UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria

Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

---

Analisi e ottimizzazione di uno schema di firma digitale basato su codici derivato dal problema di decodifica della sindrome nel caso ristretto

Analysis and Optimisation of a code-based digital signature scheme derived from the Restricted Syndrome Decoding Problem

Candidato:

Riccardo Schiavoni

Relatore:

Prof. Marco Baldi

Correlatore:

Prof. Paolo Santini

Anno Accademico 2023-2024

Abstract

This thesis provides a better understanding of the security of CROSS (Codes and Restricted Objects Signature Scheme), a digital signature scheme derived from an interactive Zero Knowledge Identification Protocol (ZKID) by applying the Fiat-Shamir Transformation.

CROSS is currently in the NIST competition for the standardization of a quantum-resistant digital signature scheme. In particular, in this thesis we focus on the security of the problem underlying CROSS, namely the Restricted Syndrome Decoding Problem (R-SDP). This is a problem based on the decoding of linear codes proven to be NP-hard.

However, R-SDP is a relatively new problem and, therefore, its security needs to be carefully studied.

The contribution of the following work is to study a possible application of the Locality Sensitive Framework, an approach for solving the Nearest Neighbour search problem in the case of binary vectors, to the best solvers for R-SDP which are, to date, the algorithms based on Information Set Decoding (ISD).

The result of this analysis is the realization of a new attack for R-SDP. By formulating a theoretical cost for this new approach, it was possible to evaluate its performance on different instances of the restricted problem.

The results obtained in this thesis allow for two important interpretations: on the one hand, the cryptanalysis of some schemes in the literature based on R-SDP is improved by reducing their security level, and on the other hand, although the performance of existing attacks is improved, it can be established that CROSS, as designed, achieves the NIST security level I even for this new attack.

Sommario

In questa tesi viene fornita una migliore compresione della sicurezza di CROSS (Codes and Restricted Objects Signature Scheme), uno schema di firma digitale derivato da un protocollo interattivo di identificazione Zero Knowledge (ZKID) applicando la Trasformazione di Fiat-Shamir.

CROSS è attualmente in gara nella competizione indetta dal NIST per la standizzazione di uno schema di firma digitale che sia sicuro in un'ottica post-quantum.

In particolare in questa tesi ci si focalizza sulla sicurezza del problema alla base di CROSS, ovvero il Restricted Syndrome Decoding Problem (R-SDP). Si tratta di un problema basato sulla decodifica di codici lineari dimostrato essere NP-hard. Tuttavia R-SDP è un problema relativamente nuovo e, pertanto, la sua sicurezza ha bisogno di essere studiata attentamente.

Il contributo del seguente lavoro è quello di studiare una possibile applicazione del Locality Sensitive Framework, un approccio per la risoluzione del Nearest Neighbor search problem nel caso di vettori binari, ai migliori risolutori per R-SDP che sono, ad oggi, gli algoritmi della famiglia Information Set Decoding (ISD).

Il risultato di questa analisi è stato la realizzazione di un nuovo attacco per R-SDP. Tramite la formulazione di un costo teorico per questo nuovo approccio è stato possibile valutarne le prestazioni su diverse istanze del problema ristretto.

I risultati ottenuti in questa tesi sono soggetti ad una duplice interpretazione: se da una parte la crittoanalisi di alcuni schemi presenti in letteratura basati su R-SDP viene migliorata riducendo il loro security level, dall'altra, pur avendo ottenuto un miglioramento in termini di performance degli attacchi esistenti, è possibile stabilire che CROSS, anche di fronte a questo nuovo approccio, risulta raggiungere i criteri di sicurezza stabiliti dal NIST.

Contents

List of Figures

List of Tables

Introduction

Cryptographic algorithms are widely employed in a variety of protocols and applications and are a fundamental asset to protect digital communications from attackers.

A very common family of cryptographic algorithms is the one normally called *public key cryptography*, which comprehends all algorithms which require an asymmetric key pair: a public key and a secret key, mathematically linked but different and with different purposes. One of the most important primitives is that of *digital signatures*, which allows to verify the authenticity of digital messages or documents.

Digital signatures provide relevant security properties such as integrity and non repudiability and are widely employed in all the protocols in which authentication is required (e.g., signatures are used to build public key certificates).

This state of affairs is currently threatened by the upcoming advent of *quantum computers*. These devices, first theorized by Feynman e Manin in the early '80 [1], [2], exploiting the phenomena of quantum mechanics, open up for the possibility of devising algorithms that follow rules which are fundamentally different from the principles of classical computation.

In particular, quantum information theory is based on the representation of data through quantum states of matter.

While a classical bit can assume two states, 0 and 1, which are mutually exclusive, a qubit, exploiting a quantum principle known as quantum superposition, can simultaneously assume the values 0 and 1, however, the value that is returned to only one of the two classical states by the effect of any measurement.

As long as the superposition state is maintained, a sequence of n qubit therefore has the capacity to represent all 2^n combinations, that would instead require 2^n bit.

This property makes it possible to speed up the execution of certain algorithms significantly, bringing their complexity from being exponential in input length when

performed on a classical computer to becoming polynomial in input length on a quantum computer.

Arguably, the most famous quantum algorithm is Shor's algorithm [3]. It can be used to factor large integers and, with some variations, can also find a solution to the discrete logarithm problems.

Such problems are at the base of the most employed digital signature schemes, RSA and ECDSA. This implies that, as soon as sufficiently large quantum computers become available, RSA and ECDSA will become obsolete, as using Shor's algorithm an attacker can retrieve the secret key.

The upcoming quantum threat is pushing the cryptographic community to the development of the so-called *post-quantum cryptography*, that is, the new generation of cryptographic algorithms, capable of resisting against quantum attacks.

The choice to use post-quantum cryptography is still preferable to quantum cryptographic solutions because they are subject to significant limitations, making them practically usable only in very limited scenarios.

Notably, these algorithms are required to run on classical devices and must be based on mathematical problems which cannot be solved efficiently by a quantum computer.

The emblema of this effort is represented by the NIST process for post-quantum cryptography standardization. Up to now, NIST has issued two calls.

The first one, announced in 2016 and started in 2017, is almost concluded and has already produced four standard algorithms, one for key encapsulation and three for digital signatures. However, the situation with digital signatures is not extremely satisfying as two of the selected algorithms (Dilithium and Falcon) are based on basically the same problem, while the third scheme (SPHINCS+) is extremely conservative and has poor performances.

For these reasons, NIST has issued a second call, specifically tailored to the proposal of digital signatures.

The new competition has started in June 2023 and more than 40 algorithms have been deemed as complete and, thus, accepted for being evaluated in the competition.

CROSS: signatures from the Restricted Syndrome Decoding Problem

CROSS is a signature scheme derived from a interactive Zero Knowledge Identification protocol by applying the Fiat-Shamir Trasformation.

The underlying principle is that a user, called prover, who wants to prove his identity to a second user, called verifier, must prove that he knows the secret key associated with his public key, but without revealing it.

Specifically, the verification of the identity of the prover is based on the ability to know how to correctly answer random challenges made by the verifier, which can only be answered correctly if the value of the secret key is known.

In particular, a ID Scheme must guarantee the following properties:

- Completeness: an honest prover has to be always accepted.
- Soundness: an user who does not know the value of the secret key should not be able to answer the challenges correctly.
- Zero Knowledge: the prover must not reveal information about the secret key.

The perfect soundness is difficult to achieve; in fact, there is certainly a probability that an attacker will be able to execute the protocol correctly without knowing the secret key. This probability is called the soundness error.

Cross is based on the so-called Restricted Syndrome Decoding Problem (R-SDP), an NP-complete problem that can be seen as a variant of the classical Syndrome Decoding Problem (SDP).

The difference from the original SDP defined over a finite filed $\mathbb{F}_q$ is that we consider an additional restriction: the entries of the solution error vectors are restricted to those living in a multiplicative subgroup $\mathbb{E} \in \mathbb{F}_q*$ of order $z \leq q-1$.

When using $R-SDP$ instead SDP the cost of ISD algorithms, the state-of-the-art ot the solvers for decoding problems, increases.

This allows to select smaller parameters to attain the desired security levels, positively impacting both signature sizes and computational complexity.

This allows to select smaller parameters to achieve the desired security levels, positively impacting both signature sizes and computational complexity, placing CROSS among the fastest schemes in the new NIST competition.

However, the $R-SDP$ is a rather new problem, many concerns arise about is actual security, then understanding the security of the recommended instances appears to be a major need.

Our contribution

In this thesis we improve our understanding about the security of $R-SDP$ and CROSS.

The security of any asymmetric cryptographic scheme relies both on an underlying problem, which is supposedly difficult to solve, and on the structure of the protocol used.

Generally, two main attack approaches are considered in security analysis:

- Structural Attacks: when exploiting the algebraic structure of the cryptosystem.
- Non-Structural Attacks: combinatorially recover the secret key without exploiting any algebraic structure.

In the next, we will focus on non-structural attack in order determine the security lever for a given choice of parameters choice of parameters for a given security level. We will start from solvers for the decoding problem in the Hamming metric realized via ISD framework and design a new algorithm to solve $R-SDP$.

In particular, we will use the approach of solving the Nearest Neighbour Search problem through locality sensitive functions. The NN Search asks to find two binary vectors of weight w_1 and w_2 respectively, such that their sum returns a restricted binary vector of weight $w_3 < w_1 + w_2$.

Trivially, a pair of solution vectors should be such that their supports must overlap in exactly $o = w_1 + w_2 - w_3$ inputs. By using locality sensitive function we are going to search for solutions among all pairs of vectors that in certain fractions of their coordinates overlap in exactly $o' \leq o$ inputs.

Seeking a solution for SDP by solving the NN search problem was proposed in the binary case in [4], but through the use of different approaches. We will show how LSF can be adapted to ISD solvers for SDP in the restricted case.

We test the performances of our algorithms and compare with the previously known best solver for $R-SDP$, namely, $BJMM$. We show that our algorithm can,

in many cases, outperform significantly the $BJMM$ algorithm. We show this by considering some of the instances that have already been attacked in [5].

Our algorithm is significantly faster than previously existing approaches (up to 2^{20} times) and, de facto, reduces further the security level of the considered schemes.

Then, we test our algorithm on the proposed CROSS instances. We show that these instances are not threatened by the new algorithm, as the running time is always larger than the claimed security level. We see as a positive result, as CROSS instances has been designed by taking into account conservative criteria (e.g., values of z that are not too small). The obtained results confirm that this choices has been wise and, de facto, give another solid confirmation about the security of CROSS.

Chapter 1

Notations

1.1 Algebraic Notation

We denote as $[a; b]$ the range of real numbers $x \in \mathbb{R}$ such that $a \leq x \leq b$. When considering a generic set as A we denote its cardinality as $|A|$ and the complete set including the value 0 as $A_0 = A \cup \{0\}$. A random uniform extraction of an element from a set A is denoted by $a \xleftarrow{\$} A$. Let p a prime number: we will represent by $\mathbb{F}_p$ the finite field of order p and by $\mathbb{F}_p^*$ its multiplicative group. We denote as $ord(g)$ the multiplicative order of an element $g \in \mathbb{F}_p^*$.The use of uppercase letters will be used to define matrices, and lowercase letters for vectors. For a generic vector v of length n and a matrix with m rows and n columns, for a given set $j \subseteq \{\ , \ldots, n\}$. The identity matrix of dimension m is denoted as Id_m

1.2 Cryptographic Notation

As for conventional cryptographic notations, first of all we denote as λ the security parameter expressed in bit, then we will introduce all the cryptographic structures:

- Hash:$\{0,1\}^* \rightarrow \{0,1\}^{2\lambda}$, which is a secure cryptographic hash function that takes an input of arbitrary size and produces a fixed-length digest of 2λ bits.
- $\mathsf{MerkleTree}$, constructed from t elements $(a^{(1)}, \ldots, a^{(t)})$ that form the leaves of the tree, denoted as T=$\mathsf{MerkleTree}(a^{(1)}, \ldots, a^{(t)})$. The root of the tree is extracted using the $\mathsf{T.Root()}$ method. The function for composing the Merkle Proof, starting from the leaves indicated by the set J, is $\mathsf{T.MerkleProof}(J)$.

The reconstruction of the root, starting from the leaves $a^{(i)}i \in J$ and the $\texttt{MerkleProof}$, is achieved through $\mathsf{VerifyMerkleRoot}(c_1^{(i)}i \notin J, \texttt{MerkleProof})$. The hash function used for constructing the tree produces digests of length 2λ since, within the structure, each leaf is a binary string of precisely 2λ bits.

- $\mathsf{SeedTree}$, created from the root $\texttt{Root}$. The function that composes the t leaves $(\texttt{Seed}^{(1)}, \ldots, \texttt{Seed}^{(t)})$ is $\mathsf{SeedTree}(\texttt{Root})$. All generated seeds have a length of λ. To construct the data structure $\texttt{SeedPath}$, which is used to regenerate the seeds indexed by the set J with $\mathsf{GetSeeds}(\texttt{SeedPath}, J)$, the function $\mathsf{SeedPath}(\texttt{Root}, J)$ is used. Additionally, it is possible to include a $\texttt{Salt}$ string of length 2λ to enhance complexity and security in the data structure.

Finally, the notation $a \xleftarrow{\text{Seed}} A$ indicates that a is sampled using a cryptographically secure deterministic random generator, which outputs elements uniformly chosen from A, and is initialized with the input $\texttt{Seed}$.

Chapter 2

Preliminaries

In this chapter we recall useful background notions about coding theory, linear codes and their use in cryptography.

2.1 Algebraic Coding Theory

For $p \in \mathbb{N}$ a prime power (i.e., $p = p^m$ for p a prime and m an integer), we denote by $\mathbb{F}_p$ the finite field with p elements. In this thesis we focus only on the case in which p is a prime, i.e., $m = 1$.

Definition 2.1.1. Linear Code Let $1 \leq k \leq n$ be integers, an $[n, k]$ linear code C over $\mathbb{F}_p$ is a k-dimensional linear subspace of $\mathbb{F}_p^n$.

The elements in the code are referred to as codewords and $R = \frac{k}{n}$ is the rate of the code. One can represent a linear code either through a generator matrix G, i.e. $C = \{uG \,|\, u \in \mathbb{F}_p^k\}$, or as the kernel of $(n-k) \times n$ parity check matrix H, i.e. $C = \{c \in \mathbb{F}_p^n \,:\, Hc = 0\}$.

In order to measure how far apart two vectors are, we endow $\mathbb{F}_p$ with a metric. The most famous and employed metric is the so-called Hamming metric.

Definition 2.1.2. Hamming Weight: For $x \in \mathbb{F}_p^n$, with n positive integer, the weight of x is given by the size of its support, i.e.,

$$wt_H = |\{i \in 1, ..., n \,|\, x_i \neq 0\}|$$

Through for $x, y \in \mathbb{F}_p^n$, the Hamming distance between x and y is given by the number of positions in which they differ, i.e.,

$$d_H(x, y) = |\{i \in \{1, ..., n\} \mid x_i \neq y_i\}|$$

One can derive the Hamming distance also from the Hamming weight, that is $d_H(x, y) = wt_H(x - y)$.

Once this metric has been defined, it is possible to consider the minimum distance of a code, which is the smallest distance achieved by any pair of distinct codewords.

Definition 2.1.3. Minimum Distance. Let C be a code over $\mathbb{F}_p$, the minimum Hamming distance of a code C is defined as follow:

$$d(C) = min\{d(x, y) \mid x, y \in C, x \neq y\}$$

Or, alternatively, as:

$$d(C) = min\{wt_H(x) \mid x \in C, x \neq 0\}$$

Through the minimum distance of a code one can define the error correction capacity of the code. We say that a code can correct up to t error, if for all $x \in \mathbb{F}_p^n$ with $d_H(x, C) \leq t$, there exists exactly one $y \in C$ such that $d_H(x, y) \leq t$.

$$\text{A code } C \text{ can correct up to } t := \lfloor \tfrac{d-1}{1} \rfloor \text{ errors.}$$

We denote as Hamming Sphere the set of all vectors of length n with entries living in $\mathbb{F}_p$ which have exactly Hamming weight r, i.e.,

$$\mathbb{W}_{n,r}^p := \{x \in \mathbb{F}_p^n \,:\, wt_H(x) = r\}$$

Futhermore, we define the Hamming Ball of radius r as the set of all vectors with entries living in $\mathbb{F}_p$ which have at most Hamming weight r, i.e.,

$$\mathbb{B}_p(n, r) := \{x \in \mathbb{F}_p^n \,:\, wt_H(x) \leq r\}$$

The Volume of $\mathbb{B}_p(n, r)$ is defined as follows:

$$Vol_p(n, r) := \textstyle\sum_{i=1}^{r} \binom{n}{i}(p-1)^i.$$

Given a p -ary code of block length n and minimum distance d, it is important to estimate which is the largest possible size $A(n, d)$, i.e. the maximum number of codewords, that a code can have.

Given a p-ary code C (not necessarily linear) of length n and minimum distance d, it is important to estimate which is its largest possible size $A(n, d)$, i.e. the maximum number of codewords, that a code can have.

First, we construct a code C with minimum distance d and maximum size through a greedy procedure: starting from a codeword, we keep on adding codewords that have a distance at least d from the previously chosen ones, until we cannot add more of them. When this happens, the Hamming balls of radius $d-1$ centered at every codeword must cover the whole space. Indeed, otherwise, we could pick one more codeword whose distance from the others is at least d, thus the procedure would not have stopped.

When the greedy procedure stops it holds that

$$A_p(n, d) \cdot Vol_p(n, d-1) \geq p^n.$$

This inequality represents the so-called Gilbert-Varshamov bound, expressed as:.

Lemma 1. *Gilbert-Varshamov bound Let C be a code over $\mathbb{F}_p$ with minimum distance d. It must hold*

$$A_p(n, d) \geq \frac{p^n}{Vol_p(n, d-1)} = \frac{p^n}{\sum_{j=0}^{d-1} \binom{n}{j}(p-1)^j}. \tag{2.1}$$

When C is linear, then $A_p(n, d) = p^k$ for a positive integer $k \leq n$ and it holds

$$k \geq \left\lfloor n - \log_p(Vol_p(n, d-1)) \right\rfloor = \left\lfloor n - \log_p \left(\sum_{j=0}^{d-1} \binom{n}{j}(p-1)^j \right) \right\rfloor.$$

The GV bound represents a lower bound as a function of the field size p, the block length n and the minimum distance d. These results can be obtained in the asymptotic version, i.e., considering exponential approximations with regard to n of the used quantities.

We denote by $h_p : [0, 1] \to [0, 1]$ the p-ary entropy function:

$$h_p(x) = x \log_p(p-1) - x \log_p(x) - (1-x) \log_p(1-x)$$

And when $p = 2$ we have the binary entropy

$$h_p(x) = -x\log_2(x) - (1-x)\log_2(1-x)$$

Then, the following lemma can be proven:

Lemma 2. *Let n and $p \geq 2$ be positive integers and $\delta \in [0, 1-1/p]$ be a real number. It holds*

$$p^{(h_p(\delta)-o(1))n} \leq Vol(n, \delta n) \leq p^{h_p(\delta)n}$$

In other words, for n big enough, the asymptotic Volume of the Hamming ball of radius δn can be approximated to

$$\frac{1}{n}\log_p(Vol(n, \delta n)) = h_p(\delta). \tag{2.2}$$

The rate of a generic code C of size $|C|$ is defined as

$$R = \frac{1}{n}\log_p(|C|), \tag{2.3}$$

which becomes the well-known

$$R = \frac{k}{n}$$

if the code is linear. We want to take the asymptotic values of 2.1. To do so, we use the result of 2.1 and the definition given in 2.3, then we obtain the asymptotic formulation of GV bound, expressed as follows:

Lemma 3. *Asymptotic Gilbert-Varshamov bound*
Let p be a positive integer and $\delta \in [0, 1-1/p]$ be a real number. For every p and δ there exists an infinite family C of p-ary codes with rate

$$R \geq 1 - h_p(\delta) \tag{2.4}$$

The asymptotic GV bound states the existence of such codes, but doesn't give information about which ones they are. Consider a random linear code $[n, k]$ over $\mathbb{F}_p$, i.e., a code where each entry of its parity-check matrix (or, equivalently, generator

matrix) is picked uniformly at random. The number of vectors of weight t in $\mathbb{F}_p^n$ is

$$\binom{n}{t}(p-1)^t.$$

A vector $\mathbf{c}$ is a codeword if it satisfies the parity-check equations, i.e., $\mathbf{c}^\top H = 0$. Given that H is a random matrix, that happens with a p^{k-n} probability. Therefore, the average number of codewords of weight t is

$$\binom{n}{t}(p-1)^t p^{k-n} \tag{2.5}$$

If this amount is greater than 1, then a codeword with weight t exists on average. Then, by definition, the minimum distance d is the minimum value for which the 2.5 is greater than 1:

$$d = \min\left\{ t \,\middle|\, \binom{n}{t}(p-1)^t p^{k-n} \geq 1 \right\} \tag{2.6}$$

By taking the asymptotic results in 2.6 we obtain the relative distance

$$\delta = \min\left\{ T \,\middle|\, h_p(T) - (1-R) \geq 0 \right\} \tag{2.7}$$

where $T = t/n$ is the relative weight.

