the views of the parties in I, where $f_I(x) := f_i(x), \forall x \in I$.*

Thanks to the linear sharing, we can easily apply the following operations on the shares:

- **Addition**: each party P_i can locally compute $[\![x+y]\!]$ by adding their shares:

  $$(x+y)^{[\![i]\!]} = x^{[\![i]\!]} + y^{[\![i]\!]} \ \forall i \in [1, N].$$

  We denote this by $[\![x+y]\!] = [\![x]\!] + [\![y]\!]$.

- **Adding a constant**: we can easily share the constant α among the parties as $[\![\alpha]\!] = (\alpha, 0, \ldots, 0)$. The parties can then compute $[\![x+\alpha]\!] = [\![x]\!] + [\![\alpha]\!]$. We denote this by $[\![x]\!] + \alpha$.

- **Multiplication by a constant**: each party P_i can locally compute $[\![\alpha x]\!]$ by multiplying their respective share by α:

  $$(\alpha x)^{[\![i]\!]} = \alpha x^{[\![i]\!]} \ \forall i \in [1, N].$$

  We denote this by $[\![\alpha x]\!] = \alpha [\![x]\!]$.

Finally, we need a function that we will compute using a MPC protocol to prove that the witness $(\vec{x}, \vec{y}) \in \mathrm{NSBC}(\vec{u}, \vec{v})$.

Let $X_1, X_2, Y_1, Y_2 \in \mathbb{F}_{q^k}$ be random values. Consider the following polynomial in t:

$$\begin{aligned} F(t) &= (X_1 + t(\vec{u} \cdot \vec{x}))(X_2 + t(\vec{v} \cdot \vec{x})) - (Y_1 + t(\vec{v} \cdot \vec{y}))(Y_2 + t(\vec{u} \cdot \vec{y})) \\ &= X_1 Y_1 - X_2 Y_2 \\ &+ [X_1(\vec{v} \cdot \vec{y}) + Y_1(\vec{u} \cdot \vec{x}) - X_2(\vec{u} \cdot \vec{y}) - Y_2(\vec{v} \cdot \vec{x})]t \\ &+ [(\vec{u} \cdot \vec{x})(\vec{v} \cdot \vec{y}) - (\vec{u} \cdot \vec{y})(\vec{v} \cdot \vec{x})]t^2. \end{aligned}$$

We notice here that the component of degree 2 is equal to zero and that $deg(F(t)) < 2$ if and only if $(\vec{x}, \vec{y}) \in \mathrm{NSBC}(\vec{u}, \vec{v})$. In that case, we rewrite $F(t)$ as:

$$F(t) = A + Bt$$

with $A = X_1Y_1 - X_2Y_2$ and $B = [X_1(\vec{v} \cdot \vec{y}) + Y_1(\vec{u} \cdot \vec{x}) - X_2(\vec{u} \cdot \vec{y}) - Y_2(\vec{v} \cdot \vec{x})]t$.

Then we compute the values of A and B and share them among the parties and prove that $F(t) = A + Bt$ which means that $(\vec{x}, \vec{y}) \in \mathrm{NSBC}(\vec{u}, \vec{v})$.

To do so we apply the following protocol:

1. The parties runs an auxiliary protocol to compute a random value $t_0 \in \mathbb{F}_{q^k}$.
2. The parties evaluate $F(t_0)$.
3. The parties evaluate $A + Bt_0$.
4. The parties output *accept* if and only if $F(t_0) = A + Bt_0$ and *reject* otherwise.

The evaluations can be efficiently performed by the parties, thanks to the linear sharing of the coefficients A, B, and the secret $(\vec{x}, \vec{y})$. For a detailed description of the underlying computations, we refer the reader to [HJ23].

One can notice that the evaluations may also coincide if t_0 is a root of the polynomial

$$(A + Bt) - F(t).$$

A malicious party P_i could exploit this by waiting to receive all the shares of $\llbracket t_0 \rrbracket$ from the other parties, and then choosing their own share $t_0^{\llbracket i \rrbracket}$ such that

$$t_0 = \sum_{j \neq i} t_0^{\llbracket j \rrbracket} + t_0^{\llbracket i \rrbracket}$$

is a root of the polynomial. This would allow P_i to bias the evaluation towards a value that satisfies the equation, which could compromise the protocol by causing it to output *accept* even though $(\vec{x}, \vec{y}) \notin \mathrm{NSBC}(\vec{u}, \vec{v})$.

To mitigate this issue, parties will not be allowed to choose their share of $\llbracket t_0 \rrbracket$ arbitrarily. Instead, each share $t_0^{\llbracket i \rrbracket}$ will be deterministically generated as the output of a hash function applied to a fixed, party-specific string concatenated with a common seed and something inherit to the party like their shares. Specifically, the share is computed as:

$$t_0^{\llbracket i \rrbracket} = \mathcal{H}(\text{"}t_0 \text{ value for party } i\text{"} \parallel \mathrm{seed} | x^{\llbracket i \rrbracket}, y^{\llbracket i \rrbracket}, X_1^{\llbracket i \rrbracket}, X_2^{\llbracket i \rrbracket}, Y_1^{\llbracket i \rrbracket}, Y_2^{\llbracket i \rrbracket}, A^{\llbracket i \rrbracket}, B^{\llbracket i \rrbracket}),$$

where $\mathcal{H}$ is a cryptographic hash function and $\parallel$ denotes concatenation. This ensures that the shares are unpredictable yet deterministic, thereby preventing any party from biasing the outcome.

However, since the value of t_0 is truly random, there remains a non-zero probability that it is a root of the polynomial. This probability is upper-bounded by $\frac{2}{q^k}$, which is negligible for appropriately chosen parameters.

2.5 Identification Protocol

We present the identification protocol, which serves as the foundation for constructing the signature scheme via the Fiat–Shamir transformation.

To do so, we transform the MPC protocol into an interactive ZKP of knowledge of the witness $(\vec{x}, \vec{y})$ such that it is a solution of $\mathrm{NSBC}[\vec{u}, \vec{v}]$.

The prover $\mathcal{P}$ simulates the MPC protocol *in their head* and commits to the views of each party $h = \mathcal{H}_V(View_1, \dots, View_N)$. The view of each party consists of all the messages they send and receive during the protocol. In addition, $\mathcal{P}$ commits to the following values:

$$X_1 + t_0(\vec{u} \cdot \vec{x}), \quad X_1 + t_0(\vec{v} \cdot \vec{x}), \quad Y_1 + t_0(\vec{v} \cdot \vec{y}), \quad Y_2 + t_0(\vec{u} \cdot \vec{y}), \quad A + Bt_0.$$

Then, the verifier $\mathcal{V}$ selects a random index $i^* \in [1, N]$. The prover $\mathcal{P}$ opens all the shares of the following values for all $i \in [1, N]$ such that $i \neq i^*$:

$$x^{[\![i]\!]}, \quad y^{[\![i]\!]}, \quad X_1^{[\![i]\!]}, \quad X_2^{[\![i]\!]}, \quad Y_1^{[\![i]\!]}, \quad Y_2^{[\![i]\!]}, \quad A^{[\![i]\!]}, \quad B^{[\![i]\!]}.$$

Using this information, $\mathcal{V}$ can reconstruct the shares of $t_0^{[\![i]\!]}$ as previously described. Moreover, they can compute the corresponding shares of the committed values:

$$(X_1{+}t_0(\vec{u}{\cdot}\vec{x}))^{[\![i]\!]}, \quad (X_1{+}t_0(\vec{v}{\cdot}\vec{x}))^{[\![i]\!]}, \quad (Y_1{+}t_0(\vec{v}{\cdot}\vec{y}))^{[\![i]\!]}, \quad (Y_2{+}t_0(\vec{u}{\cdot}\vec{y}))^{[\![i]\!]}, \quad (A{+}Bt_0)^{[\![i]\!]}, \quad \forall i \neq i^*.$$

Since the sharing is linear, the verifier can efficiently recover the missing shares corresponding to i^*. For the detailed description of these computations, we refer the reader to [HJ23].

Finally, the verifier $\mathcal{V}$ can compute $h' = \mathcal{H}_V(View_1, \ldots, View_N)$ and compare with h. If they match, they *accept* and *reject* otherwise.

We recall that, to reduce the communication cost, we use correlated GGM (cGGM) trees and transmit only the co-paths of the tree instead of all values. Additionally, the HyperCube technique is employed to reduce the computational overhead.

To convert the identification scheme into a signature scheme, we apply the Fiat-Shamir transformation by replacing the verifier's random choice of $i*$ with the output of a hash function.

The soundness of the protocol is given by $\frac{1}{N} + \frac{1}{q^k}$. This is because the underlying MPC protocol is $(N-1)$-private, so the prover $\mathcal{P}$ can potentially cheat on only one view, and the verifier catches them with probability $\frac{1}{N}$. Furthermore, we have shown that a false positive may occur when t_0 happens to be a root of a certain polynomial, which occurs with probability $\frac{1}{q^k}$.

Rather than increasing the number of parties N, which would increase computational cost, we repeat the protocol τ times such that:

$$\left(\frac{1}{N} + \frac{1}{q^k}\right)^\tau < 2^{-\lambda},$$

where λ is the targeted security level.

2.6 Keys and signature sizes

The size of the private key is just the size of the seed that is used to generate it which is derived from a PRF. So we only need to store the seed which is of size λ. So the size of the private key is 16 bytes in practice with the proposed parameters. For the public key, we saw that all of it is generated randomly using a PRF and v_n is computed. The size of v_n is 2λ so the is size of public key is 3λ which in practice with the proposed parameters is 48 bytes.

The signature is composed of the following:

- One hash value h corresponding to a global commitment to the τ round protocol of size 2λ and one salt of size λ.
- One punctured PRF key for each round.
- The valued $X_1 + t_0(\vec{u} \cdot \vec{x}), X_2 + t_0(\vec{y} \cdot \vec{x})Y_1 + t_0(\vec{v} \cdot \vec{y}), Y_2 + t_0(\vec{u} \cdot \vec{y}) \in \mathbb{F}_{q^k}$ for each round.

The total size of the signature in bits is therefore

$$\tau\lambda log_2(N) + 10\tau\lambda + 3\lambda = \lambda^2 + 10\tau\lambda + 3\lambda \text{ bits},$$

where we use the fact that $\lambda = \tau log_2(N)$ in the equation. We remind that we do not need to communicate the auxiliary values $\delta_A, \delta_B, \vec{\delta}_x$ and $\vec{\delta}_y$ if we use a correlated GGM tree.

Remark: the actual sizes depend heavily on the machine architecture. For instance, on 64-bit architectures, the machine word size is 64 bits, meaning that data structures are typically aligned to multiples of 64 bits. As a result, for the parameters $k = 257$, $q = 2$, and $n = 130$, the signature size becomes 4023 bytes instead of the theoretical 3376 bytes due to alignment and padding overhead.

