Exploring the multi-dimensional approach in code-based cryptography

Victor Dyseryn-Fostier

Thèse de Doctorat

Université de Limoges

Université de Limoges
ED 653 – Sciences et Ingénierie (SI)
Faculté des Sciences et Techniques – Institut de Recherche XLIM

Exploring the multi-dimensional approach in code-based cryptography

Exploration de l'approche multi-dimensionnelle en cryptographie fondée sur les codes correcteurs d'erreurs

Thèse pour obtenir le grade de
Docteur de l'Université de Limoges

présentée et soutenue publiquement le 30 janvier 2024 par

Victor DYSERYN

Thèse dirigée par Philippe GABORIT (directeur) et Carlos AGUILAR-MELCHOR (co-directeur)

Jury :

Président du jury
M. Jean-Pierre TILLICH, Directeur de recherche, INRIA Paris

Rapporteurs
M. Alain COUVREUR, Directeur de recherche, INRIA Saclay
M. Ayoub OTMANI, Professeur des universités, LITIS, Université de Rouen

Examinateurs
M. Nicolas ARAGON, Maître de conférences, XLIM, Université de Limoges
M. Pierre LOIDREAU, Ingénieur en chef de l'armement, DGA, IRMAR
Mme. Antonia WACHTER-ZEH, Associate Professor, Technical University of Munich

Direction de thèse
M. Philippe GABORIT, Professeur des universités, XLIM, Université de Limoges
M. Carlos AGUILAR-MELCHOR, Principal Research Scientist, Sandbox AQ

Remerciements

Un grand merci à mon directeur de thèse Philippe pour m'avoir fait confiance, pour tes conseils à la fois scientifiques et philisophiques. Merci également à mon directeur de thèse Carlos pour tes remarques pragmatiques toujours pertinentes.

Merci à Alain et Ayoub d'avoir accepté de rapporter mon manuscrit, et pour leurs conseils précieux ayant permis d'améliorer ce texte.

J'adresse également mes profonds remerciements aux autres membres du jury Jean-Pierre, Nicolas, Pierre et Antonia.

Un merci particulier à Maxime pour les discussions passionnantes et son écoute. Merci à tous les autres doctorants et collègues que j'ai pu croiser lors de mes passages à Limoges.

Un grand merci à toute mon équipe du ministère avec qui j'ai passé trois années formidables et qui me manque déjà beaucoup.

Je remercie chaleureusement tous mes amis, en particulier ceux des dîners scientifiques Yann, Clément, Hippolyte (et l'invitée occasionnelle Emmanuelle), Maël, Vincent. Un grand merci à mon ancienne coloc Elena. Merci à Polo, toujours là quand il faut.

J'ai également une pensée pour ma famille, mes grands-parents, Mathilde, Edouard et Nathalie, Etienne et Mathias, ainsi que pour mon parrain Georges.

Un immense merci à mes parents pour leur soutien sans faille et à toute épreuve.

Enfin, merci à Anne-France, sache que je mesure tout ce que tu as fait pour moi, surtout à la fin de cette thèse, et je t'en suis infiniment reconnaissant.

Contents

Motivation

Post-quantum cryptography. Cryptography is the art of exchanging data securely. A series of mathematical transformations is applied on the cleartext message, which becomes unreadable to an eavesdropper. On the other side of the network, the recipient is able to revert the transformations, thanks to some secret information: the *key*. Initially reserved for military communication, cryptography has become mainstream since the end of the twentieth century and is now integrated in widespread messaging applications such as WhatsApp.

Current cryptography has been at threat since the discovery by Peter Shor of a quantum algorithm [Sho94] that breaks some widely-used cryptosystems. Symmetric cryptography is resistant to Shor's algorithm but cannot be used alone in a big-scale network as it requires that both the sender and recipient share the same secret key. Public-key cryptography is a solution to securely exchange a secret key but currently-used public-key algorithms are precisely those targeted by Shor's algorithm. Recent developments in quantum computing demonstrate an increase in the number of qubits available in a single quantum processor. This calls for a rapid transition to a new *quantum-resistant cryptography* (also called *post-quantum cryptography*, or PQC) as opposed to the current *classical cryptography*. To progress towards this objective, an American government agency, the National Institute of Standards and Technology (NIST) launched in 2017 a post-quantum standardization project [NISb] in order to select cryptosystems that would resist both classical and quantum computers. Two categories exist in this project: (public-key) **encryption schemes** that ensure confidentiality of data, and digital **signature schemes** that ensure authenticity of the sender. Both primitives are essential to build secure communication networks. We are now in the fourth phase of the process; 4 algorithms were selected for standardization and 4 remain under evaluation for another round. Final standards are expected to be ready for 2025.

There are several families of post-quantum algorithms: three schemes based on **Euclidean lattices** (Kyber [SAB+22], Dilithium [LDK+22] and Falcon [PFH+22]) were eventually selected for standardization in 2022. An **isogeny-based** candidate SIKE [JAC+22] is still in the contest for an additional Round 4, although it was broken very recently [CD23]. **Code-based** cryptography is the main focus of this manuscript and counts three active encryption candidates in Round 4 (BIKE [ABB+22], Classic McEliece [ABC+22] and HQC [AAB+22]). **Multivariate** cryptography is another family of post-quantum algorithms.

	Encryption		**Signature**
	Selected	*Round 4*	Selected
Lattices	1		2
Codes		*3*	
Isogeny		*1*	
Multivariate			
Hash-based			1

Table 0.1: Status of the NIST PQC standardization process in July 2022. Candidates displayed in *italics* are still under consideration for an additional Round 4.

NIST opened in 2023 a **new call for additional digital signature** proposals [NISa] to be considered in the PQC standardization process. The new contest aims at mitigating the fact that the two selected general purpose signature schemes Dilithium and Falcon are both based on structured Euclidean lattices and that there are no remaining digital signature candidates under consideration for Round 4. Another signature scheme SPHINCS+ [HBD+22] was also selected for standardisation but suffers from a very slow signing algorithm. Code-based cryptography is a very active family

in this new call as there were 11 code-based[1] signature proposals.

	Signature (*Round 1*)
Lattices	7
Codes	11
Isogeny	1
Multivariate	12
Symmetric-based	4
Other	5

Table 0.2: Candidates to the NIST PQC onramp standardization process for digital signature opened in June 2023.

Code-based cryptography. This fascinating family of cryptographic primitives relies heavily on the theory of error-correcting codes, which existed prior to public-key cryptography and is used to correct errors in a noisy communication channel. Inventors of code-based cryptographic primitives have three design choices to make regarding (i) the use of a hidden decoder, (ii) the cyclic structure, and (iii) the metric. An extended introduction to code-based cryptography can be read in Chapter 1.

The first code-based encryption scheme, due to McEliece [McE78], used a **hidden decoder**: the public key is the masked version of a private description of a code. In order to depend on fewer security assumptions, Alekhnovich then designed an encryption scheme without a hidden decoder [Ale03]. Every code-based encryption scheme belongs to either McEliece or Alekhnovich framework. For signatures, the same distinction as for encryption can be found; constructions with decoding are called hash-and-sign, whereas those without one derive from the generic Fiat-Shamir transformation [FS87].

One of the challenges of post-quantum cryptography is its large key or message sizes when compared to classical cryptography. To overcome this limitation, objects with a **cyclic structure**, which can be stored in a succinct form, can be considered. This idea was first introduced for code-based cryptography in [Gab05]. This approach however implies that the security assumption does not rely on a pure instance of the usual difficult problem in code-based cryptography: the syndrome decoding problem. Although in general, no major speedups have been found on attacking the decoding problem by exploiting the cyclicity, in some specific cases an attack was found [CDW21]. This potential vulnerability led some European information security agencies, such as French ANSSI in their position paper [Age23], to recommend the unstructured FrodoKEM [NAB+20] as a more conservative option than Kyber, which features cyclicity.

The Hamming metric is the usual metric in code-based cryptography. An alternative is the **rank metric** introduced by Delsarte [Del78] and Gabidulin [Gab85]. It also enables a diminution of sizes and all the previous design choices are also available for rank metric. Codes with a cyclic structure are generalized into ideal codes in the rank metric.

Multi-dimensional approach. A fourth and somehow less studied design possibility exists in code-based cryptography. The idea of the multi-dimensional approach is to view the words as multi-dimensional vectors (i.e. matrices) whose errors are *synchronized*, in the sense that they have the same error locations (also called support). The code remains the same, so for the same key size, more information can be embedded in the ciphertext. As a result, parameters can be made smaller and the overall total size is decreased. Of course, the difficult problem in the security proof of the scheme needs to be adapted. More details about the multi-dimensional approach can be found in Chapter 2.

The first publication implementing this technique is an identity-based encryption based on rank-metric codes [GHPT17]. To prove the security of their encryption scheme, they introduce a new difficult problem, the Rank Support Learning (RSL), a variant of the syndrome decoding in

[1]The official classification by NIST adds another *MPC-in-the-Head* category. We rather chose to put the corresponding schemes into the category to which the difficult problem they are based on belongs.

rank metric (RSD) with multiple errors of the same support. They studied the security of RSL and showed that it remains exponential when the number of different errors is not too large. Several other articles were published [DT18, BB21, BBBG22, BBB+23] that refined the security analysis of RSL without calling its overall security into question. A rank metric signature scheme, Durandal [ABG+19], was also designed using the multi-dimensional approach.

The decoding of multiple syndromes that result from synchronized errors can also be referred to as the decoding of interleaved codes. Interleaved LRPC codes and their decoding were studied in [RJB19]. Furthermore, the idea of using interleaved codes for Hamming-based cryptosystems was introduced in [EWZZ18] and for rank-based cryptosystems in [RPWZ19].

Permuted Kernel Problem. The Permuted Kernel Problem (PKP), introduced by Shamir [Sha90] is a difficult problem closely linked to coding theory. Despite many cryptanalytic efforts over the years [Geo92, BCCG93, PC94, JJ01, LP11, KMP19, SBC22] the problem remains hard against classical as well as quantum attackers. The well-studied hardness, compact description and simplicity of involved objects and corresponding computations has made the PKP an attractive candidate for post-quantum secure schemes.

Although there has been no encryption scheme based on PKP so far, this problem can be used to build a proof of knowledge and consequently a digital signature scheme. Building upon the initial proposal by Shamir [Sha90], many refinements were published in the recent years [BG23, BFK+19, Beu20b].

The Permuted Kernel Problem can also be combined with the multi-dimensional approach. A multi-dimensional variant of the PKP has already been studied [LP11, SBC22] but was not used in a cryptographic scheme before our work.

Fully Homomorphic Encryption. At the end of this manuscript, we will make a foray into the world of homomorphic encryption, thanks to the multi-dimensional approach. A homomorphic encryption scheme allows to perform operations on plaintexts which are still in their encrypted form. Because it enables computations in a public cloud while keeping the data private, homomorphic encryption has numerous applications, especially in the medical or banking sector.

In the early years of homomorphic encryption after it was introduced by Rivest, Adleman and Dertouzos in 1978 [RAD78], some schemes were designed [GM82, ElG85, Pai99] but they only supported a single type of operation, either an addition or a multiplication. The scheme by Boneh, Goh and Nissim [BGN05] was the first to support unlimited additions and a single multiplication. A long-standing problem was finally solved when Gentry designed in 2009 a **fully homomorphic encryption** (FHE) scheme [Gen09], able to perform unlimited additions and multiplications on encrypted data.

A fundamental technique in achieving fully homomorphic encryption is called **bootstrapping** [Gen09]. In most systems, after some homomorphic operations, the ciphertext suffers from a large amount of noise that prevent any further operation. The bootstrapping technique consists in homomorphically applying the decryption circuit, generating a new ciphertext under a new key with a reduced amount of noise. This allows to continue with additional homomorphic operations. However, in existing systems, the bootstrapping procedure is very costly, making homomorphic encryption inpractical for generic applications.

Many improvements have been made [BV11, DM15, CGGI20] since the initial proposal by Gentry in order to make fully homomorphic encryption efficient. These efficient schemes were built with structured lattices, making them all highly at risk should an attack be found on structured lattice difficult problems.

The question of **homomorphic encryption based on codes** was first addressed in 2011: the authors of [AAPS11] present a symmetric scheme supporting additions and a limited number of multiplications. Their construction relies on a class of codes called *special evaluation codes* whose codewords have natural multiplicative properties. They instantiate their scheme with Reed-Muller codes. They do not investigate bootstrapping further than showing its impossibility. In the same year, a public-key homomorphic encryption scheme based on Reed-Solomon codes was proposed [BL11] but was broken shortly after its publication [CGG+14]. More recently, homomorphic computations in Reed-Muller codes were investigated [CKN20]. The authors present an operation

on Reed-Muller codewords so that the result represents an encoding of the multiplication of the messages. However, they did not study their techniques in the presence of noise, nor did they propose an encryption scheme. Therefore their work is not related to any notion of cryptographic security. All existing code-based homomorphic constructions thus rely on **families of codes with an efficient decoding algorithm**, which turned out in the past to be a source of numerous attacks [SS92, Ove08].

Contributions

During our doctoral work, we mainly explored the following question: **to which extent can code-based cryptography benefit from the multi-dimensional approach**? We applied this paradigm to three main categories: public-key encryption with a particular focus on rank-based schemes; digital signatures; and homomorphic encryption. For rank-based encryption schemes, we positively answered to the question. For signature and homomorphic schemes, the results are more balanced. A summary table of our contributions is presented at the end of the chapter (Table 0.6)

Public-key encryption

In Section 3.1, we apply the multi-dimensional approach on the NIST candidate ROLLO [ABD+19], a low-rank parity check (LRPC) code-based cryptosystem with a hidden decoder. This gave rise to LRPC-MS (for **LRPC** with **M**ultiple **S**yndromes) and its quasi-cyclic counterpart ILRPC-MS (**I** stands for Ideal). The difference between ROLLO and ILRPC-MS lies only in the ciphertext which in the latter is comprised of multiple noisy codewords whose errors share the same support. The trickiest part was to evaluate the decoding failure rate of LRPC codes with multiple syndromes. We could provide a proof that – with large probability – the entries of the product of two matrices, whose entries are sampled in low dimension linear spaces E and F, span the entire product space EF. This yielded an upper bound on the requested decoding failure rate. The public key size between ROLLO and ILRPC-MS (which both have cyclicity) is decreased from 1.9 kB to 0.5 kB, for the same 128-bit security and roughly the same ciphertext size. As a result, the reduction factor on the bandwith (the sum of public key and ciphertext sizes) is almost 50%. On a security perspective, on top of the indistinguishability problem of an ideal LRPC matrix (ILRPC-Ind), whereas ROLLO relies on the hardness of the Ideal Rank Syndrome Decoding (IRSD) problem, ILRPC-MS relies on the Ideal Rank Support Learning (IRSL) problem instead. LRPC-MS can be compared to other non-cyclic cryptosystems and the gain is even more spectacular; it is almost three times smaller than FrodoKEM [NAB+20] and five times smaller than Loidreau's cryptosystem [Loi17]. Similarly, LRPC-MS relies on the hardness of RSL instead of RSD for its security proof.

Then, in Section 3.2, we adapt the same technique to another rank-based encryption scheme with a hidden decoder: Loidreau's cryptosystem [Loi17]. The public key is a Gabidulin code masked with a low weight homogeneous matrix. The security assumption is weaker than the masking of an LRPC code for ROLLO, but gives potential for more efficient sizes and has no ideal structure by design. We named this new encryption scheme LowMS (for **Lo**idreau **w**ith **M**ultiple **S**yndromes). Similarly to (I)LRPC-MS, we had to prove an accurate upper bound on the decoding failure rate of Gabidulin codes with multiple syndromes. In LowMS, the public key size is decreased to 4.6 kilobytes and the ciphertext size to 1.1 kilobytes, making it one of the best performing unstructured cryptosystems up to this date.

Concurrently to our work, the same multi-dimensional approach was studied by another team on an encrytion scheme in the rank metric without a hidden decoder called RQC [AAB+19]. In their article [BBBG22], they presented Multi-RQC and its unstructured counterpart Multi-UR. We compare performance and security assumption between our schemes and existing works in Tables 0.3 and 0.4.

Finally, in Section 3.3 we present an algorithmic optimization to the key generation in ROLLO. The most time-consuming part is an inversion in the ring of polynomials $\mathbb{F}_{q^m}[X]$ modulo an irreducible polynomial of degree n, especially when implemented in constant time with the Itoh-Tsuiji algorithm [IT88]. When implemented with a traditional polynomial basis as in [AAB+21], the key generation takes about 13 million CPU cycles for the 128-bit security version. By switching to an optimal normal basis, we get a major speedup in the Itoh-Tsuiji algorithm and come down to 3.5 million cycles. This optimization is generic and would also work when the multi-dimensional approach is taken. In particular, (I)LRPC-MS would also benefit from such an improvement.

	128 bits	Security assumptions
LowMS ($\lambda = 3$)	5.76KB	RSL, Gab. codes masking
NH-Multi-UR-AG [BBBG22]	7.12KB	NHRSL
LRPC-MS	7.21KB	RSL, LRPC codes masking
LowMS ($\lambda = 4$)	10.78KB	RSL, Gab. codes masking
Multi-UR-AG [BBBG22]	11,03KB	RSL
FrodoKEM [NAB+20]	19.34KB	LWE
Classic McEliece [ABC+22]	261KB	SD, Goppa codes masking

Table 0.3: Comparison of sizes of unstructured post-quantum KEMs. The sizes represent the sum of public key and ciphertext expressed in bytes. Our schemes are written in **bold** font.

	Hamming metric		Rank metric	
	with cyclicity	no cyclicity	with cyclicity	no cyclicity
With hidden decoder	BIKE	McEliece	ROLLO **ILRPC-MS**	Loidreau **LRPC-MS** **LowMS**
Without hidden decoder	HQC	Alekhnovich	RQC Multi-RQC	Multi-UR

Table 0.4: Examples of encryption schemes for various design possibilities in code-based cryptography. Schemes implementing the multi-dimensional technique are underlined. Our schemes are written in **bold** font.