Considering that $h_p(T)$ grows monotonically from 0 to 1 for $T \in [0, 1-1/p]$, we can consider only this interval of weights. Then, the minimum relative distance is the one for which

$$h_p(\delta) = 1 - R \tag{2.8}$$

Therefore, random linear codes attain the asymptotic GV bound with high probability. Then, when using a random linear code we can assume its relative distance as the solution of 2.8.

2.2 Digital Signature Schemes

A digital signature scheme is a protocol for binding certain specific properties to a digital piece of information. Specifically, the source of a message applies a digital signature to the data sent, such that it ensures:

- Authenticity : the origin must be legit.
- Integrity : the message cannot be altered after it has been signed.
- Non-repudiation : the origin cannot deny the signing of the message.

Generally, the digital signature scheme is structured in steps in which the data source proves that such information has these properties, while the destination verifies them. For this reason we will call the two protocol actors *Prover* and $Verifier$. Specifically, these phases are:

1. Key generation: the *Signer* creates a pair of keys, public and private.
2. Signing: through a public-key cryptographic primitives a signature of the message is obtained.
3. Verification: the $Verifier$ receives the signed message and checks its validity through the same primitive.

The performance of a digital signature scheme is measured by:

- Signature size
- Public key size
- Signing and verification times

The goal is to find the right trade-off in reducing the size and time required while maintaining a certain level of security.

Zero Knowledge Identification Schemes

A Zero Knowledge identification scheme, also called Proof of Knowledge, is an interactive protocol in which a *Prover* P aims to prove to a *Verifier* V knowledge of a secret that verifies some public statement, without revealing it or giving information about it. In this case, only protocols that involve sending five messages will be examined, with the *Prover* always sending the first and last.

This type of protocols consists of 5 steps: P sends to V a commitment based on the values of the secret key s_k, next, through the exchange of 4 challenge-response messages based on two sets of random values, the identification process is realized.

PROVER		VERIFIER
Holds the secret key $\mathtt{sk}$		Knows the public key $\mathtt{pk}$
	$\xrightarrow{\mathtt{Com}}$	
		$\mathtt{Ch}_1 \xleftarrow{\$} C_1$
	$\xleftarrow{\mathtt{Ch}_1}$	
	$\xrightarrow{\mathtt{Rsp}_1}$	
		$\mathtt{Ch}_2 \xleftarrow{\$} C_2$
	$\xleftarrow{\mathtt{Ch}_2}$	
	$\xrightarrow{\mathtt{Rsp}_2}$	
		Return $\mathtt{Out} \in \{0;1\}$

A single execution of the identification process, keeping the order of interactions unchanged, can be summarized as:

$$T = (\mathtt{Com}, \mathtt{Ch}_1, \mathtt{Rsp}_1, \mathtt{Ch}_2, \mathtt{Rsp}_2).$$

We will call T as the transcript of the protocol, that is, a single correct and complete execution of the procedure.

Properties

A Zero Knowledge Identification Protocol must guarantee :

- *Completeness*: an honest *Prover* must always be accepted, this means that an execution started by a P who knows the secret always ends with a *Verifier* result of 1.
- *Zero Knowledge*: Interactions between P and V must not reveal information about the secret. This implies that knowing the challenge values a priori allows a malicious *Prover* to produce valid transcripts that are indistinguishable from those of an honest *Prover*.
- *Soundness*: when the *Verifier* is honest, and thus the challenges are sampled through uniform distributions on C_1 and C_2, a malevolent *Prover* (who does not know the secret) can convince the *Verifier* with a probability $\varepsilon < 1$. The quantity ε is called the *soundness error* and corresponds to the probability that a malevolent *Prover* can correctly guess which subset of challenges will

be chosen by the *Verifier* (in addition to a negligible quantity, related to the probability that the *Prover* succeeds in solving difficult problems).

We will consider t parallel executions of a 5-step protocol, characterized by a soundness error ε, thus obtaining a new scheme with soundness error ε^t. In order to distinguish data from different rounds, we use the notation $^{(i)}$ to denote the element of the i-th round.

Fiat-Shamir Trasformation

The Fiat-Shamir transformation [6] is a standard technique to turn an interactive ZK protocol into a signature scheme. The process aims to remove the interactions, with the *Prover* simulating the *Verifier* by generating the challenges as the output of some one-way function (e.g., a hash), using all the former messages as input. The message to be signed msg is provided as another input to the hash function; this way, the transcript becomes associated with msg. To sum up, the Fiat-Shamir transform operates as follows:

1. Commitment Generation:

$$\mathtt{Com} = (\mathtt{Com}^{(1)}, \cdots, \mathtt{Com}^{(t)})$$

2. First Challenge Generation:

$$\mathtt{Ch}_1 = (\mathtt{Ch}_1^{(1)}, \cdots, \mathtt{Ch}_1^{(t)}) = \mathtt{Hash}(\mathtt{msg}, \mathtt{Com});$$

3. First Response Computation:

$$\mathtt{Rsp}_1 = (\mathtt{Rsp}_1^{(1)}, \cdots, \mathtt{Rsp}_1^{(t)})$$

4. Second Challenge Generation:

$$\mathtt{Ch}_2 = (\mathtt{Ch}_2^{(1)}, \cdots, \mathtt{Ch}_2^{(t)}) = \mathtt{Hash}(\mathtt{msg}, \mathtt{Com}, \mathtt{Ch}_1, \mathtt{Rsp}_1);$$

5. Second Response Computation:

$$\mathtt{Rsp}_2 = (\mathtt{Rsp}_2^{(1)}, \cdots, \mathtt{Rsp}_2^{(t)}).$$

This transformations results in a signing algorithm which produces a digital signature by simulating the interaction of both P and V on t parallel executions of the protocol. The verification algorithm, on the other hand, simulates the checks performed by the $Verifier$ on the data received from the *Prover*. Depending on the transcript, the output may be 1 (if the signature is accepted) or 0 otherwise.

Intuitively, the resulting signature is secure since each challenge is generated through a pseudo-random one-way function, which receives as input the previous messages exchanged. Moreover, it is not possible to change the commitment $\texttt{Com}$ after the generation of the first challenge $\texttt{Ch}_1$, unless the attacker detects collisions in the hash function.

In order to reduce the signature size, challenges are usually omitted, as they can be regenerated during verification. In conclusion, the signature will be characterized by the following form:

$$\texttt{Sign} = (\texttt{Salt}, \texttt{Com}, \texttt{Rsp}_1, \texttt{Rsp}_2).$$

Commitment, Seed and Salt

Commitments are the means by which the identification and integrity of sent messages is ensured. They are typically implemented through hash functions: if the reference value is x, the *Prover* will compose and send $\texttt{Com} = \texttt{Hash}(x)$. However, the functions in question will have to ensure the following characteristics to be considered valid:

- *Hiding*: given the output $\texttt{Com}$ no information can be derived about the input x;
- *Binding*: it must be computationally impossible to find two distinct messages $x \neq x'$ that correspond to the same output $\texttt{Com}$. In other words, the *Prover* cannot modify x after it has chosen it as its reference.

Moreover, the *Prover* reference value is typically a seed of length λ. If certain precautions are not taken, it is possible to find collisions in the commitments in time $O(2^{\frac{\lambda}{2}})$. In this case an attacker can find consistent values, without the *Prover* propagating them. To avoid this, it suffices to sample, for each new signature generated, a new $\texttt{Salt}$ of size 2λ, which will be used as an additional input to the

hash function with the following criterion:

$$\texttt{Com} = \texttt{Hash}(x, \texttt{Salt}).$$

Security Assumptions

Using Fiat-Shamir transformations to a ZK identification protocol, with soundness error equal to ε, leads to obtaining a signature scheme that admits forgery attacks in time $O(\varepsilon^{-1})$. Typically, a malicious *Prover* can repeatedly simulate the protocol by trying to guess the challenge values and preparing commitments with consistent responses to them. The challenge values will be related to the malicious *Prover*'s attempts with probability equal to ε; this means that on average the attempts to compromise the execution must be ε^{-1}.

If we consider a signature scheme with soundness error ε, characterized by t parallel executions, the resulting cost for a forgery attack is $O(\varepsilon^{-t})$. If we follow the well-known heuristic-ε^{-t}, it is sufficient to choose t such that $\varepsilon^{-t} > 2^{\lambda}$. The only issue is related to the fact that, in the case of 5-step schemes, some instances of the protocol may fail; therefore, a complexity level of 2^{λ} cannot always be guaranteed.

In [7] the formulas for characterizing the cost of forgery attacks are further discussed. Since C_1 and C_2 are essentially fixed, based on the ZK protocol on which the scheme relies, the value of t should be chosen so that the cost is above 2^{λ}. In practice, if we rely on the bindings defined in the documentation, the actual value of t is larger than that given by the heuristic-ε^{-t}.

Additional attacks are instead based on the idea of acquiring information and exploiting the non-interactivity of the scheme to produce valid transcripts, with a higher probability of success than ε^{t}.

Below, it can be demonstrated how the signature scheme obtained from the Fiat-Shamir paradigm achieves EUF-CMA (Existential Unforgeability Under Chosen Message Attack) security, through a Zero Knowledge identification protocol with specific properties.

CROSS is based on a ZK protocol which follows the structure of q 2- Identification schemes

Lemma 4. *A ZK protocol is classified as a q 2-Identification scheme when it possesses*

the following properties:

- $|C_1| = q$*;*
- $|C_2| = 2$*;*
- *The probability of the* `Com` *assuming a given detectable value is a negligible quantity considered in the safety parameter* λ*.*

It can be shown how the execution of t instances in parallel of a ZK $q2$ protocol, transformed using the Fiat-Shamir technique, leads to a signature scheme with EUF-CMA security, which is necessary to validate the digital signature scheme against the Quantum Random Oracle Model. The proof of the claim is based on the existence of a $q2$ solver, i.e., a probabilistic polynomial algorithm ξ that computes, with non-negligible probability of success equal to $1-\varepsilon$, the secret key `sk`, given four valid transcripts of the type:

$$T = (\mathtt{Com}, \mathtt{Ch}_1, \mathtt{Rsp}_1, \mathtt{Ch}_2, \mathtt{Rsp}_2),$$

$$T' = (\mathtt{Com}, \mathtt{Ch}'_1, \mathtt{Rsp}'_1, \mathtt{Ch}'_2, \mathtt{Rsp}'_2),$$

$$T'' = (\mathtt{Com}, \mathtt{Ch}''_1, \mathtt{Rsp}''_1, \mathtt{Ch}''_2, \mathtt{Rsp}''_2),$$

$$T''' = (\mathtt{Com}, \mathtt{Ch}'''_1, \mathtt{Rsp}'''_1, \mathtt{Ch}'''_2, \mathtt{Rsp}'''_2),$$

con

$$\mathtt{Ch}_1 = \mathtt{Ch}''_1 \neq \mathtt{Ch}'_1 = \mathtt{Ch}'''_1,$$

$$\mathtt{Ch}_2 = \mathtt{Ch}''_2 \neq \mathtt{Ch}'_2 = \mathtt{Ch}'''_2.$$

So, as far as $q2$ identification patterns are concerned, proving the existence of a $q2$ solver turns out to be identical to showing that the protocol is in the form (2,2)-out-of-(q,2) special sound, which implies soundness error equal to:

$$\varepsilon = 1 - \Big(1 - \frac{1}{q}\Big)\Big(1 - \frac{1}{2}\Big) = \frac{q+1}{2q}.$$

Let it be, finally, known how the choice of a suitable value for t parallel repetitions is of paramount importance and requires to consider thoroughly the cost of the forgery attacks described in [7].

2.3 Code-Based Problems

A decoding problem asks, on input some $x \in \mathbb{F}_p^n$ and a code C, to find a codeword $c \in C$ which is sufficiently similar to x. Here, sufficiently similar is a rather vague term, but we used it purposely as the possible requirements are so heterogeneous that there is not quantitative word to classify all of them. For instance, when the Hamming metric is employed, the problem asks to find the codeword minimizing the hamming distance from the input x. In other words, the problem asks to find the codeword having the largest number of common entries with the input x.

This problem goes by the name of Syndrome Decoding Problem (SDP), as it can be easily seen that it corresponds to finding the vector $e \in \mathbb{F}_p^n$ with lowest weight, such that $Hx^\top = He^\top$. The vector $Hx^\top$ is normally called syndrome, so that the problem is normally stated as follows.

Definition 2.3.1. Syndrome Decoding Problem: Given $H \in F_p^{(n-k)\times n}$, $s \in \mathbb{F}_p^{n-k}$ and an integer $t > 0$, find a vector $e \in F_p^n$ such that $wt_H(e) \leq t$ and $eH^T = s$

The number of solutions to the SDP problem depends on the weight t. Since the code is random, assuming the syndrome is obtained from a vector e of weight t, i.e., $e^\top H = s$, the average number of solutions is

$$1 + \binom{n}{t}(p-1)^t p^{k-n} \approx 1 + \binom{n}{t}(p-1)^t p^{k-n}.$$

If there is more than one solution, it is intuitive that the SDP problem gets easier to solve. Thus, it is always required to have (on average) a unique solution. That happens when

$$\binom{n}{t}(p-1)^t p^{k-n} \leq 1,$$

or, asymptotically,

$$h_p(T) - (1-R) \leq 0. \tag{2.9}$$

Therefore, if the target relative weight t is less than the minimum relative distance δ of the random code, i.e., the one that satisfies 2.8, we expect to have on average a unique solution.

Code-based cryptography is traditionally based on the hardness of decoding a random linear code, the NP-completeness of the Syndrome Decoding Problem was proved in [8] for the case of binary linear codes equipped with the Hamming metric and then generalized in [9] to an arbitrary finite field.

2.4 Information Set Decoding

Now we will present the Information Set Decoding, a class of generic decoding algorithms for code-based problems, in particular when the problem has only a small number of solutions.

For this brief introduction we will take up the concepts presented in [10].

Let's introduce the following notation: for $x \in \mathbb{F}_p$ and $S \subseteq \{1, \dots, n\}$ we denote by x_S the vector consisting of the entries of x indexed by S. Similarly for a generic matrix $A \in \mathbb{F}_p^{m \times n}$ we denote by A_S the matrix consisting of the columns of A indexed by S.

Consequently we can define C_S, the code consisting of the codewords c_S, in particular a $[n, k]$ linear code can be completely defined by certain sets of k positions.

Definition 2.4.1. Information Set Let $k \leq n$ be positive integers and let C be an $[n, k]$ linear code over $\mathbb{F}_p$. Then, a set $I \subseteq \{1, \dots, n\}$ of size k is called an information set of C if $|C| = |C_I|$.

Furthermore, one can introduce the following definition:

Definition 2.4.2. Systematic Form Let $k \leq n$ be positive integers and let C be an $[n, k]$ linear code over $\mathbb{F}_p$, for some convertible matrix $U \in \mathbb{F}_p^{(n-k)\times(n-k)}$ and some permutation matrix $P \in \mathbb{F}_p^{n \times n}$, the systematic form of the parity check matrix $H \in \mathbb{F}_p^{(n-k)\times n}$ is:

$$UHP = [Id_{n-k}|\tilde{H}]$$

Where $\tilde{H} \in F_p^{(n-k)\times k}$

The main idea is to exploit the knowledge of the hamming weight of the error vector and reduce the search space by linear algebra, the first ISD algorithm was proposed in 1962 by Prange [11] and interestingly, all improvements display the same structure. In particular we will analyze ISD-based algorithms using Partial Gauss Elimination (PGE) framework, generally this class of algorithm follows a common pattern, that can be schematized as follows:

1: $PGE + ISD$ **Framework**

Data: The parity-check matrix $H \in \mathbb{F}_p^{(n-k)\times(n)}$ describing a $[n,k]-C$ linear code and the syndrome vector $s \in \mathbb{F}_p^{(n-k)}$.

Result: $e \in \mathbb{W}_{n,t}^p$ such that $eH^T = s$.

1. Choose an information set $I \subseteq \{1,\ldots,n\}$ of size k for C.
2. Brings H into the systematic form corrisponding to I, i.e. find an invertible matrix $U \in \mathbb{F}_p^{(n-k)\times(n-k)}$ and permutation matrix $P \in \mathbb{F}_p^{(n-k)\times(n-k)}$:
$$UHP = [Id_{n-k}|\tilde{H}]$$
3. Go through all error vector $e \in$ having the assumed weight distribution.
4. Check if the parity-check equations
$$eH^TU^T = sU^T$$
are satisfied.
5. If they are satisfied, output e, if not start over with a new choice of I.

Note that the searching phase on step 3 it should be optimized in order to reduce the number of operation necessary for finding candidate solution, futhermore the average number of overall iterations required depends by the success probability of one iteration and this probability is completely determined by the assumed weight distribution.

All the improvements that have been suggested to Prange's simplest form of ISD assume a more likely weight distribution of the error vector, which results in a higher cost of one iteration but give overall a smaller cost, since less iterations have to be performed.

The improvements split into two directions: the first direction is following the idea of Lee and Brickell [12] where they ask for v errors in the information set and $t - v$ outside. The second direction is Dumer's approach [13], which is asking for v errors in $k + \ell$ positions, which are containing an information set, and $t - v$ in the remaining $n - k - \ell$. Clearly, the first direction is a particular case of the second direction when $\ell = 0$.

In the chapter 4 we will focus on two specific approach: the one proposed by Stern [14] and then generalised by Peters to $\mathbb{F}_p$, which proposes to partition the information set into two sets and ask for v errors in each part and $t - 2v$ errors outside the information set, and $BJMM$ [15] which improves MMT [16], an algorithm based on representation technique, by introducing overlapping supports.

Operations cost over $\mathbb{F}_p$

We will assume that one addition over $\mathbb{F}_p$ costs $\lceil \log_2(p) \rceil$ binary operations and one multiplication costs $\lceil \log_2(p) \rceil^2$ binary operations.

Lemma 5. *Given $A \in \mathbb{F}_p^{(k \times n)}$ and $B \in \mathbb{F}_p^{(n \times h)}$ compute AB will cost:*

$$knr \left(\lceil \log_2(p) \rceil + \lceil \log_2(p) \rceil^2 \right) \text{ binary operations}$$

Number of Iterations

Since in one iteration of a generic algorithm, a fixed information set is considered, the success probability of an iteration is given by the fraction of how many vectors there are with the assumed weight distribution, divided by how many vectors there are in general with the target weight t.

For example if we assume that the error vector $e \in \mathbb{F}_p$ with weight t has t_i non zero entries within the information set I of size k the other t_c non zero entries on the remaining $n - k$ coordinates, the success probability of one iteration is given by:

$$Pr_{succ} = \frac{\binom{k}{t_i} p^{t_i} \binom{n-k}{t_c} p^{t_c}}{\binom{n}{t} p^t} \tag{2.10}$$

Consequently the number of iterations needed on average by an algorithm using $PGE + ISD$ framework is given by: Pr_{succ}^{-1}

Average number of Collisions

The algorithms that will be analyzed in the next sections are based on a collision search between two lists of vectors defined as:

- $L_1 := \{x \in \mathbb{F}_p^{l_x} \mid wt_H(x) = \omega_x\}$
- $L_2 := \{y \in \mathbb{F}_p^{l_y} \mid wt_H(y) = \omega_y\}$

The algorithm would go through all the vectors pairs $(x, y) \in (L_1 \times L2)$ and check if (x, y) satisfy a specific condition. We can compute the number of collision we can expect on average.

Lemma 6. *Let L_1, L_2 two lists uniformly distributed, $OP : \mathbb{F}_p \times \mathbb{F}_p \to \mathbb{F}_p$ a generic vector operation, for a given $H \in \mathbb{F}_p^{x \times w}$, a target vector $s \in \mathbb{F}_p^{\ell}$ the average number of pairs $(x, y) \in L_1 \times L_2$ such that $OP(x, y) = z$, with $zH^T = s$ and $wt_H(z) = \omega_z$, is equals to:*

$$\frac{|L_1||L_2|}{p^\ell}$$

Having defined the two lists we can explicit the amount as:

$$\frac{|L_1||L_2|}{p^\ell} = \frac{\binom{l_x}{\omega_x}(p-1)^{\omega_x}\binom{l_y}{\omega_y}(p-1)^{\omega_y}}{p^\ell}$$

Representations Technique over set $\mathbb{F}_p$

We now analyze a more flexible decomposition of vector.
Let $\tilde{x} \in \mathbb{F}_p^n$ with $wt_H(x) = \omega$ one can decompose $\tilde{x}$ into vectors $\tilde{x}_A$, $\tilde{x}_B$, whose values can overlap at a given fraction of their entries. Having fixed $\alpha \in [0, \frac{1}{2}]$, one can define:

- $supp(\tilde{x}_A) \subseteq \{1 \dots, \lfloor(\frac{1}{2} + \alpha)(n)\rfloor\}$
- $supp(\tilde{x}_B) \subseteq \{\lfloor(\frac{1}{2} - \alpha)(n)\rfloor + 1, \dots, n\}$

Every non-zero within the $2\alpha(n)$ overlapping coordinates of $\tilde{x}$ can be represented as either $\tilde{x}_{A,i} + 0$ or $0 + \tilde{x}_{B,i}$, depending on whether the error is assigned to $\tilde{x}_A$ or $\tilde{x}_B$, the consequence of using this technique is to obtain an exponential number of representations. ($\alpha = 0$ no representation, $\alpha = \frac{1}{2}$ max n rep). Furthermore, to increase the number of possible representations, one can introduce $\varepsilon \in \{0, \dots, 2\alpha(n - \omega)\}$,

a parameter that defines how many non-zero entries to be added to both vectors. If those additional values occur within the overlapping coordinates we obtain more ways to represent a value in the resulting vector as a sum of two values.