The authors also provide some practical benchmarks to evaluate the performance of their scheme in realistic settings:

			SBC cGGM$_{\text{SHA}}$		SBC cGGM$_{\text{AES}}$		SBC$_{\text{FF}}$ cGGM$_{\text{AES}}$	
D	τ	|sign|	Sign	Verify	Sign	Verify	Sign	Verify
8	16	5950 B	9.34 ms	9.23 ms	0.95 ms	0.85 ms	0.91 ms	0.81 ms
9	15	5822 B	16.88 ms	16.75 ms	1.15 ms	1.04 ms	1.06 ms	0.96 ms
10	13	5260 B	28.62 ms	28.58 ms	1.38 ms	1.29 ms	1.22 ms	1.13 ms
11	12	5051 B	52.33 ms	52.25 ms	1.95 ms	1.86 ms	1.61 ms	1.51 ms
12	11	4810 B	95.44 ms	95.15 ms	3.03 ms	2.94 ms	2.31 ms	2.22 ms
13	10	4537 B	173.44 ms	173.51 ms	5.27 ms	5.18 ms	3.64 ms	3.52 ms
15	9	4376 B	628.41 ms	627.91 ms	22.77 ms	22.54 ms	11.71 ms	11.46 ms
16	8	4023 B	1116.88 ms	1117.77 ms	40.39 ms	40.09 ms	20.94 ms	20.59 ms

Table 1: Running times for $\lambda = 128$ using $N = 2^D$ parties and τ rounds with the parameters $q = 2, k = 257$ and $n = 130$ for basic SBC comparing different version of the cGGM tree using an AMD EPYC 8374F processor running at 3.85 GHz. cGGM$_{\text{AES}}$ denotes the usage of AES as a PRF for the cGGM and cGGM$_{\text{SHA}}$ denotes the usage of the hash function SHA.

We also ran these benchmarks on a consumer grade processor and the results were approximately the double of the ones by the authors of [HJ23].

The FF notation refers to an optimization introduced by the authors in [HJ23], known as *Fast Fold*, which is applied during the GGM tree generation. While we do not describe this optimization in detail here, we refer the interested reader to [HJ23] for a comprehensive explanation.

3 Polynomial system solving

Gröbner basis theory was first introduced by Buchberger in his thesis [Buc06] and he gave an algorithm to compute the Gröbner basis of an ideal. Gröbner basis algorithms like Buchberger's Algorithm [Buc06], are used to find solutions for polynomial systems using algebraic geometry tools. For a gentle introduction to this subject [CLO10].

In [Laz83], Lazard introduced the connection between linear algebra and Gröbner basis theory via the usage of Macaulay matrices which is defined as follows.

Definition 3.1. *Let $\prec$ be an admissible monomial ordering on a given ring R. Given a sequence of homogeneous (resp. affine) polynomials $\mathcal{F} = (f_1, \ldots, f_m) \in R$, we associate to it the Macaulay matrix of degree D (resp. $\leq D$), denoted by $Mac_{D,m}(\mathcal{F})$ (resp. $Mac_{\leq D,m}(\mathcal{F})$), and defined as follows: the columns of the matrix are indexed by the monomials of degree D (resp. $\leq D$) sorted in decreasing order from the left to right using the defined monomial ordering. Each row of the matrix is labeled by a tag, also called signature, (u, f_i) where u is a monomial in R and $f_i \in \mathcal{F}$ such that $deg(uf_i) = D$ (resp. $deg(uf_i) \leq D$), and the polynomial uf_i is written as vector of coefficients of monomials on the row. We denote by $\widetilde{Mac}_{D,m}(\mathcal{F})$ (resp. $\widetilde{Mac}_{\leq D,m}(\mathcal{F})$) the row* *echelon form of $Mac_{D,m}(\mathcal{F})$ (resp. $Mac_{\leq D,m}(\mathcal{F})$) without swapping columns to preserve the monomial ordering.*

3.1 Faugère's F5

The major "issue" of Gröbner basis algorithms like Bucherberger's [Buc06], F4 [Fau02] are the reductions to 0 in the Macaulay matrix associated to a polynomial sequence $Mac_{D,m}(\mathcal{F})$, because the size of the matrices are huge, multiple millions rows and columns for practical examples, reductions to 0 can cost a lot during the linear algebra process. F5 [Fau02] was a major improvement because it provided a criterion that avoided reduction to 0 during the linear algebra part for all regular sequences.

Proposition 3.1 (General criterion [Fau02])**.** *Let d_i be the total degree of $f_i \forall i \in [1, m]$. $\forall j < m$, if a row of signature (u, f_j) in the matrix $\widetilde{Mac}_{D-d_j,m-1}(\mathcal{F})$ has as leading term t', then the row (t', f_m) in $Mac_{D,m}(\mathcal{F})$ (resp. $Mac_{\leq D,m}(\mathcal{F})$) will reduce to 0 i.e. is a linear combination of the predecessors.*

Proposition 3.2 (Frobenius criterion [Fau02])**.** *If a row of signature (t, f_m) in $\widetilde{Mac}_{D-d_m,m}(\mathcal{F})$ has as leading term t', then the row (t', f_m) in $Mac_{D,m}(\mathcal{F})$ (resp. $Mac_{\leq D,m}(\mathcal{F})$) will reduce to 0.*

The Frobenius criterion 3.2 is the analogue of the General criterion 3.1 but for polynomial systems defined over $\mathbb{F}_2$. F5 criterion and Frobenius criterion remove the rows corresponding to the trivial syzygies, i.e. the syzygies $(s_1, \ldots, s_m)$ such that for a row with signature (s_i, f_i), we get $s_i \in \langle f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_m \rangle$.

Bardet introduced a variant of F5 Algorithm that is simple to understand and analyze called Matrix F5 in [BTB04]. We give the description of Matrix F5 Algorithm.

3.2 Hilbert series

Hilbert-Poincaré series (or Hilbert series) is a very important tool in algebraic geometry to describe ideals, varieties and their dimensions.

Definition 3.2 ([CLO10])**.** *Let I be a homogeneous ideal in R. The affine Hilbert function of I is the function on the nonnegative integers is defined by*

$$^{a}HF_{R/I}(s) = dimR_s/dimI_s = dimR_s - dimI_s.$$

Proposition 3.3. *$^{a}HF_{R/I}(s)$ is the number of monomials not in I of total degree s.*

Proof. First note that the monomials $x^\alpha | \alpha \leq s$ is a basis of $R_{\leq s}$ as a vector space. We also have that $x^\alpha | \alpha \leq s | x^\alpha \in I$ is a basis of $I_{\leq s}$. Therefore, the monomials $x^\alpha | \alpha \leq s | x^\alpha \notin I$ are the ones missing to form a basis for R, so they form a basis for $R_{\leq s}/I_{\leq s}$ which concludes the proof. □

Definition 3.3. *Let I be a homogeneous ideal. The Hilbert series is defined as follows*

$$HS_{R/I}(t) = \sum_{m=0}^{\infty} HF_{R/I}(m)t^m.$$

The index of regularity is the smallest integer s_0 such that $HF_{R/I}(s_0)$ is equal to a certain polynomial called the Hilbert polynomial. It is also the smallest integer s_0 such that the coefficient of the Hilbert series is less or equal to 0. We will denote $H(I)$ as the index or degree of regularity of I.

Faugère showed that no reductions to 0 occur in the Macaulay matrix during the F5 algorithm 1 until the degree of regularity [Fau02] for regular sequences.

Algorithm 1 Matrix F5

```
Require: { $m$ homogeneous polynomials $f_1, \dots, f_m$ of degree $d_1 \leq d_2 \leq \dots \leq d_m$,
         { $D$ an integer,
         { a monomial ordering $\prec$
Ensure: { $G$ is a Gröbner basis of $\langle f_1, \dots, f_m \rangle$.
$G \leftarrow f_1, \dots, f_m$
for $d$ from $d_1$ to $D$ do
    $\widetilde{Mac}_{d,0} \leftarrow$ matrix with 0 rows
    for $i$ from 1 to $m$ do
        Construct $Mac_{d,i}$ by adding to $\widetilde{Mac}_{d,i-1}$ the following rows:
        if $d_i = d$ then
            Add the row $f_i$ with signature $(1, f_i)$
        end if
        if $d > d_i$ then For all $f$ from $\widetilde{Mac}_{d,i-1}$ with signature $(e, f_i)$, such that $x_\lambda$ is the greatest variable of $e$, add the $n - \lambda + 1$ rows $x_\lambda f, x_{\lambda+1} f, \dots, x_n f$ with the signatures $(x_\lambda e, f), (x_{\lambda+1} e, f), \dots, (x_n e, f)$ except those who satisfy the F5 criterion with $((x_{\lambda+k} e, fi), \widetilde{Mac}_{d-d_i,i-1}$
        end if
        Compute $\widetilde{Mac}_{d,i}$ the row echelon form of $Mac_{d,i}$
        Add to $G$ the polynomials corresponding to the rows of $\widetilde{Mac}_{d,i}$ such that their leading
        monomials is different from the leading monomial of the row with the
        same signature in $Mac_{d,i}$
    end for
end for
Return $G$
```

3.3 Semi-regularity

One problem with the definition of a regular sequence, is that it applies for only underdetermined polynomial systems because on of the properties of a regular sequence of a polynomial system $f_1, \dots, f_m \in \mathbb{K}[x_1, \dots, x_n]$ is that the dimension of the ideal $\langle f_1, \dots, f_m \rangle$ is $n - m$. This property can not be verified for overdetermined systems because the dimension of the ideal can not be negative.

Thus, Bardet studied a new concept called semi-regularity for overdetermined system to allow their study and analysis. The definition of a semi-regular sequence is given by Bardet [BTB04].

Definition 3.4. *A zero-dimensional homogeneous sequence $f_1, \dots, f_m \subset R$ is semi-regular if the following conditions are verified:*

- *$I = \langle f_1, \dots, f_m \rangle \neq R$,*
- *For $i \in [1 : m]$, if $g_i f_i = 0$ in $R/\langle f_1, \dots, f_m \rangle$ and deg$(g_i f_i) < H(I)$ then $g_i = 0$ in $R/\langle f_1, \dots, f_m \rangle$.*

Bardet also showed that if we apply matrix-F5 1 to a semi-regular system, there are no reductions to 0 until the degree $d = H(I) - 1$. We also have a certain reciprocity.

One interesting thing shown by Bardet [BTB04], is that a random polynomial system is very likely to be semi-regular. Most of the cryptosystems using random polynomial systems assume the semi-regularity of the system to ease their analysis using well established tools. For example, one can compute the Hilbert series of a semi-regular system easily.

Proposition 3.4 (Proposition 3.2.5 in [BTB04])**.** *Let $f_1, \dots, f_m$ a semi-regular homogeneous sequence, f_i is of degree d_i. Then the following properties are verified:*

- *The sequence $f_1, \dots, f_m$ is semi-regular if and only if its Hilbert series is the series*

  $$S_{m,n}(z) = \sum_{d \geq 0} h_{d,m}(n) z^d = \prod_{i=1}^{m} (1 - z^{d_i})/(1 - z)^n.$$

  $S_{m,n}(z)$ is called the generating series of $f_1, \dots, f_m$.