Digital signatures

During our work, we studied a rank metric digital signature, Durandal [ABG+19], which was implementing the multiple syndrome approach by design. While trying to improve this signature scheme, we found an major security flaw in Durandal. Our attack is presented in Section 4.1 and targets a problem called PSSI, which is very specific to Durandal and unlinked to the multi-dimensional approach. Unfortunately we could not find a way to repair Durandal while still taking advantage of the multi-dimensional paradigm. Our attack recovers the private key using a leakage of information coming from several signatures produced with the same key. Our approach is to combine pairs of signatures and perform Cramer-like formulas in order to build subspaces containing a secret element. We break all existing parameters of Durandal: the two published sets of parameters claiming a security of 128 bits are broken in respectively 2^{66} and 2^{73} elementary bit operations, and the number of signatures required to finalize the attack is 1,792 and 4,096 respectively. We implemented our attack and ran experiments that demonstrated its success with smaller parameters.

Among other rank metric signatures than Durandal, the best performing ones are based on the MPC-in-the-Head paradigm [IKOS07]. Combining MPC with the multi-dimensional approach had already been tried in [BGKM23] but the proof of knowledge soundness gave a reduction to an unconventional variant of RSL whose security was hard to estimate. Instead we focused on another MPC-in-the-Head signature based on a different problem: the Permuted Kernel Problem (PKP). An efficient signature based on PKP was presented in [BG23], which we refined in our work thanks to the multi-dimensional approach. In Section 4.2 we introduce PERK, a new compact digital signature scheme based on a new multi-dimensional version of PKP: the relaxed IPKP problem (r-IPKP). PERK achieves the smallest signature size amongst existing PKP-based schemes, with 6 kB for level 1 security, while obtaining competitive signing and verification timings. We also give an in-depth study of the concrete complexity to solve our variant.

The following table gives an overview of code-based signature schemes including our contributions. As cyclicity has not been used for signatures yet (see Section 1.2.2 for possible reasons), the two remaining choices are the decoding method and the metric.

	Hamming metric	Rank metric	PKP
Hash-and-sign	Wave CFS†	RankSign†	
Proof of knowledge	SDitH	**Durandal**† **RYDE**	[BG23] **PERK**

Table 0.5: Examples of signature schemes for various design possibilities in code-based cryptography. Schemes implementing the multi-dimensional technique are underlined. Schemes to which we contributed are written in **bold** font. The † symbol indicates a scheme for which there currently exists no parameters that resist cryptanalysis.

Homomorphic encryption

The multi-dimensional approach gives a natural additively homomorphic scheme: adding two errors sharing the same support gives another error with the same support again. The security of the homomorphic scheme then relies on the support learning problem, restricting the number of publishable ciphertexts. However, with this technique, a homomorphic multiplication operation is also non trivial to obtain.

Taking a different approach, we propose in Chapter 5 an Aleknovich-inspired [Ale03] construction, whose ciphertext is a pair of vectors. It can be seen as a secret key version with multiple syndromes of the NIST Round 2 candidate RQC [AAB+19], with an important difference: the message is encoded into a vector space orthogonal to the error. The ciphertext space benefits from an $\mathbb{F}_q^n$-module structure which makes addition and plaintext absorption completely straightforward, and we also propose an algorithm for multiplication. Surprisingly, the security of the resulting homomorphic scheme is not reduced to a support learning problem but rather to a traditional syndrome decoding problem for a low rate code.

Our scheme is remarkably the first code-based somewhat homomorphic encryption scheme relying on random ideal codes. It has therefore a stronger security reduction than existing approaches, all based on highly structured codes.

We also propose the first candidate bootstrapping algorithm for a code-based homomorphic scheme that homomorphically decrypts ciphertexts produced from another secret key. Remarkable for its simplicity, our algorithm enjoys no multiplicative depth, as it requires additions and plaintext multiplications only.

However, our scheme suffers from two major limitations that hamper its categorization as fully homomorphic. First, the number of multiplications is limited because each operation increases the length of the ciphertext as well as the dimension of the noise space. Second, and most importantly, there is an upper bound to the number of independent ciphertexts that can be published without a polynomial key recovery attack. In particular, the number of ciphertexts required for our bootstrapping algorithm is larger than the maximal number of publishable independent ciphertexts. To address these problems, we propose a refinement of our homomorphic decryption algorithm by introducing the notion of ciphertext packing. It reduces the number of bootstrapping ciphertexts very close yet still above the maximal limit.

Still, we give concrete parameters for our scheme that shows its efficiency as a somewhat homomorphic encryption scheme and a strong potential to be refined into a high-performing FHE scheme. For a single multiplication, the key size is 3.7 kB and the ciphertext size is only 0.9 kB, with competitive running times estimated to be a few microseconds for addition and 0.5 millisecond for multiplication. Other parameters could be found to support an arbitrary fixed number of multiplications.

Contribution	Sec.	Output	Impacts
New LRPC-based encryption schemes with multiple syndromes	3.1	[AAD+22]	Shortest LRPC-based encryption scheme without ideal structure. Improves ROLLO by a factor 2 in size.
New scheme based on Loidreau's encryption with multiple syndromes	3.2	[ADG+22]†	Shortest rank-based encryption scheme without ideal structure. Improves Loidreau's cryptosystem by a factor 6 in size.
Algorithmic optimization to ROLLO key generation	3.3	[AADG21]	Faster constant-time algorithm for ROLLO key generation.
Cryptanalysis of the PSSI problem	4.1	[ADG23b]	Breaks all 128-bit parameters of Durandal signature scheme in a bit complexity ranging from 66 to 73.
New signature scheme based on a multi-dimensional variant of PKP	4.2	[ABB+23a]†	Shortest PKP-based signature scheme, was submitted to NIST standardization call for digital signatures.
New somewhat homomorphic encryption scheme based on random ideal codes	5	[ADG23a]*	Alternative to lattice-based FHE schemes with a strong security assumption. A first step towards FHE based on codes.

Table 0.6: Summary of our contributions. The † symbole indicates an article that was accepted to a peer-reviewed journal with minor modifications, and is currently in the process of being published. The * symbol indicates a preprint that has not been peer-reviewed yet.

Notation

$[a, b]$	Set of integers i such that $a \leq i \leq b$
$[n]$	Shorthand for $[1, n]$
$\#S$	Cardinality of a finite set S
$x \xleftarrow{\$} S$	x is sampled uniformly at random from S
$x \xleftarrow{\$,\theta} S$	x is sampled pseudo-randomly from the set S, based on the seed θ
$\mathcal{S}_n$	Set of permutations of $[1, n]$
$\mathbb{F}$	Field
$\mathbb{F}^{m\times n}$ or $\mathcal{M}_{n,m}(\mathbb{F})$	Set of matrices with n rows and m columns of elements in $\mathbb{F}$
$\mathcal{M}_n(\mathbb{F})$	Ring of square matrices of size $n \times n$ of elements in $\mathbb{F}$
$\mathbb{F}_q$	Finite field with q elements
$\mathbb{F}_{q^m}$	Finite field with q^m elements (also a $\mathbb{F}_q$-linear space of dimension m)
$k \star \boldsymbol{b}$	Multiplication of a scalar $k \in \mathbb{F}$ with a vector $\boldsymbol{b} \in \mathbb{F}^n$
$\langle X \rangle_{\mathbb{F}}$	Set of $\mathbb{F}$-linear combinations of elements in X, where X is a subset of $\mathbb{L}$, an $\mathbb{F}$-linear subspace
$\langle X \rangle$	Shorthand for the above notation when there is no ambiguity on the field
$\mathbf{Gr}(\mathbb{L}, d)$	Set of $\mathbb{F}$-linear subspaces of $\mathbb{L}$ of dimension d.
λ	Security parameter
$\mathsf{negl}(\lambda)$	Negligible function, i.e. $f : \mathbb{N} \to \mathbb{R}^+$ such that for all $c \in \mathbb{N}$ there exists a $N_0 \in \mathbb{N}$ such that $f(n) < 1/n^c$ for all $n > N_0$
$\mathrm{poly}(\lambda)$	Polynomially bounded function, i.e.there exists $c, \lambda_0 \in \mathbb{N}$ such that $\mathrm{poly}(\lambda) \leq \lambda^c$ for all $\lambda \geq \lambda_0$
$o(\cdot), \mathcal{O}(\cdot), \Theta(\cdot)$	Family of Landau notations
PPT	Probabilistic polynomial-time

Table 0.7: General notation.

Vectors and matrices. Vectors will be represented with lowercase bold letters and matrices with uppercase bold letters (e.g., $\boldsymbol{v} = (v_1, \ldots, v_n)$ and $\boldsymbol{M} = (m_{ij})_{\substack{1\leqslant i\leqslant k\\ 1\leqslant j\leqslant n}}$). Vectors are assumed to be row vectors unless stated otherwise. When n equals m this set, together with classical matrix sum and product, forms a ring that we denote $\mathcal{M}_n(\mathbb{F})$. For a given vector $\boldsymbol{b}$, we will also sometimes note $\boldsymbol{b}^{(i)}$ its i-th coordinate, and for a given matrix $\boldsymbol{B}$, we will note $\boldsymbol{B}^{(i)}$ its i-th *column* vector. For a vector $\boldsymbol{x} = (x_1, \cdots, x_n) \in \mathbb{F}^n$, we denote $\pi[\boldsymbol{x}] = (x_{\pi(1)}, \cdots, x_{\pi(n)})$ the vector whose coordinates have been permuted by π.

Field extensions. Elements and computations in $\mathbb{F}_{q^m}$ are usually associated with polynomial representations and computations over $\mathbb{F}_q[X]/(P)$ for $P \in \mathbb{F}_q[X]$ an irreducible monic polynomial of degree m. We fix such a polynomial P for the rest of this document.

An element $f \in \mathbb{F}_{q^m}$ can be associated to a *column* vector of $\mathbb{F}_q^m$ using the coefficients of the polynomial representation of f. It is obviously an $\mathbb{F}_q$-vector space isomorphism that we denote $\mathbf{vec}()$:

$$\begin{aligned} \mathbf{vec} : \mathbb{F}_{q^m} &\longrightarrow \mathbb{F}_q^m \\ f = \sum_{i=0}^{m-1} f_i X^i &\longmapsto \begin{pmatrix} f_1 \\ \vdots \\ f_m \end{pmatrix}. \end{aligned}$$

Similarly, a vector $\boldsymbol{v} \in \mathbb{F}_{q^m}^n$ can be associated to an $m \times n$ matrix $\mathbf{Mat}(\boldsymbol{v}) \in \mathcal{M}_{m,n}(\mathbb{F}_q)$ whose i-th column is $\mathbf{vec}(\boldsymbol{v}^{(i)})$. (The bold capital 'M' in this transformation's name being a reminder that it outputs a matrix).

Product of vectors. We would like to give a field structure to the set of vectors in $\mathbb{F}_{q^m}^n$. To do so, we associate a vector $\boldsymbol{v}$ of $\mathbb{F}_{q^m}^n$ to a polynomial $\mathrm{poly}(\boldsymbol{v})$ in the ideal ring $\mathbb{F}_{q^m}[X]/\langle Q \rangle$ for $Q \in \mathbb{F}_{q^m}[X]$ a monic irreducible polynomial of degree n whose coefficients are in the subfield $\mathbb{F}_q$[2]. We fix such a polynomial Q for the rest of this document.

$$\begin{aligned} \mathrm{poly} : \quad \mathbb{F}_{q^m}^n &\longrightarrow \mathbb{F}_{q^m}[X]/\langle Q \rangle \\ (v_0, \ldots, v_{n-1}) &\longmapsto \sum_{i=0}^{n-1} v_i X^i \end{aligned}$$

We can now define the multiplication of two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{F}_{q^m}^n$ as

$$\boldsymbol{u} \cdot \boldsymbol{v} = \mathrm{poly}^{-1}(\mathrm{poly}(\boldsymbol{u})\mathrm{poly}(\boldsymbol{v})).$$

Note that the product of polynomials is calculated *modulo* Q.

Probabilistic notation. Let X and Y be two discrete random variables defined over a finite support D. The *statistical distance* between the two distributions is defined as

$$\Delta(X, Y) := \frac{1}{2} \sum_{d \in D} |\mathbb{P}(X = d) - \mathbb{P}(Y = d)|.$$

We say two ensembles of random variables $\{X_\lambda\}_{\lambda \in \mathbb{N}}$, $\{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are *statistically close* if there exists a negligible function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}^+$ such that $\Delta(X_\lambda, Y_\lambda) \leq \mathsf{negl}(\lambda)$ for all $\lambda \in \mathbb{N}$. We say two ensembles of random variables $\{X_x\}_{x \in \{0,1\}^*}$, $\{Y_x\}_{x \in \{0,1\}^*}$ are *statistically close* if there exists a negligible function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}^+$ such that $\Delta(X_x, Y_x) \leq \mathsf{negl}(|x|)$ for all $x \in \{0,1\}^*$.

Basic definitions

Definition 0.0.1 (Circulant matrix)**.** *A square $n \times n$ matrix $\boldsymbol{M}$ is said to be circulant if it is of the form*

$$\boldsymbol{M} = \begin{pmatrix} m_0 & m_1 & \ldots & m_{n-1} \\ m_{n-1} & m_0 & \ddots & m_{n-2} \\ \vdots & \ddots & \ddots & \vdots \\ m_1 & m_2 & \ldots & m_0 \end{pmatrix}$$

where each row is right-shifted from the above row.

[2] This is to ensure $\mathbb{F}_q^n$ is stable under multiplication.

Definition 0.0.2 (Product space). *For E and F, $\mathbb{F}_q$-linear subspaces of $\mathbb{F}_{q^m}$, the product space EF of E times F is defined as the subspace generated by all the products of an element of E with an element of F:*

$$EF = \langle \{ef \mid e \in E, f \in F\} \rangle_{\mathbb{F}_q}$$

Remark. When E and F are of $\mathbb{F}_q$-dimension r and d respectively, the dimension of the product space EF is upper bounded by rd. Indeed, for a basis $(e_1, ..., e_r)$ of E and a basis $(f_1, ..., f_d)$ of F, it is clear that the tensor product of these basis $(e_i f_j)_{1 \leq i \leq r, 1 \leq j \leq d}$ is a generating family of EF.

When r and d are small with respect to m, this family is also linearly independent with great probability, meaning that the dimension of EF is exactly rd in the typical case (see [AGH+19, Section 3] for more detailed results on this probability).

Standard cryptographic primitives

Definition 0.0.3 (Pseudorandom Generator (PRG)). *Let* p *be a polynomial and let* PRG *be a deterministic polynomial-time algorithm such that for any $\lambda \in \mathbb{N}$ and any input $s \in \{0,1\}^\lambda$, the result $\mathsf{PRG}(s)$ is a string of length $\mathsf{p}(\lambda)$. We say that* PRG *is a* pseudorandom generator *if the following conditions hold:*

1. *Expansion: For every $\lambda \in \mathbb{N}$ it holds that $\mathsf{p}(\lambda) > \lambda$.*
2. *Pseudorandomness: For any* PPT *algorithm* D*, there is a negligible function* negl *such that*

$$|\mathbb{P}[\mathsf{D}(\mathsf{PRG}(s)) = 1] - \mathbb{P}[\mathsf{D}(r) = 1]| \leq \mathsf{negl}(\lambda)$$

where the first probability is taken over the uniform choice of $s \in \{0,1\}^\lambda$ and the randomness of D*, and the second probability is taken over the choice of $r \in \{0,1\}^{\mathsf{p}(\lambda)}$ and the randomness of* D*.*

We say PRG is $(\mathsf{t}, \epsilon_{\mathsf{PRG}})$-secure if for every D running in time at most $\mathsf{t}(\lambda)$ the success probability of D is upper bounded by some function $\epsilon_{\mathsf{PRG}}(\lambda)$.

Definition 0.0.4 (Collision-Resistant Hash Functions (CRHF)). *Let ℓ, κ be polynomials and let $\mathcal{H} = \{\mathsf{H}_k : \{0,1\}^* \to \{0,1\}^{\ell(\lambda)};\ k \in \{0,1\}^{\kappa(\lambda)}\}_\lambda$ be a family of functions indexed by $\lambda \in \mathbb{N}$. We say that $\mathcal{H}$ is* collision-resistant *if there exists a negligible function* negl *such that, for any* PPT *algorithm $\mathcal{A}$ it holds that,*

$$\mathbb{P}\left[\begin{array}{l} x \neq x' \bigwedge \\ \mathsf{H}_k(x) = \mathsf{H}_k(x') \end{array} \;\middle|\; \begin{array}{l} k \xleftarrow{\$} \{0,1\}^{\kappa(\lambda)}; \\ (x, x') \longleftarrow \mathcal{A}(k) \end{array} \right] \leq \mathsf{negl}(\lambda).$$

Definition 0.0.5 (Commitment Scheme). *A commitment scheme is a tuple of algorithms* $(\mathsf{Com}, \mathsf{Open})$ *such that* $\mathsf{Com}(r, m)$ *returns a commitment c for the message m and randomness r while* $\mathsf{Open}(c, r, m)$ *returns either* 1 *(*accept*) or* 0 *(*reject*). A commitment scheme is said to be correct if:*

$$\mathbb{P}\left[b = 1 \;\middle|\; c \leftarrow \mathsf{Com}(r, m),\ b \leftarrow \mathsf{Open}(c, r, m)\ \right] = 1.$$

Definition 0.0.6 (Computationally Hiding). *Let (m_0, m_1) be a pair of messages, the advantage of $\mathcal{A}$ against the hiding experiment is defined as:*

$$\mathsf{Adv}^{\mathsf{hiding}}_{\mathcal{A}}(1^\lambda) = \left| \mathbb{P}\left[b = b' \;\middle|\; \begin{array}{l} b \xleftarrow{\$} \{0,1\},\ r \xleftarrow{\$} \{0,1\}^\lambda \\ c \longleftarrow \mathsf{Com}(r, m_b),\ b' \longleftarrow \mathcal{A}(c) \end{array} \right] - \frac{1}{2} \right|.$$

A commitment scheme is computationally hiding if for all PPT *adversaries $\mathcal{A}$ and every pair of messages (m_0, m_1), $\mathsf{Adv}^{\mathsf{hiding}}_{\mathcal{A}}(1^\lambda)$ is negligible in λ.*

We say Com is $(\mathsf{t}, \epsilon_{\mathsf{Com}})$-secure if for every $\mathcal{A}$ running in time at most $\mathsf{t}(\lambda)$ the success probability of $\mathcal{A}$ is upper bounded by some function $\epsilon_{\mathsf{Com}}(\lambda)$.