Consequently, we require that $\tilde{x}_A \in \mathbb{W}_A$ and $\tilde{x}_B \in \mathbb{W}_B$, The two sets are defined as follows:

- $\mathbb{W}_A := \mathbb{W}^p_{(\frac{1}{2}+\alpha)(n),\frac{\omega}{2}+\varepsilon} \times \{0\}^{(\frac{1}{2}-\alpha)(n)}$
- $\mathbb{W}_B := \{0\}^{(\frac{1}{2}-\alpha)(n)} \times \mathbb{W}^p_{(\frac{1}{2}+\alpha)(n),\frac{\omega}{2}+\varepsilon}$

The number of representation for an error vector depends on its weight-distribution. It is known that $\tilde{x} \in \mathbb{F}_p^n$ with $wt_H(e) = \omega$ is balanced when is $\left(\left(\frac{1}{2}-\alpha\right)(\omega), \left(\frac{1}{2}-\alpha\right)(\omega)\right)$-distributed, which means it has Hamming weight exactly i and j on its first and last $\left(\frac{1}{2}-\alpha\right)(n)$ coordinates respectively. This implies that $\tilde{x}$ has weight $\omega-(i+j)$ within the remaining $2\alpha(n)$ coordinates.

Futhermore, one can observe that for a given $a \in \mathbb{F}_p$ it is possible to define:

$$\alpha_p(a) = |\{b \in \mathbb{F}_p \,|\, \exists\, c \in \mathbb{F}_p \, s.t. \, b + c = a\}|$$

It should be noted that $\alpha_p(a)$ is the same for all $a \in \mathbb{F}_p$, so we can denote it as α_p.

Lemma 7. *Let us now consider a balanced vector $\tilde{x}$, one can estimate the number of pairs $(\tilde{x}_A, \tilde{x}_B) \in \mathbb{W}_A \times \mathbb{W}_B$ such that $\tilde{x} = \tilde{x}_A + \tilde{x}_B$.*

$$r_{n,\alpha,\varepsilon} = \binom{2\alpha\omega}{\alpha\omega}\binom{2\alpha(n-\omega)}{2\varepsilon}\alpha_p^{2\varepsilon}$$

Proof. The two vectors $\tilde{x}_A$ and $\tilde{x}_B$ must have 2ε common non-zero coordinates. Furthermore, $\tilde{x}$ is balanced, then they have weight $\left(\frac{1}{2}-\alpha\right)(\omega)$ on the first and last $\left(\frac{1}{2}-\alpha\right)(n)$ coordinates, respectively. It is therefore possible to arbitrarily assign $\alpha\omega$ of the $2\alpha\omega$ many non zero values within the overlapping coordinates of $\tilde{x}$ to vector $\tilde{x}_A$ (with the remaining being assigned to vector $\tilde{x}_B$). Subsequently, it is possible to choose 2ε number of intersecting positions among the remaining 0-positions within the overlapping coordinates, for each we can choose α_p possible entries. □

The computational effort lies in computing the list of candidate solutions for parity check equation in 1, in the next sections we will show how the representations technique adapted to ISD algorithms allows to implement a more efficient approach for generating candidate solutions list.

Merge Lists Functions

An important part of ISD algorithms is to merge two lists based on a certain condition, in the following for a vector x, we will denote by $x|_{\tilde{\ell}}$ the vector consisting of the first $\tilde{\ell}$ entries of x.

The following algorithm takes as input two lists $L_1, L_2 \subset \mathbb{F}_p^n$, containing vectors of weight u_1 and u_2, respectively, a positive integer $\tilde{\ell}$, which denotes the number of symbols on which one merges and a target vector $t \in \mathbb{F}_p^{\tilde{\ell}}$. The output is a list of the sums of all pairs of vectors that satisfy a specific condition on the vector t and whose sum has a specific Hamming weight.

Algorithm 2: MergeLists

Input: L_1, L_2, the parity check matrix $H \in \mathbb{F}_p^{n \times k}$, the integers $0 < \tilde{\ell} < k$, $0 \leq \omega \leq n$ and the target vector $t \in \mathbb{F}_p^{\tilde{\ell}}$

Output: L

Lexicographically sort L_1 and L_2 according to $(x_i H^T)|_{\tilde{\ell}}$ and $(y_j H^T)|_{\tilde{\ell}} + t$ respectively, for $x_i \in L_1$ and $y_j \in L_2$

$L := \varnothing$

for $(x_i, y_j) \in L_1 \times L_2$ *with* $(x_i H^T)|_{\tilde{\ell}} = (y_j H^T)|_{\tilde{\ell}} + t$ **do**

  if $wt_H(x_i + y_i) = \omega$ **then**

    $L = L \cup \{x_i + y_j\}$

  end

end

return L

Lemma 8. *The cost of the the Algortihm 2 is given by:*

$$C_{RM}(L_1, L_2, H, \tilde{\ell}, \omega, t) = (|L_1|u_1 + |L_2|u_2)\,\tilde{\ell}\,(\lceil \log_2(p) \rceil + \lceil \log_2(p) \rceil^2) + L_1 \log_2(L_1) + L_2 \log(L_2) + (\tfrac{L_1 L_2}{p^{\tilde{\ell}}})\,\lceil (n) \log_2(p) \rceil$$

Proof. The algorithm must compute the partial value of the syndrome on $\tilde{\ell}$ symbols, thus it is sufficient to consider only $\tilde{\ell}$ columns of the matrix H. This operation must be performed for each vector in both lists, while the cost of the vector-matrix multiplication over $\mathbb{F}_p$ is given by lemma 5. Subsequently, both lists must be sorted according to the label. Sorting a list L costs $|L| \log_2(|L|)$, which explains the second and third terms. Finally, for each collision found, it is necessary to check if their sum

has the desired weight, that requires adding two vectors of length n and weight u_1 and u_2, respectively, remembering how the average number of collisions is defined by lemma 6, this justifies the last term. □

We are also interested in evaluating the size of the output list, considering that the algorithm for each collision pair on the syndrome conditions stores the sum vector only if it has a certain weight, one can state that:

Lemma 9. *The algorithm 2 returns a list L of size* $|L| = min\left\{\frac{\binom{n}{\omega}(p-1)^{\omega}}{p^{\tilde{\ell}}}, \frac{L_1 L_2}{p^{\tilde{\ell}}}\right\}$

2.5 Locality Sensitive Functions

The main idea of LSF is to apply a certain relation to two vectors that when collide under this relation, they are likely to be a solution. In particular, one can define choose a set of vectors:

$$\mathbb{C}_f \subseteq S_f^n \quad \text{with size} \quad |\mathbb{C}_f| = \binom{n}{f}(p-1)^f$$

And its definition allows, by choosing an integer value γ, to divide the Hamming space into several regions, which can be overlapped, such regions can be defined as follows:

$$Region_{(c,\gamma)} = \{x \in \mathbb{F}_2^n : (x \wedge c) = \mathbb{S}_{n,\gamma}\} \quad \forall c \in \mathbb{C}_f$$

In other words, each vector $c \in \mathbb{C}_f$ define a subset of $\mathbb{F}_2^n$, which contains all the binary vectors which have exactly γ overlaps with c.

One may be interested in finding all the vectors of a specific set S that live in the region defined by a certain $c \in \mathbb{C}_f$, then defining:

$$Bucket_{(c,\gamma)} = S \cap Region_{(c,\gamma)}$$

In the next we will analyze how this concepts can be used to speed up the collision search. In particular we will use LSF to reduce the number of comparisons by checking only those vectors with particular weight distributions. In fact, a necessary criterion for the sum of two vectors to have a particular weight is that the two vectors have an exact number of overlaps.

We will exploit the fact that if two uniform random vectors $x, y \in F_2^n$ happen to be assigned to the same bucket, they have a certain overlap in support with c, so they are more likely to have overlap in support with each other.

In particular we will adapt the approach proposed in [17] that use locality sensitive functions for solving the following problem:

Definition 2.5.1. ω-Nearest Neighbor Search Given two lists of vectors $L_1 \subseteq \mathbb{W}^2_{n,w_1}$, $L_2 \subseteq \mathbb{W}^2_{n,w_1}$ and a target weight $\omega_t n$, find all pairs $(x, y) \in (L_1 \times L_2)$, find all pairs s.t. $wt_H(x + y) = \omega_t$

Chapter 3

CROSS: Codes and Restricted Objects Signature Scheme

In this chapter we will analyze the main features and properies of CROSS, the signature scheme derived, via Fiat-Shamir transformation from the Zero Knowledge CROSS-ID identification protocol. This protocol, which is a $q2$-identification protocol with a $q2$ extractor, is based on the CVE identification logic discussed in [18] originally proposed for SPD. Similarly, the use of the Fiat-Shamir transformation guarantees, for t parallel executions of CROSS-ID ($q2$), the achievement of a signature scheme with Existential Unforgeability under Chosen Message Attack (EUF-CMA).

3.1 Restricted Syndrome Decoding Problem (R-SDP)

Let's analyze the underlying problem. The $R-SDP$ was first introduced for $z=2$ in [19] and then for any z [20] , the difference from the original SDP is that we consider an additional restriction: the entries of the error vector solution are defined in a particular subset of the finite field.

By choosing a generator $g \in \mathbb{F}_p^*$ of multiplicative order z it is possible to define the Restricted Set as

$$\mathbb{E} = \{g^i \ (mod\, p) \,|\, i \in 0, ..., z-1\} \ \subseteq \ F_p^*.$$

When considering a parity check matrix $H \in \mathbb{F}_p^{(n-k)\times n}$ and a syndrome $s \in \mathbb{F}_p^{n-k}$ the Restricted Syndrome Decoding Problem is defined as follows:

Definition 3.1.1. $R-SDP$: Given $t \in \mathbb{N}$, $H \in \mathbb{F}_p^{(n-k)\times n}$, $s \in \mathbb{F}_p^{n-k}$ find a vector $e \in \mathbb{E}^n$ such that $eH^\intercal = s$ and $wt_H(e) = t$

NP-Completeness of the underlying problem

The $R-SDP$ is strongly related to other well-known hard problems. For example, when $z = p-1$, the $R-SDP$ is close to the classical SDP, if $z = 1$, the $R-SDP$ is similar to the Subset Sum Problem (SSP) over finite fields. Consequently, the $R-SDP$ is NP-complete for any choice of $\mathbb{E}$.

Lemma 10. *The Restricted Syndrome Decoding Problem in NP-Complete*

The NP-completeness of the Restricted Syndrome Decoding Problem is proven in [21].

Uniqueness of the solution $R-SDP$

Both the parity-check matrix H and the error vector e have been generated by sampling from $\mathbb{F}_p^{(n-k)\times n}$ and $\mathbb{E}^n$ uniformly at random, this means that the $R-SDP$ instance is chosen uniformly at random. Consequently we expect to have on average a unique solution if the weight t is such that:

$$\binom{n}{t} z^t p^{k-n} \leq 1 \tag{3.1}$$

Considering this asymptotically, we have for $T = \frac{t}{n}$ the following condition:

$$2^{n(H_2(T)+T\log(z)-(1-R)\log_2(p))} \leq 1,$$

It happens when for z and R such that:

$$T\log(z) + H(T) - (1-R)\log(p) \leq 0.$$

Let T^* be the maximum value of T for which a random instance of $R-SDP$ is expected to have a unique solution, that is

$$T = max\{T \in [0;1] \,/\, T\log_2(z) + H(T) - (1-R)\, log_2(p) \leq 0\} \tag{3.2}$$

Comparing this to the condition in 2.9, we can see that with the $R-SDP$, we are allowed to choose a much larger weight t and still guarantee the uniqueness of

the solution. Notice that if $\log_2(z) \leq (1-R)\log_2(p)$, we even have uniqueness for full-weight vectors.

We will see in the next section how considering the full-weight version of the $R-SDP$ leads to reduced signature size.

3.2 Zero Knowledge Protocol Structure

What differentiates this scheme from classical CVE is that the *Prover* first samples a transformed error vector $e' \in \mathbb{E}^n$ and a random vector $u' \in \mathbb{F}_p^n$. Only then a transformation $\sigma : \mathbb{E} \mapsto \mathbb{E}$ is found such that $e = \sigma(e')$. Let it be known that σ is a bjection and is uniformly random on $\mathbb{E}$, this guarantees that e' is randomly and uniformly sampled from $\mathbb{E}$. Moreover, since u' is uniformly random on $\mathbb{F}_p^n$, the response vector $y = u' + \beta e'$, where $\beta \in \mathbb{F}_p^*$ represents the vector of the first challenge, follows the uniform distribution on $\mathbb{F}_p^n$.

Another difference with the CVE scheme is the first answer, which in CROSS-ID is $h = \mathsf{Hash}(y)$. The vector of the second challenge is denoted by b, with binary values. When $b = 1$, the *Prover* must report the seed it used to sample both u' and e': this shows how y is generated as the sum of a masked vector and a restricted vector, multiplied by β. This strategy provides savings in communication cost, since sending h requires fewer bits than y. When $b = 0$, the *Prover* reveals y, along with σ (which, not being a random transformation, hence it cannot be compressed using the seeds).

Private Key	$e \in \mathbb{E}^n$
Public Data	$\mathbb{E}^n$, $H \in \mathbb{F}_p^{(n-k)\times n}$, $s = eH^\top \in \mathbb{F}_p^{n-k}$

PROVER		VERIFIER
Sample $\mathtt{Seed} \xleftarrow{\$} \{0;1\}^\lambda$		
Sample $(\mathtt{Seed}^{(\mathtt{u'})}, \mathtt{Seed}^{(\mathtt{e'})}) \xleftarrow{\mathtt{Seed}} \{0;1\}^{2\lambda}$		
Sample $\mathtt{u}' \xleftarrow{\mathtt{Seed}^{(\mathtt{u'})}} \mathbb{F}_p^n$		
Sample $\mathtt{e}' \xleftarrow{\mathtt{Seed}^{(\mathtt{e'})}} \mathbb{E}^n$		
Sample $\sigma \in \mathbb{E}^n$ such that $\sigma(\mathtt{e'}) = \mathtt{e}$		
Set $\mathtt{u} = \sigma(\mathtt{u'})$		
Compute $\tilde{\mathtt{s}} = \mathtt{u}\mathtt{H}^\top$		
Set $c_0 = \mathtt{Hash}(\tilde{\mathtt{s}}, \sigma)$		
Set $c_1 = \mathtt{Hash}(\mathtt{u'}, \mathtt{e'})$		
	$\xrightarrow{(c_0,c_1)}$	
		Sample $\beta \xleftarrow{\$} \mathbb{F}_p^*$
	$\xleftarrow{\beta}$	
Compute $\mathtt{y} = \mathtt{u'} + \beta\mathtt{e'}$		
Set $h = \mathtt{Hash(y)}$		
	$\xrightarrow{h}$	
		Sample $b \xleftarrow{\$} \{0;1\}$
	$\xleftarrow{b}$	
If $b = 0$, set $f := (\mathtt{y}, \sigma)$		
If $b = 1$, set $f := \mathtt{Seed}$		
	$\xrightarrow{f}$	
		If $b = 0$: Compute $\tilde{\mathtt{y}} = \sigma(\mathtt{y})$ e $\tilde{\mathtt{s}} = \tilde{\mathtt{y}}\mathtt{H}^\top - \beta\mathtt{s}$ Accept if: 1) $\mathtt{Hash}(\mathtt{y}) = h$ 2) $\mathtt{Hash}(\tilde{\mathtt{s}}, \sigma) = c_0$ 3) $\sigma \in E^n$
		If $b = 1$: Sample $\mathtt{Seed}^{(\mathtt{u'})}, \mathtt{Seed}^{(\mathtt{e'})} \xleftarrow{\mathtt{Seed}} \{0;1\}^{2\lambda}$ Set $\mathtt{y} = \mathtt{u'} + \beta\mathtt{e'}$ Accept if: 1) $\mathtt{Hash}(\mathtt{y}) = h$ 2) $\mathtt{Hash}(\mathtt{u'}, \mathtt{e'}) = c_1$

Figure 3.1: ZK identification protocol CROSS-ID

3.3 CROSS-ID Properties

One can demonstrate how the CROSS-ID protocol guarantees the characteristics of: zero knowledge, completeness and (2,2)-out-of-$(p-1,2)$ special soundness. In this way it will be possible to state the achievement of the q2 form, with $q = p - 1$ and soundness error equal to $\frac{p}{2(p-1)}$.

Completeness

The protocol Fig.3.1 is complete .

Proof. It must be shown how an honest $Prover$ is always accepted. When $b = 0$, one will have:

$$\tilde{y} = \sigma(\mathbf{y}) = \sigma(\mathbf{u}') + \beta\sigma(\mathbf{e}') = \mathbf{u} + \beta\mathbf{e}.$$

From this, it is observed that:

$$\tilde{\mathbf{y}}\mathbf{H}^\top - \beta\mathbf{s} = \mathbf{u}\mathbf{H}^\top + \beta\mathbf{e}\mathbf{H}^\top - \beta\mathbf{s} = \tilde{\mathbf{s}} + \beta\mathbf{s} - \beta\mathbf{s} = \mathbf{u}\mathbf{H}^\top.$$

This result correspond to the syndrome used to generate the commitment c_0, considering $\sigma \in \mathbb{E}^n$. When $b = 1$, the $Prover$ sends only the seeds and, since PRNGs are deterministic, the $Verifier$ obtains the very same quantities that have been used to generate the commitments.

Zero Knowledge

The protocol 3.1 achieves Zero-Knowledge.

Proof. It must be proven that a simulator S with knowledge of the challenges, can simulate the interaction between the $Prover$ and the $Verifier$. Formally, we show how S produces a transcript T^* that is indistinguishable from a transcript T, resulting from the interaction $\langle P, V \rangle$.

One can define two strategies for S, depending on the values of b (vector of the second challenge):

- Strategy when $b = 0$. The simulator S searches, by linear algebra, for a vector $\mathbf{e}^\star \in \mathbb{F}_p^n$ such that $\mathbf{e}^\star\mathbb{H}^\top = \mathbf{s}$. Then, S chooses $\sigma^\star \in \mathbb{E}$ and a vector $\mathbf{u}^\star \in \mathbb{F}_p^n$ and compute $\mathbf{u}^{\star'} = \sigma^{\star-1}(\mathbf{e}^\star)$. Finally, S computes $\tilde{\mathbf{s}^\star} = \mathbf{u}^\star\mathbf{H}^\top$ and commitment $c_0 = \texttt{Hash}(\tilde{\mathbf{s}^\star}, \sigma^\star)$. When S will receive the answer vector $\mathbf{y}^\star = \mathbf{u}^{\star'} + \beta\mathbf{e}^{\star'}$. It is easy to verify how the transcript produced by S follows the same statistical distribution as a transcript produced by an honest $Prover$ considering how $\mathbf{y}^\star$ and $\sigma^\star$ are computed. Indeed, in an honest execution, $\mathbf{y}$ is uniformly random

over $\mathbb{F}_p^*$, since $\mathbf{u}'$ is uniformly random over $\mathbb{F}_p^n$. This guarantees that $\mathbf{u}' + \beta\mathbf{e}'$ is uniformly random over $\mathbb{F}_p^n$ and the same holds for the σ transformation. So, in an honest execution of the protocol, σ is uniformly distributed over $\mathbb{E}^n$. That is, it is observed that, for every $\mathbf{e}' \in \mathbb{E}^n$, there exists a unique $\sigma \in \mathbb{E}^n$ such that $\sigma(\mathbf{e}') = \mathbf{e}$. If $\mathbf{e}'$ is uniformly random on $\mathbb{E}^n$, then σ follows the same distribution. The commitment, which is not verified, can be chosen as a binary string of length 2λ. Under the Random Oracle Model assumption, it will have the same statistical distribution as an honestly computed c_1 commitment.

– Strategy when $b = 1$. In this case, the simulator simply needs to execute the protocol by sampling the seeds and computing c_0, analogous to what an honest *Prover* would do.As for the other commitment, simply use a random binary string as in the previous case.

Soundness

The protocol in Fig.3.1 ensure the soundness, with error $\varepsilon = \frac{p}{2(p-1)}$.

Proof. Let's consider an adversary A who tries to imitate the $Prover$. His goal is to correctly replicate the answers, relative to the $Verifier$'s challenges. Two attack strategies will be shown that achieve a probability of success equal to $\varepsilon = \frac{p}{2(p-1)}$.

Next, it will be observed why these strategies are optimal, namely, that the probability of success is maximal and corresponds to the soundness error. In order to prove these claims, it will be necessary to observe that the protocol guarantees the (2,2)-out-of-$(p-1$,2) special sound characteristic, and then compute the error through the formulas defined in [22, 23].