- *The sequence $f_1, \dots, f_m$ is semi-regular on $\mathbb{F}_2$, if and only if its Hilbert series is the series*

  $$S_{m,n}(z) = \sum_{d \geq 0} h_{d,m}(n) z^d = (1 + z)^n / \prod_{i=1}^{m} (1 + z^{d_i}).$$

  $S_{m,n}(z)$ is called the generating series of $f_1, \dots, f_m$.

- *Every permutation $f_{\sigma(1)}, \dots, f_{\sigma(m)}$ is a semi-regular sequence.*
- *When the associated variety is zero-dimensional, $H(I)$ is characterized by*

  $$\forall d < H(I), h_{d,m}(n) > 0 \text{ and } h_{H(I),m}(n) \leq 0.$$

- *The sequence $f_1, \dots, f_m$ is semi-regular $\not\Rightarrow \forall i \in [1; m]$, the sequence $f_1, \dots, f_i$ is semi-regular.*

3.4 Bilinear systems case

Our NSBC instances are quadratic systems that are bilinear in the variables of degree 2 and overdetermined.

Definition 3.5. *A homogeneous polynomial $f \in \mathbb{K}[x_1, \ldots, x_{n_x}, y_1, \ldots, y_{n_y}]$ is called bilinear if we have:*

$$\forall \lambda, \mu \in \mathbb{K}, f(\lambda x_1, \ldots, \lambda x_{n_x}, \mu y_1, \ldots, \mu y_{n_y}) = \lambda \mu f(x_1, \ldots, x_{n_x}, y_1, \ldots, y_{n_y}).$$

A bilinear system is a polynomial system where each equation is a bilinear form.

We will denote n_x (resp. n_y) the number of variables in x (resp. in y). In our case, $n_x = n_y = (m-1)/2$ and $m = k$.

The article [FSS11] and the thesis [Spa12] provide a new criterion for bilinear determined systems and a complexity analysis of their algorithm (the proofs in the end of [FSS11] are not exact, the authors recommended to refer to [Spa12] for better proofs) that removes all reductions to 0 in F5 Algorithm for bilinear systems which are not semi-regular.

3.4.1 Jacobian matrices and syzygies

First we will need to introduce some important notations.

- Let $f_1, ..., f_m \in R$ be a bilinear polynomial. We denote by F_i the polynomial sequence $(f_1, ..., f_i)$ and I_i the ideal spanned by this sequence $\langle F_i \rangle$.
- $jac_x(F_{i-1})$ (resp. $jac_y(F_{i-1})$) is the jacobian matrix with respect to the two subsets of variables $x_1, ..., x_{n_x}$ (resp. $y_1, ..., y_{n_y}$).
- Let M be a $l \times c$ matrix with $l > c$. We call maximal minors of M the determinants of the $c \times c$ sub-matrices of M. We denote the maximal minors of M by $MaxMinors(M)$.
- $I_{i-1} : f_i$ is the ideal spanned by $\{g \in R \mid gf_i \in I_{i-1}\}$.

We will give important results of this article before discussing them further.

Theorem 3.1 (Theorem 2 in [FSS11])**.** *Let $i > n_x + 1$ (resp. $i > n_y + 1$) and let s be a linear combination of maximal minors of $jac_x(F_{i-1})$ (resp. $jac_y(F_{i-1})$). Then $s \in I_{i-1} : f_i$.*

As Theorem 3.1 indicates, reductions to 0 may also arise from the maximal minors of the jacobian matrix, so the signatures (s, f_i) where s is a linear combination of maximal minors of $jac_x(F_{i-1})$ or $jac_y(F_{i-1})$ is a reduction to 0. This explains Algorithm 3 where we compute the maximal minors of jacobian matrices before applying Algorithm 2 where we compute linear combinations of these maximal minors. This means that Algorithm 3 computes all the signatures that will arise in reductions to 0 that involve the maximal minors of jacobian matrices instead of being incremental like F5 criterions [Fau02].

Algorithm 2 Reduce

```
Require: A monomial ordering $\prec$ and $S$ a set of homogeneous polynomials and $q$ a degree.
Ensure: $T$ a reduced set of homogeneous polynomials of degree $q$.
$M \leftarrow Mac_{\prec}(S, q)$.
$M \leftarrow RowEchelonForm(M)$.
Return $T$ the set of polynomials corresponding to the rows of M.
```

Algorithm 3 BL_Criterion

```
Require: $m$ bilinear polynomials $f1, ...f_m$ such that $m \leq n_x + n_y$.
    $\prec$ a monomial ordering overs $R$.
Ensure: $V$ a set of pairs $(h, f_i)$ such that $h \in I_{i-1} : f_i$ and $h \notin I_{i-1}$.
$V \leftarrow \emptyset$
for $i$ from 2 to $m$ do
    if $i > n_y$ then
        $T \leftarrow$ Reduce$(MaxMinors(jac_y(F_{i-1})), n_y + 1)$
        for $h$ in $T$ do
            $V \leftarrow V \cup \{(h, f_i)\}$
        end for
    end if
    if $i > n_x$ then
        $T' \leftarrow$ Reduce$(MaxMinors(jac_x(F_{i-1})), n_x + 1)$
        for $h$ in $T'$ do
            $V \leftarrow V \cup \{(h, f_i)\}$
        end for
    end if
end for
Return $V$
```

Algorithm 4 BilinF5Criterion

```
Require: $(t, f_i)$ the signature of a row.
    A matrix $M$ in row echelon form.
Ensure: $True$ if the row will reduce to 0 otherwise $False$.
if $t$ is the leading monomial of a row of $M$ or
$\exists (h, f_i) \in V$ such that $LM(h) = t$ then
    Return $True$.
else Return $False$.
end if
```

Definition 3.6 (Definition 8 in [FSS11]). *Let $\prec$ be a monomial ordering such that its restriction to $\mathbb{K}[x_1, \dots, x_{n_x}]$ (resp. $\mathbb{K}[y_1, \dots, y_{n_y}]$) is the grevlex ordering. Let $m \leq n_x + n_y$ and $f_1, \dots, f_m$ be bilinear polynomials or R. We say that the polynomial sequence $(f_1, \dots, f_m)$ is a bi-regular sequence if $m = 1$ or if $(f_1, \dots, f_m)$ is a bi-regular sequence and*

$$LM(I_{m-1} : f_m) = \langle Monomials^x_{m-n_y-2}(n_y+1)\rangle + \langle Monomials^y_{m-n_x-2}(n_x+1) + LM(I_{m-1})\rangle.$$

We can notice that the output of algorithm 2 is polynomials of degree q. Here q is always $n_x + 1$ or $n_y + 1$. We deduce that the only non trivial syzygies are of degree $n_x + 1$ or $n_y + 1$. **c'est pas $n_x + 2$ ou $n_y + 2$ du coup ?**

The following theorem, ensures that there are no reductions to 0 with extended the F5 criterion. **Ce théorème est en suspens pour utiliser une autre formulation sans utiliser la topologie de Zariski**

Theorem 3.2 (Theorem 4 in [FSS11]). *Let $m, n_x, n_y \in \mathbb{N}$ such that $m < n_x + n_y$. If Conjecture 1 of [FSS11] is true, then the set of bi-regular sequences $(f_1, ..., f_m)$ contains a nonempty Zariski open set. Moreover, if $(f_1, ..., f_m)$ is a bi-regular sequence, then there are no reductions to zero with the extended F5 criterion.*

An important result of [FSS11] is on the index of regularity of bi-regular systems.

Theorem 3.3. *If the system $\mathcal{F}$ is a generic bilinear system, then the degree of regularity of $\mathcal{V}(I)$ is upper bounded by $d_{reg} \leq min(n_x, n_y)$.*

Finally, the authors give the following corollary which is the consequence of 3.3 by applying the complexity of computing a Gröbner basis given by [BTB04].

Corollary 3.1 (Corollary 3 in [FSS11]). *The arithmetic complexity of computing a Gröbner basis of a generic affine bilinear system $f_1, ..., f_{n_x+n_y}$ with the F5 algorithm is bounded by*

$$\mathcal{O}\left(\binom{n_x + n_y + min(n_x+1, n_y+1)}{min(n_x+1, n_y+1)}^{\omega}\right).$$

Corollary 3.2.

First, we can clearly see that the complexity of Algorithm 3 is exponential and we will show that in the following proposition.

Proposition 3.5. *The complexity of Algorithm 3 is*

$$\mathcal{O}\left(\left(\sum_{i=n_x}^{m}\binom{i}{n_x}n_x^{\omega} + \binom{2n_x+1}{n_x}^{\omega}\right) + \left(\sum_{i=n_y}^{m}\binom{i}{n_y}n_y^{\omega} + \binom{2n_y+1}{n_y}^{\omega}\right)\right).$$

Proof. First, if $i > n_y$ (resp. $i > n_x$), we construct the jacobian matrix $jac_y(F_{i-1})$ (resp. $jac_x(F_{i-1})$) which is of size $n_y \times n_y$ (resp. $n_x \times n_x$) for the first one and $m - 1 \times n_y$ (resp. $m-1 \times n_x$) for the last one i.e. $i = m$. We suppose here that the construction of such matrices is free. Computing the $MaxMinors(jac_y(F_{i-1}))$ (resp. $MaxMinors(jac_x(F_{i-1}))$)means that we have to compute the determinants of all the sub-matrices of size $n_y \times n_y$ (resp. $n_x \times n_x$) and for a matrix M of size $l \times c$ with $l > c$ there are $\binom{l}{c}$ sub-matrices of size $c \times c$, hence the term $\binom{i}{n_y}n_y^{\omega}$ in the sum if we suppose the complexity of computing the determinant of a square multivariate polynomial matrix of size $c \times c$ to be $\mathcal{O}(c^{\omega})$, which is not the case but let us suppose the best case ever. **faudrait trouver la vraie complexité de ça pour être plus juste.** For now, we have a sequence of polynomials and we apply the algorithm **Reduce** 2 on them with degree $n_y + 1$ (resp. $n_x + 1$). In **Reduce**, we construct the Macaulay matrix of the sequence of degree $n_y + 1$ (resp. $n_x + 1$) and compute it's row echelon form. The size of Macauly matrix is bounded by $\binom{n+D}{D}$ where D is the degree and n the number variables. So the complexity of **Reduce** is

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

(resp.