Definition 0.0.7 (Computationally Binding)**.** *The advantage of an adversary $\mathcal{A}$ against the commitment binding experiment is defined as:*

$$\mathsf{Adv}^{\mathsf{binding}}_{\mathcal{A}}(1^\lambda) = \mathbb{P}\left[\begin{array}{l} m_0 \neq m_1 \\ 1 \longleftarrow \mathsf{Open}(c, r_0, m_0) \\ 1 \longleftarrow \mathsf{Open}(c, r_1, m_1) \end{array} \;\middle|\; (c, r_0, r_1, m_0, m_1) \longleftarrow \mathcal{A}(1^\lambda) \right].$$

A commitment scheme is computationally binding if for all PPT *adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{binding}}_{\mathcal{A}}(1^\lambda)$ is negligible in λ.*

Definition 0.0.8 (Key Encapsulation Mechanism)**.** *A Key Encapsulation Mechanism KEM =* $(\mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$ *is a triple of probabilistic algorithms together with a key space $\mathcal{K}$. The key generation algorithm* KeyGen *generates a pair of public and secret keys* $(\mathsf{pk}, \mathsf{sk})$*. The encapsulation algorithm* Encap *uses the public key* pk *to produce an encapsulation c and a key $K \in \mathcal{K}$. Finally* Decap*, using the secret key* sk *and an encapsulation c, recovers the key $K \in \mathcal{K}$, or fails and returns $\perp$.*

Definition 0.0.9 (Signature Scheme)**.** *A signature scheme consists of three probabilistic polynomial time algorithms* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vf})$ *which work as follows:*

- $\mathsf{KeyGen}\left(1^\lambda\right)$*: The* key generation *algorithm takes a security parameter as input and outputs a pair of keys* $(\mathsf{pk}, \mathsf{sk})$*. The key* sk *is the private (secret)* signing key *and* pk *is the* public key *used for verification.*
- $\mathsf{Sign}_{\mathsf{sk}}(m)$*: The* signing *algorithm takes as input a secret signing key* sk *and a message m from some message space (that may depend on* pk*). It outputs a signature $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m)$.*
- $\mathsf{Vf}_{\mathsf{pk}}(m, \sigma)$*: The deterministic* verification *algorithm takes as input a public key* pk*, a message m, and a signature σ. It outputs a bit $b := \mathsf{Vf}_{\mathsf{pk}}(m, \sigma)$, with $b = 1$ meaning the signature-message pair is valid and $b = 0$ meaning it is invalid.*

Security notions

The notion of indistinguishability under chosen plaintext attack (IND-CPA) for a KEM is debatable because the Encap algorithm in a KEM offers no possibility to the attacker to choose a plaintext. In this document we follow the definitions of [BBF+19, Section 2], where the attacker in an IND-CPA experiment (defined in Figure 1) has to guess whether the given key is a valid encapsulation of the given ciphertext.

$\mathbf{Exp}^{\mathsf{ind-cpa}}(\mathcal{A})$:

- $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{KeyGen}(1^\lambda)$
- $b \xleftarrow{\$} \{0, 1\}$
- $(c^*, k_0^*) \longleftarrow \mathsf{Encap}(\mathsf{pk})$
- $k_1^* \xleftarrow{\$} \mathcal{K}$
- $b' \longleftarrow \mathcal{A}(\mathsf{pk}, c^*, k_b^*)$
- Return $b == b'$

Figure 1: IND-CPA game for KEMs

Definition 0.0.10 (IND-CPA Security)**.** *A key encapsulation scheme KEM is IND-CPA-secure if for every PPT (probabilistic polynomial time) adversary $\mathcal{A}$, its advantage in the IND-CPA experiment*

$$\mathsf{Adv}^{\mathsf{ind-cpa}}(\mathcal{A}) := 2 \times \mathbb{P}[\mathbf{Exp}^{\mathsf{ind-cpa}}(\mathcal{A}) = 1] - 1$$

is negligible.

Definition 0.0.11 (EUF-CMA Security)**.** *A signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vf})$ *is* EUF-CMA *secure if, for all* PPT *adversaries* $\mathcal{A}$ *there is a negligible function* $\mathsf{negl}(\cdot)$ *such that,*

$$\mathbb{P}\left[\begin{array}{l|l} \mathsf{Vf}_{\mathsf{pk}}(m^*, \sigma^*) = 1 \bigwedge & (\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{KeyGen}(1^\lambda), \\ (m^*, \cdot) \notin \mathcal{Q}^{\mathsf{Sign}} & (m^*, \sigma^*) \longleftarrow \mathcal{A}(\mathsf{Sign}_{\mathsf{sk}}(\cdot), \mathsf{pk}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

where the environment keeps track of the queries to and from the signing oracle via $\mathcal{Q}^{\mathsf{Sign}}$.

Chapter 1

Introduction to code-based cryptography

Contents

1.1 Codes and decoding problem

Error-correcting codes (or simply codes) are an old mathematical concept that dates back to the first electronical communication devices. They are a way to encode a message with redundancy so that potential errors can be corrected. The reference textbook in coding theory is [MS77] by MacWilliams and Sloane. A (linear) error-correcting code is simply defined as a linear subspace of $\mathbb{F}^n$.

Definition 1.1.1 ($\mathbb{F}$-linear code). *Let $\mathbb{F}$ be a field. An $\mathbb{F}$-linear code $\mathcal{C}$ of dimension k and length n is a subspace of dimension k of $\mathbb{F}^n$ seen as an $\mathbb{F}$-linear space. The notation $[n,k]_{\mathbb{F}}$ is used to denote its parameters (or simply $[n,k]$ when there is no ambiguity on the field).*

The code $\mathcal{C}$ can be represented by two equivalent ways:

- *by a generator matrix $\boldsymbol{G} \in \mathbb{F}^{k\times n}$. Each row of $\boldsymbol{G}$ is an element of a basis of $\mathcal{C}$,*

$$\mathcal{C} = \{\boldsymbol{x}\boldsymbol{G}, \boldsymbol{x} \in \mathbb{F}^k\}.$$

- *by a parity-check matrix $\boldsymbol{H} \in \mathbb{F}^{(n-k)\times n}$. Each row of $\boldsymbol{H}$ determines a parity-check equation verified by the elements of $\mathcal{C}$:*

$$\mathcal{C} = \{\boldsymbol{x} \in \mathbb{F}^n : \boldsymbol{H}\boldsymbol{x}^T = \boldsymbol{0}\}.$$

We say that $\boldsymbol{G}$ (respectively $\boldsymbol{H}$) is under systematic form if and only if it is of the form $(\boldsymbol{I}_k|\boldsymbol{A})$ (respectively $(\boldsymbol{I}_{n-k}|\boldsymbol{B})$).

The set $\mathbb{F}^k$ is said to be the set of **messages** and $\mathbb{F}^n$ the set of **words**. The set of words $\mathbb{F}^n$ is equipped with a distance δ. Since $\mathbb{F}^n$ has a natural vector space structure, the distance $\delta(\cdot,\cdot)$ can be derived from a definition of weight $\|\cdot\|$. We define below the usual **Hamming metric** for which $\mathbb{F} = \mathbb{F}_q$ is a finite field.

Definition 1.1.2 (Hamming metric over $\mathbb{F}_q^n$)**.** *Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$. The Hamming weight $\|\boldsymbol{x}\|$ of $\boldsymbol{x}$ is defined as the number of its non-zero coordinates*

$$\|\boldsymbol{x}\| = \#\{i \in [1, n] \,|\, x_i \neq 0\}.$$

The associated distance $\delta(\boldsymbol{x}, \boldsymbol{y})$ between elements $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{F}_q^n$ is defined by $\delta(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$.

Definition 1.1.3 (Support in Hamming metric)**.** *The Hamming support of a word $\boldsymbol{x} \in \mathbb{F}_q^n$ is the set of indexes of its non-zero coordinates:*

$$\mathbf{Supp}(\boldsymbol{x}) = \{i \in [1, n] \,|\, x_i \neq 0\}.$$

The weight of a word can be calculated from its support using

$$\|\boldsymbol{x}\| = \#\mathbf{Supp}(\boldsymbol{x}).$$

Now that the space $\mathbb{F}^n$ is equipped with a metric, we can define the minimal distance for a code.

Definition 1.1.4 (Minimal distance)**.** *Let $\mathcal{C}$ be an $[n, k]_{\mathbb{F}}$ code. The minimal distance d of $\mathcal{C}$ is the minimum distance between two distinct words in $\mathcal{C}$*

$$d = \min\{\delta(\boldsymbol{x}, \boldsymbol{y}) \,|\, (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{C}^2, \boldsymbol{x} \neq \boldsymbol{y}\}.$$

The code $\mathcal{C}$ is then said to be an $[n, k, d]_{\mathbb{F}}$ code.

An error-correcting code in a noisy communication is used as follows. The sender encodes a message $\boldsymbol{m} \in \mathbb{F}^k$ into the corresponding codeword $\boldsymbol{mG} \in \mathbb{F}^n$. Because of imperfections in the transmission channel, the recipient gets a word with errors $\boldsymbol{y} = \boldsymbol{mG} + \boldsymbol{e}$. If the weight of the error $\|\boldsymbol{e}\|$ is not too large, the recipient may nonetheless recover the codeword $\boldsymbol{mG}$ with a decoding algorithm, and consequently the correct message $\boldsymbol{m}$. Good codes for communication have low redundancy (i.e the rate $R = k/n$ is close to 1) and efficient decoding algorithms.

For cryptography, the perspective is different. Errors are voluntarily added by the sender, instead of being created by an imperfect channel. The code itself is also different: it is (from the point of view of the attacker) a random code. As no efficient decoding algorithm exists for a random code, in order to decrypt the message, an attacker has to solve the decoding problem defined below.

Problem 1.1.1 (Decoding problem)**.** *Given $\boldsymbol{G} \in \mathbb{F}^{k\times n}$, $\boldsymbol{y} \in \mathbb{F}^n$ and a target weight w, the decoding problem* $\mathsf{Decode}_{n,k,w}$ *asks to find $\boldsymbol{m} \in \mathbb{F}^k$ such that $\|\boldsymbol{y} - \boldsymbol{mG}\| \leq w$.*

Noting that for a parity-check matrix $\boldsymbol{H}$ of the code generated by $\boldsymbol{G}$, we have $\boldsymbol{Hy}^T = \boldsymbol{H}(\boldsymbol{y} - \boldsymbol{mG})^T$, the decoding problem is often solved in its dual equivalent version, the *syndrome decoding problem.*

Problem 1.1.2 (Syndrome decoding problem SD)**.** *Given $\boldsymbol{H} \in \mathbb{F}^{(n-k)\times n}$, $\boldsymbol{y} \in \mathbb{F}^{n-k}$ and a target weight w, the syndrome decoding problem $SD_{n,k,w}$ asks to find $\boldsymbol{e} \in \mathbb{F}^n$ such that $\|\boldsymbol{e}\| \leq w$ and $\boldsymbol{eH}^T = \boldsymbol{y}$.*

These problems have been proven NP-complete [BMvT78] in the worst case, and up to this date the best solving algorithms for random instances are exponential in n (i.e. the problem is also hard in the average case), provided that the weight w is properly chosen. Indeed, if there was no constraint on the weight of the error, the syndrome decoding problem would be easily solved with linear algebra. More precisely, the SD problem becomes polynomial when w is greater than the *Singleton* bound.

Definition 1.1.5 (Singleton bound). *For a $[n,k]_{\mathbb{F}_q}$ code in Hamming metric, the Singleton bound is defined as*

$$d_{Sg} = n - k + 1.$$

The weight should neither be too small otherwise an attacker could bruteforce the problem. In practice, the best regime is when the weight is equal to the Gilbert-Varshamov bound, for which the SD problem has exactly one solution on average.

Definition 1.1.6 (Gilbert-Varshamov bound). *For a $[n,k]_{\mathbb{F}_q}$ code in Hamming metric, the Gilbert-Varshamov bound d_{GV} is defined as the smallest positive integer t such that*

$$q^{n-k} \leq \sum_{i=0}^{t} \binom{n}{i} (q-1)^i.$$

Asymptotically, the ratio between the Gilbert-Varshamov bound and the length of the code tends to a constant that depends on the field size q and the rate $R = k/n$,

$$\frac{d_{GV}}{n} \sim \gamma_{q,R}.$$

For example, for a binary field and half-rate codes, the Gilbert-Varshamov bound is approximately a ninth of the length ($\gamma_{2,1/2} \approx \frac{1}{9}$).

The question is now: if the sent data appears like a random vector that is not decodable in reasonable time, how can the legitimate recipient decrypt the message? We will explore some possibilities in the next section.

1.2 Design choices

1.2.1 With or without a hidden decoder?

Encryption schemes

The first code-based scheme [McE78] is an encryption scheme, named after its author McEliece, and works with a **hidden decoder**. The key generation algorithm samples a secret code $\mathcal{C}$ from a family of non truly random but efficiently decodable codes. Let us denote $\boldsymbol{G}$ a generating matrix of $\mathcal{C}$. The private code is then masked with a permutation $\pi \in \mathcal{S}_n$ acting on the n coordinates of vectors in $\mathcal{C}$, giving a code $\mathcal{C}' = \pi(\mathcal{C})$. A generating matrix of $\mathcal{C}'$ is given by $\boldsymbol{SGP}$, where $\boldsymbol{S}$ is a non-singular matrix and $\boldsymbol{P}$ a permutation matrix representing π. The private key is the triple $(\boldsymbol{S}, \boldsymbol{G}, \boldsymbol{P})$ and the public key is $\boldsymbol{G}'$. Because of the permutation, the public key $\boldsymbol{G}'$ looks like a random generating matrix and no efficient decoding algorithm can be used on it.

When the legitimate recipient receives the noisy codeword $\boldsymbol{y} = \boldsymbol{mG}' + \boldsymbol{e}$, she can apply the inverse of the secret permutation (the *trapdoor*). Because a permutation is an isometry for the Hamming metric, the resulting word has the same distance to the private code $\mathcal{C}$, allowing to take advantage of the efficient decoding algorithm for $\mathcal{C}$ and recover the message $\boldsymbol{m}$. On the contrary, an eavesdropper can only apply a decoding algorithm for random codes and thus faces an exponential difficulty.

The above steps are represented in Figure 1.1. All these constructions with a hidden decoder are based on the **assumption** that the masked public code $\mathcal{C}'$ is indistinguishable from a random code, and designers must closely evaluate the security of this assumption.

In [McE78], the family of efficient codes is the set of Goppa codes. They have not been broken so far but have a large public key. Numerous attempts have been made to instantiate the scheme with other families (e.g. generalized Reed-Solomon codes in [Nie86], or Reed-Muller codes in [Sid94]) in order to decrease the key size, but most of them have been broken because the private code $\mathcal{C}$ could be recovered from $\mathcal{C}'$.

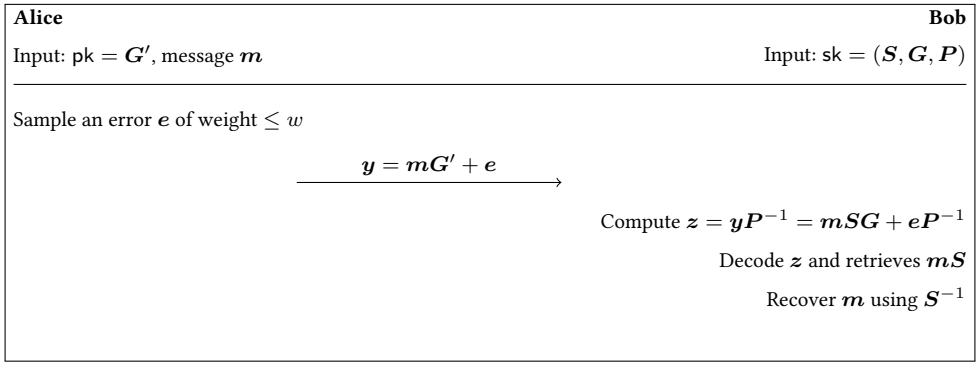

Figure 1.1: McEliece encryption framework.

Another approach **without a hidden decoder** was introduced by Alekhnovich [Ale03]. The public key is a purely random instance of the SD problem $(\boldsymbol{H}, \boldsymbol{y})$ and the secret key is a solution $\boldsymbol{e}$ to that instance. On decryption, the ciphertext is combined with the secret key to recover the message. Contrary to systems with a hidden decoder, in Alekhnovich-like cryptosystems both key recovery and message recovery attacks require to solve the SD problem hence there is no additional assumption on the masking of a private code. It has therefore strong guarantees of security, but in return has a high **decoding failure** that necessitates repeating the encryption operation several times. As a result, it is not totally practical except with an additional ingredient that will be covered in the next subsection: quasi-cyclic structure.

Signature schemes

Before that, a few words about the design of digital signatures. Just like encryption, there are two possible approaches: the first one with a hidden decoder is called *hash-and-sign* and the second one without decoding is called *Fiat-Shamir* style.

In a **hash-and-sign** signature scheme, the message is hashed and interpreted as the noisy syndrome of a private code with an efficient decoding. When the noisy vector can be decoded, the signature is computed as the closest codeword. The verifier can then hash the message and verify that the signature is close enough to the hash; it proves that the signer owned an efficient way to decode the hash. In order for that approach to work, the probability that a noisy syndrome can be decoded must be non negligible, which can be achieved with only a limited set of code families. Many attempts of code-based hash-and-sign signatures have been broken, for example CFS [CFS01] and RankSign [GRSZ14b]. Wave [DST19] is a code-based hash-and-sign signature scheme that seems to resist to cryptanalysis. Its signature size is small, although it has a large public key (greater than 3MB for 128-bit security).

Another possibility is to turn a **zero-knowledge proof** of knowledge into a signature scheme thanks to the Fiat-Shamir transformation [FS87]. The challenge is chosen to be equal to the hash of the message, making the proof of knowledge non interactive and suitable for digital signing. This is the preferred approach to build code-based signatures with a smaller public key size. 8 out of the 11 code-based candidate signature schemes to the recent NIST onramp call rely on the Fiat-Shamir transformation. In the recent years, there has been a considerable activity in the field of proofs of knowledge for coding theory difficult statements. The smaller signatures build thanks to the *MPC-in-the-Head* paradigm that was introduced in 2007 [IKOS07]. This is a generic transform that turns a secure multiparty computation (MPC) protocol into a proof of knowledge. In Hamming metric, the most efficient signature scheme based on MPC-in-the-Head for syndrome decoding is called SDitH [FJR22].