Strategy 0: The adversary A aims to answer correctly always, for the case $b = 0$, but still tries to guess the values of $\beta^\star$ when $b = 1$. A first decides the value for $\beta^\star \in \mathbb{F}_p^*$ and a seed `Seed`, which will be used to sample $\mathbf{u}' \in \mathbb{F}_p^n$ and $\mathbf{e}' \in \mathbb{E}^n$. The adversary, chooses a random $\sigma \in \mathbb{E}^n$ and computes $\mathbf{y}^\star = \mathbf{u}' + \beta^\star\mathbf{e}'$. Subsequently, the adversary computes $\tilde{\mathbf{s}} = \sigma(\mathbf{y}^\star)\mathbf{H}^\top$ and sets $c_0 = \mathtt{Hash}(\tilde{\mathbf{s}} - \beta^\star\mathbf{s}, \sigma)$. Finally, the adversary choose $\tilde{\mathbf{e}} \in \mathbb{F}_p^n$ such that $\tilde{\mathbf{e}}\mathbf{H}^\top = \mathbf{s}$ and $\tilde{\mathbf{u}} \in \mathbb{F}_p^n$ such that $\tilde{\mathbf{u}}\mathbf{H}^\top = \sigma(\mathbf{y}^\star)\mathbf{H}^\top - \beta^\star\mathbf{s}$. The adversary sends c_0 e c_1 to the $Verifier$s and sends β. If $\beta = \beta^\star$, the adversary

answer with $h = \texttt{Hash}(\mathbf{y}^\star) = h$ with $\sigma \in \mathbb{E}$ e

$$c_0 = \texttt{Hash}(\sigma(\mathbf{y}^*\mathbf{H}^\top - \beta\mathbf{s}), \sigma) = \texttt{Hash}(\tilde{\mathbf{s}} - \beta^\star\mathbf{s}, \sigma).$$

If $b = 1$, the adversary answer with a seed to compute **e'** and **u**'. Furthermore the adversary is accepted as $h = \texttt{Hash}(\mathbf{u'} + \beta^\star\mathbf{e'})$ e $c_1 = \texttt{Hash}(\mathbf{u'}, \mathbf{e'})$. If $\beta \neq \beta^\star$, the adversary sends a different h. Formally, the adversary compute $\mathbf{y} = \sigma^{-1}(\tilde{\mathbf{u}}) + \beta\sigma^{-1}(\tilde{\mathbf{e}})$ and sends $h = \texttt{Hash}(\mathbf{y})$. If the $Verifier$ asks for $b = 0$, the adversary sends the $(\mathbf{y}, \sigma)$ and is accepted because $h = \texttt{Hash}(\mathbf{y})$, $\sigma \in \mathbb{E}^n$, e

$$c_0 = \texttt{Hash}(\sigma(\mathbf{y})\mathbf{H}^\top - \beta\mathbf{s}, \sigma) = \texttt{Hash}(\tilde{\mathbf{u}}\mathbf{H}^\top, \sigma) = \texttt{Hash}(\sigma(\mathbf{y}^\star)\mathbf{H}^\top - \beta^\star\mathbf{s}, \sigma).$$

If the $Verifier$ sends $b = 1$, the adversary cannot be accepted. Consequently, this strategy has a success probability equal to:

$$Pr[b = 0] + Pr[(b = 1) \wedge (\beta = \beta^\star)] = \frac{1}{2} + \frac{1}{2(p-1)} = \frac{p}{2(p-1)}.$$

Strategy 1: The adversary hopes to receive the the challenge $b = 1$ but, again, prepare to answer also if obtain $b = 0$, by guessing the value of β. The adversary initially chooses a value $\beta^\star \in \mathbb{F}_p^*$, then it selects a seed from which $\mathbf{u'} \in \mathbb{F}_p^n$ and $\mathbf{e'} \in \mathbb{E}^n$ are generated. The adversary also chooses $\sigma \in \mathbb{E}^n$ and computes $\mathbf{u} = \sigma(\mathbf{u'})$, $\tilde{\mathbf{e}} = \sigma(\mathbf{e'}) \in \mathbb{E}^n$. The adversary computes $\tilde{\mathbf{s}} = \mathbf{u}\mathbf{H}^\top + \beta^\star\tilde{\mathbf{e}}\mathbf{H}^\top - \beta^\star\mathbf{s}$. The adversary will send the commitments $c_0 = \texttt{Hash}(\tilde{\mathbf{s}}, \sigma)$ and $c_1 = \texttt{Hash}(\mathbf{u'} + \mathbf{e'})$. When the adversary receives $\beta \in \mathbb{F}_p^*$, it computes $\mathbf{y} = \mathbf{u'} + \mathbf{e'}\beta$ and sends the Hash h.

If the adversary receives $b = 1$, he sends the seeds to compute $\mathtt{u}$' and $\mathtt{e}$', and he will definitely be accepted. This is because the $Verifier$ uses the seeds to reconstruct $\mathtt{u}$' and $\mathtt{e}$', which are used to compute and check the values of $h = \texttt{Hash}(\mathbf{u'} + e')$ and $c_1 = \texttt{Hash}(\mathbf{u'}, \mathbf{e'})$.

However, if the opponent receives the challenge $b = 0$, then it will send the pair $(\mathbf{y}, \sigma)$ and will only be accepted if $\beta = \beta^\star$, since

$$\sigma(\mathbf{y})\mathbf{H}^\top - \beta\mathbf{s} = \mathbf{u}\mathbf{H}^\top + \beta\tilde{\mathbf{e}}\mathbf{H}^\top - \beta\mathbf{s} = \tilde{\mathbf{s}}.$$

Thus, in the case where $c_0 = \texttt{Hash}(\mathbf{y})\mathbf{H}^\top) - s, \sigma)$ and $h = \texttt{Hash}(\mathbf{y})$, with $sigmain\mathbb{E}^n$.

Thus, this strategy has a probability of success

$$Pr[b = 1] + Pr[(b = 0) \wedge (\beta = \beta^\star)] = \frac{1}{2} + \frac{1}{2(p-1)} = \frac{p}{2(p-1)}.$$

(2,2)-out-of-(p-1,2) special soundness

Consider four transcripts T_1, T_2, T_3, T_4, all associated with the same pairs of commitments c_0, c_1. The commitment c_0 allowsto fix $(\tilde{\mathbf{s}}, \sigma)$, while c_1 fixes the pair $(\mathbf{u}', \mathbf{e}')$. In the following we will identify the transcripts based on the values of the challenge: $(\beta, 0)$, $(\beta, 1)$, $(\beta^\star, 0)$ e $(\beta^\star, 1)$. Taking the *Prover*'s responses as a reference, the transcript structures are as follows:

$$T_1 : (c_0, c_1, \beta, h, \mathbf{y}, \sigma);$$

$$T_2 : (c_0, c_1, \beta, h, \mathbf{Seed});$$

$$T_3 : (c_0, c_1, \beta^\star, h^\star, \mathbf{y}^\star, \sigma^\star);$$

$$T_4 : (c_0, c_1, \beta^\star, h^\star, \mathtt{Seed}^\star).$$

Next we show, based on knowledge of the four transcripts, a solution for an instance of R-SDP with $\{s, H\}$ that can be easily computed in polynomial time. We focus initially on T_2 and T_4. Let $\mathbf{u}'$, $\mathbf{e}'$ the vector generated from $\mathtt{Seed}$ and let $\mathbf{u}^{\star'}$, $\mathbf{e}^{\star'}$ generated from $\mathtt{Seed}^\star$. Since c_1 is verified in both cases, care must be taken toward the collisions of the functions $\mathtt{Hash}(\mathbf{u}', \mathbf{e}') = \mathtt{Hash}(\mathbf{u}^{\star'}, \mathbf{e}^{\star'})$ with $\mathbf{u}' \neq \mathbf{u}^{\star'}$ and $\mathbf{e}' \neq \mathbf{e}^{\star'}$, or $\mathbf{u}' = \mathbf{u}^{\star'}$ and $\mathbf{e}' = \mathbf{e}^{\star'}$. Since h and $h^\star$ are checked and no collisions are detected in the hash functions, we get $h = \mathtt{Hash}(\mathbf{y})$, with $\mathbf{y} = \mathbf{u}' + \beta\mathbf{e}'$ ed $h^\star = \mathbf{Hash}(\mathbf{y}^\star)$, with $\mathbf{y}^\star = \mathbf{u}^{\star'} + \beta^\star\mathbf{e}^{\star'} = \mathbf{u}' + \beta^\star\mathbf{e}'$. It implies $\mathbf{y} - \mathbf{y}^\star = \mathbf{e}'(\beta - \beta^\star)$.

Finally, we focus on the transcript pair T_1 and T_3. If no collisions are detected, when considering $\sigma = \sigma^\star$, we obtain:

$$\sigma(\mathbf{y})\mathbf{H}^\top - \beta\mathbf{s} = \tilde{\mathbf{s}}, \sigma(\mathbf{y}^\star)\mathbf{H}^\top - \beta^\star\mathbf{s} = \tilde{\mathbf{s}},$$

from which it follows that

$$\sigma(\mathbf{y} - \mathbf{y}^\star)\mathbf{H}^\top = (\beta - \beta^\star)\mathbf{s}.$$

By analyzing the relationships derived from the pair (T_2, T_4), one can compute $\mathbf{y} - \mathbf{y}^\star = (\beta - \beta^\star)\mathbf{e}'$, where $\mathbf{e}'$ is a restricted vector such that

$$(\beta - \beta^\star)\sigma(\mathbf{e}')\mathbf{H}^\top = (\beta - \beta^\star)\mathbf{s} \Rightarrow \sigma(\mathbf{e}')\mathbf{H}^\top = \mathbf{s}.$$

Since σ and $\mathbf{e}'$ have been verified, since $\sigma, \mathbf{e}' \in \mathbb{E}^n$, then $\sigma(\mathbf{e}') \in \mathbb{E}^n$. This result allows us to state that $\sigma(\mathbf{e}')$ solves the R-SDP problem for the instance $\mathbf{s}$, $\mathbf{H}$.

3.4 Fiat-Shamir Trasformation

Since the protocol in 3.1 is classified $q2$, as specified in the documentation [24], if t parallel executions of the algorithm are taken into account by applying the Fiat-Shamir transformation, what is derived is a signature scheme that guarantees EUF-CMA security.

Lemma 11. *CROSS, the signature scheme resulting from the application of the Fiat-Shamir transform on t parallel executions of the Zero Knowledge q2 CROSS-ID protocol, guarantees EUF-CMA security.*

This follows from the fact that CROSS applies the Fiat-Shamir transform on t parallel executions of a q2-Identification protocol and by the arguments from [24].

Protocol Optimizations

All messages exchanged in the i-th round will be denoted by superscripts $^{(i)}$. To obtain a clear and compact notation, we group the exchanged messages in the following representation:

$$\begin{array}{rccccccc} & \textbf{Round 1} & \cdots & \textbf{Round i} & \cdots & \textbf{Round t} \\ \textit{Commitment} = & c_0^{(1)}\ c_1^{(1)} & \cdots & c_0^{(i)}\ c_1^{(i)} & \cdots & c_0^{(t)}\ c_1^{(t)} \\ \textit{First Challenge} = & \beta^{(1)} & \cdots & \beta^{(i)} & \cdots & \beta^{(t)} \\ \textit{First Response} = & h^{(1)} & \cdots & h^{(i)} & \cdots & h^{(t)} \\ \textit{Second Challenge} = & b^{(1)} & \cdots & b^{(i)} & \cdots & b^{(t)} \end{array}$$

$$\textit{Second Response} = \quad f^{(1)} \quad \cdots \quad f^{(i)} \quad \cdots \quad f^{(t)}$$

On the other hand, as far as a security perspective is concerned, in order to prevent attacks based on hash function collisions, 2λ bits of `Salt` are introduced, as suggested in the documentation [25]. Next, we focus on deepening the configurations employed in order to ensure improvements to the scheme, both in terms of efficiency and security.

Fixing Second Challenge Weight

Considering the second challenge, consisting of $(b^{(1)},\cdots,b^{(t)})$, it will always have fixed weight equal to ω. This value represents the number of rounds in which the $Verifier$ requires the values associated with $b = 1$, while $t - \omega$ are the rounds in which $b = 0$.

In the first case, the $Prover$ is required to send only a seed of length λ and without revealing $\mathbf{y}$ (since the $Verifier$ can recompute it independently). A valid choice it so choose a value of ω close t. Thus, the goal is to reduce communication costs as much as possible, for as many rounds as possible. Such an intervention, however, changes the cost of forgery attacks, as an adversary may take advantage of the dominance of $b = 1$-valued rounds.

Using Seed Tree

For each execution of the signature algorithm, t seeds are generated $\texttt{Seed}^{(1)},\cdots,\texttt{Seed}^{(t)}$, which will be used to sample, at each round, $\mathbf{u'}^{(i)}$ and $\mathbf{e'}^{(i)}$. These seeds are obtained through a tree-data structure, composed from the root `MSeed||Salt`, with $\texttt{MSeed} \xleftarrow{\$} \{0;1\}^{\lambda}$. This list of seeds $\texttt{Seed}^{(1)},\cdots,\texttt{Seed}^{(t)}$ represents the leaves of the tree, as it characterizes its lowest level.

Let it be known that the $Prover$ is required to send the seed in ω rounds, while in the remaining $t - \omega$ rounds it is obliged to reveal all data. In conclusion, the maximum number of nodes to be revealed is equal to $(t-\omega)\log_2(\frac{t}{t-\omega})$. So, sending

all seeds has a communication cost equal to

$$|\texttt{SeedPath}| = \lambda(t-\omega)\log_2(\frac{t}{t-\omega}).$$

Postponing first response verification

The verification of the first challenge can be postponed until the end of the control algorithm.

Indeed, at each round, the $Verifier$ can get $\mathbf{y}^{(i)}$ (whether he received it from the $Prover$ or recomputed it locally from $\texttt{Seed}^{(i)}$).

Rather than replying with $(h^{(1)},\cdots,h^{(t)})$, the $Prover$ conveniently can send

$$h = \texttt{Hash}(\mathbf{y}^{(1)},\mathbf{y}^{(2)},\cdots,\mathbf{y}^{(t)}).$$

To generate the second challenge, the $Prover$ uses h (since this value is included in the signature). After the execution of all rounds, the $Verifier$ can locally recompute h. If the $Verifier$'s recomputation coincides with the given value of h, then either a collision has been found in the hash function or the t vectors $\mathbf{y}^{(1)},\mathbf{y}^{(2)},\cdots,\mathbf{y}^{(t)}$ are with certainty valid.

Reducing commitment size

In each round, the $Verifier$ can always recompute one of the two commitments locally. Since the second challenge vector has fixed weight ω close to t, the $Verifier$ will recompute most of the commitments $c_1^{(i)}$ and only a few commitments $c_0^{(i)}$. As for commitments $c_1^{(i)}$, the $Prover$ can conveniently associate with a single hash digest

$$c_1 = \texttt{Hash}(c_1^{(i)},\cdots,c_1^{(t)}).$$

Let $J \subseteq \{1,\cdots,t\}$ be the support of $(b^{(1)},\cdots,b^{(t)})$ (i.e., the set of indices i such that $b^{(j)} = 1$): the $Verifier$ will know all $c_1^{(i)}$ with $i \in J$ and will not know those for which $i \notin J$. For each of the latter indices, the $Prover$ will have to include $c_1^{(i)}$ in the second answer. In this way, the total cost associated with the commitments

$c_1^{(i)},\cdots,c_1^{(t)}$ will be

$$|\texttt{Com(1)}| = \underbrace{2\lambda}_{c_1} + \underbrace{2(t-\omega)\lambda}_{c_1^{(i)},\ i\notin J} = 2\lambda(t-\omega-1).$$

For the commitments $c_0^{(i)}$, the *Prover* can construct a Merkle tree T, using $c_0^{(1)},\cdots,c_0^{(t)}$ as leaves, with d_0 as the root. The $Verifier$ will be able to recompute all $c_0^{(i)}$ with $i \notin J$; furthermore, to certify that the *Prover* has bound to correct values, the $Verifier$ will require a Merkle Proof for all remaining $c_0^{(i)}$.

Let it be known that $t-\omega$ rounds are characterized by a second zero-valued challenge. Normally, sending all the proofs would require $(t-\omega)\log_2(t)$ hash digests (there being $\log_2(t)$ digests for each of the $t-\omega$ leaves for which the proof is required). More conveniently, it is possible to consider how these proofs have paths in common: the number of different hashes that are required is no greater than $(t-\omega)\log_2(\frac{t}{t-\omega})$.

In doing so, the total cost associated with the commitments $c_0^{(1)},\cdots,c_0^{(t)}$ will be limited superiorly by

$$|\texttt{Com(0)}| = \underbrace{2\lambda}_{c_0} + \underbrace{2\lambda\log_2(\frac{t}{t-\omega})}_{Proof\ di\ c_0^{(i)},\ i\notin J} = 2\lambda(1+(t-\omega)\log_2(\frac{t}{t-\omega})).$$

3.5 CROSS Signature Scheme

After carefully evaluating all previously optimizations, the actual signing scheme consists of the following steps (Fig. 3.2, 3.3)

Signing

1. sample $\texttt{Salt} \xleftarrow{\$} \{0;1\}^{2\lambda}$;
2. sample $\texttt{MSeed} \xleftarrow{\$} 0;1^{\lambda}$ and build a seed tree, which has as its leaves the t elements $\texttt{Seed}^{(1)},\cdots,\texttt{Seed}^{(t)}$. The single $\texttt{Seed}^{(i)}$ is needed to sample $\mathbf{u}'^{(i)}$ and $\mathbf{e}'^{(i)}$, which are actually employed in round i;
3. for rounds $i = 1,\cdots,t$ computes the restricted transformation $\sigma^{(i)}$ and the commitments $c_0^{(i)}$ and $c_1^{(i)}$, as defined in CROSS-ID. It also uses the $\texttt{Salt}$ and round index i within the hash functions to add security;

4. builds the Merkle T tree with the commitments $c_0^{(1)}, \cdots, c_0^{(t)}$;

5. generates the vector of the first challenge $(\beta^{(1)}, \cdots, \beta^{(t)})$ using: the message, the `Salt` and the commitments;

6. compute $(\mathbf{y}^{(1)}, \cdots, \mathbf{y}^{(t)})$ as described in CROSS-ID, and then generate h by hashing;

7. generates the vector of the second challenge $\mathbf{b} = (b^{(1)}, \cdots, b^{(t)}) \in \{0;1\}^t$ from the hash of: message, `Salt`, commitment, response and h. Such a vector has fixed Hamming weight equal to ω. In addition, the set J is defined as the support of b, characterizing the indices i such that $b^{(i)} = 1$;

8. computes `SeedPath` as the set of intermediate nodes of the seed tree, which are needed to recompose the $\texttt{Seed}^{(i)}$, for $i \in J$;

9. configures `MerkleProof` as a structure containing the proofs for the leaves $c_{0\ i \notin J}^{(i)}$, in order to recompute the root of the Merkle tree;

10. the signature obtained will be

$$\texttt{Signature} = \{\texttt{Salt}, c_0, c_1, h, \texttt{SeedPath}, \texttt{MerkleProof}(T_0), \{\mathbf{y}^{(i)}, \sigma^{(i)}, c_1^{(i)}\}_{i \notin J}\}.$$

Private Key $\mathbf{e} \in \mathbb{E}^n$
Public Key $\mathbb{E}^n$, $\mathbf{H} \in \mathbb{F}_p^{(n-k)\times n}$, $\mathbf{s} = \mathbf{e}\mathbf{H}^\top \in \mathbb{F}_p^{n-k}$

Input Message $\mathtt{Msg}$
Output Signature $\mathtt{Signature}$

SIGNER	VERIFIER
Sample $\mathtt{MSeed} \xleftarrow{\$} \{0;1\}^\lambda$, $\mathtt{Salt} \xleftarrow{\$} \{0;1\}^{2\lambda}$	
Generate $(\mathtt{Seed}^{(1)},\cdots,\mathtt{Seed}^{(t)}) = \mathtt{SeedTree(MSeed,Salt)}$	
For $i = 1,\cdots,t$:	
Sample $(\mathtt{Seed}^{(\mathtt{u}')},\mathtt{Seed}^{(\mathtt{e}')}) \xleftarrow{\mathtt{Seed}^{(i)}} \{0;1\}^{2\lambda}$	
Sample $\mathtt{u}'^{(i)} \xleftarrow{\mathtt{Seed}^{(\mathtt{u})'}} \mathbb{F}_p^n$, $\mathtt{e}'^{(i)} \xleftarrow{\mathtt{Seed}^{(\mathtt{e}')}} \mathbb{E}^n$	
Compute $\sigma^{(i)} \in \mathbb{E}^n$ such that $\sigma^{(i)}(\mathtt{e}'^{(i)}) = \mathtt{e}$	
Set $\mathtt{u}^{(i)} = \sigma^{(i)}(\mathtt{u}'^{(i)})$	
Compute $\tilde{\mathtt{s}}^{(i)} = \mathtt{u}^{(i)}\mathtt{H}^\top$	
Set $c_0^{(i)} = \mathtt{Hash}(\tilde{\mathtt{s}}^{(i)}, \sigma^{(i)}, \mathtt{Salt}, i)$	
Set $c_1^{(i)} = \mathtt{Hash}(\mathtt{u}'^{(i)}, \mathtt{e}'^{(i)}, \mathtt{Salt}, i)$	
Set $T = \mathtt{MerkleTreee}(c_o^{(1)},\cdots,c_o^{(t)})$	
Compute $c_o = T.\mathtt{Root}()$	
Compute $c_1 = \mathtt{Hash}(c_1^{(1)},\cdots,c_1^{(t)})c$	
Generate $(\beta^{(1)},\cdots,\beta^{(t)}) = \mathtt{GenCh}_1(c_o, c_1, \mathtt{Msg}, \mathtt{Salt})$	
For $i = 1,\cdots,t$:	
Compute $\mathbf{y}^{(i)} = \mathbf{u}'^{(i)} + \beta^{(i)}\mathbf{e}'^{(i)}$	
Compute $h^{(i)} = \mathtt{Hash}(\mathbf{y}^{(i)})$	
Compute $h = \mathtt{Hash}(h^{(1)},\cdots,h^{(t)})$	
Generate $(b^{(1)},\cdots,b^{(t)}) = \mathtt{GenCh}_2(c_o, c_1, \beta^{(1)},\cdots,\beta^{(t)}, h, \mathtt{Msg}, \mathtt{Salt})$	
Set $J = \{i \mid b^{(i)} = 1\}$	
Set $\mathtt{SeedPath} = \mathtt{SeedPath(MSeed,\ Salt,\ J)}$	
For $i \notin J$:	
Set $f^{(i)} := (\mathbf{y}^{(i)}, \sigma^{(i)}, c_1^{(i)})$	
Compute $\mathtt{MerkleProofs} = T.\mathtt{Proofs}(\{1,\cdots,t\} \setminus J)$	
Set $\mathtt{Signature} = \{\mathtt{Salt}, c_0, c_1, h, \mathtt{SeedPath}, \mathtt{MerkleProofs}, \{f^{(i)}\}_{i\notin J}\}$	
$\xrightarrow{\mathtt{Signature}}$	

Figure 3.2: Signature Generation

Verification:

1. generates the vector of the first challenge $(\beta^{(1)},\cdots,\beta^{(t)})$ from $\mathtt{Msg}$, $\mathtt{Salt}$, c_0 and c_1;

2. generates the vector of the second challenge $(b^{(1)},\cdots,b^{(t)})$ from $\mathtt{Msg}$, $\mathtt{Salt}$, c_0, c_1, $\beta^{(1)},\cdots,\beta^{(t)}$ and h;

3. using $\mathtt{SeedPath}$, generates i seeds $\mathtt{Seed}_{i\in J}^{(i)}$;

4. for $i \in J$, recomputes $c_1^{(i)}$, $\mathbf{y}^{(1)}$, and $h^{(1)}$;

5. For $i \notin J$, compute $h^{(i)} = \text{Hash}(\mathbf{y}^{(i)})$ and $c_1^{(i)}$;
6. uses the `MerkleProof`, along with the $c_0^{(i)}{}_{i \notin J}$, to recompute and verify the c_0 root;
7. verifies $c_1 = \texttt{Hash}(c_1^{(1)}, \cdots, c_1^{(t)})$;
8. verifies $h = \texttt{Hash}(h^{(1)}, \cdots, h^{(t)})$.