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

). Hence the second term in the sum, because we call **Reduce** at each iteration of the for loop. We finally suppose that the iteration on the elements of the output of **Reduce** is negligible in the big $\mathcal{O}$ notation. □

It is an asymptotic complexity which does not take into account the constant factor in implementations that could slow down a lot the algorithm. We also recall that in our case with NSBC, $n_x = n_y$. So we get

$$\mathcal{O}\left(2n_x^{\omega}\left(\sum_{i=n_x}^{m}\binom{i}{n_x} + \binom{2n_x + 1}{n_x}^{\omega}\right)\right).$$

Since we have quadratic systems, reductions to 0 would not appear before degree 4 of the Macaulay matrix. Also, in practice we are using $q = 2$ and there are much better algorithms in the literature for boolean systems than basic Gröbner basis algorithms like F5. The best algorithm used at the moment is CrossBred [JV17]. Most of the time in practical implementations, we stop at degree 4 of the Macaulay in pre-processing step. So, to know what advantage we could have if we use the extended F5 criterion, we will suppose the worst, meaning that all of our entries will reduce to 0 at degree 4 of the Macaulay matrix, which is unlikely.

In following, we will denote $n_x + n_y$ by simply n.

If this case arises, the only computations that would be done are for extended F5 criterion. The complexity for F5-Matrix algorithm with only the Frobenius criterion 3.2 is

$$\mathcal{O}\left(\binom{n+4}{4}^{\omega}\right).$$

If we only take into consideration the big O notation and not the constants, which is not realistic at all for efficient implementations to really break a cryptographic scheme, we get

$$Adv(n, m) = 2\left(\sum_{i=n/2}^{m}\binom{i}{n/2}\left(\frac{n}{2}\right)^{\omega} + \binom{n+1}{n/2}^{\omega}\right) - \binom{n+4}{4}^{\omega}. \tag{1}$$

We used the computer algebra system SageMath [Theyy] to test numerically when using the F5 extended criterion of [FSS11] becomes negligible compared to the computations of reductions to 0, we see experimentally that $Adv(10, 12) = 2498903$, i.e. the theoretical advantage becomes negligible and it would be advantageous to simply compute reductions to 0 instead of applying the F5 extended criterion. This number grows exponentially with n and m. For those who wonder, what would be the estimated theoretical advantage computing reductions to 0 for the parameters given by [HJ23] $Adv(256, 257)$ is a 514 bit number.

3.5 Boolean systems

Sometimes, the finite field used is $\mathbb{F}_2$ because it is still supposedly hard to solve but is very easy to use and manipulate on computers. But in such small fields one can begin to use exhaustive search to obtain the solution to the system. The best algorithm known to the authors for exhaustive search for multivariate polynomial system $f_1, \ldots, f_m$ over $\mathbb{F}_2[x_1, \ldots, x_n]$ is [Bou+10] with a complexity $\mathcal{O}(ln(n)2^n)$. What is remarkable is that the complexity does not implicate the number of polynomials in the system, m. There is also a software made by one of the authors, Charles Bouillaguet, that is available online and easy to use, *libFES-lite* [Bou22] and there have been implementations on FPGA that broke real world cryptography [Din+20].

Another type of algorithms uses exhaustive search combined with probabilistic algorithms. [Lok+17] gave an algorithm that finds the solution to a polynomial system with complexity $\tilde{\mathcal{O}}(2^{0.8765n})$ which is better than exhaustive search. Even though it is asymptotically better, there are no known implementations because of the huge constant factors present in the complexity that makes it worse than exhaustive search for any systems that can be solved on modern computers. New algorithms have been proposed by Itai Dinur [Din21b; Din21a] where the complexity is brought down to $\tilde{\mathcal{O}}(2^{0.6943n})$, but still no implementation known to the authors.

The last type is a hybrid algorithms using Gröbner basis algorithms and exhaustive search to find solutions. The most known and the best one is BooleanSolve hybrid by Bardet, Faugère, Salvy and Spaenlehauer [Bar+13] where we guess a certain number of variables k and apply F5 algorithm on the specialized systems. Under semi-regularity assumption, the algorithm finds the solution in $\mathcal{O}^{(}2^{0.792n})$ if $n = m$ and $k = 0.55n$. This is better than exhaustive search, but still, the constant factors are too big for it to be better for systems smaller with less than 200 variables.

The algorithm on which I focused on during my internship is Crossbred by Joux and Vitse [JV17]. It mixes the usage of exhaustive search and algebraic algorithms like BooleanSolve hybrid but with a pre-processing step before doing the exhaustive search and applying algebraic algorithms like XL [Cou+00] or it's variants or F5 [Fau02].

To understand this algorithm, we will need definitions of special Macaulay matrices used by it.

We will denote by $deg^k(p)$ the total degree of p in the first k variables.

Definition 3.7. *For a given $\mathcal{F} = \{f_1, \ldots, f_m\}$, $Mac^k_{\leq D, \geq d, m}$ is the submatrix of $Mac_{\leq D,m}$ whose rows correspond to a product uf_i, $1 \leq i \leq m$, with $deg^k(u) \geq d - 1$. Let $\mathcal{M}^k_{\leq D, \geq d, m}$ be the submatrix of $Mac^k_{\leq D, \geq d, m}$ with columns correspond to monomials m such that, $deg^k(m) \geq d + 1$.*

This algorithm have shown it's effectiveness in practice with good implementations like Bouillaguet and Sauvage's implementation [CB23] that broke records for solving boolean systems on the Fukuaka MQ challenge [Yas15]. Despite that, there are no proven complexity analysis for this algorithm due to it's huge number of possible parameters but we can do some estimations with admissible parameters as we will see in the next section.

4 Contributions

4.1 System Modeling

One of the known attacks given by the authors is through algebraic cryptanalysis. One can model the attack by taking g as the following polynomial

$$g(x_1, ..., x_n, y_1, ..., y_n) := (\vec{u} \cdot \vec{x})(\vec{v} \cdot \vec{y}) - (\vec{u} \cdot \vec{y})(\vec{v} \cdot \vec{x}).$$

Algorithm 5 The CrossBred algorithm

```
Require: Polynomials $\mathcal{F} = f_1, \dots, f_m$ of $n$ variables and integers $D, d, k$
Ensure: A solution for the system if it exists or nothing otherwise.
Construct $Mac^k_{\leq D, \geq d, m}(\mathcal{F})$ and $\mathcal{M}^k_{\leq D, \geq d, m}(\mathcal{F})$
Compute a basis $(v_1, \dots, v_r)$ if the left kernel of $\mathcal{M}^k_{\leq D, \geq d, m}(\mathcal{F})$
Construct the polynomials $p_1, \dots, p_r$ corresponding to $v_i \cdot Mac^k_{\leq D, \geq d, m}(\mathcal{F})$.
for $a = (a_{k+1}, \dots, a_n) \in \mathbb{F}_2^{n-k}$ do
    Evaluate the last $n - k$ variables in each $f_i \in \mathcal{F}$ at $(a_{k+1}, \dots, a_n)$ and compute $\mathcal{F}*$
    Compute $Mac_{\leq d, m}(\mathcal{F}*)$
    Compute $\mathcal{F}'*$ as the partial evaluation of $F' = p_1, \dots, p_r$ at $(a_{k+1}, \dots, a_n)$
    Consider the system $S*$ consisting of $Mac_{\leq d, m}(\mathcal{F}*) \cup \mathcal{F}'*$
    if $S*$ is consistent using XL for example then
        Return solution
    end if
end for
```

We consider

$$g(x_1, .., x_{n-2}, y_1, .., y_{n-2}) := (\sum_{i=1}^{n-2} u_i x_i + u_{n-1})(\sum_{i=1}^{n-2} v_i y_i + v_n) - (\sum_{i=1}^{n-2} u_i y_i + u_n)(\sum_{i=1}^{n-2} v_i x_i + v_{n-1}).$$

Then, one can perform a Weil descent on the polynomial g because $\mathbb{F}_{q^k}$ is a $\mathbb{F}_q$-vector space and obtain the polynomial system

$$\begin{gathered} g_1(x_1, \dots, x_{n-2}, y_1, \dots, y_{n-2}) = 0 \\ \vdots \\ g_k(x_1, \dots, x_{n-2}, y_1, \dots, y_{n-2}) = 0 \end{gathered}$$

where g_i are polynomials with coefficients in $\mathbb{F}_q$.

The most straightforward cryptanalysis of this type of systems is to perform a Gröbner basis algorithm like Buchberger's algorithm [Buc06], Faugère's F4 [Fau99], F5 [Fau02] or their multiple variants [**EF17**].

Huth and Joux cite a complexity result in [FSS11] to argue their parameters choice, we will dive deeper into this article results in a later section.

4.2 Algebraic cryptanalysis

Bardet showed in [BTB04] that F5 criterion removed reductions to 0 for semi-regular sequences until the degree $H(I) - 1$, where I is the ideal generated by the polynomials of the system.

Theorem 4.1 (Theorem 3.2.10 in [BTB04])**.** *Let $f_1, ..., f_m$ a homogeneous sequence, such that $\langle f_1, ..., f_m \rangle$ is 0-dimensional and $\prec$ an admissible graded monomial ordering. We have,*

- *If the sequence is semi-regular, then no reductions to 0 are performed during the F5-matrix algorithm until the degree* $d = H(I) - 1$
- *If there are no reductions to 0 during the F5-matrix algorithm until the degree $D - 1$, and if the matrix of degree D is full rank and is the first matrix to have more rows than columns, then the sequence is semi-regular and its index of regularity is $H(I) = D$.*

In [BFS15], the authors show that the complexity of computing the Gröbner basis of a semi-regular system $f_1, ..., f_m \in \mathbb{K}[x_1, ..., x_n]$ is $O\left(mD\binom{n+D-1}{D}^{\omega}\right)$.

We will also need the next definition of γ-strong semi-regularity.

Definition 4.1 (Definition 4 of [VID24])**.** *Let $\mathcal{F} = f_1, ..., f_m$ be a semi-regular sequence of polynomials in $\mathbb{F}_2[x_1, ..., x_n]$ and let $0 \leq \gamma \leq 1$ such that $k = (1-\gamma)n$. We say that this sequence is γ-strong semi-regular if*

$$\mathcal{S}(I) = \{(a_{k+1}, ..., a_n) \in \mathbb{F}_2^{n-k} |$$

$$\{f_1(x_1, ..., x_k, a_{k+1}, ..., a_n), ..., f_m(x_1, ..., x_k, a_{k+1}, ..., a_n)\} \textit{ is not semi-regular}\}$$

has cardinality $\mathcal{O}(2^{-\gamma n})$.

In our setting, we consider bilinear systems that are slightly overdetermined. We argue that when using hybrid methods such as *Crossbred*, which avoid the need to construct Macaulay matrices up to high degrees—as required by algorithms like F5—the only reductions to zero that are likely to occur are those explained by the F5 and Frobenius criterion. Indeed, other types of reductions to zero typically appear at degrees $n_x + 1$ and $n_y + 1$ for bilinear systems, which are not reached in our practical use cases. Consequently, we consider these systems to behave in a semi-regular manner, although we believe they are, in practice, easier to solve.