1.2.2 Adding cyclic structure for a smaller public key

Code-based cryptography usually suffers from having large key sizes. A way to mitigate this issue was initially proposed by Gaborit [Gab05]; the principle is to build codes with variations of a

circulant matrix, which can be fully deduced from one of its rows.

Definition 1.2.1 (Quasi-cyclic codes). *Let $n \geq 1$ and $s \geq \ell \geq 1$. A quasi-cyclic linear code of block length n, index s and rate ℓ/s is an $[ns, n\ell]_{\mathbb{F}_q}$ code admitting a generating matrix of the form:*

$$\boldsymbol{G} = \begin{pmatrix} \boldsymbol{G}_{1,1} & \cdots & \cdots & \boldsymbol{G}_{1,s} \\ \vdots & & & \vdots \\ \boldsymbol{G}_{\ell,1} & \cdots & \cdots & \boldsymbol{G}_{\ell,s} \end{pmatrix}$$

where blocks $\boldsymbol{G}_{i,j}$ are circulant matrices (Definition 0.0.1) of size $n \times n$.

In practice, we additionally require the existence of a systematic quasi-cyclic parity-check matrix. To lighten the notation, we only define systematic quasi-cyclic codes for rates of the form $1/s$.

Definition 1.2.2 (Systematic Quasi-cyclic codes). *Let $n \geq 1$ and $s \geq 1$. A systematic quasi-cyclic linear code of block length n, index s and rate $1/s$ is an $[ns, n]_{\mathbb{F}_q}$ code admitting a partiy-check matrix of the form:*

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{I}_n & 0 & \cdots & 0 & \boldsymbol{H}_1 \\ 0 & \boldsymbol{I}_n & & & \boldsymbol{H}_2 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & \boldsymbol{I}_n & \boldsymbol{H}_{s-1} \end{pmatrix}$$

where blocks $\boldsymbol{H}_i$ are circulant matrices of size $n \times n$. The set of matrices of the form above is denoted $QC(\mathbb{F}_q^{(s-1)n \times sn})$.

When codes used in a cryptosystem are quasi-cyclic, the difficult problem needs to be adapted and is defined below.

Problem 1.2.1 (Quasi-cyclic syndrome decoding problem QCSD). *Given a block length $n \geq 1$, an index $s \geq 1$, $\boldsymbol{H} \in QC(\mathbb{F}^{(s-1)n \times sn})$, $\boldsymbol{y} \in \mathbb{F}^{(s-1)n}$ and a target weight w, the quasi-cyclic syndrome decoding problem $QCSD_{n,s,w}$ asks to find $\boldsymbol{e} \in \mathbb{F}^{sn}$ such that $\|\boldsymbol{e}\| \leq w$ and $\boldsymbol{e}\boldsymbol{H}^T = \boldsymbol{y}$.*

Although there is no known reduction from QCSD to SD, the former problem is considered as hard as the latter – up to a polynomial factor – in the context of random codes. In practice, QCSD is easier than SD by a factor of at most $\sqrt{n}$, using the Decoding Out Of Many (DOOM) approach [Sen11]. Indeed, quasi-cyclic shifts on the syndrome produce n equivalent instances of the decoding problem and the DOOM strategy gives a square root speedup in the number of instances.

The positive impact of cyclicity on code-based signature schemes is less obvious. On one hand, regarding hash-and-sign signatures, it is still an open problem to take advantage from a quasi-cyclic structure. Designing a hash-and-sign scheme requires to find a delicate balance, which seems uncompatible with the slight perturbation induced by adding structure. For example, a quasi-dyadic structure was suggested in [BCMN11] for the CFS signature, but was later broken in [FOP+16]. Designing a quasi-cyclic variant of Wave [DST19] is also a challenge because the structure must be adapted to the very specific masking of the scheme; to date, no solution was found. On the other hand, for proofs of knowledge, public keys are already quite small, and the reduction induced by the quasi-cyclic structure only gives a limited advantage.

1.2.3 Rank metric

Rank metric is an alternative to the Hamming metric that enables building cryptosystems with smaller sizes. It was invented by Delsarte [Del78] for matrix codes whose words are matrices in $\mathbb{F}_q^{m \times n}$. Later, Gabidulin [Gab85] extended the metric to vector codes with $\mathbb{F}_{q^m}$-linearity. In this document we will adopt the viewpoint of Gabidulin: codes embedded in the rank metric are $\mathbb{F}_{q^m}$-linear subspaces of $\mathbb{F}_{q^m}^n$ and are denoted $[n, k]_{\mathbb{F}_{q^m}}$. The rank metric on the set $\mathbb{F}_{q^m}^n$ is defined as follows.

Definition 1.2.3 (Rank metric over $\mathbb{F}_{q^m}^n$). *Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}_{q^m}^n$. As explained in the notation section in preamble, $\boldsymbol{x}$ can be associated to an $m \times n$ matrix $\mathbf{Mat}(\boldsymbol{x})$. The rank weight $\|\boldsymbol{x}\|$ of $\boldsymbol{x}$ is then defined as the rank of $\mathbf{Mat}(\boldsymbol{x})$. This definition does not depend on the choice of the basis. The associated distance $\delta(\boldsymbol{x}, \boldsymbol{y})$ between elements $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{F}_{q^m}^n$ is defined by $\delta(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$.*

Remark. The Hamming metric is also well defined on $\mathbb{F}_{q^m}^n$. When there is an ambiguity, the Hamming weight will be denoted $\|\cdot\|_h$ and the rank weight $\|\cdot\|_r$. One can easily establish that the rank metric is dominated by the Hamming metric: for all $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$, $\|\boldsymbol{x}\|_r \leq \|\boldsymbol{x}\|_h$.

Similar to the Hamming metric, we can define a notion of support in rank metric. In a rank metric context, the rank weight is now calculated by taking the dimension of the support (instead of cardinality for Hamming metric).

Definition 1.2.4 (Support in rank metric). *The rank support of a word $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ is the $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$ generated by the entries of $\boldsymbol{x}$:*

$$\mathbf{Supp}(\boldsymbol{x}) = \langle x_1, \ldots, x_n \rangle_{\mathbb{F}_q}.$$

The rank weight of a word can be calculated from its support using

$$\|\boldsymbol{x}\| = \dim(\mathbf{Supp}(\boldsymbol{x})).$$

The same techniques as for the Hamming metric can be used to build cryptographic primitives with rank metric codes. The hardness assumption then relates to an adaptation of SD problem to the rank metric, the RSD problem.

Problem 1.2.2 (Rank syndrome decoding problem RSD). *Given $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$, $\boldsymbol{y} \in \mathbb{F}_{q^m}^{n-k}$ and a target rank weight w, the rank syndrome decoding problem $\text{RSD}_{m,n,k,w}$ asks to find $\boldsymbol{e} \in \mathbb{F}_{q^m}^n$ such that $\|\boldsymbol{e}\|_r \leq w$ and $\boldsymbol{e}\boldsymbol{H}^T = \boldsymbol{y}$.*

Similar to the Hamming metric, there exists a rank Singleton bound above which the RSD problem becomes polynomial, and a rank Gilbert-Varshamov bound which usually gives the highest attack complexity of RSD.

Definition 1.2.5 (Rank Singleton bound [Gab85]). *For a $[n, k, d]_{\mathbb{F}_{q^m}}$ code, the rank Singleton bound is defined as*

$$d_{RSg} = \left\lfloor \frac{m(n-k)}{\max(m,n)} \right\rfloor + 1.$$

Remark. When $m \geq n$, the rank Singleton bound is equal to its Hamming counterpart.

Definition 1.2.6 (Rank Gilbert-Varshamov bound [ABG+19, Section 2.4]). *The rank Gilbert-Varshamov bound d_{RGV} of an $[n, k]_{\mathbb{F}_q}$ code is the smallest positive integer t such that*

$$q^{m(n-k)} \leq \sum_{i=0}^{t} \prod_{j=0}^{i-1} \frac{(q^m - q^j)(q^n - q^j)}{q^i - q^j}.$$

The rank Gibert-Varshamov bound admits a simple asymtotic equivalent when $m = n$:

$$\frac{d_{RGV}}{n} \sim 1 - \sqrt{R}.$$

Contrary to the (Hamming) SD problem, the RSD problem is not known to be NP-complete. There exists a polynomial probabilistic reduction from the SD problem that was presented in [GZ16], which however works for a restricted range of parameters only. Despite this absence of a proper reduction, practical attacks against RSD are exponential. The reference attacks against RSD are the combinatorial attack in [AGHT18] and the algebraic attack in [BBB+23]. For $m = n$, a constant rate R and weight w on the rank Gilbert-Varshamov bound, the exponential factor in those attacks are in the order of n^2, which gives the following asymptotic comparison with the Hamming metric.

	Hamming metric	Rank metric
Cost of attack ($\log_q$ scale)	$\Theta(n)$	$\Theta(n^2)$
Size of public key	$\Theta(n^2)$	$\Theta(n^3)$

Table 1.1: Comparison between Hamming and rank metrics. For a constant rate $R = k/n$ and a weight on the Gilbert-Varshamov bound, the first line approximates the exponential factor in the cost of attacking the generic decoding problem (SD or RSD depending on the metric), and the second row approximates the size of the public key.

Because the cost of attacks must be chosen above the security level λ, in Hamming metric $\Theta(n) = \Theta(\lambda)$ and therefore the size of the public key will be $\Theta(\lambda^2)$, whereas in rank metric $\Theta(n^2) = \Theta(\lambda)$ and the size of the public key will be $\Theta(\lambda^{1.5})$, which gives a clear advantage to the latter metric for practical cryptography.

This is not magic: $\mathbb{F}_{q^m}$-linearity is a very strong property for matricial codes embedded with the rank metric, and RSD can be seen as a quasi-cyclic variant of the well-known NP-complete problem MinRank.

Definition 1.2.7 (MinRank). *Given $K+1$ matrices $(\boldsymbol{M}_0, \boldsymbol{M}_1, \ldots, \boldsymbol{M}_K) \in (\mathbb{F}_q^{m\times n})^{K+1}$ and a target rank w, the MinRank$_{m,n,K,w}$ problem asks to find a non-zero linear combination of those matrices of rank $\leq w$, i.e. a vector $\boldsymbol{x} \in \mathbb{F}_q^K$ such that*

$$\operatorname{rank}\left(\boldsymbol{M}_0 + \sum_{i=1}^{K} x_i \boldsymbol{M}_i\right) \leq w.$$

Existing attacks on RSD are not mere transpositions of attacks against MinRank; they take advantage of the $\mathbb{F}_{q^m}$-linearity but the speedup is less than the gain obtained by the decrease in the public key size.

Another fundamental difference between Hamming and rank metrics is the behaviour of the weight when concatenating two identical words: in the first case, the weight is doubled, whereas in the second case, it remains the same.

$$\|(\boldsymbol{x} \mid \boldsymbol{x})\|_h = 2\|\boldsymbol{x}\|_h \qquad\qquad \|(\boldsymbol{x} \mid \boldsymbol{x})\|_r = \|\boldsymbol{x}\|_r$$

Drawing a parallel with physics, the Hamming metric can be qualified as *extensive* and the rank metric *intensive*. This will have implications in the next chapter regarding the multi-dimensional approach.

For scheme with hidden decoders, we need efficiently decodable rank metric codes. We present below two examples of such codes as well as their decoding algorithm. Low-rank parity-check (LRPC) codes will be used in our scheme LRPC-MS in Section 3.1 and Gabidulin codes will be used in our scheme LowMS in Section 3.2.

1.2.3.1 LRPC codes

LRPC codes admit a parity-check matrix whose entries generate a small-dimensional subspace of $\mathbb{F}_{q^m}$. They were introduced in [GMRZ13].

Definition 1.2.8 (Homogeneous matrix of weight d). *Let $d \leq m$ be an integer. A matrix $\boldsymbol{H} \in \mathbb{F}_{q^m}^{k\times n}$ is an homogeneous matrix of weight d when it is full-rank and its entries generate an $\mathbb{F}_q$-subspace $F = \langle \{h_{ij} \mid 1 \leq i \leq k, 1 \leq j \leq n\} \rangle_{\mathbb{F}_q}$ of dimension d.*

Definition 1.2.9 (LRPC code). *An $[n,k]_{q^m}$ code $\mathcal{C}$ is an LRPC code of dual weight d when it admits a homogeneous parity-check matrix $\boldsymbol{H}$ of weight d.*

LRPC codes have a polynomial time probabilistic decoding algorithm that allows to decode errors up to a weight of $\frac{n-k}{d}$ with a fairly good probability. We present below the details of this decoding algorithm described in Algorithm 1.1. When $\boldsymbol{s} = \boldsymbol{e}\boldsymbol{H}^T$ where $\boldsymbol{H}$ is an homogeneous matrix of support F and weight d, and the error $\boldsymbol{e}$ has a support E of dimension r, the entries of $\boldsymbol{s}$ are in the product space EF. The general idea of the algorithm is to use the fact that the subspace $S = \langle s_1, \dots, s_{n-k} \rangle$ generated by the entries of the syndrome enables one to recover the whole product space EF. The knowledge of both EF and F enables to recover E. Then, knowing the support E of the error vector $\boldsymbol{e}$, it is easy to recover the value of each of its entires by solving a linear system.

Algorithm 1.1 Rank Support Recovery (RSR) algorithm

Input: $F = \langle f_1, ..., f_d \rangle$ an $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$, $\boldsymbol{s} = (s_1, \dots, s_{n-k}) \in \mathbb{F}_{q^m}^{(n-k)}$ a syndrome of an error $\boldsymbol{e}$ of weight r and of support E
Output: A candidate for the vector space E

▷ **Part 1:** Compute the vector space EF
1: Compute $S = \langle s_1, \dots, s_{n-k} \rangle$
▷ **Part 2:** Recover the vector space E
2: $E \leftarrow \bigcap_{i=1}^{d} f_i^{-1} S$
3: **return** E

Notation. For all i we denote S_i the space $f_i^{-1} S$.

Probability of failure. There are two cases for which the decoding algorithm might fail:

- $S \subsetneq EF$, the syndrome entries do not generate the entire space EF, or
- $E \subsetneq S_1 \cap \cdots \cap S_d$, the chain of intersections generates a space of larger dimension than E.

From [AGH+19] we have that the probability of the first failure case $S \subsetneq EF$ is less than $q^{rd-(n-k)-1}$. In [ABD+19], under the assumption that the S_i's behave as random subspaces containing E (which is validated by simulations), it is proven that the probability of the second failure case $E \subsetneq S_1 \cap \cdots \cap S_d$ is less than $q^{-(d-1)(m-rd-r)}$. This leads to the following proposition:

Proposition 1.2.1 ([ABD+19]). *The Decoding Failure Rate of algorithm 1.1 is bounded from above by:*

$$q^{-(d-1)(m-rd-r)} + q^{rd-(n-k)-1}.$$

A very recent result [BO23] intended to rigorously prove the above proposition without any assumption on the behaviour of the subspaces S_i but the best upper bound they could obtain is much less tight ($\approx q^{-(m-2rd+r)} + q^{rd-(n-k)-1}$) and non consistent with simulations, suggesting possible improvements.

Computational cost of decoding. According to [AGH+19], the computational cost of the decoding algorithm is in $\mathcal{O}(4r^2d^2m + n^2r)$ operations in the base field $\mathbb{F}_q$. There is an improved version of this decoding algorithm which was also presented in [AGH+19]. We do not need these improvements in the present document.

LRPC codes and cryptography. Similar to McEliece encryption, when used for cryptography, the structure of the private parity-check matrix $\boldsymbol{H}$ of an LRPC code is masked and the public key is given as $\boldsymbol{H}' = \boldsymbol{S}\boldsymbol{H}$, where $\boldsymbol{S}$ is a non-singular matrix. In practice, $\boldsymbol{S}$ is chosen such that the public key $\boldsymbol{H}'$ is in a systematic form. The security of an LRPC-based cryptosystem is thus reduced to the following indistinguishability problem:

Problem 1.2.3 (LRPC codes decisional problem - LRPC-Ind). *Given a matrix $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times k}$, distinguish whether the code $\mathcal{C}$ with the parity-check matrix $(\boldsymbol{I}_{n-k}|\boldsymbol{H})$ is a random code or an LRPC code of weight d.*

The best attacks against the above problem are described in Section 3.1.5.3 are usually not decisive for the choice of parameters.

1.2.3.2 Gabidulin codes

Gabidulin codes [Gab85] are a well-known class of rank-metric codes and can be seen as the rank-metric analogs of Reed–Solomon codes.

Definition 1.2.10 (Gabidulin Code). *A Gabidulin code $\mathcal{G}[n,k]$ over $\mathbb{F}_{q^m}$ of length $n \leq m$ and dimension k is defined by its $k \times n$ generator matrix*

$$\boldsymbol{G} = \begin{pmatrix} g_1 & g_2 & \dots & g_n \\ g_1^{[1]} & g_2^{[1]} & \dots & g_n^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ g_1^{[k-1]} & g_2^{[k-1]} & \dots & g_n^{[k-1]} \end{pmatrix},$$

where $\boldsymbol{g} = (g_1, g_2, \dots, g_n) \in \mathbb{F}_{q^m}^n$, $\|\boldsymbol{g}\| = n$ and $[i] = q^i$. The vector $\boldsymbol{g}$ is called the generator of the code $\mathcal{G}$.

The dual of a Gabidulin code is also a Gabidulin code.

Proposition 1.2.2 ([Loi07]). *A Gabidulin code $\mathcal{G}[n,k]$ generated by $\boldsymbol{g}$ admits as a parity-check matrix*

$$\boldsymbol{H} = \begin{pmatrix} h_1 & h_2 & \dots & h_n \\ h_1^{[1]} & h_2^{[1]} & \dots & h_n^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ h_1^{[n-k-1]} & h_2^{[n-k-1]} & \dots & h_n^{[n-k-1]} \end{pmatrix},$$

where $(h_1, ..., h_n) = (\alpha_1^{[n-k+1]}, ..., \alpha_n^{[n-k+1]})$ with the α_i verifying

$$\sum_{i=1}^{n} \alpha_i g_i^{[j]} = 0$$

for $j \in \{0, 1, ..., n-2\}$. We note $\mathcal{G}_{(n,k)}^T$ the set of all parity-check matrices of $[n,k]$ Gabidulin codes.