Futhermore, the specific functions for the realization of complex data structures, which are implicit for now, will be made explicit in the next section.

Private Key $\mathbf{e} \in \mathbb{E}^n$
Public Key $\mathbb{E}^n$, $\mathbf{H} \in \mathbb{F}_p^{(n-k)\times n}$, $\mathbf{s} = \mathbf{e}\mathbf{H}^\top \in \mathbb{F}_p^{n-k}$
Input *Message* `Msg`
Output *Signature* `Signature`

PROVER		VERIFIER
	$\xrightarrow{\texttt{Signature}}$	
		Generate $(\beta^{(1)}, \cdots, \beta^{(t)}) = \texttt{GenCh}_1(c_0, c_1, \texttt{Msg}, \texttt{Salt})$
		Generate $(b^{(1)}, \cdots, b^{(t)}) = \texttt{GenCh}_2(c_0, c_1, \beta^{(1)}, \beta^{(t)}, h, \texttt{Msg}, \texttt{Salt})$
		Set $J = \{i \mid b^{(i)} = 1\}$
		Generate $\{\texttt{Seed}_{i\in J}^{(i)} = \texttt{GetSeeds(SeedPath, Salt)}\}$
		For $i \in J$:
		Compute $(\texttt{Seed}^{(\mathbf{u}')}, \texttt{Seed}^{(\mathbf{e}')}) \xleftarrow{\texttt{Seed}^{(i)}} \{0;1\}^{2\lambda}$
		Sample $\mathbf{u}'^{(i)} \xleftarrow{\texttt{Seed}^{(\mathbf{u}')}} \mathbb{F}_p^n$, $\mathbf{e}'^{(i)} \xleftarrow{\texttt{Seed}^{(\mathbf{e})'}} \mathbb{E}^n$
		Set $c_1^{(i)} = \texttt{Hash}(\mathbf{u}'^{(i)}, \mathbf{e}'^{(i)}, \texttt{Salt}, i)$
		Compute $\mathbf{y}^{(i)} = \mathbf{u}'^{(i)} + \beta^{(i)}\mathbf{e}'^{(i)}$
		Compute $h^{(i)} = \texttt{Hash}(\mathbf{y}^{(i)})$
		For $i \notin J$:
		Set $h^{(i)} = \texttt{Hash}(\mathbf{y}^{(i)})$
		Compute $\tilde{\mathbf{s}}^{(i)} = \sigma^{(i)}(\mathbf{y}^{(i)})\mathbf{H}^\top - \beta^{(i)}\mathbf{s}$
		Set $c_1^{(i)} = \texttt{Hash}(h^{(1)}, \cdots, h^{(t)})$
		Verify $h = \texttt{Hash}(h^{(1)}, \cdots, h^{(t)})$
		Verify $c_0 = \texttt{VerifyMerkleRoot}(\{c_0^{(i)}\}_{i\notin J}, \texttt{MerkleProof})$
		Verify $c_1 = \texttt{Hash}(c_1^{(1)}, \cdots, c_1^{(t)})$

Figure 3.3: Signature Verification

Size Analysis

The scheme just described is characterised by a public key `pk` with dimension

$$|\texttt{pk}| = \underbrace{(n-k)\lceil \log_2(p) \rceil}_{\mathbf{s}} + \underbrace{\lambda}_{\texttt{Seed}_{pk}} .$$

Similarly, the signature size is

$$|\texttt{Signature}| = \underbrace{8\lambda}_{h,c_0,c_1,\texttt{Salt}} + \underbrace{\lambda(t-\omega)\log_2\left(\frac{t}{t-\omega}\right)}_{\texttt{SeedPath}} + \underbrace{2\lambda\Big(1+(t-\omega)\log_2\left(\frac{t}{t-\omega}\right)\Big)}_{\texttt{MerkleProof}} + \\ + \underbrace{(t-\omega)\Big(\underbrace{2\lambda}_{c_1^{(i)}} + \underbrace{n\lceil\log_2(p)\rceil}_{\mathbf{y}^{(i)}} + \underbrace{m\lceil\log_2(z)\rceil}_{\sigma^{(i)}}\Big)}_{f^{(i)},\ i\notin J}.$$

Chapter 4

Security Analysis

Recall that we provided both the EUF-CMA security statement in lemma 11, ensuring that the scheme is designed to resist forgery attacks, and the hardness of the underlying problem in lemma 10; in the next, we will focus on non-structural attack realized via ISD framework in order determine the choice of parameters for a given security level.

Specifically, we will first provide a general description of the the best-known algorithms for the subset sum problem adopted to $R-SDP$ in [5] for the particular case of $z \in 2, 4, 6$ and in [20] for arbitrary values of z , and then we will show how to optimize the collision search subroutine using the approach proposed in [17] for solving the Nearest Neighbor Search problem using the Locality Sensitive Framework. Finally, an estimate of the security level of $R-SDP$ will be given when $z = 2$ and $q = 127$, i.e., the one chosen in [26].

4.1 Partial Gauss Elimination Step

As already explained, the algorithms covered in the following analysis will follow the $PGE + ISD$ framework presented in Algorithm 1.

Quasi-Systematic Form

Given the parity-check matrix $H \in \mathbb{F}_p^{(n-k)\times(n)}$, an information set $I \subseteq \{1, \ldots, n\}$ of size k is chosen and H is brought into quasi-systematic form. For this, let I' be a set of size $k + \ell$ which contains the information set I and transform H as

$$UHP = \tilde{H} = \begin{bmatrix} Id_{n-k-\ell} & H_1 \\ 0 & H_2 \end{bmatrix}$$

where $U \in \mathbb{F}^{(n-k)\times(n-k)}$ is an invertible matrix and $P \in \mathbb{F}^{(n-k)\times(n-k)}$ is a permutation matrix.

Definition 4.1.1. Given the parity check matrix $H \in \mathbb{F}_p^{(n-k)\times(n)}$, an information set $I' \subseteq \{1, \ldots, n\}$ of size $k + \ell$ such that $I \subseteq I'$, the cost of computing the quasi-systematic form of H, following the PGE framework is given by:

$$C_{QSF}(p, n, k, \ell) = (n - k - \ell)^2)(n + 1)\left(\lceil \log_2(p) \rceil + \lceil \log_2(p) \rceil^2\right)$$

This splits the unknown error vector e into the positions indexed by I', i.e., $eP^T = (e_1, e_2)$. Thus, we get the system of two equations.

$$\begin{aligned} e_1 + e_2 H_1^\intercal &= s_1 \\ e_2 H_2^\intercal &= s_2 \end{aligned}$$

Recalling that we assume that e has t non zero entries within I' and $\omega - t$ on the other $n - (k + \ell)$ coordinates , thus $e_1 \in \mathbb{E}_{n-(k+\ell),\omega-t}$ and $e_2 \in \mathbb{E}_{k+\ell,\omega}$. To solve the system, one enumerates solutions e_2 of the second equation $e_2 H_2^\intercal = s_2$ and checks for each one if the remaining $e_1 = s_1 - e_2 H_1^\intercal$ completes it to a valid, i.e., restricted, solution.

In the next we will analyze some approach that can be used to efficiently compute the list of vectors e_2 that are solution of

$$s_2 = e_2 H_2^T \tag{4.1}$$

In particolar one can state that:

Lemma 12. *An algorithm using the PGE setup and ISD framework finds a valid solution for* 3.1.1 *with time complexity*

$$C_{PGE+ISD}(p, z, n, k, \omega) = C_{QSF}(p, n, k) + C_{Solver}(p, z, n, k, t)$$

and probability:

$$P_{succ} = \left(\frac{\binom{k+\ell}{t}\binom{n-(k+\ell)}{\omega-t}}{\binom{n}{\omega}} \right)$$

Proof. Each execution of the algorithm first computes the Gaussian partial elimination setting and then uses a particular approach to find the solution of 4.1, this explain the cost of a single iteration. The average number of iteration, recalling 2.10, is the inverse of the success probability Pr_{succ} for the assumed distribution of weight t. □

4.2 Stern Algorithm

Let us begin by analyzing an algorithm based on meet-in-the-middle, this approach was adopted for the syndrome decoding problem by Stern and Dumer in [13] and [14].

This approach considers a partition of $[k+\ell]$ by choosing uniformly at randomly two subsets P_1 and P_2, with size $\frac{\lfloor k+\ell \rfloor}{2}$ and $\frac{\lceil k+\ell \rceil}{2}$ respectively, such that $[k+\ell] = P_1 \cup P_2$, then enumerates the solutions for 4.1 by a collision search of the two lists defined as follows.

- $L_1 = \{(x_A, (x_A, 0)H_2^T) \,|\, x_A \in \mathbb{E}^{\lfloor \frac{k+\ell}{2} \rfloor}\}$ with size $|L_1| = z^{\lfloor \frac{k+\ell}{2} \rfloor}$
- $L_2 = \{(x_B, s_2 - (0, x_B)H_2^T) \,|\, x_A \in \mathbb{E}^{\lceil \frac{k+\ell}{2} \rceil}\}$ with size $|L_2| = z^{\lceil \frac{k+\ell}{2} \rceil}$

We find a solution (x_A, x_B) for each pair x_A and x_B such that

$$(x_A, 0)H_2^T \;=\; s_2 - (0, x_B)H_2^T$$

Algorithm 3: Restricted Stern

Input: n, k, p, z, $s \in F_p^{n-k}$, $H \in F_p^{n \times (n-k)}$
Internal Parameters: ℓ
Output: Partial error solution e_2
s_2, $H_2 \leftarrow ComputeQuasiStandardForm(s, H, \ell)$
$L \leftarrow MergeList(L_1, L_2, \ell, s_2, k+\ell)$

Lemma 13. *The cost of a single iteration of Stern's algorithm on restricted set requires on average:*

$$C_{Stern}(p,z,n,k,t) = \frac{C_{MergeLists}(L_1,L_2,\ell,s)}{z^n p^{k-n}}$$

where

$$C_{MergeLists}(L_1,L_2,\ell,s) = |L_1|\left(\left\lfloor\frac{k+\ell}{2}\right\rfloor(\lceil\log_2(p)\rceil + \lceil\log_2(p)\rceil^2) + \log_2(|L_1|)\right) + |L_2|\left(\left\lceil\frac{k+\ell}{2}\right\rceil(\lceil\log_2(p)\rceil + \lceil\log_2(p)\rceil^2) + \log_2(|L_2|)\right) + \frac{|L_1||L_2|}{p^\ell}$$

Proof. This cost derives directly from 8, the only difference is that a concatenation merge is performed here and not a sum merge, this is because it is not necessary to check the weight condition because the collisions depend only on the syndrome value. The overall cost is divided for the number of solutions. □

Lemma 14. *The Stern algorithm requires an amount of memory equal to:*

$$M_{Stern}(p,z,n,k,t) = min\{M_1, M_2\}$$

Where $|M_i| = |L_i|(u_i \log_2(z))$ *for* $i = 1, 2$

4.3 Properties of the Restricted Set Structure

Let us first discuss some possible approaches for exploiting the properties of values living in the restricted set; in particular, following the analysis carried out in [5], we will focus on how different choices of z affect the number of representations for a vector with a generic weight and values living in the restricted set.

Exploiting Additive Structure of Restricted Set

To increase the number of representations was proposed a slight modification to the additive structure of the restricted set: the search is not carried out within lists containing vectors with values in $\mathbb{E}_0$ but in $\mathbb{E}_0 \cup \mathbb{D}$, where $\mathbb{D}$ contains elements defined from original set $\mathbb{E}$.

In the following we will use $\mathbb{E} \cup \mathbb{D}_{n,u}$ to denote the set of vectors of length n with u and entries in $\mathbb{E}$ and d_i in $\mathbb{D}$, as $(\mathbb{E} \cup \mathbb{D})_{n,u,d}$. If the entries live only within the original restricted set we will simply refer to this set as $\mathbb{E}_{n,u}$.

The new algebraic set has a different additive structure from that of $\mathbb{E}$, more specifically for a given $x \in \mathbb{E}$, one can define:

$$\alpha_E(x) := |\{b \in \mathbb{E} \,|\, \exists\, c \in \mathbb{E} \; s.t. \; b + c = x\}| \tag{4.2}$$

$$\alpha_D(x) := |\{b \in \mathbb{E} \,|\, \exists\, c \in \mathbb{D} \; s.t. \; b + c = x\}| \tag{4.3}$$

These two amounts denote the number of possibilities to write an element $x \in \mathbb{E}$ as $b+c$, with both $b, c \in \mathbb{E}$ or $b \in \mathbb{E}$ and $c \in \mathbb{D}$. Since for our choice of $\mathbb{E}$ these quantities do not depend from the particular $x \in \mathbb{E}$, we simply denote them as α_E and α_D.

Furthermore, given an $x \in \mathbb{D}$ we will use the following notation:

$$\beta_x := |\{b \in \mathbb{E} \,|\, \exists\, c \in \mathbb{E} \; s.t. \; b + c = x\}| \; for \, x \in \mathbb{D} \tag{4.4}$$

This refers to the number of possible pairs in $\mathbb{E}$ that when summed return x; since this amount does not depend on x, so we will simply refer to it as β.

Shifting Restricted Set Values

In the case of error vectors with large weight, as proposed in [26], given $x \in \mathbb{E}$ it is possible to define the new set

$$\mathbb{E}_x = \{a - x \,|\, a \in \mathbb{E}\} \setminus \{0\}$$

Chosen $\tilde{x} = (-x, \ldots, -x)$, with $\tilde{x}H^T = s_x$, one can consider a new instance of the problem by changing the target vector, we ask to find the vector $\tilde{e} = e + \tilde{x} \in (\mathbb{E}_x \cup \{0\})^n$ such that $\tilde{e}H^T = s + s_x$. Since the entries of e are picked independently, the Hamming weight of the new instance follows a binomial distribution:

$$Pr(wt_H(\tilde{e}) = \omega) = \frac{\binom{k+\ell}{\omega}(z-1)^w}{z^{k+\ell}}$$

Having assumed that the target error weight e for 4.1 has weight u, this is the probability that u_x entries of e are equals to the shifting$x$ while the other $u - u_x = \omega$ don't, it follows that the total cost of the generic solver is then the cost of a single iteration divided by the success probability.

It should be noted, however, that the sets obtained by shifting may not preserve their own additive structure: one can build from the $\mathbb{D}$ corresponding to $\mathbb{E}$ a $\mathbb{D}_x$

which fits $\mathbb{E}_x$. This is done by shifting the elements of $\mathbb{D}$ and neglecting those that would represent zero.

Estimate the number of representations

We now go on to analyze our reference instance of the restricted set, where for $g = 2$ non primitive element of $\mathbb{F}_p$ with multiplicative order $z = 7$ we have:

$$\mathbb{E} = \{g^i (\mod p) \,|\, i \in \{0, \ldots, 6\}\}.$$

The supporting additive set is defined as:

$$\mathbb{D} = \{a - b \,|\, a, b \in \mathbb{E}\} \setminus \mathbb{E}_0$$

Lemma 15. *Given $e \in (\mathbb{E}_0 \cup \mathbb{D})_{(k+\ell,u_i,d_i)}$, the overall numbers of pairs $e_A, e_B \in (\mathbb{E}_0 \cup \mathbb{D})_{(k+\ell,u_{i+1},d_{i+1})}$, with $u_{i+1} = \frac{u_i}{2} + \varepsilon_{i+1}$ and $d_{i+1} = \frac{d_i}{2} + \delta_{i+1}$, such that $e = e_A, e_B$ is given by:*

$$\rho_i = \binom{u_i}{u_{i+1}} \binom{u_{i+1}}{2\varepsilon_{i+1}} \alpha_{\mathbb{E}}^{2\varepsilon_{i+1}} \binom{(u_i/2)-\varepsilon_{i+1}}{\delta_{i+1}}^2 \alpha_{\mathbb{D}}^{2\delta_{i+1}} \binom{d_i}{(d_i/2)}$$

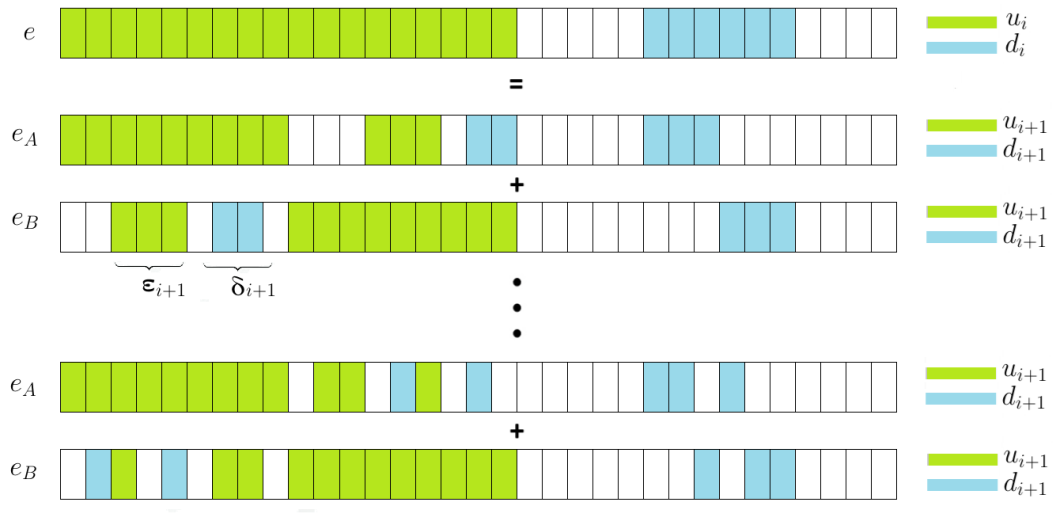

Figure 4.1: Representations of a generic vector when $z = 7$

Proof. This formula, shown in Figure 4.1, can be explained as follows: the first three terms describe all the possible ways in which we can distribute the entries with values living to the set $\mathbb{E}$ of the two addend vectors, such that, when summed, they overlap in $2\varepsilon_{i+1}$ coordinates, resulting in u_i values in set $\mathbb{E}$ while α_E, the additivity of $\mathbb{E}$,

specify the number of possible ways we can choose the $2\varepsilon_{i+1}$. The following two terms describe all the possible way to choose the overlaps of the δ_{i+1} extra values in $\mathbb{D}$ of one vectors in the remaining $(u_i/2) - 2\varepsilon_{i+1}$ coordinates of the other one, such that when summed they result in values in $\mathbb{E}$, for each we can choose α_D different values (and then do the same for the other vector). The last term represents all possible ways to obtain the d_i coordinates in $\mathbb{D}$ by splitting. □

Smaller Restricted Sets

Let us analyze the case of the restricted sets for small and even values of z, in particular focusing on how it changes the additive structure, and consequently the number of representations.