4.3 Admissible parameters for Crossbred

We tested if small instances of our problem are semi-regular or not using Proposition 3.4 and they were not semi-regular. We also remarked that the bilinear criterion 4 detected reductions to 0 using the original F5 criterion 3.1 and Frobenius criterion for systems over $\mathbb{F}_2$ but also detected reductions to 0 on polynomials of degree n_x or n_y according to the algorithms 3 and 2. In our case, we will use the Crossbred algorithm and we do not think that we will get to these degrees in the Macaulay matrices, so no need to use the bilinear criterion.

Even though no concrete complexity analysis is available for the Crossbred algorithm because of it's wide parameters choice, there are some articles that give formal tools to analyze admissible parameters and try to estimate the complexity for these parameters. We choose to focus on the article [VID24] to select valid parameters and estimate the complexity of the algorithm using them.

Theorem 4.2. *Let $H^k_{m,n}(X,Y) = \sum_{d_1 \geq 0, d_2 \geq 0} h^k_{d_1,d_2,m} X^{d_1} Y^{d_2}$ be the bivariate series with coefficient which is the number of new "independent" polynomials generated during the pre-processing. This series is given by:*

$$H^k_{m,n}(X,Y) = \frac{1}{Y}\left((1+X)^{n-k} - \frac{(1+XY)^k(1+X)^{n-k}}{(1+X^2Y^2)^m}\right).$$

Proposition 4.1. *For fixed values of m, n and k the bivariate series $G^k_{m,n}(X,Y) = \sum_{d_1 \geq 0, d_2 \geq 0}(\sum_{d_1' \leq d_1, d_2' \geq d_2} g^k_{d_1',d_2',m}) X^{d_1} Y^{d_2}$ is given by the formula:*

$$G^k_{m,n}(X,Y) = -\frac{Y H^k_{m,n}(X,Y) - H^k_{m,n}(X,1)}{(1-X)(1-Y)}.$$

the coefficient in front of $X^{d_1}Y^{d_2}$ is the number of polynomials generated during the pre-processing step for a given pair (d_1, d_2).

We also have that by definition $h^k_{d_1,d_2,m}$ is defined as:

$$h^k_{d_1,d_2,m} = \begin{cases} U^k_{d_1,d_2,m} - M^k_{d_1,d_2+1}, & \text{if } d_1 \geq d_2 \geq 0,\ m \geq 1, \\ -M_{d_1,0}, & \text{if } d_1 \geq 0,\ d_2 = -1,\ m \geq 0, \\ -M^k_{d_1,d_2+2}, & \text{if } d_1 \geq d_2 \geq 0,\ m = 0, \\ 0, & \text{in all other cases.} \end{cases} \tag{2}$$

Theorem 4.3. *Let $\mathcal{F} = f_1, \ldots, f_m$ be a γ-strong semi-regular sequence of polynomials in $\mathbb{F}_2[x1, \ldots, x_n]$ and consider $D < D_{reg}, d < d_{reg}(k)$. Then k, D and d are potentially admissible parameters for the Crossbred algorithm if the coefficient corresponding to $X^D Y^d$ of the following bivariate series*

$$J^k_{m,n}(X,Y) = \frac{1}{(1-X)(1-Y)} \left(\frac{(1+X^{n-k})(1+XY)^k}{(1+X^2Y^2)^m} - \frac{(1+X)^n}{(1+X^2)^m} - \frac{(1+Y)^k}{(1+Y^2)^m} \right)$$

is non-negative.

We wrote SageMath scripts that automated the search of admissible parameters using the bivariate series given in [VID24]. We searched for different key sizes to have a rough estimate on the complexity.

We discuss more the attacks complexity using Crossbred and different algorithms in the next section.

4.4 Security proofs

The security of the signature scheme is demonstrated under standard assumptions regarding the adversary's capabilities, ensuring that the scheme meets well-defined security goals.

In the literature, there are three main security goals:

- *Unbreakability (**UB**)*: the attacker recovers the secret key from the verification key (or public key).
- *Universal Unforgeability (**UUF**)*: the attacker can produce a valid signature of any message in the message space without necessarily recovering the verification key.
- *Existential Unforgeability (**EUF**)*: the attacker creates a message and produces a valid signature of it without necessarily having a choice for the message.

Finally, there are three main adversarial models:

- *Key-Only Attacks (**KOA**)*: The adversary has only access to the verification keys, which is the simplest and unavoidable scenario.
- *Known-Message Attacks (**KMA**)*: The adversary has access to a set of messages and their valid signatures.
- *Chosen-Message Attacks (**CMA**)*: The adversary can probe an oracle that can sign any message of his choice and he can request as many signatures as he wants.

In constructing a signature scheme, the objective is to achieve **EUF-CMA** security, which represents the strongest and most desirable level of security. As a first step, we demonstrate that the scheme satisfies the **UB-KOA** property, which is a weaker but more tractable security notion.

Lemma 4.1. *Let (q, n, k) be a set of parameters for the NSBC problem. Assume that the NSBC function $NSBC\colon \mathbb{F}_{q^k}^{2n} \to \mathbb{F}_q^{2n}$ is (τ, ϵ_{OWF})-hard, meaning that no probabilistic polynomial-time ($\mathcal{PPT}$) adversary mathbbA running in time at most τ can solve the NSBC problem with probability greater than ϵ_{OWF}.*

$$Succ(\mathcal{A}) = \mathbb{P}\left[\begin{array}{l}(u,v) \leftarrow \mathbb{F}_{q^k}^{2n},\\ (x,y) \leftarrow \mathcal{A}(u,v),\\ (x,y) \in SBC[u,v]\end{array}\right] < \epsilon_{OWF}.$$

*Then the signature scheme is **UB-KOA**.*

The proof of 4.1 can be found in B.

Theorem 4.4. *Assume that no adversary can distinguish the output of the pseudorandom function (PRF) used in the protocol from truly random bits. The adversary outputs a bit b, where $b = 0$ indicates that they believe the bits originate from the PRF, and $b = 1$ indicates they believe the bits are uniformly random*

$$|\mathbb{P}_{seed \in \{0,1\}^n}[b = 0 \mid p \leftarrow PRF(seed)] - \mathbb{P}[b = 1 \mid p \xleftarrow{\$} \{0,1\}^{2n}]| \leq \frac{1}{2}.$$

Suppose also that the protocol uses a cryptographic hash function which is one-way and does not have collision attacks with better complexity than 2^λ using a $\mathcal{PPT}$, where λ is the number security bits needed for the signature scheme.

*Finally, suppose that the NSBC problem is hard, i.e. the complexity of the best attack of the NSBC problem is 2^λ. Then, the NSBC signature scheme is **EUF-CMA**.*

The proof of 4.4 can be found in the appendix B.

4.5 Security analysis

This section is heavily inspired from the security analysis of the MQOM signature scheme [RB25].

First we need to enumerate the parameters for the signature scheme. The finite field used is $\mathbb{F}_2$ and the vector length n is 130 but we consider the NSBC version so it is 128 in practice with the 2 last coordinates implicitly defined. The field extension used k is 257 which is prime to avoid subfield attacks [Pé24].

For the linear algebra constant ω, we take $\omega = 2.81$ for dense linear algebra using the Strassen algorithm, which is implemented in dense linear boolean algebra software like M4RI and M4RIE [AB].

For sparse linear algebra, we suppose the use of block-Wiedemann algorithm which is the most used in software like CADO-NFS [Tea17] and in XL parallel implementation [NNY17]. The complexity of this algorithm depends on the "blocking factor" that we will denote $\tilde{m}$ and $\tilde{n}$, the size of the matrix denoted N and the number of non-zero elements denoted nnz. Most of the algorithm boils down to (sparse $N \times N$ matrix) $\times$ (dense N vector) products. There are 3 phases in the block-Wiedemann algorithm BW1, BW2 and BW3. The reader can refer to [NNY17] for a clear description of these phases. The most costly phase is BW1 where it does $(1/\tilde{n} + 1/\tilde{m})N$ (dense $\tilde{m} \times N$ matrix) x vector products that require $2N\tilde{m}$ operations. The complexity of BW2 and BW3 phases can be negligible compared to BW1. So we estimate the total number of arithmetic operations of the block-Wiedemann algorithm to be $(2(1 + \tilde{n}/\tilde{m})N(nnz + \tilde{m}N))$. In [NNY17], which describes the only and best implementation of XL using also a good implementation of the block-Wiedemann algorithm, they suggest $\tilde{n} = \tilde{m} = 512$ for $\mathbb{F}_2$ and that is what we used for our estimations. **A préciser et comprendre si possible**.

We also need to clarify some properties that we use for the Macaulay matrices. First we know that the number of monomials of degree exactly d (resp. of $\leq d$) in n variables, which is the number of columns of the Macaulay matrix of degree d (resp. of degree $\leq d$), is $\binom{n}{d}$ (resp. $\sum_{i=0}^{d} \binom{n}{d}$) because of the Frobenius identity that reduces their number. A good remark is that the number of monomials of system modelisation of SBC problems contain exactly $\frac{n^2}{4} + \frac{n}{2} + 1$ monomials. For the number of rows, we will use the bivariate series G 4.1. Finally, we know that Macaulay matrices are very sparse and each row contain at most $(n+1)(n+2)/2$ non zero coefficients.

Now we will estimate attacks based on the Crossbred algorithm. Even though we do not have tight complexity analysis of it, we can do estimations based on some algorithms and common assumptions. We need to first compute the number of rows of $\mathcal{M}^k_{\leq D, \geq d,m}(\mathcal{F})$ and the number of nonzero elements per column for our systems. We focus on this matrix because it is the one on which we apply block-Lancszos to find r vectors in its co-kernel. The number of rows is bounded by

$$nb\_monomials\_\mathcal{M}^k_{\leq D, \geq d,m} = \sum_{i=d+1}^{D} \sum_{j=0}^{d-i} \binom{k}{i} \cdot \binom{n-k}{j}$$

Finally, we saw by experiment that the number of monomials present in the our system polynomials is $\frac{n^2}{4} + \frac{n}{2} + 1$ which is the upper-bound for the number of nonzero elements per row instead of the standard $\mathcal{O}(n^2)$. The time complexity of this operation is

$$\mathcal{O}\left(\frac{r}{W} max(\frac{n^2}{4} + \frac{n}{2} + 1)c^2/W, c^2\right),$$

where W is the machine word size which is 64 on most common machines nowadays and c is the number of rows of $\mathcal{M}(\leq D, \geq d, m)^k$ which is bounded by **Citer article quand il sera sortie**

$$c \leq \left(\sum_{i=0}^{D} \binom{n}{i} - \sum_{i=0}^{d-2} \binom{k}{i}\right) m.$$

The authors of [RB25] noticed that if $d \geq 1$, then $\mathcal{M}^k_{\leq D, \geq d,m}$ is a dense matrix, so it is better to just apply Strassen to find these vectors. We compute both complexities and take the minimum for our estimations. After finding the vectors, comes the exhaustive search part. First, we compute the specialized system $\mathcal{F}*$ and compute $Mac_{\leq d,m}(\mathcal{F}*)$ and concatenate to the Macaulay matrix the polynomials $\{p_1, \ldots, p_r\}$ evaluated and finally we check if the system is full rank by applying XL algorithm using block-Wiedemann or just Strassen or simply Gaussian elimination which is done in *hpXbred* [CB23]. The most computationally intensive part here is the Gaussian elimination that we apply on a matrix almost square matrix of size $\sum_{i=0}^{d} \binom{n-k}{i}$ with complexity $\mathcal{O}(n^3)$ or $\mathcal{O}(n^{2.81})$ using Strassen or M4RI algorithms. We then apply these algorithms 2^{n-k} times to get an estimation for the time complexity of exhaustive search part.