In [Gab85], it is shown that Gabidulin codes are Maximum Rank Distance (MRD) codes, i.e., their minimum distance satisfies $d = n - k + 1$, and can be decoded uniquely in polynomial time up to a rank weight $w \leq \lfloor \frac{d-1}{2} \rfloor$. No efficient algorithms are known to decode in a Gabidulin code when $w > \lfloor \frac{d-1}{2} \rfloor$.

The first rank metric cryptosystem, GPT [GPT91], was based on a masking of a Gabidulin code under the McEliece framework. It was broken by structural attacks [Ove05] that recover the private Gabidulin code. Many reparations were suggested, and the only one that has not been broken so far is the cryptosystem of Loidreau [Loi17] (there exists an attack [CC20] but it does not break all the parameters of the cryptosystem). The masking consists of an homogeneous matrix of small weight. The difficult problem resulting from this masking is presented in Section 3.2.5.1.

1.2.3.3 Ideal codes: cyclicity in the rank metric

In rank metric, cyclicity is obtained with ideal matrices, a generalization of circulant matrices modulo a polynomial.

Definition 1.2.11 (Ideal Matrices). *Let $Q \in \mathbb{F}_{q^m}[X]$ be a polynomial of degree n and $\boldsymbol{v} \in \mathbb{F}_{q^m}^n$. The ideal matrix generated by $\boldsymbol{v}$ modulo Q is the matrix denoted $\mathcal{IM}_Q(\boldsymbol{v}) \in \mathcal{M}_n(\mathbb{F}_{q^m})$ of the form:*

$$\mathcal{IM}_Q(\boldsymbol{v}) = \begin{pmatrix} \boldsymbol{v} \\ X\text{poly}(\boldsymbol{v}) \\ \vdots \\ X^{n-1}\text{poly}(\boldsymbol{v}) \end{pmatrix}.$$

The products of polynomials are computed *modulo* Q and the transformation poly^{-1} has been omitted for simplicity.

In Hamming metric, the quotient polynomial must be $Q = X^n - 1$ in order to preserve the support, with n such that the polynomial $\sum_{k<n} X^k$ is irreducible in $\mathbb{F}_q$, to avoid attacks exploiting the additional structure introduced by two large divisors of Q (see [LJS+16] for an example of attack in a particular case). In rank metric, any polynomial in $\mathbb{F}_q[X]$ preserves the support, allowing for more granularity than the Hamming metric. The polynomial must still be chosen irreducible in order to avoid the same kind of attacks.

Definition 1.2.12 (Systematic ideal codes). *Let $n \geq 1$ and $s \geq 1$. A systematic ideal linear code of index s and rate $1/s$ is an $[ns, n]_{\mathbb{F}_{q^m}}$ code admitting a partiy-check matrix of the form:*

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{I}_n & 0 & \cdots & 0 & \boldsymbol{H}_1 \\ 0 & \boldsymbol{I}_n & & & \boldsymbol{H}_2 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & \boldsymbol{I}_n & \boldsymbol{H}_{s-1} \end{pmatrix}$$

where blocks $\boldsymbol{H}_i$ are ideal matrices of size $n \times n$ modulo an irreducible polynomial $Q \in \mathbb{F}_q[X]$ of degree n. The set of matrices of the form above is denoted $\mathcal{IM}_Q(\mathbb{F}_{q^m}^{(s-1)n \times sn})$.

When using ideal codes in a cryptosystem, the difficult problem also needs to be adapted.

Problem 1.2.4 (Ideal rank syndrome decoding problem IRSD). *Given a block length n, an index $s \geq 1$, a polynomial $Q \in \mathbb{F}_q[X]$, $\boldsymbol{H} \in \mathcal{IM}_Q(\mathbb{F}_{q^m}^{(s-1)n \times sn})$, $\boldsymbol{y} \in \mathbb{F}_{q^m}^{(s-1)n}$ and a target weight w, the ideal rank syndrome decoding problem $\text{IRSD}_{n,s,w}$ asks to find $\boldsymbol{e} \in \mathbb{F}_{q^m}^{sn}$ such that $\|\boldsymbol{e}\| \leq w$ and $\boldsymbol{e}\boldsymbol{H}^T = \boldsymbol{y}$.*

Two rank-metric NIST candidates featured ideal codes in their design: ROLLO [ABD+19] is a McEliece-like encryption scheme based on a masking of ideal LRPC codes, whereas RQC [AAB+19] is an Alekhnovich-like encryption scheme based on random ideal codes.

The indistinguishability problem between a random parity-check matrix and an LRPC matrix also has an ideal variant.

Problem 1.2.5 (Ideal LRPC codes decisional problem - ILRPC-Ind). *Given a polynomial $Q \in \mathbb{F}_q[X]$ of degree n and a vector $\boldsymbol{h} \in \mathbb{F}_{q^m}^n$, distinguish whether the ideal code $\mathcal{C}$ with the ideal parity-check matrix generated by $(\boldsymbol{1}, \boldsymbol{h})$ modulo Q is a random ideal code or an ideal LRPC code of weight d.*

1.2.3.4 Cryptography in the rank metric

In this subsection we give a short historical panorama of rank metric cryptography, with recent results but excluding new developments regarding the multi-dimensional approach that will be covered in Chapter 2.

Encryption schemes. Cryptosystems using a hidden decoder in the rank metric either rely on the weak structure of LRPC codes with a light masking; or on the strong structure of Gabidulin with a heavy masking. For the first LRPC category, the initial proposal in [GRSZ14a] has been refined with ideal cyclicity in the NIST submission ROLLO [ABD+19]. For the second Gabidulin

category, all variants of GPT [GPT91] were broken by structural attacks, except the cryptosystem of Loidreau [Loi17] which seems to resist to cryptanalysis when the distortion parameter λ is chosen > 2. Without a hidden decoder, the only proposal is RQC [AAB+19] that features ideal codes. A non-cyclic version of RQC would have too large sizes to be seriously considered.

Although very promising, rank metric schemes were not admitted to the third round of the NIST standardization process. The reason given by NIST in their Round 2 report [AASA+20] was the impact of algebraic attacks [BBB+20,BBC+20] that demanded an update on the parameter sets of rank metric submissions. Since 2020, the rank metric has however gained in maturity and cryptanalysis has stabilized, with some complexities being revised upwards in very recent results [BBB+23].

Signature schemes. In rank metric, the first signature scheme was a hash-and-sign proposal with LRPC codes, RankSign [GRSZ14b], that was broken in [DT18]. Other rank-based signature schemes derive from a Fiat-Shamir transformation of a proof of knowledge: Durandal [ABG+19] is a signature following the approach of Lyubashevsky [Lyu09] that we investigate in Section 4.1, and RYDE [ABB+23b], a NIST submission to the onramp call for additional signatures.

1.2.4 Summary

We summarize different approaches for designing code-based cryptography in a three-dimensional table.

	Hamming metric		Rank metric	
	with cyclicity	no cyclicity	with cyclicity	no cyclicity
With hidden decoder	BIKE	McEliece	ROLLO	Loidreau
Without hidden decoder	HQC	Alekhnovich	RQC	

Table 1.2: Examples of encryption schemes for various design possibilities in code-based cryptography. A more complete panorama, including the multi-dimensional approach, is presented in Table 0.4.

	Hamming metric	Rank metric
Hash-and-sign	Wave CFS†	RankSign†
Proof of knowledge	SDitH	Durandal† RYDE

Table 1.3: Examples of signature schemes for various design possibilities in code-based cryptography. A more complete panorama, including the multi-dimensional approach, is presented in Table 0.5. The † symbol indicates a scheme for which there currently exists no parameters that resist cryptanalysis.

1.3 Permuted kernels

The Permuted Kernel Problem (PKP) is a problem introduced by Shamir in 1990 [Sha90]. Although it is not a decoding problem *per se*, it is closely linked to some coding theory problems. The original PKP consists in finding a permutation that sends a given vector onto the kernel of a given matrix. We retain here a generalized form of PKP, called *inhomogeneous*.

Definition 1.3.1 (IPKP problem). *Let (q, m, n) be positive integers such that $m < n$. Given $\boldsymbol{H} \in \mathbb{F}_q^{m\times n}$, $\boldsymbol{x} \in \mathbb{F}_q^n$ and $\boldsymbol{y} \in \mathbb{F}_q^m$, the Inhomogeneous Permuted Kernel Problem IPKP(q, m, n, t) asks to find a permutation $\pi \in \mathcal{S}_n$ such that $\pi[\boldsymbol{x}]\boldsymbol{H}^T = \boldsymbol{y}$.*

The expected number of solutions for an random instance of IPKP is

$$\frac{n!}{q^m},$$

allowing to define a *"Gilbert-Varshmov"-like* bound for IPKP, that is the largest integer m for which $\frac{n!}{q^m} \leq 1$.

PKP has been proven NP hard in [GJ79]. When $q = 2$, the SD problem can be seen as a particular case of PKP with the vector $\boldsymbol{x} = (1, \cdots, 1, 0, \cdots, \cdots, 0) \in \mathbb{F}_2^n$ containing w ones and $n - w$ zeros. We will also see in the next chapter that PKP is linked to a subcode equivalence problem (see Section 2.4). The foundational cryptanalytic efforts on PKP were the method of Georgiades [Geo92] that made use of linear algebra, as well as a time-memory trade-off [BCCG93]. These attacks can be seen as the equivalent of, respectively, Prange algorithm [Pra62] and Dumer's birthday decoding [Dum89], showing once again a strong connection of PKP with coding theory. Recently, the most efficient approaches to solve the permuted kernel problem have been hybrid methods [JJ01, KMP19] between the two initial proposals (just like for codes).

Up to this date, the only cryptographic usage of the permuted kernel problem has been to build a proof of knowledge, then turned into a signature scheme. In particular, no encryption scheme based on PKP was found until now. The well-studied hardness, compact description and simplicity of involved objects and corresponding computations has made the PKP an attractive candidate for post-quantum signature schemes in recent years [BG23, BFK+19, Beu20b].

Chapter 2

Overview on the multi-dimensional approach

Contents

2.1 General idea

The multi-dimensional approach is a non canonical terminology; we endeavor to gather into a single paradigm various approaches, which all have in common a *multiple usage of parts of a secret element*, in order to improve the efficiency of existing primitives or to build new advanced cryptosystems. Consequently, security assumptions are slightly modified and new problems must be thoroughly studied. The multi-dimensional approach conveys several names in the litterature: a few examples are the (rank) support learning problem, interleaved codes, and synchronized decoding.

Let us try to expose simply the benefits of this approach; suppose we are in a McEliece-style encryption scheme with a ciphertext of the form:

$$\boldsymbol{c} = \boldsymbol{x}\boldsymbol{G} + \boldsymbol{e}.$$

The weight w of the error $\boldsymbol{e}$ must be chosen high enough so that an attacker cannot recover the message with decoding techniques on random codes, such as ISD and variants. However, this weight cannot be too large, otherwise it would result in a decryption failure. The main idea of the multi-dimensional approach is to modify these constraints on the weight by sending multiple copies of ciphertexts:

$$\begin{aligned} \boldsymbol{c}_1 &= \boldsymbol{x}_1\boldsymbol{G} + \boldsymbol{e}_1 \\ &\vdots \\ \boldsymbol{c}_\ell &= \boldsymbol{x}_\ell\boldsymbol{G} + \boldsymbol{e}_\ell \end{aligned}$$

not with unrelated errors – that would make no difference at all – but rather with **correlated** errors (usually, it means that they share the same support). For some special codes, it happens that having correlated errors improves the decoding capacity. The weight w can then be increased without hampering decryption. Consequently, it may happen that the scheme is better resisting to attacks thanks to this increase, allowing to a decrease in the size of matrix $\boldsymbol{G}$ while keeping the same transmission rate k/n. Because most encryption schemes based on coding theory especially suffer from a large public key size, any improvement on the size of $\boldsymbol{G}$ can be a game changer. Improved encryption schemes based on this principle will be presented in Chapter 3. We propose to call this approach **multi-dimensional**, because the error can now be seen as a bidimensional matrix instead of being a flat vector. This approach is specific to coding theory and is not immediately transposable to lattices, as it heavily depends on the notion of support, which does not exist in the latter family.

Of course, this modification requires a change in the security assumption. The attacker now gets several noisy codewords whose errors share the same support, hence needs to solve a variant of the syndrome decoding problem that is usually called in the literature the **support learning problem**. This new problem is described in detail for both rank and Hamming metrics in Sections 2.2 and 2.3, respectively.

For **signature schemes**, the contribution of the multi-dimensional approach is debatable, although not for the same reason as the quasi-cyclic structure mentioned in Section 1.1.

The case of hash-and-sign signatures is the most interesting. Unlike encryption, for which the sender chooses to sample multiple errors with the same support, the signer receives a random syndrome and does not control the shape of the error. The hash of the message, turned into multiple syndromes, can be decoded with a multi-dimensional algorithm only if the corresponding errors share the same support, which represents an additional constraint. Let us take the example of a Hamming metric code that can be decoded up to weight t_1 for a single syndrome and weight t_2 for multiple ℓ syndromes. In the first case, the probability of decoding a random syndrome is the number of errors divided by the number of possible syndromes, i.e.

$$p_1 \approx \frac{\binom{n}{t_1} q^{t_1}}{q^{n-k}}.$$

In the second case of ℓ syndromes, this probabitlity becomes

$$p_\ell \approx \frac{\binom{n}{t_2} q^{\ell t_2}}{q^{\ell(n-k)}},$$

the binomial coefficient being counted only once because all the errors must have the same support. Even though $t_2 > t_1$, the same inequality is therefore not guaranteed for the probabilities. A suitable scheme for which $p_\ell > p_1$ has not been found yet.

When it comes to signatures based on a proof of knowledge, an adaptation of the proof to the multi-dimensional case often leads to more unusual security assumptions than in multi-dimensional encryption schemes (see [BGKM23, BG23]) and parameters must be therefore chosen too large to be competitive. However, in some very specific situations, the use of multiple syndromes that share the same error support gives an additional flexibility that enables to build a signature scheme such as Durandal [ABG+19], whose security is studied in Section 4.1. Because it is structurally different from decoding instances, a multi-dimensional instance can however be useful for PKP-based proofs of knowledge. It gave rise to the PERK signature scheme that is presented in Section 4.2.

Finally, the multi-dimensional approach gives a natural **additively homomorphic** scheme: adding two errors sharing the same support gives another error still with the same support. The security of the homomorphic scheme relies on the support learning problem, restricting the number of publishable ciphertexts. A homomorphic multiplication operation is also non trivial to obtain. We explore a way to overcome these limitations in Chapter 5. Surprisingly, the security of our resulting homomorphic scheme is not reduced to a support learning problem but rather to a tradi-

tional syndrome decoding problem for a low rate code.

The rest of this chapter is dedicated to the study of difficult problems related to the multi-dimensional approach as well as a literature review of their applications. We start in the rank metric and we focus especially on the rank support learning problem that has attracted a lot of attention in the recent years. Then we inspect the case of the Hamming metric, which was less studied but historically the first one to appear. Finally, we explore previous multi-dimensional variants on the permuted kernel problem. All existing analyses tend to indicate that **the multi-dimensional versions of these problems remain difficult**, as long as the number of correlated instances is small.

2.2 The multi-dimensional approach in rank metric

The most emblematic problem of the multi-dimensional approach is the Rank Support Learning (RSL) problem. It asks to decode multiple syndromes whose errors share the same rank support.

Problem 2.2.1 (Rank support learning problem RSL [GHPT17]). *Let ℓ be a positive integer. Given $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}, (\boldsymbol{y}_1, \dots, \boldsymbol{y}_\ell) \in (\mathbb{F}_{q^m}^{n-k})^\ell$ and a target rank weight w, the rank support learning problem $\mathrm{RSL}_{m,n,k,w,\ell}$ asks to find $(\boldsymbol{e}_1, \dots, \boldsymbol{e}_\ell) \in (\mathbb{F}_{q^m}^n)^\ell$ such that there exists an $\mathbb{F}_q$-subspace $E \subset \mathbb{F}_{q^m}$, such that $\dim E \leq r$ and for all $1 \leq i \leq \ell$, $\boldsymbol{e}_i \in E^n$ and $\boldsymbol{e}_i \boldsymbol{H}^T = \boldsymbol{y}_i$.*

One immediately sees that RSL is a generalized version of RSD; the additional flexibility induced by this new definition allows to build more efficient schemes. This will be covered mainly in Chapter 3. But before that, the burning question is: how can we compare the respective hardness of RSD and RSL?

2.2.1 Comparison with RSD and best attacks

It could seem that RSD is always harder than RSL because an instance of RSD can be extracted from an instance of RSL. It is actually more subtle than that and depends on the choice of parameters; it may happen that RSD has more solutions on average than RSL. In that case the solver for RSD may not find the required solution for RSL. For a random RSD instance, the expected number of solutions is given by

$$N_1 = \frac{\begin{bmatrix} m \\ w \end{bmatrix}_q q^{wn}}{q^{m(n-k)}}. \tag{2.1}$$

This expected number of solution is approximately 1 when w is on the rank Gilbert-Varshamov bound d_{RGV} (Definition 1.2.6).

For a random RSL instance with the same parameters, the expected number of solutions is given by

$$N_\ell = \frac{\begin{bmatrix} m \\ w \end{bmatrix}_q q^{w\ell n}}{q^{m\ell(n-k)}}. \tag{2.2}$$

We can define a new "*multi-rank* Gilbert-Varshamov" bound d_{mRGV} that is the smallest w such that $N_\ell > 1$. It depends on ℓ, on top of q, m, k and n. When $m = n$ and $\ell \to \infty$, this bound has an asymptotic equivalent

$$d_{mRGV} \sim n\left(1 - \frac{k}{n}\right).$$

The proof can be read in Appendix A. This is to be compared with the expression of the mono-dimensional bound when $m = n$, $d_{RGV} \sim n\left(1 - \sqrt{\frac{k}{n}}\right)$. Because the most advantageous zone

of parameters for a better resistance to attacks is usually to choose the weight on the Gilbert-Varshamov bound, we can see that switching to a multi-dimensional problem leads to an increase in the optimal choice of weight w.