We first see that the following lemma holds:

Lemma 16. *Let $\mathbb{E} = \{g^j | j = 0, \ldots, z-1\} \subseteq \mathbb{F}_p$ be a restricted set, $g \in \mathbb{F}_q^*$ a non primitive element of multiplicative order z. When z is even, the restricted set can be expressed as*

$$\mathbb{E} = \{\pm g^i | i \in \{0, \ldots, \tfrac{z}{2} - 1\}$$

Proof. Since the product of every pair of elements of the restricted set is still an element of $\mathbb{E}$, it is enough to prove that -1 is in $\mathbb{E}$.

By definition $g^z = 1$ when $i = z$ and $g^i \neq 1$ if $i < z$, then when we consider $x = g^{\frac{z}{2}}$ we have:

$x^2 = g^z = 1 \implies x^2 - 1 = 0 \implies (x+1)(x-1) = 0$, since $z/2 < z$, then $x = g^{\frac{z}{2}} = -1$ □

Next we will consider in the case of restricted sets constructed with small and even values of z, where their will depend on the factorization of $(x^z - 1)$ and supporting set $\mathbb{D} \subseteq \{a + b \,|\, a, b \in \mathbb{E}\}$. Moreover, according to lemma 16, the definition of these particular sets allows to build sum to 0 pairs of values in some of the overlapping entries.

z = 2

In this first case we have the sets

$$\mathbb{E} = \{\pm 1\} \qquad \mathbb{D} = \{\pm 2\}$$

To introduce the representations for this particular case we must first note that the way the sets are defined implies that $\mathbb{E}$ does not possess an additive structure, consequently $\alpha_E = 0$, while $\alpha_D = 1$ and $\beta = 1$.

Lemma 17. *For $e \in (\mathbb{E} \cup \mathbb{D})_{k+\ell,u,d}$ the number of $e_1, e_2 \in (\mathbb{E} \cup \mathbb{D})_{k+\ell,u_i,d_i}$ such that $e_1 + e_2 = e$ is given by:*

$$\rho_i = \sum_{\delta_E \in U_\delta}^{n} \binom{u}{u/2} \binom{u}{\delta_E}^2 \binom{d_i}{\varepsilon_D+\delta_D,\varepsilon_D} \binom{k+\ell-(u+d)}{o_E}$$

Where: $U_\delta = \left[max\{0, di - \frac{d}{2}\}, min\{d - \frac{\varepsilon - d_i}{2}, d_i, \frac{u}{2}\}\right]$, $d_{i+1} = \delta_D + \delta_E$ and $u_i = \frac{u}{2} + o_E + \varepsilon_D$.

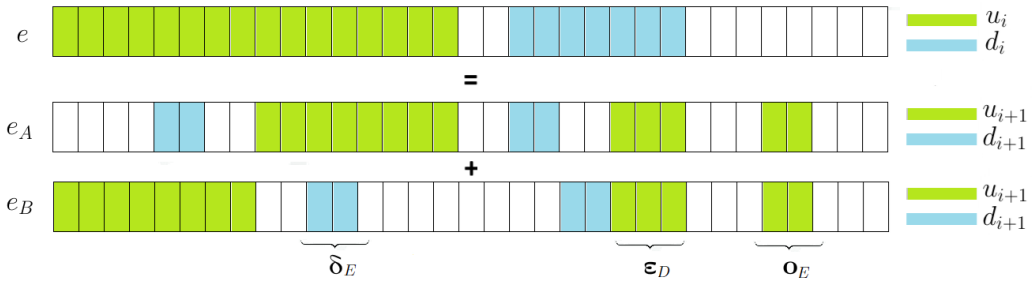

Figure 4.2: Representations of a generic vector when $z = 2$

Proof. This formula, shown in Figure 4.1, can be explained as follows: the first term represents all possible ways of obtaining the elements into $\mathbb{E}$, without overlaps. From both of the $u_i/2$ entries we can choose δ_E position for the overlaps, which explain the second binomial coefficient. Then the d_i entries are obtained via the non-overlapping distribution of the remaining δ_D values in $\mathbb{D}$ while the other δ_E entries from the overlap of ε_D entries in $\mathbb{D}$. The last term represents all possible distributions of the zero-sum o_E pairs in $\mathbb{D}$ on the remaining coordinates. □

z = 4

In this case we have

$$\mathbb{E} = \{\pm 1, \pm g\} \qquad \mathbb{D} = \{\pm(g \pm 1)\} \subseteq \{a + b \,|\, a, b \in \mathbb{E}\}$$

While $\alpha = 0$ remains, unlike the previous case there are two possible ways of representing an $x \in \mathbb{E}$ as a sum of $a \in \mathbb{E}$ and $b \in \mathbb{D}$, so $\alpha_D = 2$, as well as for each $x \in \mathbb{D}$ there are two possible pairs of $a \in \mathbb{E}$ and $b \in \mathbb{D}$ such that $x = a + b$, then $\beta = 2$.

Consequently, the number of representations is the same as 17 but increased by $\alpha_D^{2\delta_E}\beta^{\varepsilon_D}$.

z = 6

In this case, we have $\mathbb{E} = \{\pm 1, \pm g, \pm(g-1)\}$. Note that $\mathbb{E}$ already possesses additive structure: any element $e \in \mathbb{E}$ can be obtained as $e = e_1 + e_2 = e_2 + e_1$ with $e_2, e_1 \in \mathbb{E}$, $e_2 \neq e_1$, then $\alpha_E = 2$

$$\mathbb{E} = \{\pm 1, \pm g \pm (g-1)\} \qquad \mathbb{D} = \emptyset \tag{4.5}$$

Lemma 18. *For $e \in (\mathbb{E} \cup \mathbb{D})_{k+\ell,u,d}$ the number of $e_1, e_2 \in (\mathbb{E} \cup \mathbb{D})_{k+\ell,u_i,d_i}$ such that $e_1 + e_2 = e$, when $z = 6$ is given by:*

$$\rho_i = \binom{u}{u/2+\varepsilon_i, 2\varepsilon_i} \alpha_E^{2\varepsilon_i}$$

4.4 BJMM Algorithm

$BJMM$ is a the multi-level algorithm which uses representations from a sum partition instead of a set partition and decompose the search for the error vector by constructing partial solutions. In each level, the algorithm identifies vectors with a certain weight whose syndrome is consistent with the equation for a specific number of input. Using the representations technique allows to improve the collision search process: for each level it is possible to choose the number of syndrome symbols on which to merge so as to reduce the number of intermediate results but expecting that a representation of the final solution will be found on average.

The purpose of this analysis is to provide a generalized description of the $BJMM$ algorithm with a generic number of levels M, valid for the different instances of the restricted set previously analyzed.

Parameters Description

For the vectors on the i-th level we will denote u_i and d_i as the number of entries in $\mathbb{E}$ and $\mathbb{D}$, respectively.

Then, recalling that we are searching for $e \in \mathbb{E}_{k+\ell,u_0}$, the weights for the M levels are:

$$\begin{array}{lllll} u_0 = u_0 & \dots & u_i = \frac{u_{i-1}}{2} + \varepsilon_i & \dots & u_M = \frac{u_{m-1}}{2} \\ d_0 = 0 & \dots & d_i = \frac{d_{i-1}}{2} + \delta i & \dots & d_M = \frac{d_{m-1}}{2} \end{array}$$

So, the internal parameters which can be optimized are: $\varepsilon_1, \dots, \varepsilon_{m-1}$, $\delta_1, \dots, \delta_{m-1}$, corresponding, respectively, to the overlapping entries in $\mathbb{E}$ and $\mathbb{D}$ on each level and ℓ, the redundancy of the small instance due to the partial Gaussian elimination.

When the set is shifted, the algorithm must also identify the optimal weight distribution, which entails optimizing $u_0 \in \{0, \dots, k+\ell\}$. Otherwise, it always starts from full weight vectors.

It is also important to note that when the elements from $\mathbb{D}$ are not involved in the construction of intermediate lists, thus $\delta_i = 0$ for $i \in 1, \dots, M-1$.

Find error vector via partial solutions

On the i-th level, the algorithm will merge the input lists on a specific number of entries of the syndrome, taking into account a specific target vector. Specifically, on level i $BJMM$ perform a merging subroutine on ℓ_i symbols of the syndrome:

$$\ell_i = \lfloor log_p(\rho_i) \rfloor \text{ - } \textstyle\sum_{j=i}^{M-1} \ell_j$$

From 6 we know that algorithm 2 when merge two lists L_1, L_2 on ℓ_i with a target vector $t \in F_p^{\ell_i}$ obtain an average number of collision equal to $(|L_1||L_2|)/p^{\ell_i}$, the choice of ℓ_i guarantees that at least one representation of the final solution will survive each merge on average.

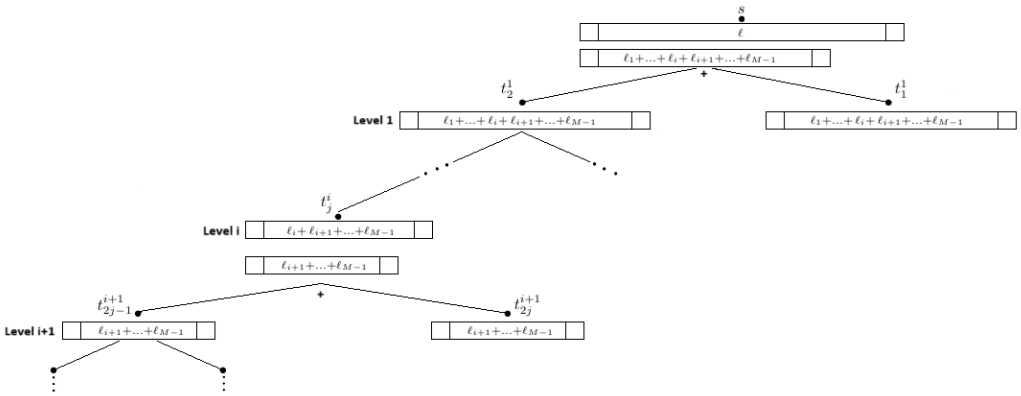

Figure 4.3: Tree decomposition of the target vectors

The syndrome vector can be decomposed at each level through a top-down procedure as shown in Figure 4.3. We start from the first level by choosing two vectors t_1^1, t_2^2 whose sum satisfies the syndrome over a specific number of values. Iterating this process, we obtain a binary tree structure of depth M, in which each node is a vector. The sum of each pair of child nodes satisfies the syndrome value of the parent node over a certain number of values.

In particular on the i-th level, the algorithm merges 2^i pairs of input lists on ℓ_i symbols using the target vectors $t_1^i, \ldots, t_{2^i}^i$, each can be decomposed choosing $t_{2j-1}^{i+1}, t_{2j}^{i+1} \in \mathbb{F}_p^{\tilde{\ell}_i}$ such that $t_j^i|_{\tilde{\ell}_i} = t_{2j-1}^{i+1} + t_{2j}^{i+1}$ for $j \in \{1, \ldots, 2^i\}$ with $\tilde{\ell}_i = \sum_{j=i}^{M}(\ell_j)$

Next we will show how by choosing this approach, all vectors in the output lists of the i-th level partially satisfy the equation on $\sum_{j=i}^{M-1} \ell_j$ symbols.

Algorithm Description

The algorithm starts by computing 2^M base lists to be merged in the upper level. The process for defining base list pair begins with the selection of a partition of $[k+\ell]$. This is done, by randomly choosing two subset P_1 and P_2, with size $\frac{\lfloor k+\ell \rfloor}{2}$ and $\frac{\lceil k+\ell \rceil}{2}$ respectively, such that $[k+\ell] = P_1 \cup P_2$, and then one can define:

$$L_{B_i} = \{e \in (\mathbb{E} \cup \mathbb{D})^{k+\ell} \,|\, wt_H(e) = u_M + d_M \;\; supp(e) \subset P_i\} \quad \text{for } i \in (1,2)$$
$$\text{with size } |L_{B_i}| = \tbinom{|P_i|}{u_M, d_M} z^{u_M} z_{\mathbb{D}}^{d_M}$$

Lemma 19. *The cost of single iteration on BJMM algorithm with M level on restricted set 4 requires on average:*

Algorithm 4: Restricted $BJMM$ with depth-M

Input: n, k, p, z, $z_{\mathbb{D}}$, $\alpha_{\mathbb{E}}$, $\alpha_{\mathbb{D}}$, $s \in F_p^{n-k}$, $H \in F_p^{n\times(n-k)}$
Internal Parameters: u_0, ℓ, $\varepsilon_1, \ldots, \varepsilon_{M-1}$, $\delta_1, \ldots, \delta_{M-1}$
Output: Partial error solution vector
s_2, $H_2 \leftarrow ComputeQuasiStandardForm(s, H, \ell)$
$u_1, \ldots, u_M \leftarrow ComputeWeightsE(u_0, \varepsilon_1, \ldots, \varepsilon_{m-1})$
$d_1, \ldots, d_M \leftarrow ComputeWeightsD(d_0, \delta_1, \ldots, \delta_{m-1})$
$\ell_0, \ldots, \ell_{m-2} \leftarrow ComputeSymbols(u_0, u_1, \ldots, u_M, \varepsilon_1, \ldots, \varepsilon_{m-1}, d_1, \ldots, d_M, \delta_1, \ldots, \delta_{m-1})$
for $i = 0$ *to* $M-1$ **do**
  for $j = 1$ *to* 2^i **do**
    $t^{i+1}_{2j-1}, t^{i+1}_{2j} \leftarrow ComputeTargetVectors(t^i_j)$
  end
end
$i = M$
for $i = M-1$ *to* 1 **do**
  for $j = 1$ *to* 2^i **do**
    $L_{(i,j)} \leftarrow MergeList(L_{(i+1,2j-1)}, L_{(i+1,2j)}, \ell_i, t_{(i,j)}, u_i, d_i)$
  end
end

$$C_{BJMM}(p, z, n, k, t) = \frac{\sum_{i=0}^{M-1}\sum_{j=1}^{2^i} C_{MergeLists}(L^{i+1}_{2j-1}, L^{i+1}_{2j}, \ell_i, t_{i,j})}{z^{(k+\ell)}p^{k-n}}$$

Where, from Lemma 2, we have:

$$\begin{aligned}
&C_{MergeLists}(L^{i+1}_{2j-1}, L^{i+1}_{2j}, \ell_i, t_{i,j}) = \\
&(|L^{i+1}_{2j-1}| + |L^{i+1}_{2j}|)\left((u_{i+1}+d_{i+1})\ell_i(\lceil\log_2(p)\rceil + \lceil\log_2(p)\rceil^2)\right) + \\
&(|L^{i+1}_{2j-1}|)\log_2(|L^{i+1}_{2j-1}|) + \\
&(|L^{i+1}_{2j}|)\log_2(|L^{i+1}_{2j}|) + \\
&\frac{|L^{i+1}_{2j-1}||L^{i+1}_{2j}|}{p^{\ell_i}}(k+\ell)\lceil\log_2(p)\rceil
\end{aligned}$$

Proof. On the level i one need to merge on ℓ_i symbols of the syndrome 2^i pairs of lists from the lower level, as shown in figure 4.4, containing vector with u_{i+1} values in $\mathbb{E}$ and d_{i+1} in $\mathbb{D}$. Then, for each level from $M-1$ to 0 the algorithm performs the procedure 2 for each input lists pair. The overall cost is divided for the number of solutions. □

It is important to note that the vectors of weight $u_{i+1} + d_{i+1}$ within the input

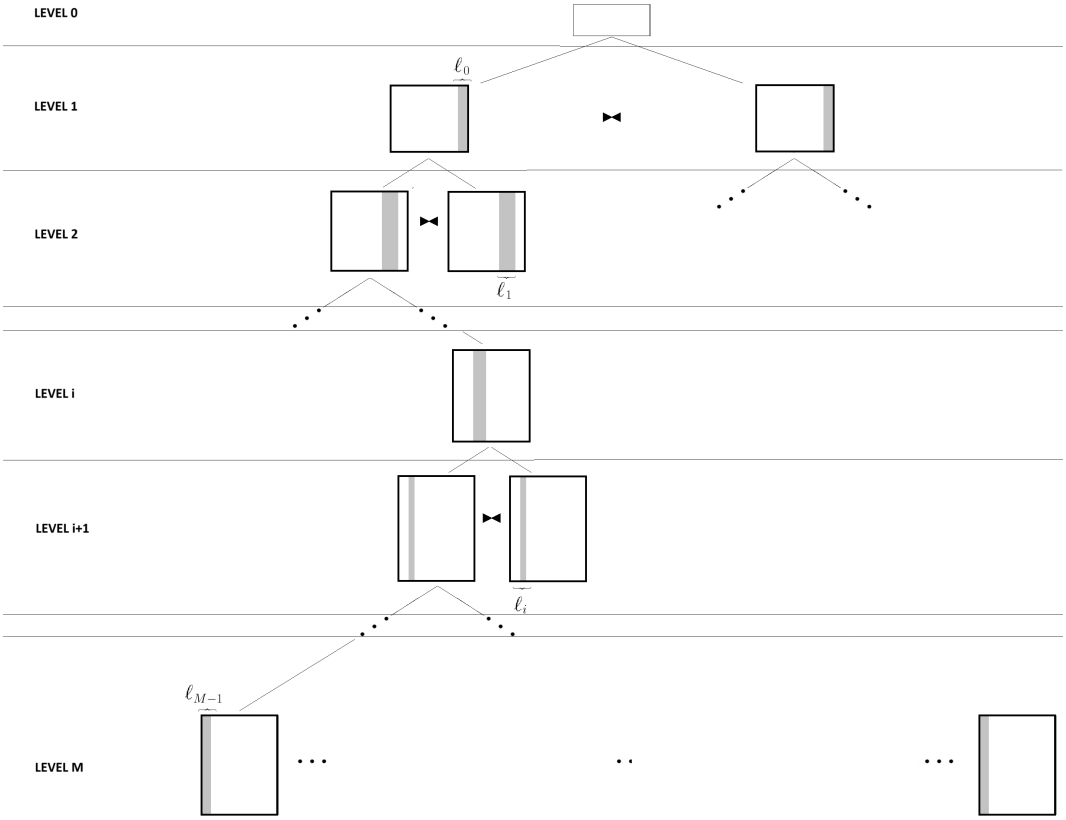

Figure 4.4: BJMM with M levels

lists at the i-th level partially satisfy the equation for $\sum_{j=i+1}^{M-1} \ell_j$ symbols, while the vectors within the output list at the i-th level are those of weights $u_i + d_i$ that partially satisfy it on additional ℓ_i symbols, resulting in a total of $\sum_{j=i}^{M-1} \ell_j$ symbols of the syndrome satisfied. iterating this procedure and remembering how the symbols where to merge are defined, this ensures that the solution found in the last level is of weight u_0 and satisfies the equation on ℓ symbols.

Assuming from the lemma 9, we can state that the size of the j-th list on level i , for $j \in \{1, \ldots, 2^i\}$, is equal to:

$$|L_{i,j}| \;=\; min\left\{ \frac{|L_{2j-1}^{i+1}||L_{2j}^{i+1}|}{p^{\ell_i}} \,,\; \binom{k+\ell}{u_i+d_i}\binom{u_i+d_i}{d_i} z^{u_i} z_{\mathbb{D}}^{d_i} p^{-\sum_{j=i}^{M-1}\ell_j} \right\}$$

Having defined the lists size for each level, one can estimate the amount of memory required:

Lemma 20. *The BJMM algorithm with depth M requires an amount of memory*

equal to:

$$M_{BJMM}(p,z,n,k,t) = max_{i\in\{1,\dots,M\}}\{min\{M_{i,j}, M_{i,j+1}\}\ for\ odds\ j\ \in\{1,\dots,2^i\}\}$$

Where $|M_{i,j}| = |L_{i,j}|\,(u_i\ \log_2(z) + d_i\ \log_2(z_D))$

Proof. Each merging subroutine requires to store at least one of the two input lists, for each pair we can store the smaller one. The cost is given my the size of the larger memorized list. □

4.5 Improve Collision Search via LSF

In this section, we will demonstrate how the definition of problem 2.5.1 can be adapted to the search for the error vector that satisfies equation 4.1. Subsequently, we will analyze how a solver LSF-based for the the NN-search problem can be adapted to the $R-SDP$.

The search aims to find a solution vector $e \in \mathbb{E}_{k+\ell,u}$ from the collision search between two lists, each containing vectors with values only in $\mathbb{E}$ or constructed using also the set $\mathbb{D}$. Since a pair $(e_A, e_B) \in (\mathbb{E}\cup\mathbb{D})_{k+\ell,u_A,d} \times (\mathbb{E}\cup\mathbb{D})_{k+\ell,u_B,d}$ is a solution $e_A + e_B \in \mathbb{E}_{k+\ell,u}$, they must have exactly $o = u_A + u_B - u$ overlaps, then the search for a pair whose sum satisfies the condition on weight can be carried out among those that have exactly o overlaps.

We want to analyze how this approach introduces an improvement for the $BJMM$ algorithm for the different versions of the restricted set previously discussed. In fact different choices of z and g imply restricted sets with different properties in the additive structures resulting for each in different types of values that must be summed to obtain the desired solution.