With all of these estimation, we get a broad estimation for the Crossbred algorithm for a admissible parameters. We wrote SageMath scripts to compute these estimation which are available in a public github repository

http://www.github.com/paolo/MPCitH_SBC.

Our best estimations are...**A suivre**.

Finally, the simplest and best attack to our knowledge is to do exhaustive search on all of the y or x variables, then the specialized systems $\mathcal{F}*$ are just linear systems and we could just apply Gaussian elimination or system solving using M4RI [AB] on it in $\mathcal{O}(n^{log(7)})$ operations.

The system is of size 257 × 129. If the system is full rank, we give the solution, else we continue the exhaustive search. The cost of this attack is

$$\mathcal{O}(2^{128} \cdot 257^{log(7)}) \approx \mathcal{O}(2^{150.5}).$$

One advantage of this attack is that it's memory footprint is negligible.

4.6 Space complexity of Crossbred

One important consideration for practical implementations of cryptographic attacks is the memory footprint. In the literature, many algorithms are classified as *galactic*, meaning that although they exhibit favorable asymptotic complexity, they are unlikely to be used in practice due to large constant factors hidden in the big-O notation. Well-known examples of galactic algorithms include matrix multiplication in $\mathcal{O}(n^{2.371552})$ [Wil+23] and integer multiplication in $\mathcal{O}(n \log n)$ [HH21].

On the other hand, some algorithms are practically efficient in terms of computation but require an enormous amount of memory, such as birthday attacks or rainbow table attacks on hash functions, which often renders them impractical in real-world scenarios.

In this section we discuss the memory needed by the Crossbred algorithm to attack the SBC signature scheme.

We first need to construct $Mac^k_{\leq D, \geq d, m}$, and then consider its submatrix $\mathcal{M}^k_{\leq D, \geq d, m}$. In practice, $\mathcal{M}^k_{\leq D, \geq d, m}$ can be obtained by manipulating pointers to $Mac^k_{\leq D, \geq d, m}$, without duplicating data. Therefore, the primary memory footprint to consider at this stage is that of $Mac^k_{\leq D, \geq d, m}$.

Let us now analyze the following cases:

- We use a sparse matrix representation based on the Compressed Sparse Row (CSR) format, which requires storing two arrays: the array of column indices of non-zero elements, denoted as `col_index`, and the array that indicates the number of non-zero elements per row, denoted as `row_index`. A third array for storing the actual coefficients is not needed, since all non-zero entries are equal to 1.

  The size of `col_index` is nnz and the size of `row_index` is the number of rows of the matrix. So, the memory footprint for the storage of $Mac^k_{\leq D, \geq d, m}$ is in our case

  $$memory\_footprint \leq \left(\left(\frac{n^2}{4} + \frac{n}{2} + 1\right) + 1\right) \times \left(\sum_{i=0}^{D} \binom{n}{i} - \sum_{i=0}^{d-2} \binom{k}{i}\right) m \times 4 \text{ bytes.}$$

  One important thing to take into account is the size of the coefficients in the arrays. They have to be at least of size $log\left(\sum_{i=0}^{D} \binom{n}{i}\right)$ **à affiner comme borne**, because it's the number of columns of $Mac^k_{\leq D, \geq d, m}$. So the most accurate estimation is rather

  $$\left(\left(\frac{n^2}{4} + \frac{n}{2} + 1\right) + 1\right) \times \left(\sum_{i=0}^{D} \binom{n}{i} - \sum_{i=0}^{d-2} \binom{k}{i}\right) m \times \lceil\lceil log\left(\sum_{i=0}^{D} \binom{n}{i}\right)\rceil \times \frac{1}{8}\rceil \text{ bytes.}$$

  Example 4.1. *If we have $D = 12, d = 10, n = 256, m = 257, k = 176$, the memory_footprint would be:*

  $$\left(\left(\frac{256^2}{4} + \frac{256}{2} + 1\right) + 1\right) \times \left(\sum_{i=0}^{12} \binom{256}{i} - \sum_{i=0}^{8} \binom{176}{i}\right) \times 257 \times \lceil\lceil log\left(\sum_{i=0}^{12} \binom{256}{i}\right)\rceil \times \frac{1}{8}\rceil =$$

  $$5112102637167689483266493844 \approx 2^{92} \textit{ bytes.}$$

- If we use dense representation, then we could represent each row as a single integer of size $log\left(\sum_{i=0}^{D}\binom{n}{i}\right)$ bits and we would have $\left(\sum_{i=0}^{D}\binom{n}{i} - \sum_{i=0}^{d-2}\binom{k}{i}\right) m$ of these integers, which is the number of rows. So the memory_footprint would be

$$memory\_footprint \leq \lceil \lceil log\left(\sum_{i=0}^{D}\binom{n}{i}\right)\rceil \times \frac{1}{8}\rceil \times \left(\sum_{i=0}^{D}\binom{n}{i} - \sum_{i=0}^{d-2}\binom{k}{i}\right) m \text{ bytes.}$$

Example 4.2. *For the parameter choice in Example 4.1, the memory footprint of the dense representation would be:*

$$\lceil \lceil log\left(\sum_{i=0}^{12}\binom{256}{i}\right)\rceil \times \frac{1}{8}\rceil \times \left(\sum_{i=0}^{12}\binom{256}{i} - \sum_{i=0}^{8}\binom{176}{i}\right) \times 257 =$$

$$309561743803299593270346 \approx 2^{78} \textit{ bytes.}$$

These are theoretical estimations and in practice it can be much more. For example, to represent a 67 bit number using the bignum implementation in the GMP library, we would need at least 32 bytes instead of the 9 that we estimated because of memory alignement and the total coast of the structure, but this can be optimized to use less memory for an array usage.

4.7 Attacks summary

Algorithm	Parameters	Time	Memory
Exhaustive search	/	260	24
Dinur First	/	229	200
Dinur Second	/	221	172
Crossbred	D=12, d=10, k=130	159	78
Naive	/	150	15

Table 2: Time and memory complexity of given algorithms in powers of 2. For Exhaustive search, Dinur First and Dinur second, the estimations were obtained using CryptographicEstimator [Ess+24]

References

[AB] Martin Albrecht and Gregory Bard. *The M4RI Library – Version **version***. The M4RI Team. URL: \url{https://bitbucket.org/malb/m4ri} (cited on pages 21, 22).

[AM+22] Carlos Aguilar-Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. *The Return of the SDitH*. Cryptology ePrint Archive, Paper 2022/1645. 2022. URL: https://eprint.iacr.org/2022/1645 (cited on page 5).

[Bar+13] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Pierre-Jean Spaenlehauer. “On the complexity of solving quadratic Boolean systems”. In: *Journal of Complexity* 29.1 (2013), pages 53–75. ISSN: 0885-064X. DOI: https://doi.org/10.1016/j.jco.2012.07.001. URL: https://www.sciencedirect.com/science/article/pii/S0885064X12000611 (cited on page 17).

[Beu22] Ward Beullens. *Breaking Rainbow Takes a Weekend on a Laptop.* Cryptology ePrint Archive, Paper 2022/214. 2022. URL: `https://eprint.iacr.org/2022/214` (cited on page 2).

[BFS15] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. "On the complexity of the F5 Gröbner basis algorithm". In: *Journal of Symbolic Computation* 70 (2015), pages 49–70. ISSN: 0747-7171. DOI: `https://doi.org/10.1016/j.jsc.2014.09.025`. URL: `https://www.sciencedirect.com/science/article/pii/S0747717114000935` (cited on page 19).

[Bou+10] Charles Bouillaguet, Chen-Mou Cheng, Tony (Tung) Chou, Ruben Niederhagen, Adi Shamir, and Bo-Yin Yang. *Fast Exhaustive Search for Polynomial Systems in F_2.* Cryptology ePrint Archive, Paper 2010/313. 2010. URL: `https://eprint.iacr.org/2010/313` (cited on page 17).

[Bou22] Charles Bouillaguet. *libFES-lite.* 2022. URL: `https://github.com/cbouilla/libfes-lite` (cited on page 17).

[BTB04] Magali Bardet (Turrel Bardet). "Etude des systèmes algébriques surdéterminés : applications aux codes correcteurs et à la cryptographie". Thèse de doctorat dirigée par Faugère, Jean-Charles Informatique Paris 6 2004. PhD thesis. 2004, 1 vol., [XIII] –157 p. URL: `http://www.theses.fr/2004PA066404` (cited on pages 10, 12, 15, 18).

[Buc06] Bruno Buchberger. "Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal". In: *Journal of Symbolic Computation* 41.3 (2006). Logic, Mathematics and Computer Science: Interactions in honor of Bruno Buchberger (60th birthday), pages 475–511. ISSN: 0747-7171. DOI: `https://doi.org/10.1016/j.jsc.2005.09.007`. URL: `https://www.sciencedirect.com/science/article/pii/S0747717105001483` (cited on pages 9, 10, 18).

[Bui+24] Dung Bui, Eliana Carozza, Geoffroy Couteau, Dahmun Goudarzi, and Antoine Joux. *Faster Signatures from MPC-in-the-Head.* Cryptology ePrint Archive, Paper 2024/252. 2024. URL: `https://eprint.iacr.org/2024/252` (cited on page 5).

[CB23] Julia Sauvage Charles Bouillaguet. *High-Performance Xbred.* 2023. URL: `https://gitlab.lip6.fr/almasty/hpXbred` (cited on pages 17, 22).

[CLO10] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra.* 3rd. Springer Publishing Company, Incorporated, 2010. ISBN: 1441922571 (cited on pages 9, 10).

[Cou+00] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. "Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations". In: *Advances in Cryptology — EUROCRYPT 2000.* Edited by Bart Preneel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pages 392–407. ISBN: 978-3-540-45539-4 (cited on page 17).

[Din+20] Jintai Ding, Joshua Deaton, Vishakha, and Bo-Yin Yang. *The Nested Subset Differential Attack: A Practical Direct Attack Against LUOV which Forges a Signature within 210 Minutes.* Cryptology ePrint Archive, Paper 2020/967. 2020. URL: `https://eprint.iacr.org/2020/967` (cited on page 17).