It is also readily seen from Equations (2.1) and (2.2) that

$$N_\ell < N_1 \iff w < m(1-R)$$

where $R = k/n$ is the rate of the code. For usual parameters in cryptography, this inequality will be verified and we will have $N_\ell < N_1 \leq 1$, and in this regime RSD is indeed harder than RSL. We will now assume both conditions are verified:

$$\begin{aligned} w &< m(1-R) \\ w &\leq d_{RGV} \end{aligned}$$

and when this holds, does it exist better attacks to RSL than those directed to RSD?

In [GHPT17], a **polynomial attack** against RSL is presented when $\ell \geq nw$. In [BBBG22], another polynomial attack was presented when $\ell > kw\frac{m}{m-w}$. In most cases, the latter bound is sharper.

Informally, the principle of this attack works as follows. When $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_\ell) \in (E^n)^\ell$ with $E \subset \mathbb{F}_{q^m}$ of dimension $\leq w$, there exists a non-zero $\mathbb{F}_q$-linear combination of the tuple with $a = \lfloor (\ell-1)/w \rfloor$ zeros. In other words, there exist scalars $(\lambda_1, \ldots, \lambda_\ell) \in \mathbb{F}_q^\ell$ and $\breve{\boldsymbol{e}} \in E^{n-a}$ such that

$$(\boldsymbol{0} \,|\, \breve{\boldsymbol{e}}) = \sum_{i=1}^{\ell} \lambda_i \boldsymbol{e}_i.$$

By setting $\breve{\boldsymbol{H}} = \boldsymbol{H}_{*,[a+1,n]}$, an attacker can solve the linear system of $(n-k)m$ equations over $\mathbb{F}_q$

$$\breve{\boldsymbol{e}}\breve{\boldsymbol{H}}^T = \sum_{i=1}^{\ell} \lambda_i \boldsymbol{y}_i$$

whose $(n-a)m + \ell$ unknowns are the coordinates of $\breve{\boldsymbol{e}}$ and the λ_i.

The non trivial solution of small weight $\breve{\boldsymbol{e}}$ will be the unique solution on average as long as it has more equations than variables, i.e.

$$\ell + \left(n - \left\lfloor \frac{\ell-1}{w} \right\rfloor\right) m \leq (n-k)m.$$

The above inequality can be simplified to the bound obtained by [BBBG22]. Then the attacker can recover the support E from the entries of $\breve{\boldsymbol{e}}$ and finish the attack in polynomial time.

When the condition is not met, the technique explained above can be turned into a combinatorial attack, as explained in [BBBG22], with a work factor of

$$\mathcal{O}\left(q^{w\left(m - \left\lfloor \frac{m(n-k)-\ell}{n-a} \right\rfloor\right)}\right) \tag{2.3}$$

operations in $\mathbb{F}_q$, where $a = \left\lfloor \frac{\ell-1}{w} \right\rfloor$.

A **subexponential attack** was found in [DT18] when $\ell > kw$. The principle is, when the matrix $\boldsymbol{H}$ from the RSL instance in systematic form, to define the following $\mathbb{F}_q$-subspace

$$\mathcal{C} = \{\boldsymbol{x}\boldsymbol{E}\boldsymbol{H}^T \,|\, \boldsymbol{x} \in \mathbb{F}_q^\ell\}$$

with $\boldsymbol{E}$ being the $\ell \times n$ matrix whose rows are the errors $\boldsymbol{e}_i$ of support E from the RSL instance. The authors show that $\mathcal{C} \cap E^{n-k}$ is of $\mathbb{F}_q$-dimension $\geq \ell - wk$ ([DT18, Theorem 1]), which means

that $\mathcal{C}$ contains many codewords revealing the secret support E. By solving a superdetermined bilinear system of equations given by a basis of $\mathcal{C}$, they show that the support E can be recovered in subexponential time using standard Gröbner basis techniques.

Algebraic attacks against RSL were initially studied in [DT18, BB21]. When compared to the best algebraic attacks against RSD, an algebraic approach specific to RSL seems to be useful only in some specific cases. When the number of syndromes is large enough, i.e. $\ell > n-k-w$, a closed formula for the complexity was computed in [BBBG22] and is equal to:

$$\min_{\substack{1\leq b\leq w+1\\ 0\leq \alpha_R<n-a-w\\ 0\leq \alpha_\lambda<\ell'-b}} \left(2^{w\alpha_R+\alpha_\lambda} \min\left(\mathcal{N}_{\leq b}^{\mathbb{F}_2}(\mathcal{M}_{\leq b}^{\mathbb{F}_2})^{\omega-1}, (\ell'-\alpha_\lambda)\binom{k-a+1+w}{w}(\mathcal{M}_{\leq b}^{\mathbb{F}_2})^2\right)\right) \tag{2.4}$$

with ω the linear algebra constant, a the unique integer such that $aw < \ell \leq (a+1)w$, $\ell' = aw+1$ and $\mathcal{M}_{\leq b}^{\mathbb{F}_2}$ and $\mathcal{N}_{\leq b}^{\mathbb{F}_2}$ defined as follows:

$$\mathcal{M}_{\leq b}^{\mathbb{F}_2} = \sum_{i=1}^{b} \binom{n-a-\alpha_R}{r}\binom{\ell'-\alpha_\lambda}{i},$$

$$\mathcal{N}_{\leq b}^{\mathbb{F}_2} = m\sum_{i=1}^{b}\sum_{d=1}^{i}\sum_{j=1}^{n-k}\binom{j-1}{d-1}\binom{n-k-j}{r-d+1}\binom{\ell'-\alpha_\lambda-j}{i-d}.$$

For the case $\ell \leq n-k-w$, no closed formula is known yet.

Summary

A summary is represented in Figure 2.1:

- When $\ell \leq kw$, the RSL problem is believed to be of exponential complexity, the security must be estimated by taking into account both combinatorial complexity (2.3) and algebraic complexity (2.4), as well as attacks against RSD than can be, for some parameters, surprisingly better than specific RSL attacks;
- When $kw < \ell \leq \min(nw, kw\frac{m}{m-w})$, the RSL problem is of subexponential complexity;
- When $\ell > \min(nw, kw\frac{m}{m-w})$, the RSL problem can be solved in polynomial time.

2.2.2 Applications

The first cryptographic scheme to rely on the RSL problem is RankPKE [GHPT17]. It can be seen as a way to turn the originally inefficient Alekhnovich scheme [Ale03] into a deterministic scheme without using quasi-cyclic or ideal structure. It was broken by [DT18], due to a weakness in another problem than RSL, namely the indistinguishability of simple codes. Two years later, a signature scheme based on RSL was published, Durandal [ABG+19]. We break the scheme in Section 4.1 but, once again, not because of RSL but of another unrelated problem PSSI. More recently, RSL was used to build Multi-RQC [BBBG22], the multi-dimensional version of RQC [AAB+19]. Thanks to this modification, the public key size was divided by a factor 4, for approximatively the same ciphertext size. Other efficient schemes based on RSL will be presented in Section 3.1 and Section 3.2.

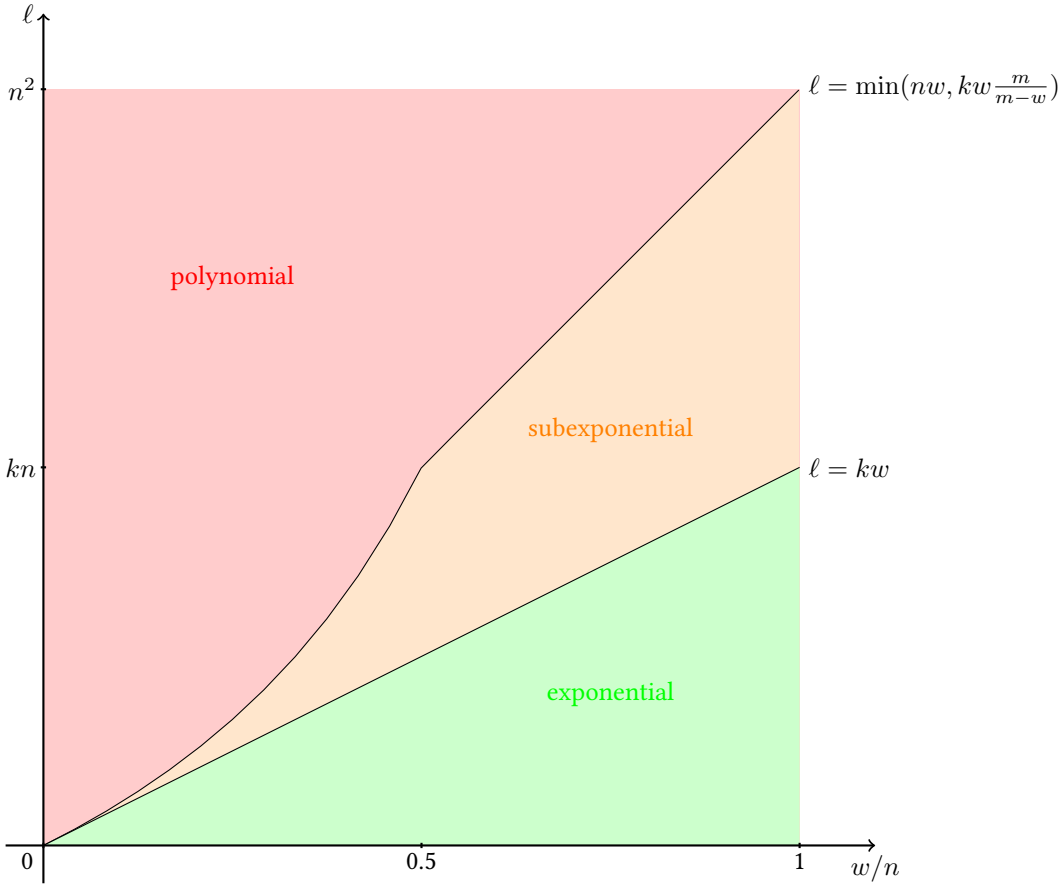

Figure 2.1: Difficulty zones for an RSL instance with an $[n,k]_{\mathbb{F}_{q^m}}$ code, $n = m = 2k$, and ℓ errors of the same support of weight w.

2.2.3 Another point of view: interleaved codes

The RSL problem can be interpreted as the standard rank syndrome decoding of an interleaved code. An interleaved code of order ℓ of a code $\mathcal{C}$ is a code defined by the concatenation of ℓ (possibly different) words in $\mathcal{C}$.

Definition 2.2.1 (Interleaved code). *Let $\mathcal{C}$ be an $[n,k]_{\mathbb{F}_{q^m}}$ code. The (horizontal) interleaved code of $\mathcal{C}$ of order ℓ is the code defined by*

$$\mathcal{I}(\ell;\mathcal{C}) = \{(\boldsymbol{c}_1 \mid \ldots \mid \boldsymbol{c}_\ell) \mid \forall i \in [1,\ell], \boldsymbol{c}_i \in \mathcal{C}\}.$$

Lemma 2.2.1. *Let $\mathcal{C}$ be an $[n,k]_{\mathbb{F}_{q^m}}$ code of parity-check matrix $\boldsymbol{H}$. The interleaved code $\mathcal{I}(\ell;\mathcal{C})$ is an $[\ell n, \ell k]_{\mathbb{F}_{q^m}}$ code and admits as a parity-check the following matrix*

$$\mathcal{I}(\ell;\boldsymbol{H}) = \begin{pmatrix} \boldsymbol{H} & & 0 \\ & \ddots & \\ 0 & & \boldsymbol{H} \end{pmatrix}$$

Obviously, the RSL problem is equivalent to the RSD problem on the correspoding interleaved code.

Lemma 2.2.2. *Let $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ and $(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_\ell) \in (\mathbb{F}_{q^m}^{n-k})^\ell$. $(\boldsymbol{e}_1,\ldots,\boldsymbol{e}_\ell)$ is a solution to the $\mathrm{RSL}_{m,n,k,w,\ell}$ instance $\big(\boldsymbol{H},(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_\ell)\big)$ if and only if $(\boldsymbol{e}_1 \mid \ldots \mid \boldsymbol{e}_\ell) \in \mathbb{F}_{q^m}^{n\ell}$ is a solution to the $\mathrm{RSD}_{m,\ell n,\ell k,w}$ instance $\big(\mathcal{I}(\ell;\boldsymbol{H}),(\boldsymbol{y}_1 \mid \ldots \mid \boldsymbol{y}_\ell)\big)$.*

There exists another definition of interleaving that concatenates the codewords *vertically*, yielding a matricial code. This approach is called vertical interleaving, where the one from Definition 2.2.1 is called horizontal interleaving.

Definition 2.2.2 (Vertical interleaved code). *Let $\mathcal{C}$ be an $[n,k]_{\mathbb{F}_{q^m}}$ code. The vertical interleaved code of $\mathcal{C}$ of order ℓ is the matrix code defined by*

$$\mathcal{I}_{vert}(\ell;\mathcal{C}) = \left\{ \begin{pmatrix} \boldsymbol{c}_1 \\ \vdots \\ \boldsymbol{c}_\ell \end{pmatrix} \;\middle|\; \forall i \in [1,\ell], \boldsymbol{c}_i \in \mathcal{C} \right\} \subset \mathbb{F}_{q^m}^{\ell \times n}.$$

Vertical interleaving was historically the first definition [BKY03] to appear as it was adapted to the case of the Hamming metric. In a rank metric context, the problem with vertical interleaving is that the usual rank metric on matrix codes is not adapted to reflect the metric of the original code. Therefore an alternative metric was defined for vertical interleaved codes.

Definition 2.2.3 (Vertical rank norm [RPWZ19]). *The vertical rank norm $\|\boldsymbol{A}\|_{vert}$ of a matrix $\boldsymbol{A} \in \mathbb{F}_{q^m}^{\ell \times n}$ is the rank of the vertical concatenation of the expansions of rows from $\boldsymbol{A}$.*

$$\|\boldsymbol{A}\|_{vert} = rank \begin{pmatrix} \mathbf{Mat}(\boldsymbol{A}_1) \\ \hline \vdots \\ \hline \mathbf{Mat}(\boldsymbol{A}_\ell) \end{pmatrix},$$

where $\boldsymbol{A}_1, ..., \boldsymbol{A}_\ell$ are the rows of $\boldsymbol{A}$.

A *horizontal* rank norm can be defined on $\mathbb{F}_{q^m}$-matrix codes that is equivalent to the usual rank norm of a horizontal interleaved code.

Definition 2.2.4 (Horizontal rank norm). *The horizontal rank norm $\|\boldsymbol{A}\|_{horiz}$ of a matrix $\boldsymbol{A} \in \mathbb{F}_{q^m}^{\ell \times n}$ is the rank of the horizontal concatenation of the expansions of rows from $\boldsymbol{A}$:*

$$\|\boldsymbol{A}\|_{horiz} = rank\left(\mathbf{Mat}(\boldsymbol{A}_1) \,|\, \ldots \,|\, \mathbf{Mat}(\boldsymbol{A}_\ell)\right),$$

where $\boldsymbol{A}_1, ..., \boldsymbol{A}_\ell$ are the rows of $\boldsymbol{A}$.

Some cryptographic systems were defined with the vertical rank norm [RPWZ19]. We claim that such an approach is not appropriate for rank-based cryptography and that horizontal interleaving should be preferred, see Section 3.2. Moreover, the problem of syndrome decoding in the vertical rank norm is not equivalent to RSL but is a different problem that has been studied in [RPWZ21] and that can be attacked when the interleaving order is greater or equal to the weight ($\ell \geq w$). This condition severely restricts the possibilities for building efficient schemes with vertical interleaving in the rank metric.

2.3 The curious case of the Hamming metric

The RSL problem has a Hamming metric equivalent that is defined below.

Problem 2.3.1 (Support learning problem SL [GHPT17]). *Let ℓ be a positive integer. Given $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}, (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\ell) \in (\mathbb{F}_q^{n-k})^\ell$ and a target weight w, the support learning problem $SL_{n,k,w,\ell}$ asks to find $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_\ell) \in (\mathbb{F}_q^n)^\ell$ such that there exists a set $J \subset [1,n]$, such that $\#J = w$ and for all $1 \leq i \leq \ell$, $\boldsymbol{H}\boldsymbol{e}_i^T = \boldsymbol{y}_i$ and $\forall j \in [1,n] \setminus I, \boldsymbol{e}_i^{(j)} = 0$.*

The problem was formalized for the first time in [GHPT17] but appeared non explicitely as early as 1997. In the KKS signature scheme [KKS97], the public key is a pair of matrices $(\boldsymbol{H}, \boldsymbol{F})$ where $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$ and $\boldsymbol{F} = \boldsymbol{H}_{*,J}\boldsymbol{G}^T$ where $J \subset [1,n]$ is a private support of size w and $\boldsymbol{G}$ is a private matrix of size $\ell \times w$. Recovering the private key from the public key exactly amounts

to solving an instance of the support learning problem. All parameters of [KKS97] were broken by [OT11] with an attack on the public key, i.e. precisely an attack on the SL problem. As the authors point out, they succeeded in breaking the KKS scheme because it was in a range of parameters favorable to an attack, but the security of the whole KKS signature scheme is not compromised. This was a preliminary witness of the existence of different zones of difficulty for the SL problem.

Similar to the rank metric, the expected number of solutions to SL can be approximated to

$$N_\ell = \frac{\binom{n}{w} q^{\ell w}}{q^{\ell(n-k)}}$$

yielding a "*multi(-Hamming)* Gilbert-Varshamov" bound d_{mGV} that could be defined as the smallest w such that $N_\ell > 1$.

However, somewhat surprisingly, the polynomial bound for the Hamming version of the support learning problem is lower than for the rank version: when $\ell \geq w$, the SL problem is solvable in polynomial time. This fact was proven in [GHPT17]. When $\ell < w$, the speedup of SL over SD is believed to be around q^ℓ, which means that for small values of ℓ, the SL problem can be robust.

It has not been used in many Hamming code-based cryptosystems though, hence the Hamming support learning problem is somewhat less studied than its rank metric counterpart.