Therefore, given the mapping function $B : (\mathbb{E}\cup\mathbb{D})^{k+\ell} \to \mathbb{F}_2^{k+\ell}$, such that $b = B(e)$ is the binary support of the restricted vector e, we will search for the pairs $e_A, e_B \in L_1 \times L_2$ such that $e_A + e_B \in \mathbb{E}_{k+\ell,u}$ among those for which $B(e_A) \wedge B(e_B) \in \mathbb{W}^2_{k+\ell,o}$.

Definition 4.5.1. γ-Nearest Neighbor Search Given two lists $L_1, \subseteq \mathbb{W}^2_{k+\ell,u_1,d}$ and $L_2 \subseteq \mathbb{W}^2_{k+\ell,u_2,d}$ find all pairs $(x,y) \in (L_1 \times L_2)$ s.t. $x \wedge y \in \mathbb{W}^2_{,k+\ell,o}$.

So we will show how to solve this problem on input lists $B(L_1), B(L_2)$, and then check if the original vector of one of the solutions is a valid restricted error.

Note that it holds for several choices of z, since it refers only to the number of non-zero entries that must overlap in a pair of fixed solutions for their sum to return a vector with desired weight and values in restricted set.

For example, when $z = 2$ the set $\mathbb{E}$ does not have an additive structure, but each value has an inverse, so in the overlapping region the entries of $\mathbb{E}$ add up to 0, deleting extra entries, while in order to get a value in $\mathbb{E}$ in the resulting vector it is necessary that one entry with value in $\mathbb{E}$ and one with value in $\mathbb{D}$ be added together. On the other hand, when $z = 7$ values in $\mathbb{E}$ don't have an inverse but there are more possibilities to obtain a value in $\mathbb{E}$ by the sum of two entries.

Geometric Interpretation of the Algorithm

Let us first recall the definition of a region for a generic $v \in \mathbb{W}^2_{k+\ell,u+d}$:

$$Region_{v,\gamma} = \{x \in F_2^n \,:\, (x \wedge v) \in \mathbb{W}^2_{k+\ell,\gamma}\}$$

Given $0 \leq f \leq k+\ell$ we are interested in those vectors in the region the region whose weight is equal f.

Definition 4.5.2. Spherical Cap For integers $0 \leq \gamma \leq f \leq k+\ell$ and $v \in \mathbb{W}^2_{k+\ell,f}$, a Spherical Cap is defined as:

$$\mathbb{C}_{v,f,\gamma} := \mathbb{W}^2_{k+\ell,u+d} \cap Region_{v,\gamma}$$

$\mathbb{C}_{v,f,\gamma}$ includes all elements on the f-Sphere with exactly γ overlaps with v.

The spherical cap definition allows the identification of the set of valid filters for v. This is achieved by considering the subset $\mathbb{C}_f \subseteq \mathbb{W}^2_{k+\ell,f}$:

$$B_{v,f,\gamma} = \mathbb{C}_f \cap \mathbb{C}_{v,\omega,\gamma} = \mathbb{C}_f \cap Region_{v,\gamma}$$

Lemma 21. *For integers $0 \leq \gamma \leq f \leq n$ and a fixed vector $v \in (\mathbb{E} \cup \mathbb{D})_{k+\ell,u_i,d}$ the number of $c \in \mathbb{C}_f$ such that $c \wedge B(v) \in \mathbb{W}^2_{k+\ell,\gamma}$ is given by:*

$$P_{u_i+d,\gamma} = Vol(B_{v,f,\gamma}) = \binom{u_i+d}{\gamma} \binom{(k+\ell)-(u+d)}{f-\gamma}$$

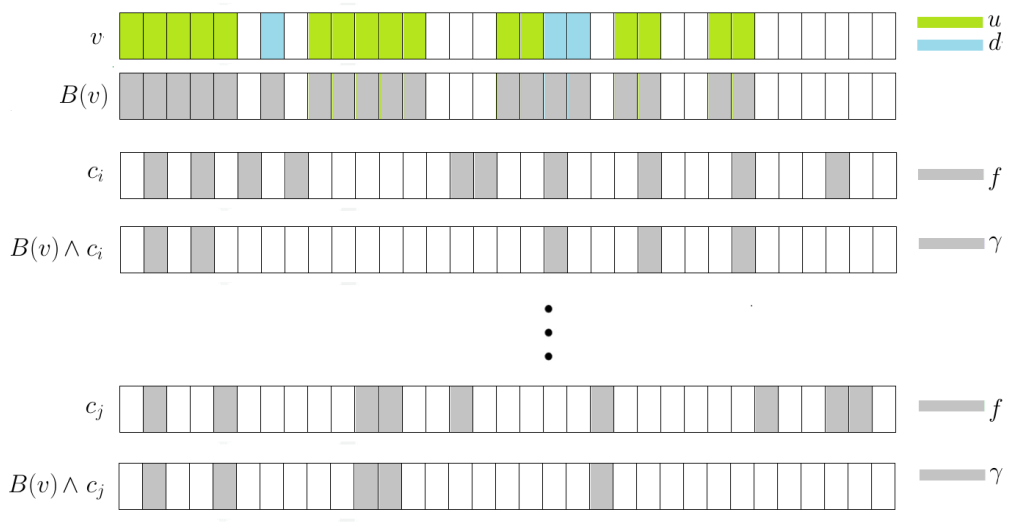

Figure 4.5: Valid Filters for a generic vector

Proof. As previously mentioned, the valid centers must have exactly γ overlaps in the binary support support of the restricted vector e. Thus, the first term represent all the possible ways to choose γ overlap out the non zero entries of v and the second term the remaining many 1's in the 0-coordinates. □

Next, we will analyze the application of a search algorithm that is designed to identify collisions between two lists of vectors, where the vectors within each list have the same Hamming weight, but the Hamming weights could differ slightly, by a maximum of one unit, between the two lists.

Furthermore, since we are interested in finding pairs with a certain number of overlaps, an efficient strategy seems to be to choose those pairs that have fewer or the same number of overlaps as required in one fraction of the coordinates and none in another fraction.

This search is realized through filters, in particular we can use two different bucketing parameters for the two lists to be optimized, i.e. γ_A, γ_B and look for those pairs that have a certain number of common overlaps with the same filter and a certain number of exclusive overlaps (i.e. where the other vector does not have them).

The following definition introduces the set of bucket centers that detect a fixed solution pair (e_A,e_B), i.e., all the centers $c \in \mathbb{C}_f$ such that $B(e_A) \wedge c \wedge B(e_B) \in \mathbb{W}^2_{k+\ell,\gamma_u}$ where $\gamma_u \leq \gamma$.

Definition 4.5.3. Solution-detecting filters Given $e_A \in (\mathbb{E} \cup \mathbb{D})_{k+\ell,u_A,d}$ and $e_B \in (\mathbb{E} \cup \mathbb{D})_{k+\ell,u_B,d}$ such that $(e_A + e_B) \in \mathbb{E}_{k+\ell,u}$, for integers $0 \leq \gamma_1, \gamma_2 \leq u_A, u_B, f \leq n$ with $x = B(e_A)$ and $y = B(e_B)$ the set of filters which detect a fixed solution (e_A, e_B) is defined as:

$$W_{x,y,f,\gamma_1,\gamma_2} := \{c \in \mathbb{C}_F : (c \wedge x) \in \mathbb{W}^2_{k+\ell,\gamma_A} \wedge (c \wedge y) \in \mathbb{W}^2_{k+\ell,\gamma_B}\} :=$$
$$B_{x,f,\gamma_1} \cap B_{y,f,\gamma_1} := \mathbb{C}_f \cap Region_{x,\gamma_1} \cap Region_{y,\gamma_2}$$

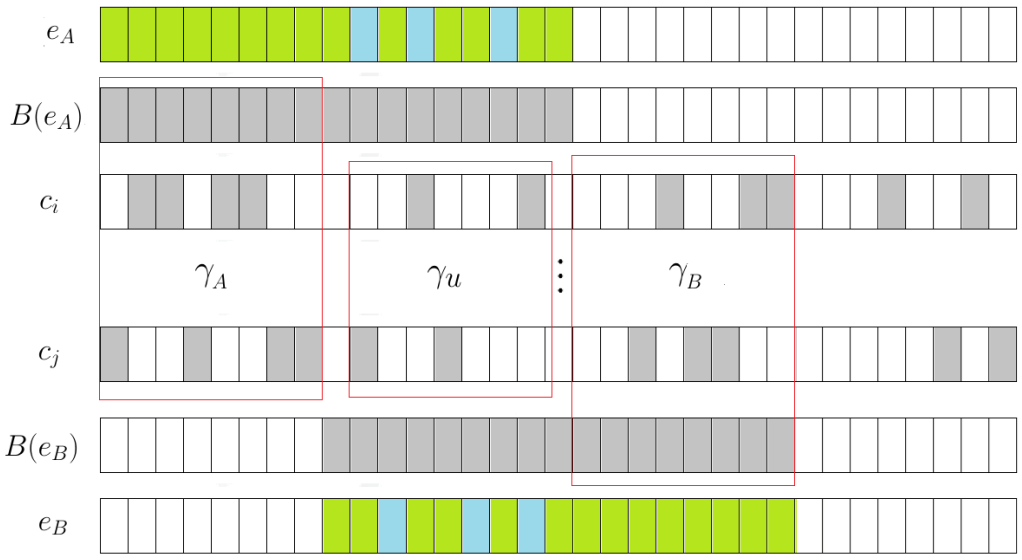

Figure 4.6: Valid Filters for fixed solution pair

Lemma 22. *For a fixed solution* (e_A, e_B) *with* $e_A \in (\mathbb{E} \cup \mathbb{D})_{k+\ell,u_A,d}$ *and* $e_B \in (\mathbb{E} \cup \mathbb{D})_{k+\ell,u_B,d}$ *the set defined in 4.5.3 has size:*

$$D_{x,y} = \sum_{\gamma_u \in \Gamma_u} \binom{o}{\gamma_u} \binom{(u_A+d)-o}{\gamma_A-\gamma_u} \binom{(u_B+d)-o}{\gamma_B-\gamma_u} \binom{k+\ell-((u_A+d)+(u_B+d)-o)}{f-(\gamma_A+\gamma_B-\gamma_U)}$$

Where :

$o = u_A + d + u_B + d - u$

$\Gamma_u = [max\{0, (\gamma_A + o) - (u_A + d - \frac{o}{2}), (\gamma_B + o) - (u_B + d - \frac{o}{2})\}, min\{\gamma_A, \gamma_B, o\}]$

Proof. The first term represents all the possible ways in which i can choose $gamma_u$ entries of a filter such that it overlaps with both only in the common region, while the range specifies for a given choice of γ_A and γ_B how many common overlaps we can identify still detecting both vector. The second and third terms represent the

exclusive overlaps with the two vectors, respectively, while the last term represents those in which the filter doesn't overlap with either. □

In order to determine what the bucketing parameter values are such that the algorithm actually returns a solution, we recall that a given solution must have exactly o overlaps, consequently for it to be able to detect it must hold that:

$$\gamma_1 + \gamma_2 \geq max\{0, (f+u) - (k+\ell)\}$$

This relationship, given the hamming weight of the filters, establishes a lower bound on the minimum number of overlaps it must necessarily have with both vectors.

Reducing Filter Set Size

In [17] is proposed a the construction of the set of filters by Random Product Codes (RPC) which, taking into account the number of filters that detect a given solution, allows to minimize the size of $\mathbb{C}_f$ such that each solution is still detect with a certain probability P.

For a fixed size of the filters set the probability that exist at least one filter $c_i \in \mathbb{C}_f$ such that $c_i \in W_{x,y,f,\gamma_A,\gamma_B}$ is given by:

$$P_{succ} = 1 - \left(1 - \frac{D_{x,y}}{\mathbb{W}^2_{k+\ell,f}}\right)^{|\mathbb{C}_f|} \tag{4.6}$$

Then one can reduce the size of filters set taking into account this probability and the size of the set W as defined in 22, in particular by choosing:

$$|\mathbb{C}_f| = \frac{\mathbb{W}^2_{k+\ell,f}}{D_{x,y}} \tag{4.7}$$

We can expect that a solution is found with good probability.

LSF Solver for R-SDP

Now we will show how to improve the collision search by using the Locality Sensitive Functions. The solver performs two steps: first assigns each $x \in L_1$ to $Bucket_{(c,\gamma)}$ for all $c \in B_{\gamma_1,x}$, we will refer to this phase as bucketing, and the repeat the procedure

for each $x \in L_2$ searching for a matching element in each $Bucket_{(c,\gamma_2)}$ for all $c \in B_{\gamma,y}$, this phase is called Checking phase.

Recall that this solver enumerates all the valid solution pairs in $L_1 \times L_2$ for the problem 2.5.1 and search among them a solution for 4.1.

Algorithm 5: *NN* search solver via *LSF*

Input : Two lists $L_1 \in (\mathbb{E} \cup \mathbb{D}_{n,u_a,d}), L_2 \in (\mathbb{E} \cup \mathbb{D}_{n,u_a,d})$, a target weight u, the filters set $\mathbb{C}_f$ of size $|\mathbb{C}_f|$, and two bucketing parameters γ_1, γ_2

Output : list L_{pairs} containing pairs $(x, y) \in (L_1 \times L_2)$ with $wt_H(x + y) = \omega_t$

$L_{pairs} = \emptyset$
for $e_1 \in L_1$ **do**
  for $c \in B_{B(e_1),f,\gamma}$ **do**
    store e_1 in $Bucket_{c,\gamma}$
  end
end
for $e_2 \in L_2$ **do**
  for $c \in B_{B(e_1),f,\gamma}$ **do**
    for $e_1 \in Bucket_{c,\gamma}$ **do**
      if $e_1 + e_2 \in \mathbb{E}_{n,u}$ **then**
        store (e_1, e_2) in L_{pairs}
      end
    end
  end
end
return L_{pairs}

The next lemma gives the cost of the algorithm:

Lemma 23. *Given two lists $L_1 \subseteq (\mathbb{E} \cup \mathbb{D}_{k+\ell,u_A,d})$ and $L_1 \subseteq (\mathbb{E} \cup \mathbb{D}_{k+\ell,u_B,d})$, and a filters set $\mathbb{C}_F \subseteq \mathbb{W}^2_{n,f}$ of size $|\mathbb{C}_F| = \frac{\mathbb{W}^2_{k+\ell,f}}{D_{x,y}}$, one iteration the algorithm 5, on average, returns a list of candidate solution pairs for 4.1, with probability P_{succ} as defined in 4.6 and time complexity:*

$$C_{LSF}(L_1, L_2, u) = |L_1| \frac{P_{u_A+d,\gamma_A}}{D_\gamma} min(u_{A+d}, f) + |L_2| \frac{P_{u_B+d,\gamma_B}}{D_\gamma} \frac{P_{u_B+d,\gamma_B}}{\mathbb{W}^2_{k+\ell,f}} (u_A + u_B - u)$$

Where γ_A and γ_B are the internal parameters to be optimized.

Proof. Assume that we start bucketing L_1, the probability of a vector being stored in a generic filter is given by P_{u_A+d,γ_A}, and since we consider only a $D_{x,y}$-fraction of all possible filters and that we have to repeat this operation for each vector we can justify the cost of first phase as $|L_1| \left(P_{u_A+d,\gamma_A}\right)/(D_\gamma)) \, min(u_{A+d}, f)$. The second step require to perform the same operation for each vector in L_2, with a cost equal

to $(|L_2|)(P_{u_2+d,\gamma_B})/(D_\gamma)$, and for each vector check whether there is a solution in the center where it is stored, i.e. for each vector if their sum in the overlapping region returns a full weight vector, where the average size of a filter is given by $(P_{u_2+d,\gamma_B})/(\mathbb{W}^2_{k+\ell,f})$ this explains the cost of the second phase. □

In terms of memory cost, the algorithm will have to store only the buckets containing the elements of the first list, consequently we can state that:

Lemma 24. *One iteration the algorithm 5 require, on average*

$$M_{LSF}(L_1, L_2, u) = |L_1|P_{u_1+d_1}$$

Representation Technique and Locality Sensitive Functions

One can improve the 4 using 5, replacing the merging search on last level for a vectors pair $(e_A, e_B) \in (L_1 \times Ł_2) \subseteq (\mathbb{E} \cup \mathbb{D})_{k+\ell,u_A,d} \times (\mathbb{E} \cup \mathbb{D})_{k+\ell,u_B,d}$ whose sum is a restricted vector of weight t with the NN-search for the pairs with $o = u_A + u_B - u$ overlaps. as shown in Figure.

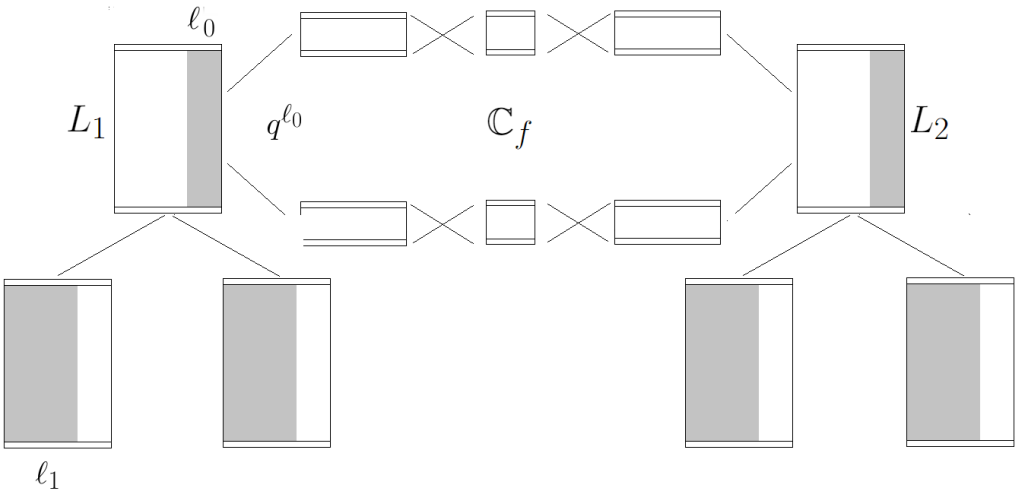

Figure 4.7: $BJMM$ with depth 2 using LSF

Lemma 25. *The cost of a single iteration on $BJMM$ algorithm with M level on restricted set with LSF solver on last level requires on average:*

$$C_{Bjmm+LSF}(p, z, n, k, t) = \frac{\left(\sum_{i=1}^{M-1}\sum_{j=1}^{2^i} C_{MergeLists}(L_{2j-1}^{i+1}, L_{2j}^{i+1}, \ell_i, t_{i,j})\right) + C_{LSF}(L_1, L_2, u)}{z^{k+\ell}p^{k-n}}$$

Chapter 5

Performances

In this chapter we evaluate the time complexity of the algorithms discussed in the previous chapter. In particular, we will show two main applications:

- cryptanalysis: we will consider some $R-SDP$ instances that have already appeared in the literature and that have already been attacked in [5]. We show that, using the new algorithm we have devised, the security level gets lowered as the new algorithm significantly outperforms all previously known solvers;
- security of CROSS: we consider the cost of the new algorithm on CROSS instances. We show that the recommended CROSS instance preserve the claimed security level, even when considering the new $R-SDP$ solver.

5.1 Cryptanalysis of R-SDP instances

We will now evaluate the time complexity of the discussed algorithms, focusing on the performance of the algorithm 4 when using the solver 5 at the last level.

We show that this new approach outperforms the classical $BJMM$ for several choices of z, p, n, k.

To support this claim, we consider some of the instances which have already been attacked [5]. For these instances, $BJMM$ with two levels resulted in the best attack. We show that, using the new algorithm with LSF , we are able to further reduce the security level by a significant amount. The considered instances, together with the old security levels and the new ones (recomputed according to our new algorithm), are shown in Table 5.1.

z	p	n	R	W	claim (bit)	$Bjmm_2$	$Bjmm_{2+LSF}$
2	16381	500	0.75	0.13	128	76	51
4	197	384	0.5	0.34	177	103	94

Table 5.1: Comparison of attack performance on some instances of $R - SDP$

It can be seen that this new approach significantly improves attacks on $R - SDP$ instances with different parameters. Indeed, while we will later analyze the case with a full-weight error vector, here we search for larger but sparse vectors defined over smaller restricted sets.

5.2 Confirmation of the security level for CROSS instances

We now consider CROSS instances and evaluate the performances of our algorithm. We focus on the instance recommended for category 1, whose parameters are as follows: $p = 127$, $z = 7$,$n = 127$, $R = 0.66$, $W = 1$.

Stern

For completeness we show the computational cost using Stern. In Figure 5.1 we show, for fixed p, z, n, k, w, the overall complexity of algorithm 3 for different values of ℓ, i.e. the cost of solving the reduced instance 4.1 for $0 \leq \ell \leq n - k$.

We can see that the optimum value is above the threshold of 143 bit. It should be noted that attempting to optimize the search using the LSF framework won't produce any improvement, because the vectors of the two lists are obtained by set partition, all possible pairs when summed return a full weight vector.

BJMM with depth 2 on set $\mathbb{E} \cup \mathbb{D}$

We now discuss how using the LSF framework improves $BJMM$ performance. In Figure 5.2 we can see that using our proposed new version results in a gain of 12 bit, reducing the time complexity of the attack but remaining above the claimed security level.