[Din21a] Itai Dinur. "Cryptanalytic Applications of the Polynomial Method for Solving Multivariate Equation Systems over GF(2)". In: *Advances in Cryptology – EUROCRYPT 2021.* Edited by Anne Canteaut and François-Xavier Standaert. Cham: Springer International Publishing, 2021, pages 374–403. ISBN: 978-3-030-77870-5 (cited on page 17).

[Din21b] Itai Dinur. "Improved Algorithms for Solving Polynomial Systems over GF(2) by Multiple Parity-Counting". In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA).* 2021, pages 2550–2564. DOI: 10.1137/1.9781611976465.151. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611976465.151. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611976465.151 (cited on page 17).

[DS05] Jintai Ding and Dieter Schmidt. "Rainbow, a new multivariable polynomial signature scheme". In: *Proceedings of the Third International Conference on Applied Cryptography and Network Security.* ACNS'05. New York, NY: Springer-Verlag, 2005, 164–175. ISBN: 3540262237. DOI: 10.1007/11496137_12. URL: https://doi.org/10.1007/11496137_12 (cited on page 2).

[EF17] Christian Eder and Jean-Charles Faugère. "A survey on signature-based algorithms for computing Gröbner bases". In: *Journal of Symbolic Computation* 80 (2017), pages 719–784. ISSN: 0747-7171. DOI: https://doi.org/10.1016/j.jsc.2016.07.031. URL: https://www.sciencedirect.com/science/article/pii/S0747717116300785.

[Ess+24] Andre Esser, Javier A. Verbel, Floyd Zweydinger, and Emanuele Bellini. "CryptographicEstimators - a Software Library for Cryptographic Hardness Estimation". In: *AsiaCCS.* {ACM}, 2024 (cited on page 24).

[Fau02] Jean Charles Faugère. "A new efficient algorithm for computing Gröbner bases without reduction to zero (F5)". In: *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation.* ISSAC '02. Lille, France: Association for Computing Machinery, 2002, 75–83. ISBN: 1581134843. DOI: 10.1145/780506.780516. URL: https://doi.org/10.1145/780506.780516 (cited on pages 10, 13, 17, 18).

[Fau99] Jean-Charles Faugère. "A new efficient algorithm for computing Gröbner bases (F4)". In: *Journal of Pure and Applied Algebra* 139.1-3 (June 1999), pages 61–88. DOI: 10.1016/S0022-4049(99)00005-5. URL: https://hal.science/hal-01148855 (cited on page 18).

[Fen23] Thibauld Feneuil. "Post-Quantum Signatures from Secure Multiparty Computation". Thèse de doctorat dirigée par Bajard, Jean-ClaudeJoux, Antoine et Rivain, Matthieu Informatique Sorbonne université 2023. PhD thesis. 2023. URL: http://www.theses.fr/2023SORUS324 (cited on pages 6, 28, 30, 31).

[FSS11] Jean-Charles Faugère, Mohab Safey El Din, and Pierre-Jean Spaenlehauer. "Gröbner bases of bihomogeneous ideals generated by polynomials of bidegree (1,1): Algorithms and complexity". In: *Journal of Symbolic Computation* 46.4 (2011), pages 406–437. ISSN: 0747-7171. DOI: https://doi.org/10.1016/j.jsc.2010.10.014. URL: https://www.sciencedirect.com/science/article/pii/S0747717110001902 (cited on pages 5, 13, 15, 16, 18).

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to construct random functions". In: *J. ACM* 33.4 (Aug. 1986), 792–807. ISSN: 0004-5411. DOI: 10.1145/6490.6503. URL: https://doi.org/10.1145/6490.6503 (cited on page 3).

[Gro96] Lov K. Grover. *A fast quantum mechanical algorithm for database search.* 1996. arXiv: quant-ph/9605043 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/9605043 (cited on pages 2, 5).

[Guo+22] Xiaojie Guo, Kang Yang, Xiao Wang, Wenhao Zhang, Xiang Xie, Jiang Zhang, and Zheli Liu. *Half-Tree: Halving the Cost of Tree Expansion in COT and DPF.* Cryptology ePrint Archive, Paper 2022/1431. 2022. URL: https://eprint.iacr.org/2022/1431 (cited on page 4).

[HH21] David Harvey and Joris van der Hoeven. "Integer multiplication in time O(n log n)". In: *Annals of Mathematics* (Mar. 2021). DOI: 10.4007/annals.2021.193.2.4. URL: https://hal.science/hal-02070778 (cited on page 23).

[HJ23] Janik Huth and Antoine Joux. *MPC in the head using the subfield bilinear collision problem.* Cryptology ePrint Archive, Paper 2023/1685. 2023. URL: https://eprint.iacr.org/2023/1685 (cited on pages 2, 5, 7–9, 16, 28, 31).

[Ish+07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. "Zero-knowledge from secure multiparty computation". In: *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing.* STOC '07. San Diego, California, USA: Association for Computing Machinery, 2007, 21–30. ISBN: 9781595936318. DOI: 10.1145/1250790.1250794. URL: https://doi.org/10.1145/1250790.1250794 (cited on pages 2, 3).

[Jou13] Antoine Joux. *A new index calculus algorithm with complexity* $L(1/4+o(1))$ *in very small characteristic.* Cryptology ePrint Archive, Paper 2013/095. 2013. URL: https://eprint.iacr.org/2013/095 (cited on page 2).

[JV17] Antoine Joux and Vanessa Vitse. *A crossbred algorithm for solving Boolean polynomial systems.* Cryptology ePrint Archive, Paper 2017/372. 2017. URL: https://eprint.iacr.org/2017/372 (cited on pages 16, 17).

[Laz83] D. Lazard. "Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations". In: *Computer Algebra.* Edited by J. A. van Hulzen. Berlin, Heidelberg: Springer Berlin Heidelberg, 1983, pages 146–156. ISBN: 978-3-540-38756-5 (cited on page 9).

[Lok+17] Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, Ryan Williams, and Huacheng Yu. "Beating brute force for systems of polynomial equations over finite fields". In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms.* SODA '17. Barcelona, Spain: Society for Industrial and Applied Mathematics, 2017, 2190–2202 (cited on page 17).

[Mac16] F.S. Macaulay. *The Algebraic Theory of Modular Systems.* Cambridge tracts in mathematics and mathematical physics. University Press, 1916. URL: https://books.google.fr/books?id=uA7vAAAAMAAJ.

[NNY17] Ruben Niederhagen, Kai-Chun Ning, and Bo-Yin Yang. *Implementing Joux-Vitse's Crossbred Algorithm for Solving MQ Systems over GF(2) on GPUs.* Cryptology ePrint Archive, Paper 2017/1181. 2017. URL: https://eprint.iacr.org/2017/1181 (cited on page 21).

[Pé24] Pierre Pébereau. *Subfield attack: leveraging composite-degree extensions in the Quotient Ring transform.* Cryptology ePrint Archive, Paper 2024/196. 2024. URL: https://eprint.iacr.org/2024/196 (cited on page 21).

[RB25] Thibauld Feneuil Matthieu Rivain Ryad Benadjila Charles Bouillaguet. *MQOM: MQ on my Mind Algorithm Specifications and Supporting Documentation (Version 2.0).* Technical report. 2025 (cited on pages 21, 22).

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. "A method for obtaining digital signatures and public-key cryptosystems". In: *Commun. ACM* 21.2 (Feb. 1978), 120–126. ISSN: 0001-0782. DOI: `10.1145/359340.359342`. URL: `https://doi.org/10.1145/359340.359342` (cited on page 2).

[Sho97] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (Oct. 1997), 1484–1509. ISSN: 1095-7111. DOI: `10.1137/s0097539795293172`. URL: `http://dx.doi.org/10.1137/S0097539795293172` (cited on page 2).

[Spa12] Pierre-Jean Spaenlehauer. "Résolution de systèmes multi-homogènes et déterminantiels algorithmes - complexité - applications". Thèse de doctorat dirigée par Faugère, Jean-Charles Informatique Paris 6 2012. PhD thesis. 2012, 1 vol. (207 p.) URL: `http://www.theses.fr/2012PA066467` (cited on page 13).

[Tea17] The CADO-NFS Development Team. *CADO-NFS, An Implementation of the Number Field Sieve Algorithm.* Release 2.3.0. 2017. URL: `http://cado-nfs.inria.fr/` (cited on page 21).

[Theyy] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version x.y.z).* `https://www.sagemath.org`. YYYY (cited on page 16).

[VID24] Damien Vidal, Sorina Ionica, and Claire Delaplace. *An analysis of the Crossbred Algorithm for the MQ Problem.* Cryptology ePrint Archive, Paper 2024/992. 2024. DOI: `https://doi.org/10.62056/ak86cy7qiu`. URL: `https://eprint.iacr.org/2024/992` (cited on pages 19, 20).

[Wil+23] Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. *New Bounds for Matrix Multiplication: from Alpha to Omega.* 2023. arXiv: `2307.07970 [cs.DS]`. URL: `https://arxiv.org/abs/2307.07970` (cited on page 23).

[Yas15] Takanori Yasuda. *Fukuoka MQ Challenge.* 2015. URL: `https://www.mqchallenge.org/` (cited on page 17).

A Signature protocol

Here we will show to signature protocol adapted from the identification protocol from [HJ23] in the same format as in [Fen23] as a 5-round protocol to be able to apply his theorem on EUF-CMA security proof of such protocol.

To simplify the notation, we omit the round index when it is clear from context. Unless stated otherwise, all sharing-related elements are defined per round $e \in [1, \tau]$.

We denote by $\mathcal{K}^e$ the punctured key corresponding to round e. This punctured key represents the co-path in the GGM tree used to derive the associated random values.

The function Expand is a generic function, that expands the hash to extract different psuedo-random values from it.

Protocol 1: Signature scheme - Signing Algorithm

Inputs: A public key $(\vec{u}, \vec{v})$, a private key $(\vec{x}, \vec{y}) \in \mathrm{NSBC}[\vec{u}, \vec{v}]$ and a message $m \in \{0,1\}^*$.

Phase 0: Initialization.

1. Sample a random salt $\texttt{salt} \leftarrow \{0,1\}^\lambda$.
2. Sample a root seed $\rho \leftarrow \{0,1\}^\lambda$.