The multi-dimensional approach is indeed only known to improve the decoding capacity of Reed-Solomon [BKY03] and Goppa codes [HLPWZ19] so far. An improvement of the McEliece cryptosystem based on interleaved Goppa codes was presented in [EWZZ18], then repaired in [HLPWZ19] and led to notable reduction in the public key size; for some parameters the decrease being more than 50%. However, no result was found up to this date regarding the multi-dimensional approach on RMRS and MDPC codes, which are featured into the most compact NIST Round 4 candidates, respectively HQC [AAB+22] and BIKE [ABB+22].

Another reason that could prevent the adoption of the Hamming SL problem is that the polynomial bound of this problem depends only on the weight w of the code, contrary to the rank metric where the length n is also a parameter. This reduced flexibility could restrict the practical use of the SL problem in cryptography. Finding another concrete application to the SL problem would be an interesting open research perspective.

Similar to the rank metric, the multi-dimensional approach in the Hamming metric is linked to the notion of interleaved codes. However, contrary to the rank metric, horizontal interleaving is unpractical in Hamming metric as the horizontal concatenation of words sharing the same small support yields a word of large support and all the information about the original support is lost. On the other hand, vertical concatenation seems to be the correct method, when defining the Hamming weight of a matrix as the number of its non-zero columns. We recall below the definition of vertical interleaving, adapting the definition to the context of $\mathbb{F}_q$-linear codes.

Definition 2.3.1 (Vertical interleaved code)**.** *Let $\mathcal{C}$ be an $[n,k]_{\mathbb{F}_q}$ code. The vertical interleaved code of $\mathcal{C}$ of order ℓ is the matrix code defined by*

$$\mathcal{I}_{vert}(\ell;\mathcal{C}) = \left\{ \begin{pmatrix} \boldsymbol{c}_1 \\ \vdots \\ \boldsymbol{c}_\ell \end{pmatrix} \;\middle|\; \forall i \in [1,\ell], \boldsymbol{c}_i \in \mathcal{C} \right\} \subset \mathbb{F}_q^{\ell \times n}.$$

Definition 2.3.2 (Vertical Hamming norm)**.** *The vertical Hamming norm $\|\boldsymbol{A}\|_{h,vert}$ of a matrix $\boldsymbol{A} \in \mathbb{F}_q^{\ell \times n}$ is the number of its non-zero columns.*

The SL problem corresponds to the syndrome decoding problem in the (matricial) vertical interleaved code equipped with the vertical Hamming norm. However, contrary to the rank metric, it is not possible to view the SL problem as the SD problem of a traditional vector code with the Hamming metric. Indeed, the matrix code $\mathcal{I}_{vert}(\ell;\mathcal{C})$ could be seen as a vector code of length n over the alphabet $\mathbb{F}_{q^\ell}$ but it would not be $\mathbb{F}_{q^\ell}$ linear, hence would not be a properly defined linear code. This underlines what appears to be a fundamental difference between the rank and Hamming

metric, namely that the multi-dimensional approach in the Hamming metric forces one to change the norm.

Lastly, the multi-dimensional approach was also used in [AAPS11] to build a homomorphic encryption based on codes in the Hamming metric. In their work the authors call codewords sharing the same support *synchronized*. The security of their encryption scheme is reduced to the Decisional Synchronized Codeword Problem (DCSP) that they introduce in their paper. They studied the security of the DCSP in the particular case of Reed-Muller codes and chose their parameters so as to avoid the best attacks against DCSP for this class of codes. They did not study the reduction of their DCSP to the SL problem as it was not formalized yet, however it is likely that the two problems are highly related.

2.4 Multi-dimensional variants of PKP

The permuted kernel problem can be naturally extended to a multi-dimensional version. It was suggested for the first time in [LP11].

Definition 2.4.1 ((Multi-dimesional) IPKP problem)**.** *Let (q, m, n, ℓ) be positive integers such that $m < n$. Given a matrix $\boldsymbol{H} \in \mathbb{F}_q^{m\times n}$, linearly independent vectors $(\boldsymbol{x}_1, \dots, \boldsymbol{x}_\ell) \in (\mathbb{F}_q^n)^\ell$ and $(\boldsymbol{y}_1, \dots, \boldsymbol{y}_\ell) \in (\mathbb{F}_q^m)^\ell$, the Inhomogeneous Permuted Kernel Problem IPKP(q, m, n, ℓ) asks to find a permutation $\pi \in \mathcal{S}_n$ such that for all $1 \leq i \leq \ell, \pi[\boldsymbol{x}_i]\boldsymbol{H}^T = \boldsymbol{y}_i$.*

Remark. This multi-dimensional variant of IPKP is the most natural but not exactly the one we will consider for our signature scheme PERK, presented in Section 4.2. Indeed, in the security proof, the problem that arises is another variant that can be found in Definition 4.2.12.

The above definition can be retained as the most generic form of the IPKP problem; this is the point of view adopted in [SBC22]. The requirement on the linear independence of vectors $\boldsymbol{x}_i$ makes sense because should there be a non trivial linear combination $\sum_i \lambda_i \boldsymbol{x}_i = 0$, two cases may happen:

- $\sum_i \lambda_i \boldsymbol{y}_i \neq 0$, the same linear combination on the $\boldsymbol{y}_i$ is non zero, in which case the IPKP instance has no solution; or
- $\sum_i \lambda_i \boldsymbol{y}_i = 0$, and then the instance can be reduced to an equivalent instance by removing one of the $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$.

The expected number of solutions of a random multi-dimensional IPKP instance is this time

$$N_\ell = \frac{n!}{q^{m\ell}}.$$

Contrary to the multi-dimensional versions of (rank) syndrome decoding problems, this number of solutions decreases very rapidly with ℓ. Also note that, in contrast with the case $\ell = 1$ for which the solving strategies are the same for PKP (in which $\boldsymbol{y} = 0$) and IPKP, there seem to be a fundamental difference between multi-dimensional PKP and IPKP. For the homogeneous PKP, it makes no sense to take $\ell > n - m$, since if there exists a solution π to the problem, all the $\boldsymbol{x}_i$ belong to the code $\pi^{-1}[\ker(\boldsymbol{H})]$ of dimension $n - m$. In the inhomogeneous IPKP, ℓ can take any value from 1 to n.

As usual, it is legitimate to ask how do mono- and multi-dimensional variants of IPKP compare to each other, and whether there exists a value for ℓ above which the IPKP becomes polynomial.

Both questions were adressed partially in [SBC22]. First, they link the *homogeneous* PKP with a code equivalence problem. Second, they adapt the state-of-the-art algorithm for mono-dimensional IPKP, the KMP algorithm, to the multi-dimensional case.

After that, we will finally present a simple polynomial attack on the inhomogeneous multi-dimensional IPKP when $\ell = n$.

Link between PKP and code equivalence problems

Definition 2.4.2 (Subcode Permutation Equivalence Problem (SEP)). *Let $k' < k \leq n$. Given two codes $\mathcal{C}[n,k]$ and $\mathcal{C}'[n,k']$, does there exists a permutation π such that*

$$\pi[\mathcal{C}'] \subset \mathcal{C}?$$

When the two given codes are of the same dimension, the problem is renamed PEP.

Definition 2.4.3 (Permutation Equivalence Problem (PEP)). *Let $k \leq n$. Given two codes $\mathcal{C}[n,k]$ and $\mathcal{C}'[n,k]$, does there exists a permutation π such that*

$$\pi[\mathcal{C}'] = \mathcal{C}?$$

Authors of [SBC22] show that the homogeneous PKP is exactly equivalent to SEP or PEP with the following correspondence on parameters:

PKP	Equivalent problem	Parameters
$\ell < n-m$	SEP	$k = n-m, k' = \ell$
$\ell = n-m$	PEP	$k = k' = n-m$

Table 2.1: Relations between PKP, SEP and PEP, and corresponding parameters.

Both problems are well-known in coding theory. While SEP was proven NP-complete in [BGK17], the NP-completeness of PEP would imply a collapse of the polynomial hierarchy [PR97]. For random codes, with overwhelming probability the algorithms in [BOS19, Sen00] solve PEP in polynomial time: for this reason, the homogeneous PKP is considered easy on average when $\ell = n-m$. For IPKP, for which when $\ell = n$, a simple polynomial attack can be mounted (see below).

Best attacks against IPKP: KMP and SBC algorithms

The two natural yet non naive attacks against mono-dimensional IPKP are Georgiades attack [Geo92] and a time-memory trade-off [BCCG93]. In the first attack, all possibilities for the first $n-m$ coordinates of $\pi[\boldsymbol{x}]$ are explored while the last m are deduced by solving a linear system. This can be seen as the equivalent of Prange algorithm [Pra62]. The second attack consists in splitting the vector $\pi[\boldsymbol{x}]$ in two halves, then building lists of all possible values for the two half vectors, and finally reconciling the lists by a collision search. As the lists can be large, it costs more memory but is usually more efficient.

Until very recently, the best attack against mono-dimensional IPKP was the KMP algorithm [KMP19]. It is a meet-in-the-middle approach between Georgiades' attack and the time-memory trade-off. It can be straightforwardly adapted to the multi-dimensional case, as shown by the authors of [SBC22]. The total cost of the attack is the minimum for a parameter $1 \leq u \leq m$ of the algorithm:

$$\mathcal{T}_{\text{KMP}} = \mathcal{O}\left(\min_{1\leq u\leq m}\left(|L_1| + |L_2| + |L_1 \bowtie L_2|\right)\right)$$

with

$$|L_1| = |L_2| = \binom{n}{\frac{n-m+u}{2}} \left(\frac{n-m+u}{2}\right)!$$

$$|L_1 \bowtie L_2| = \frac{|L_1| \times |L_2|}{q^{\ell u}}$$

The authors of [SBC22] introduced an algorithmic improvement to the KMP algorithm by a preprocessing step. The resulting SBC algorithm is the most effective compared to KMP algorithm

when the parameter ℓ increases. Similarly to the KMP algorithm though, there exists a value for ℓ after which the complexity of the algorithm does not decrease any more. Figure 2.2 depicts this lower limit attained by the SBC algorithm.

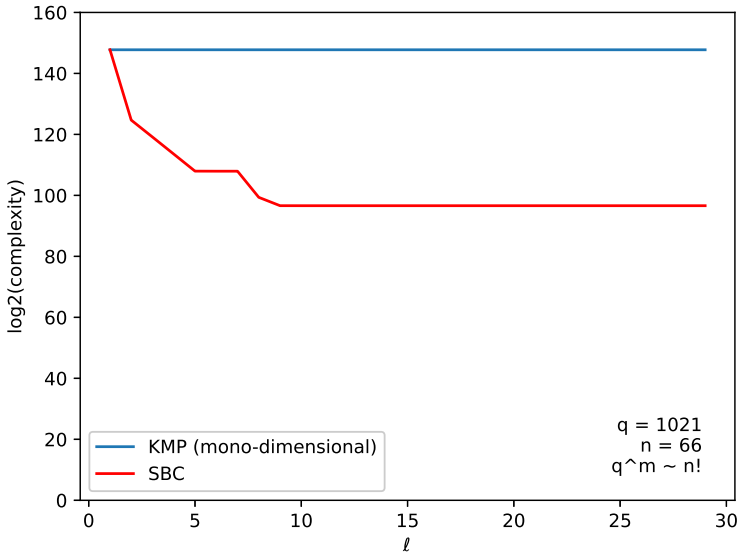

Figure 2.2: Complexity of mono-dimensional KMP and multi-dimensional SBC algorithms for increasing parameter ℓ. The figure was drawn for $q = 1021$, $n = 66$ and $m = 31$, corresponding to the zone where the expected number of solutions is close to 1.

A polynomial attack when $\ell = n$

We describe in this paragraph a simple polynomial attack on the inhomogeneous IPKP problem when $\ell = n$. In that case, the vectors $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell)$ from the IPKP instance form a basis of $\mathbb{F}_q^n$. For any index $1 \leq i \leq n$, there exists a linear combination such that $\sum_j \lambda_j \boldsymbol{x}_j = (\ldots, 0, 1, 0, \ldots)$, the unique non-zero coordinate being taken at index i. We can then obtain a vector by taking the same linear combination on the $\boldsymbol{y}_j$: $\boldsymbol{y} = \sum_j \lambda_j \boldsymbol{y}_j$. If there exists a solution π to the given instance, then $\boldsymbol{y}$ is bound to be equal to a column of $\boldsymbol{H}$ whose index yields $\pi(i)$. The attacker can then reconstruct the solution permutation π in polynomial time since it requires linear algebra only.

Summary

The Figure 2.3 below summarizes the main takeways of this section: when $\ell = 1$, both PKP and IPKP can be solved with the KMP algorithm. When $\ell > 1$, the best algorithm becomes the SBC algorithm which has a lower limit with increasing ℓ. The difference beetween PKP and IPKP appears for high ℓ. In the homogeneous case, when $\ell = n - m$ the problem is reduced to the hard problem PEP of coding theory; $\ell > n - m$ is impossible. In the inhomogeneous case, the problem becomes polynomial for $\ell = n$.

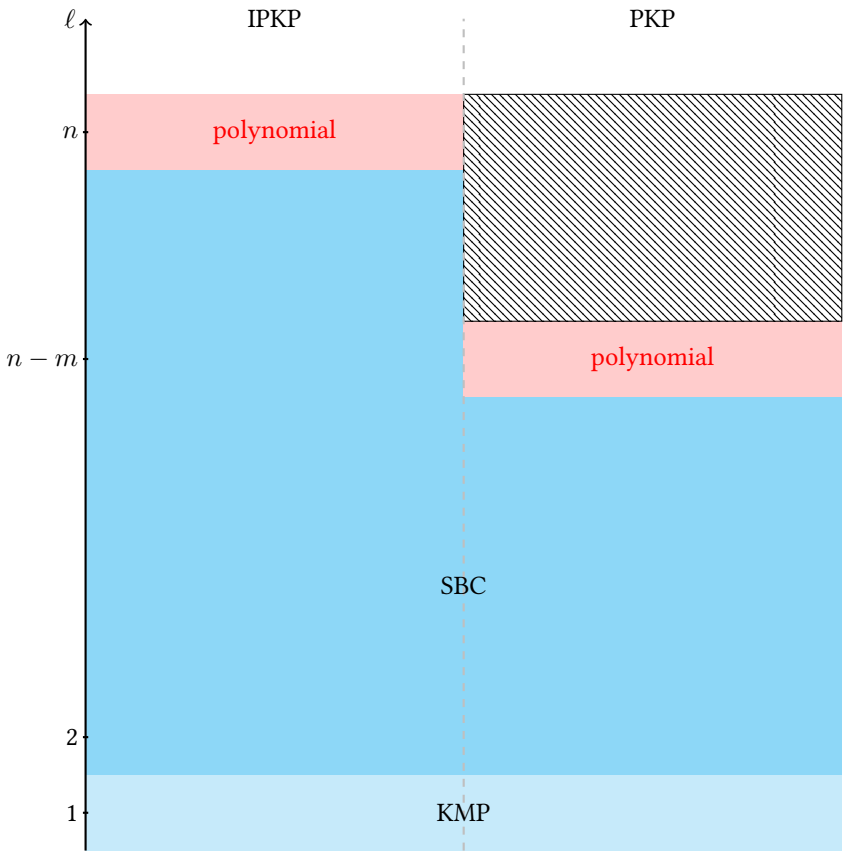

Figure 2.3: Best algorithms for solving inhomogeneous and homogeneous permuted kernel problems depending on the dimension ℓ.

Chapter 3

Design of encryption schemes

Contents

In this chapter, we apply the multi-dimensional approach to two rank metric encryption schemes: ROLLO [ABD+19] and the cryptosystem of Loidreau [Loi17]. We conclude the chapter with an improvement on the constant time key generation algorithm for LRPC code-based cryptosystems.

3.1 LRPC-MS: ROLLO with multiple syndromes

ROLLO [ABD+19] is a second-round NIST PQC standardization candidate relying on LRPC codes (Definition 1.2.9). The public key is a masked version of a low weight homogeneous parity-check matrix. LRPC parameters led to quite efficient cryptosystems for Decoding Failure Rates (DFRs) around 2^{-30}, but for DFRs below 2^{-128} there was a significant efficiency drop, as to obtain such DFRs, codes needed to be quite long. The present section shows how to avoid larger code lengths while still obtaining very low DFRs. This results in a significant improvement of the associated cryptosystems, both for the structured and unstructured case, without compromising a precise analysis of the DFR.

3.1.1 Improving LRPC decoding by using multiple syndromes

The decoding algorithm of LRPC codes presented in Section 1.2.3 has a probability of failure whose main component is $q^{rd-(n-k)-1}$ (where r is the weight of the error and d the dual weight of the LRPC code, see Proposition 1.2.1) so it forces one to have a large n in an LRPC-cryptosystem in order to obtain a DFR below 2^{-128}. To overcome this constraint, we made the observation that when several syndromes with same error support $(\boldsymbol{s}_1, ..., \boldsymbol{s}_\ell)$ were used in the decoding algorithm, the DFR was decreasing. This fact is the cornerstone of our new cryptosystem. We describe below the associated decoding problem.

Problem 3.1.1 (Decoding LRPC codes with mutliple syndromes). *Given $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ a parity-check matrix of an LRPC code of dimension d and support $F \subset \mathbb{F}_{q^m}$, a set of ℓ syndromes $\boldsymbol{s}_i \in \mathbb{F}_{q^m}^{n-k}$ for $1 \leqslant i \leqslant \ell$, and an integer r, the problem is to find a subspace E of dimension at most r such that there exists an error matrix $\boldsymbol{V} \in E^{n\times \ell}$ satisfying $\boldsymbol{HV} = \boldsymbol{S}$ where the i-th column of $\boldsymbol{S}$ is equal to $\boldsymbol{s}_i^T$.*

In order to solve this decoding problem, we introduce the Rank Support Recovery algorithm with multiple syndromes (Algorithm 3.1). It is exactly the same as Algorithm 1.1, but several columns are given to compute the syndrome space S. Intuitively, because the syndrome matrix $\boldsymbol{HV}$ has $(n-k)\times\ell$ entries inside the space EF of dimension rd, we would expect the Decoding Failure Rate of this new algorithm to be approximately $q^{rd-(n-k)\ell}$. Actually, because the entries of $\boldsymbol{HV}$ are not independent between each other, the result is not trivially established and requires technical lemmas which are presented in Appendix B.1.