It is also noted that after a certain value of ℓ the two algorithms have the same cost, which is not surprising since collision search can be said to be a special case of NN-search via LSF when full-weight filters are considered, that is, there is only one

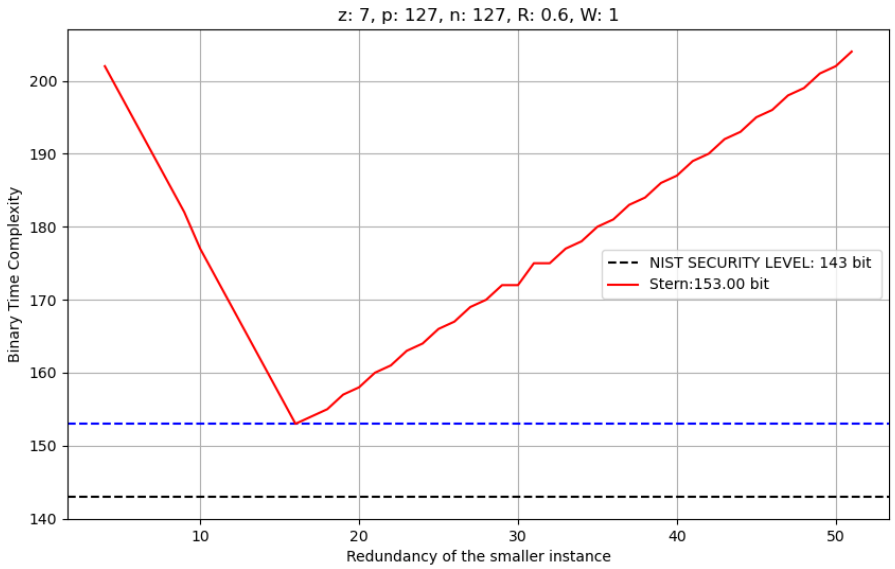

Figure 5.1: Stern performances as redundancy of the smaller instance changes

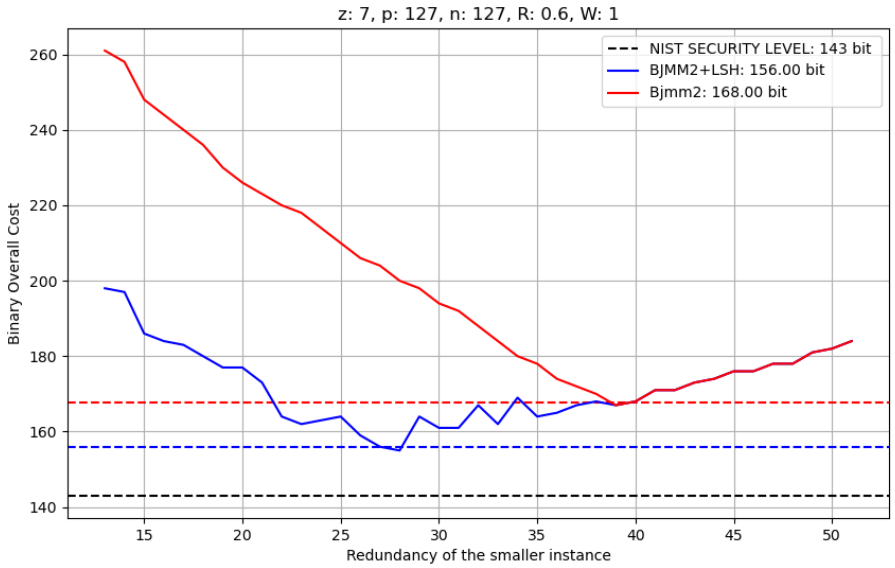

Figure 5.2: Comparison of the two approaches for $BJMM$ on the set $\mathbb{E} \cup \mathbb{D}$.

filter.

However, by focusing on the ℓ interval for which our approach obtains significantly better results, we can show how collision search is improved:

In Figure 5.3 we have shown the number of checks we have to make for each element in the lists. In particular we show this result for the second list, for the first

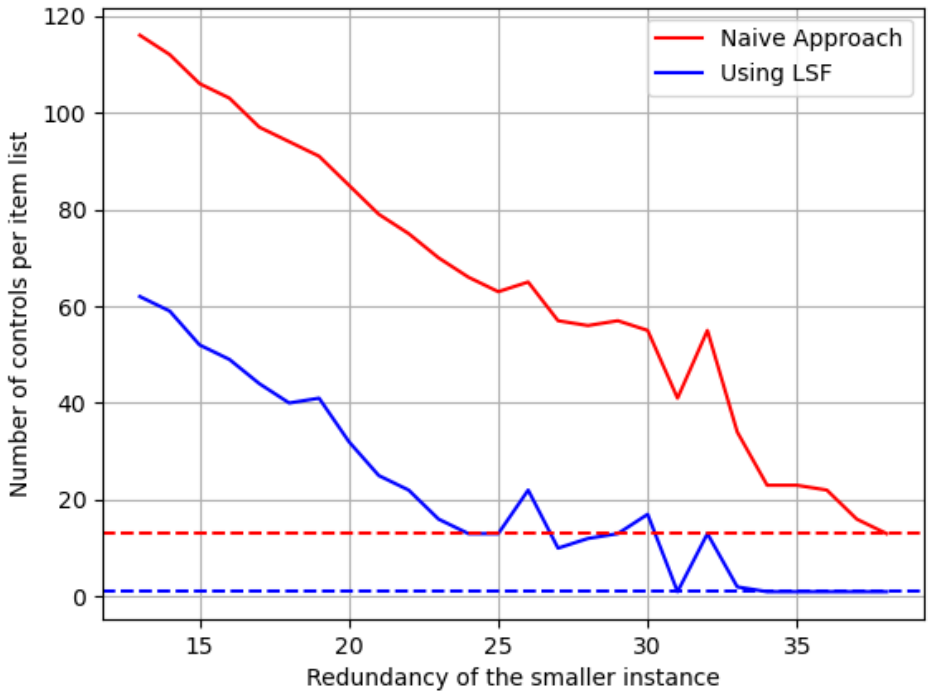

Figure 5.3: Comparison of the number of controls per element with LSF vs naive approach.

one it will be similar.

Recall that before searching for collisions in the algorithm $MergeLists$ the two lists are sorted according to the syndrome value over a certain number of symbols, which in our case is ell_0, thus comparisons are made between elements that satisfy the requirement on the syndrome value, this means that each element of the second list on average is compared with $|L_1|/p^{\ell_0}$ elements of the first list.

When using LSF solver 5, before performing the collision search, the vectors are first sorted on ℓ_0 syndrome symbols then filtered according to the distribution of their binary support, then comparisons are made only among vectors with satisfies the requirement on the syndrome value and on weight distribution.

Then recalling the proof of lemma 23, the size of valid filters given in and the size of the filters set chosen via 4.7 , each element of second list is compared with a

number of elements of the first list, on average, equal to:

$$\frac{P_{u_b+d}}{D}\left(\frac{P_{u_a+d}}{|C_f|}\frac{L_1}{p^{\ell_0}}\right)$$

This through the optimization of the internal parameters of the LSF solver is less than that obtained by using the naive approach.

We can also show that using our approaches also results in a memory gain, remembering how this costs are defined in 20 and 24, in Figure 5.4 we show the memory used by the two approaches.

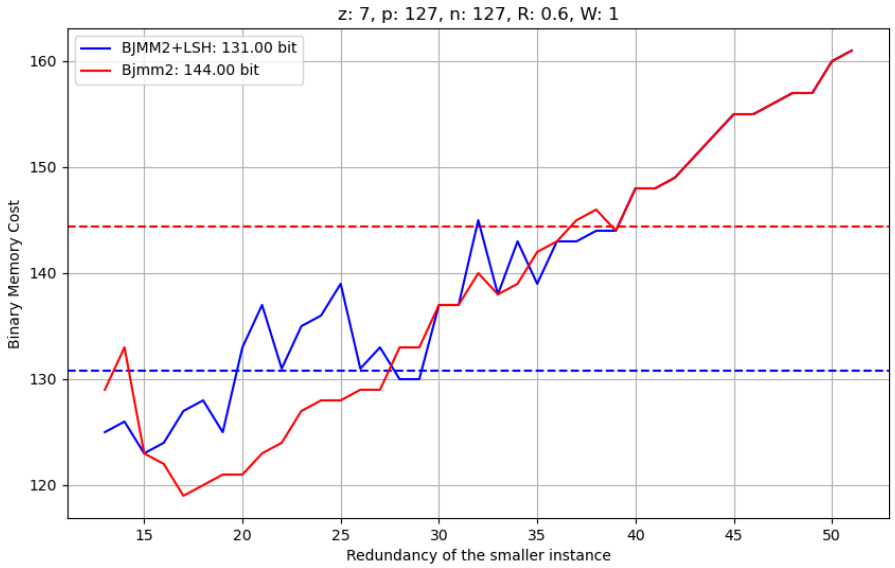

Figure 5.4: Comparison of required memory for the two approaches.

In particular, we indicate the amount of memory used in correspondence with the value of optimal ℓ values of the two approaches.

BJMM with depth 2 on shifted set $\mathbb{E} \cup \mathbb{D}$

We now show how using the LSF framework improves $BJMM$ performance when we shift the restricted set, i.e. when considering the case described in section 4.3:

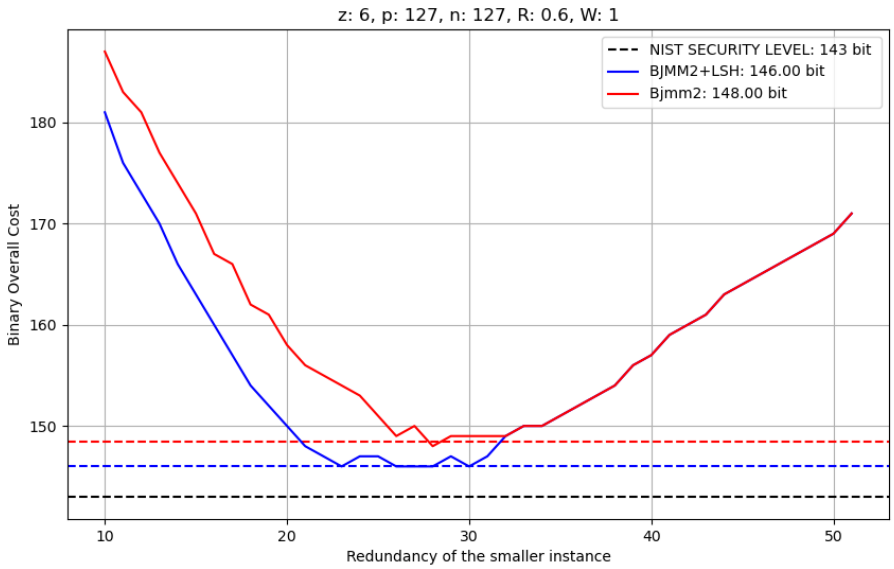

Figure 5.5: Comparison of the two approaches for $BJMM$ when restricted set is shifted.

As we can see in Figure 5.5 this case while achieving an improvement in performance, the gain in bit is not as significant as in the previous case.

Indeed, as we can in figure 5.6 in the case of the shifted set, the graphs representing the cost of collision search as ℓ varies results in a very similar trend in both cases. In particular, the optimal values differ by a few bit.

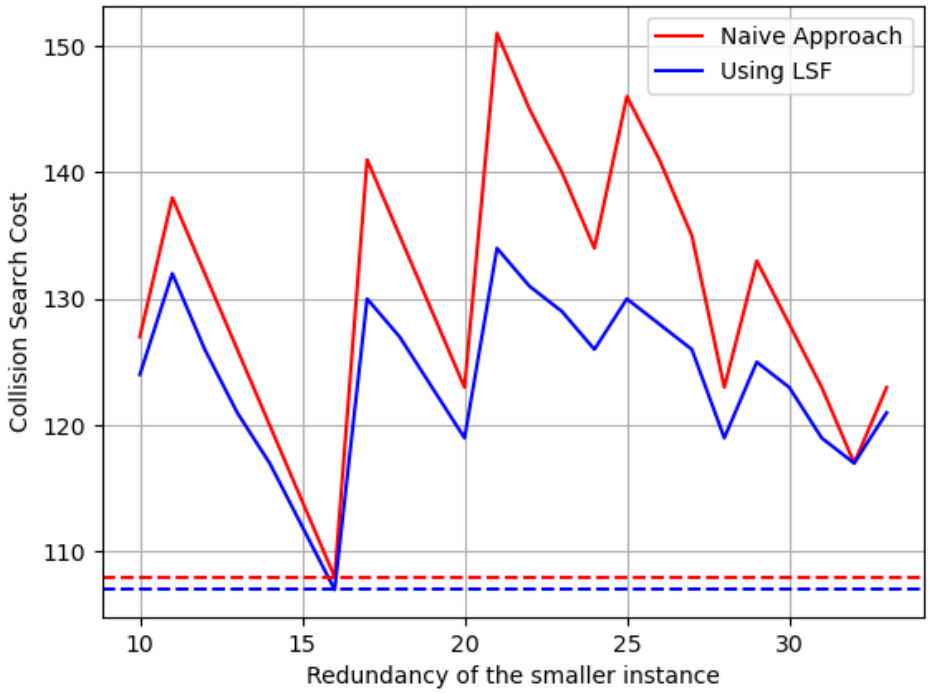

Figure 5.6: Collision search performances on shifted set $\mathbb{E} \cup \mathbb{D}$ when using the LSF vs Naive approach.

CROSS Security

Attack	Overall Cost (bit)
$Stern$	153
$Bjmm_2$	168
$Bjmm_{2+LSH}$	156
$Shifted\, Bjmm_2$	148
$Shifted\, Bjmm_{2+LSF}$	146

Table 5.2: Cost of Attacks for $p = 127$, $z = 7$, $n = 127$, $R = 0.66$, $W = 1$

In conclusion, as shown in the Table 5.2, our approach significantly improves $BJMM$ when used on the set $\mathbb{E} \cup \mathbb{D}$, gaining 12 bits of memory while providing improvements in memory utilization as well. However, it still remains slightly worse than Stern while $Bjmm_{2+LSF}$ on the shifted set $\mathbb{E} \cup \mathbb{D}$ results in the best time complexity among the attacks proposed for $RSDP$, requiring 2^{146} operation on average, which means that CROSS achieves NIST security level I even after the analysis of this new attack.

Chapter 6

Conclusions and Future Works

In this thesis, we first provided some general notions about algebraic coding theory and cryptographic concepts related to zero-knowledge identification protocols. Then we described the main features and properties of CROSS, the digital signature scheme derived via Fiat-Shamir transformation of t parallel executions zero-knowledge identification protocol CROSS ID.

In parallel, we presented the syndrome decoding problem (SDP) defined over a finite prime field and its variant defined over a the restricted set, namely the $R-SDP$, which is the NP-hard underlying problem of CROSS.

Finally, we presented the Information set decoding and the best ISD-based solvers for $R-SDP$ and in particular, we analyzed how to improve $BJMM$, an algorithm based on the representations technique, through the use of a solver for the Nearest Neighbour Search Problem based on Locality Sensitive Functions.

Indeed, we have seen that the application of this particular approach makes it possible to optimize the collision search proposed in $BJMM$: solutions for $R-SDP$ are found among those whose binary support is a solution for the NN search problem.

The results obtained allow for two important interpretations : on the one hand, the cryptanalysis of some schemes in the literature based on $R-SDP$ is improved by reducing their security level, on the other hand, it is shown that CROSS, as designed, achieves the NIST security level I even for this new attack.

Combining the LSF approach with the representations technique, on which $BJMM$ is based, has made it possible to improve the search for collisions of two lists of vectors: establishing a positional prerequisite of the support of vectors so that their sum returns a vector of the desired weight, as has been shown, to decrease

the number of comparisons.

Starting from these considerations, an interesting development could be to analyze the asymptotic behaviour of this new approach in order to analyze how performance changes as code size increases.

Furthermore, as we have already discussed, optimization was achieved through the use of a positional relation between the supports of the vector, of particular interest would be to explore the possibility of using the relations between the values of the restricted set as search criteria, trying to see if they could provide approaches to improve collision search.

As for the relationship between the performance of our approach and the parameters of CROSS, let us remember that the particularity of this scheme lies in the reduced size of the signature and in fast performances, it would be necessary to understand whether for another choice of parameters that improve the performance of the scheme, the required security level would still be achieved.

Similarly, some versions of CROSS use a particular version of $R - SDP$, i.e. $R - SDP(G)$, whereby solutions take value from a particular subset of the restricted set, it would be worth analyzing how our approach performs in the case of this particular instance.

Finally, it should be emphasised the importance to understand the performance of these algorithms when they are implemented. Let us recall that in this thesis a theoretical evaluation has been provided. For these attacks to provide good performance in real contexts, they must be subject to optimization both in terms of the execution of the individual operations and sub-routines that realize the analyzed algorithms, and in terms of the memory used. The cost in terms of memory, in particular, would require a more detailed analysis that takes into account the implementation strategies and the used devices.

Bibliography

[1] Richard P Feynman. Simulating physics with computers. In *Feynman and computation*, pages 133–153. cRc Press, 2018.

[2] Y. Manin. Computable and uncomputable. *Sovetskoye Radio*, 128, 1980.

[3] Peter W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[4] Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 203–228, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[5] Marco Baldi, Sebastian Bitzer, Alessio Pavoni, Paolo Santini, Antonia Wachter-Zeh, and Violetta Weger. Generic decoding of restricted errors. 03 2023.

[6] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. volume 263, 03 1999.

[7] Daniel Kales and Greg Zaverucha. An attack on some signature schemes constructed from five-pass identification schemes. Cryptology ePrint Archive, Paper 2020/837, 2020. `https://eprint.iacr.org/2020/837`.

[8] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[9] S. Barg. Some new np-complete coding problems. *Probl. Peredachi Inf.*, 30(3):23–28, 1994. Translated from Problems Inform. Transmission **30** (1994), no. 3, 209–214.

[10] V. Weger. *Information Set Decoding in the Lee Metric and the Local to Global Principle for Densities*. PhD thesis, University of Zurich, 2020.

[11] E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

[12] Pil Joong Lee and Ernest F. Brickell. An observation on the security of mceliece's public-key cryptosystem. In *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings*, volume 330 of *Lecture Notes in Computer Science*, pages 275–280. Springer, 1988.

[13] Ilya Dumer. Two decoding algorithms for linear codes. *Problems of Information Transmission*, 25, 03 1989.

[14] Jacques Stern. A method for finding codewords of small weight. In Gérard D. Cohen and Jacques Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer, 1988.

[15] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in 2 n/20: How 1 + 1 = 0 improves information set decoding. volume 2012, pages 520–536, 04 2012.

[16] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{\gamma}(2^{0.054n})$. pages 107–124, 01 2011.

[17] Léo Ducas, Andre Esser, Simona Etinski, and Elena Kirshanova. Asymptotics and improvements of sieving for codes. Cryptology ePrint Archive, Paper 2023/1577, 2023. `https://eprint.iacr.org/2023/1577`.

[18] Pierre-Louis Cayrel, Pascal Véron, and Sidi Mohamed El Yousfi Alaoui. A zero-knowledge identification scheme based on the q-ary syndrome decoding problem. In *International Workshop on Selected Areas in Cryptography*, pages 171–186. Springer, 2010.

[19] Marco Baldi, Massimo Battaglioni, Franco Chiaraluce, Anna-Lena Horlemann, Edoardo Persichetti, Paolo Santini, and Violetta Weger. A new path to code-based signatures via identification schemes with restricted errors. 08 2020.

[20] Marco Baldi, Sebastian Bitzer, Alessio Pavoni, Paolo Santini, Antonia Wachter-Zeh, and Violetta Weger. Zero knowledge protocols and signatures from the restricted syndrome decoding problem. Cryptology ePrint Archive, Paper 2023/385, 2023. `https://eprint.iacr.org/2023/385`.

[21] Violetta Weger, Karan Khathuria, Anna-Lena Horlemann, Massimo Battaglioni, Paolo Santini, and Edoardo Persichetti. On the hardness of the lee syndrome decoding problem. *Advances in Mathematics of Communications*, 18, 01 2022.

[22] Thomas Attema, Ronald Cramer, and Lisa Kohl. A compressed σ-protocol theory for lattices. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part II*, pages 549–579. Springer, 2021.

[23] Thomas Attema, Serge Fehr, and Michael Klooß. Fiat-Shamir transformation of multi-round interactive proofs. In *Theory of Cryptography: 20th International Conference, TCC 2022, Chicago, IL, USA, November 7–10, 2022, Proceedings, Part I*, pages 113–142. Springer, 2022.

[24] Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe. From 5-pass MQ-based identification to MQ-based signatures. Cryptology ePrint Archive, Paper 2016/708, 2016. `https://eprint.iacr.org/2016/708`.

[25] Andre Chailloux. On the (in) security of optimized stern-like signature schemes. In *In Proceedings of WCC 2022: The Twelfth International Workshop on Coding and Cryptography*, March 7-11,2022.

[26] Marco Baldi, Alessandro Barenghi, Sebastian Bitzer, Patrick Karl, Felice Manganiello, Alessio Pavoni, Gerardo Pelosi, Santini Paolo, Jonas Schupp, Freeman Slaughter, Antonia Wachter-Zeh, and Violetta Weger. Cross codes and restricted objects signature scheme. In *Submission to the NIST Post-Quantum Cryptography Standardization Process*, 2023.