Phase 1: Preparation of the MPC-in-the-Head inputs. For each iteration $e \in [1 : \tau]$:

1. Share the witness $(\vec{x}, \vec{y})$ into an $(N-1)$-private linear secret sharing $\vec{x}^{\llbracket i \rrbracket}, \vec{y}^{\llbracket i \rrbracket}$:
   $\vec{R}_x^{\llbracket i \rrbracket} = \mathrm{PRG}(\texttt{"}R_x\ \texttt{i"} \parallel \mathit{salt} \parallel e \parallel \rho)$
   $\vec{R}_y^{\llbracket i \rrbracket} = \mathrm{PRG}(\texttt{"}R_y\ \texttt{i} \parallel \mathit{salt} \parallel e \parallel \rho)$
   $\vec{\delta}_x \leftarrow \vec{x} - \sum_{i=1}^{N} \vec{R}_x^{\llbracket i \rrbracket}$.
   $\vec{\delta}_y \leftarrow \vec{y} - \sum_{i=1}^{N} \vec{R}_y^{\llbracket i \rrbracket}$.
2. Prepare MPC protocol inputs
   $X_1^{\llbracket i \rrbracket} = \mathrm{PRG}(\texttt{"}X_1\ \texttt{i} \parallel \mathit{salt} \parallel e \parallel \rho)$
   $X_2^{\llbracket i \rrbracket} = \mathrm{PRG}(\texttt{"}X_2\ \texttt{i} \parallel \mathit{salt} \parallel e \parallel \rho)$
   $Y_1^{\llbracket i \rrbracket} = \mathrm{PRG}(\texttt{"}Y_1\ \texttt{i} \parallel \mathit{salt} \parallel e \parallel \rho)$
   $Y_2^{\llbracket i \rrbracket} = \mathrm{PRG}(\texttt{"}Y_2\ \texttt{i} \parallel \mathit{salt} \parallel e \parallel \rho)$
   $R_A^{\llbracket i \rrbracket} = \mathrm{PRG}(\texttt{"}R_A\ \texttt{i} \parallel \mathit{salt} \parallel e \parallel \rho)$
   $R_B^{\llbracket i \rrbracket} = \mathrm{PRG}(\texttt{"}R_B\ \texttt{i} \parallel \mathit{salt} \parallel e \parallel \rho)$
3. Compute X_1, X_2, Y_1, Y_2
   $X_1 \leftarrow \sum_{i=1}^{N} X_1^{\llbracket i \rrbracket}$.
   $X_2 \leftarrow \sum_{i=1}^{N} X_2^{\llbracket i \rrbracket}$.
   $Y_1 \leftarrow \sum_{i=1}^{N} Y_1^{\llbracket i \rrbracket}$.
   $Y_2 \leftarrow \sum_{i=1}^{N} Y_2^{\llbracket i \rrbracket}$.
4. Compute A and B and share them:
   $A \leftarrow X_1 Y_1 - X_2 Y_2$.
   $B \leftarrow X_1(\vec{v} \cdot \vec{y}) + Y_1(\vec{u} \cdot \vec{x}) - X_2(\vec{u} \cdot \vec{y}) - Y_2(\vec{v} \cdot \vec{x})$.
   $\delta_A \leftarrow A - \sum_{i=1}^{N} R_A^{\llbracket i \rrbracket}$.
   $\delta_B \leftarrow B - \sum_{i=1}^{N} R_B^{\llbracket i \rrbracket}$.

Phase 2: First challenge (randomness for the MPC protocol).

1. Compute random value $t_0^{\llbracket i \rrbracket}$ for each round and each party.
   $t_0^{\llbracket i \rrbracket} \leftarrow Hash_0(i \parallel \vec{R}_x^{\llbracket i \rrbracket}, \vec{R}_y^{\llbracket i \rrbracket}, R_A^{\llbracket i \rrbracket}, R_B^{\llbracket i \rrbracket}, X_1^{\llbracket i \rrbracket}, X_2^{\llbracket i \rrbracket}, Y_1^{\llbracket i \rrbracket}, Y_2^{\llbracket i \rrbracket} \parallel \vec{\delta}_x^{\llbracket i \rrbracket}, \vec{\delta}_y^{\llbracket i \rrbracket}, \delta_A^{\llbracket i \rrbracket}, \delta_B^{\llbracket i \rrbracket})$.
2. Compute $t_0 \leftarrow \sum_{i=1}^{N} t_0^{\llbracket i \rrbracket}$.

Phase 3: Simulation of the MPC protocol. For each $e \in [1 : \tau]$:

1. Compute the commitments
   $View_i \leftarrow (t_0^{\llbracket i \rrbracket}, (X_1 + t_0(\vec{u} \cdot \vec{x}))^{\llbracket i \rrbracket}, (X_2 + t_0(\vec{v} \cdot \vec{x}))^{\llbracket i \rrbracket}, (Y_1 + t_0(\vec{v} \cdot \vec{y}))^{\llbracket i \rrbracket}, (Y_2 + t_0(\vec{u} \cdot \vec{y}))^{\llbracket i \rrbracket}, (A + t_0 B)^{\llbracket i \rrbracket})$.
   $\texttt{com}^{[e]} := \mathrm{Hash}_0(e, View_1, \ldots, View_N, \vec{\delta}_x, \vec{\delta}_y, \delta_A, \delta_B)$

Phase 4: Second challenge (parties to be opened).

1. Compute $h := \text{Hash}(m, \texttt{salt}, \texttt{comm}^{[1]}, \dots, \texttt{comm}^{[\tau]})$.
2. Expand h as $(I^{[e]})_{e\in[1:\tau]} \leftarrow \text{Expand}(h)$ where, for every $e, I^{[e]} \in [1:N]$ the party that will not be opened.

Phase 5: Building of the signature. Output the signature σ built as:

$$\sigma := \texttt{salt} \parallel h \parallel \mathit{punctured\_keys} \parallel [\delta_A^{[e]}]_{e\in[1,\tau]} \parallel [\delta_B^{[e]}]_{e\in[1,\tau]} \parallel$$

$$[\vec{\delta}_x^{[e]}]_{e\in[1,\tau]} \parallel [\vec{\delta}_y^{[e]}]_{e\in[1,\tau]} \parallel [(X_1 + t_0(\vec{u}\cdot\vec{x}))^{[e]}]_{e\in[1,\tau]} \parallel [(X_2 + t_0(\vec{v}\cdot\vec{x}))^{[e]}]_{e\in[1,\tau]} \parallel$$

$$[(Y_1 + t_0(\vec{v}\cdot\vec{y}))^{[e]}]_{e\in[1,\tau]} \parallel [(Y_2 + t_0(\vec{u}\cdot\vec{y}))^{[e]}]_{e\in[1,\tau]}$$

Protocol 2: Signature scheme - Verification Algorithm

Inputs: A public key $(\vec{u}, \vec{v})$, a message $m \in \{0,1\}^*$ and a signature σ.

1. Parse the signature σ as

$$\texttt{salt} \parallel h \parallel \mathit{punctured\_keys} \parallel [\delta_A^{[e]}]_{e\in[1,\tau]} \parallel [\delta_B^{[e]}]_{e\in[1,\tau]} \parallel$$

$$[\vec{\delta}_x^{[e]}]_{e\in[1,\tau]} \parallel [\vec{\delta}_y^{[e]}]_{e\in[1,\tau]} \parallel [(X_1 + t_0(\vec{u}\cdot\vec{x}))^{[e]}]_{e\in[1,\tau]} \parallel [(X_2 + t_0(\vec{v}\cdot\vec{x}))^{[e]}]_{e\in[1,\tau]} \parallel$$

$$[(Y_1 + t_0(\vec{v}\cdot\vec{y}))^{[e]}]_{e\in[1,\tau]} \parallel [(Y_2 + t_0(\vec{u}\cdot\vec{y}))^{[e]}]_{e\in[1,\tau]}$$

2. Expand h as $(I^{[e]})_{e\in[1:\tau]} \leftarrow \text{Expand}(h)$ where, for every $e, I^{[e]} \in [1:N]$ the party that we do not have.

B Security proof

Proof of Lemma 4.1:

Proof. We need to show that if a $\mathcal{PPT}$ adversary $\mathcal{A}$ can retrieve the private key from the public key, then we can break the SBC function. It is straightforward to show, because if $\mathcal{A}$ can retrieve the private key $(x, y) \in \mathbb{F}_q^{2n}$ of a given valid public key $(u, v) \in \mathbb{F}_{q^k}^{2n}$ then no need to convert anything because $(x, y) \in \text{SBC}[u, v]$ directly. So the signature scheme is **UB-KOA** if the SBC function is (τ, ϵ)-hard. □

We rely on the following theorem from Thibault Feneuil's PhD thesis [Fen23], which states that any 5-round signing protocol instantiated within the MPC-in-the-Head (MPCitH) framework is **EUF-CMA** secure, under certain assumptions specified in the theorem.

This type of proof typically proceeds by simulating a game in which an adversary interacts with a signing oracle that responds to arbitrary signing queries. The adversary's goal is to produce a valid signature on a message that was not previously queried to the oracle. A signature scheme is said to be **EUF-CMA** (Existential Unforgeability under Chosen Message Attack) if the probability that the adversary succeeds in forging such a signature is at most $2^{-\lambda}$, where λ denotes the security parameter. This probability is referred to as the adversary's advantage.

Theorem B.1 ([Fen23])**.** *Let $Hash_0, Hash_1, Hash_m$ and Expand be modeled as random oracles, and let (N, τ, λ, p) be parameters of the signature scheme. Let $\mathcal{A}$ be an adversary against the* ***EUF-CMA*** *security of the scheme running in time $t_{\mathcal{A}}$ and making a total of Q_{RO} random oracle queries and Q_{Sig} signing queries. Assuming that F is a $(t_{\mathcal{A}}, \epsilon_{OWF})$-hard one-way function and that PRG is a $(t_{\mathcal{A}}, \epsilon_{PRG})$-secure psuedorandom generator then $\mathcal{A}$'s advantage in the* ***EUF-CMA*** *game is*

$$\epsilon_{EUF-CMA} \leq \epsilon_{OWF} + \epsilon_{PRG} + \frac{(\tau N + 2)Q^2}{2^{2\lambda}} + Pr[X + Y = \tau],$$

where $Q = Q_{RO} + N_{Hash} \cdot Q_{Sig}$ with $N_{Hash} = 2 + \tau(2N - 1)$ the number of hash computations in a signature generation, and where X, Y are defined as

- $X = max_{q1 \in \mathcal{Q}_1}\{X_{q1}\}$ *with* $X_{q1} \sim \mathcal{B}\left(\tau, \binom{N}{l+1}p\right)$ *and*
- $Y = max_{q2 \in \mathcal{Q}_2}\{Y_{q2}\}$ *with* $Y_{q2} \sim \mathcal{B}\left(\tau - X, 1/\binom{N}{l}\right)$,

where $\mathcal{B}(n_0, p_0)$ denotes the binomial distribution with n_0 the number of trials and p_0 the success probability of each trial.

Proof of theorem 4.4:

Proof. We model the SBC signature scheme [HJ23] as a 5-round protocol (see Protocol A). By applying Theorem B.1 from [Fen23], and under the assumption that the underlying PRG is a $(t_{\mathcal{A}}, \epsilon_{\text{PRG}})$-secure pseudorandom generator, and as established in Lemma 4.1 that the SBC function is a $(t_{\mathcal{A}}, \epsilon_{\text{OWF}})$-secure one-way function, we conclude that the SBC signature scheme achieves *EUF-CMA* security. □