Algorithm 3.1 Rank Support Recovery (RSR) algorithm with multiple syndromes

Input: $F = \langle f_1, ..., f_d\rangle$ an $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$, $\boldsymbol{S} = (s_{ij}) \in \mathbb{F}_{q^m}^{(n-k)\times\ell}$ the ℓ syndromes of error vectors of weight r and support E
Output: A candidate for the vector space E

▷ **Part 1:** Compute the vector space EF
Compute $S = \langle s_{11}, \cdots, s_{(n-k)\ell}\rangle$
▷ **Part 2:** Recover the vector space E
$E \leftarrow \bigcap_{i=1}^{d} f_i^{-1} S$
return E

In the following subsection, we describe our new scheme and its ideal variant, then study the Decoding Failure Rate.

3.1.2 Description of the schemes

3.1.2.1 LRPC-MS

Our scheme contains a hash function G modeled as a random oracle.

- $\mathsf{KeyGen}(1^\lambda)$:
  - choose uniformly at random a subspace F of $\mathbb{F}_{q^m}$ of dimension d and sample an couple of homogeneous matrices of same support $\boldsymbol{U} = (\boldsymbol{A}|\boldsymbol{B}) \xleftarrow{\$} F^{(n-k)\times(n-k)} \times F^{(n-k)\times k}$ such that $\boldsymbol{A}$ is invertible.
  - compute $\boldsymbol{H} = (\boldsymbol{I}_{n-k}|\boldsymbol{A}^{-1}\boldsymbol{B})$.
  - define $\mathsf{pk} = \boldsymbol{H}$ and $\mathsf{sk} = (F, \boldsymbol{A})$.
- $\mathsf{Encap}(\mathsf{pk})$:
  - choose uniformly at random a subspace E of $\mathbb{F}_{q^m}$ of dimension r and sample a matrix $\boldsymbol{V} \xleftarrow{\$} E^{n\times\ell}$.
  - compute $\boldsymbol{C} = \boldsymbol{H}\boldsymbol{V}$.
  - define $K = G(E)$ and return $\boldsymbol{C}$.
- $\mathsf{Decap}(\mathsf{sk})$:
  - compute $\boldsymbol{S} = \boldsymbol{A}\boldsymbol{C}$ $(= \boldsymbol{U}\boldsymbol{V})$
  - recover $E \leftarrow \mathsf{RSR}(F, \boldsymbol{S}, r)$ (Algorithm 3.1).
  - return $K = G(E)$ or $\perp$ (if RSR failed).

We need to have a common representation of a subspace of dimension r of $\mathbb{F}_{q^m}$. The natural way is to choose the unique matrix $\boldsymbol{M} \in \mathbb{F}_q^{r\times m}$ of size $r \times m$ in its reduced row echelon form such that the rows of $\boldsymbol{M}$ are a basis of E (represented in the canonical polynomial basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$).

An informal description of this scheme can be found in Figure 3.1. We deal with the semantic security of the KEM in Section 3.1.5.

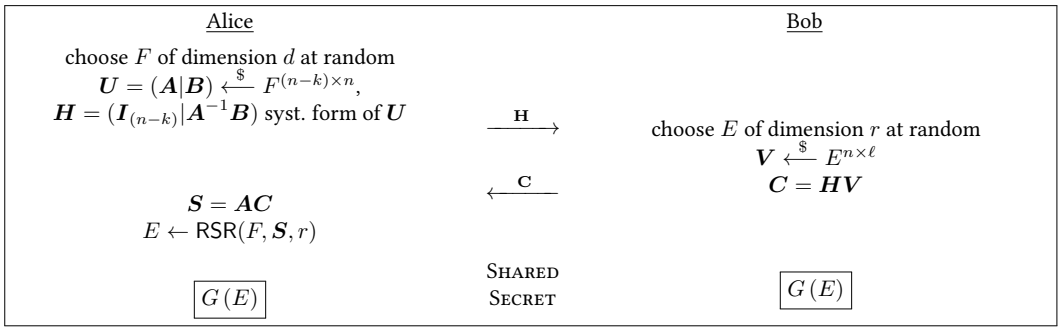

Figure 3.1: Informal description of our new Key Encapsulation Mechanism LRPC-MS. $\boldsymbol{H}$ constitutes the public key.

3.1.2.2 ILRPC-MS

An informal description of this scheme is found in Figure 3.2. As for the non-ideal scheme, we deal with the semantic security of the KEM in Section 3.1.5.

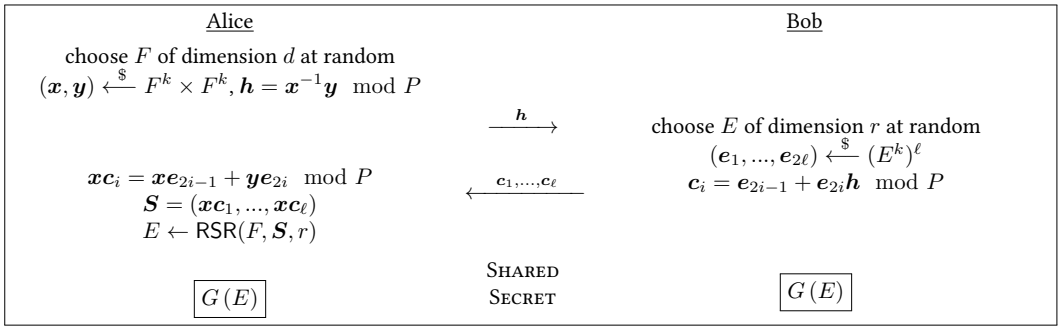

Figure 3.2: Informal description of our new Key Encapsulation Mechanism with ideal structure ILRPC-MS. $\boldsymbol{h}$ constitutes the public key.

3.1.3 Decoding Failure Rate of the schemes

The Decoding Failure Rate (DFR) of our scheme is the probability of failure of the Rank Support Recovery algorithm with multiple syndromes described in Algorithm 3.1. As stated in Section 1.2.3, the two cases that can provoke a failure of the algorithm are:

- $S \subsetneq EF$, the entries of the matrix $\boldsymbol{UV}$ do not generate the entire space EF, or
- $E \subsetneq S_1 \cap \cdots \cap S_d$, the chain of intersections generate a space of larger dimension than E.

To study the probability of each case, we restrict ourselves to the case $\dim(EF) = rd$. Indeed, when $\dim(EF) < rd$, the correctness of the algorithm is preserved, and the probabilities associated to the two sources of decoding failures are lower than in the case $\dim(EF) = rd$, since all the vector spaces will be of smaller dimensions. Hence this restriction will lead to an upper bound on the failure probability.

The first case of failure can be dealt with the following theorem, which is fully proven in Appendix B.1. Its immediate corollary yields the probability of failure for the first case. We will assume for the rest of this document that $q = 2$ since the theorem is only proven in that case.

Theorem 3.1.1. *For $q = 2$, $n_1 + n_2 \leq n$ and for $\boldsymbol{U}$ and $\boldsymbol{V}$ random variables chosen uniformly in $F^{n_1 \times n}$ and $E^{n \times n_2}$ respectively, $\mathbb{P}(\mathbf{Supp}(\boldsymbol{UV}) \neq EF) \leq (n_1 + 1)q^{rd - n_1 n_2}$*

Corollary 3.1.1. *For $q = 2$, $k \geq \ell$ and for $\boldsymbol{U}$ and $\boldsymbol{V}$ random variables chosen uniformly in $F^{(n-k) \times n}$ and $E^{n \times \ell}$ respectively, the probability that the syndrome space S computed by the algorithm $\mathsf{RSR}(F, \boldsymbol{UV}, r)$ is not equal to EF is bounded by above by $(n - k + 1)q^{rd-(n-k)\ell}$*

As for the second failure case, $E \subsetneq S_1 \cap \cdots \cap S_d$, we apply again the upper-bound $q^{-(d-1)(m-rd-r)}$, used in Section 1.2.3 for Proposition 1.2.1. This leads to the following proposition:

Proposition 3.1.1. *For $q = 2$, $k \geq \ell$ and for $\boldsymbol{U}$ and $\boldsymbol{V}$ random variables chosen uniformly in $F^{(n-k) \times n}$ and $E^{n \times \ell}$ respectively, the Decoding Failure Rate of Algorithm 3.1 $\mathsf{RSR}(F, \boldsymbol{UV}, r)$ is bounded from above by:*

$$q^{-(d-1)(m-rd-r)} + (n - k + 1)q^{rd-(n-k)\ell}$$

This proposition extends immediately to the ideal case without modifications.

3.1.4 Impact on the asymptotic range of parameters

By reducing the decoding failure rate, the multiple syndrome approach fundamentally changes the zone of parameters that we consider for our cryptosystem.

In previous LRPC code-based cryptosystems, the decoding failure rate imposed the choice of r and d to be below $\sqrt{n}$ because of the need for $rd < n - k$ (cf. Proposition 1.2.1).

In this work, we can choose r and d bigger than $\sqrt{n}$. To simplify the rest of the analysis we will consider half-rate codes only, for which $k = n/2$. We will show that it is even asymptotically possible to choose r and d on the rank Gilbert-Varshamov bound d_{RGV}.

The DFR formula leads to the choice of a large ℓ such that when m and n tend to infinity, $rd = n\ell/2 + o(1)$. Because we chose $r = d = d_{RGV}$, we get

$$\ell \sim \frac{2d_{RGV}^2}{n}.$$

When applying the asymptotic formula of d_{RGV} ([ABG+19], §2.4) to the case $k = n/2$, we get $d_{RGV} \leq n/2$. As a result, we obtain that ℓ is asymptotically upper bounded by $d_{RGV} = r$. To the best of our knowledge, the range where $\ell \leq r$ is a hard parameter range for which the RSL problem has no known polynomial attacks. In practice, for the parameters considered at the end of this document, when choosing r and d on the Gilbert-Varshamov bound, ℓ has to be chosen slightly greater than r and the RSL problem is still in a difficult zone.

The fact that we can choose r and d on the rank Gilbert-Varshamov bound has two major implications:

- Algebraic attacks against the RSD problem are more difficult when r gets closer to d_{RGV}.
- The secret parity check matrix $\boldsymbol{U}$ is homogeneous of weight d_{RGV} so the minimal distance of the dual of the resulting LRPC code is about d_{RGV}, just like a random code. It gives more confidence in the indistinguishably of the public matrix $\boldsymbol{H}$ (LRPC-Ind problem).

Our proposal is the only code-based cryptosystem with structural masking that has such an interesting property for the distinguishing problem.

3.1.5 Security

3.1.5.1 Preliminary definition

For all the problems RSD, IRSD, RSL and IRSL defined in Chapters 1 and 2, we can give a decisional version whose goal is to distinguish (for the example of RSD) between a random input $(\boldsymbol{H}, \boldsymbol{s})$ or an actual syndrome input $(\boldsymbol{H}, \boldsymbol{H}\boldsymbol{e}^T)$. We denote these decisional versions DRSD, DIRSD, DRSL and DIRSL. The reader is referred to [AGH+19] for more details about decisional problems.

3.1.5.2 IND-CPA proof

Theorem 3.1.2. *Under the hardness of the LRPC-Ind and DRSL$_{k,n,r,\ell}$ problems, the KEM LRPC-MS presented in section 3.1.2 is indistinguishable against Chosen Plaintext Attack in the Random Oracle Model.*

Proof. We are going to proceed in a sequence of games. The simulator first starts from the real scheme. First we replace the public key matrix by a random matrix, and then we use the ROM to solve Rank Support Learning.

We start from the normal game G_0: We generate the public key $\boldsymbol{H}$ honestly, as well as E, and $\boldsymbol{C}$.

- In game G_1, we now replace $\boldsymbol{H}$ by a random matrix, the rest is identical to the previous game. From an adversary point of view, the only difference is the distribution of $\boldsymbol{H}$, which

is either generated at random, or the systematic form of a low weight parity matrix. This is exactly the *LRPC codes decisional* problem, hence

$$\mathsf{Adv}_{\mathcal{A}}^{G_0} \leq \mathsf{Adv}_{\mathcal{A}}^{G_1} + \mathsf{Adv}_{\mathcal{A}}^{\text{LRPC-Ind}}.$$

- In game G_2, we now proceed as earlier except we receive $\boldsymbol{H}, \boldsymbol{C}$ from a Rank Support Learning challenger. After sending $\boldsymbol{C}$ to the adversary, we monitor the adversary queries to the Random Oracle, and pick a random one that we forward as our simulator answer to the $\text{DRSL}_{k,n,r,\ell}$ problem. Either the adversary was able to predict the random oracle output, or with probably $1/q_G$, we picked the query associated with the support E (by q_G we denote the number of queries to the random oracle G), hence

$$\mathsf{Adv}_{\mathcal{A}}^{G_1} \leq 2^{-\lambda} + 1/q_G \cdot \mathsf{Adv}_{\mathcal{A}}^{\text{DRSL}}$$

which leads to the conclusion.

□

For the ideal version of our scheme, the security proof is exactly the same except that ideal versions of hard problems appear. The IND-CPA property follows immediately.

Theorem 3.1.3. *Under the hardness of problems ILRPC-Ind and $\text{DIRSL}_{k,n,r,\ell}$, the KEM ILRPC-MS presented in section 3.1.2 is indistinguishable against Chosen Plaintext Attack in the Random Oracle Model.*

The maximal value of ℓ for which $\text{DIRSL}_{k,n,r,\ell}$ is hard is way lower than its non-ideal counterpart. Indeed, a single ideal syndrome can be expanded in k traditional syndromes by performing ideal rotations. That is why the value of ℓ is lower in the parameter sets for the ideal version.

3.1.5.3 Known attacks

Attacks against the RSL problem were covered in Section 2.2, therefore in this paragraph we only recall a specific attack against the LRPC-Ind problem.

Given $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times k}$ such that $(\boldsymbol{I}_{n-k}|\boldsymbol{H})$ is the parity-check matrix of a code $\mathcal{C}$, the problem of distinguishing LRPC codes is to decide whether $\mathcal{C}$ is a random code or an LRPC code.

The best known attack against this problem for almost ten years ([GRSZ14a]) consists in using the underlying homogeneous structure of the LRPC code to find a codeword of weight d in a $[n - \lfloor \frac{n-k}{d} \rfloor, n - k - \lfloor \frac{n-k}{d} \rfloor]_{q^m}$ subcode $\mathcal{C}'$ of the dual code $\mathcal{C}^\perp$ generated by $(\boldsymbol{I}_{n-k}|\boldsymbol{H})$ rather than a codeword of weight d in the $\mathcal{C}^\perp$ $[n, n-k]$ code. Then one can consider the previously described algebraic or combinatorial attacks for this slightly smaller code (but for the same weight d).

3.1.6 Parameters

Choice of parameters. In Section 3.1.5, the security of the protocol is reduced to the LRPC-Ind and DRSL problems (or their ideal variants). The best known attacks on these problems are thus used to define our parameters. We also chose our parameters in order to have the Decoding Failure Rate (DFR) below or very close to $2^{-\lambda}$, where λ is the security parameter, using Proposition 3.1.1. We only considered parameters with $k \geq \ell$ as required by these propositions.

Size of parameters. One may use seeds to represent the random data in order to decrease the keysize. We use the NIST seed expander with 40 bytes long seeds.

The public key pk is composed of a matrix of size $(n-k) \times n$ in a systematic form, so its size is $\left\lceil \frac{k(n-k)m}{8} \right\rceil$ bytes. The size is reduced to $\left\lceil \frac{(n-k)m}{8} \right\rceil$ bytes in the ideal case. The secret key sk is composed of two random matrices that can be generated from a seed, so its size is 40 bytes. The ciphertext ct is composed of a matrix of size $(n-k) \times \ell$, so its size is $\left\lceil \frac{(n-k)\ell m}{8} \right\rceil$ bytes. The shared secret ss is composed of $K = G(E)$, so its size is 64 bytes.

Parameters are given in Table 3.1. The "structure" column indicates whether this parameter uses unstructured (random) matrices or ideal ones. The number indicated in the "DFR" column is actually $-\log_2(\text{DFR})$.

Instance	Structure	q	n	k	m	r	d	ℓ	Security	DFR	pk size	ct size	pk + ct
LRPC-MS-128	Random	2	34	17	113	9	10	13	128	126	4,083	3,122	7,205
LRPC-MS-192	Random	2	42	21	151	11	11	15	192	190	8,324	5,946	14,270
ILRPC-MS-128	Ideal	2	94	47	83	7	8	4	128	126	488	1,951	2,439
ILRPC-MS-192	Ideal	2	188	89	109	9	8	3	192	189	1,213	3,638	4,851

Table 3.1: Parameters for our unstructured and ideal LRPC-MS cryptosystem. The security is expressed in bits and sizes are expressed in bytes.

Comparison with other unstructured cryptosystems We compare our cryptosystem to other structured and unstructured proposals. Our comparison metric is the usual TLS-oriented communication size (public key + ciphertext).

For Loong.CCAKEM [Wan19], we consider only the third set of parameters since the other sets of parameters have an error weight below 6 and thus are vulnerable to algebraic attacks. For Loidreau cryptosystem, we consider the parameters presented in the conclusion of [Pha21] which take into account the recent improvements on algebraic attacks. For both cryptosystems mentioned in this paragraph, parameters were not available (N/A) for 192 bits of security.

Instance	128 bits	192 bits
LRPC-MS	7,205	14,270
Loong.CCAKEM-III	18,522	N/A
FrodoKEM	19,336	31,376
Loidreau cryptosystem	36,300	N/A
Classic McEliece	261,248	524,348

Instance	128 bits	192 bits
ILRPC-MS	2,439	4,851
BIKE	3,113	6,197
ROLLO-II	4,030	N/A
HQC	6,730	13,548

Table 3.2: Comparison of sizes of unstructured KEMs (left table) and structured code-based KEMs (right table). The sizes represent the sum of public key and ciphertext expressed in bytes.

Performance. We provide indicative performance measurements of an implementation of some of the LRPC-MS cryptosystem parameters. Benchmarks were realized on an Intel® Core™ i7-11850H CPU by averaging 1000 executions.

Instance	KeyGen	Encap	Decap
LRPC-MS-128	383	137	3,195
ILRPC-MS-128	214	107	1,213

Table 3.3: Performances of our LRPC-MS cryptosystems in thousands of CPU cycles.

As for other code-based schemes, the decapsulation algorithm has a higher computational cost than key generation and encapsulation. Note however, that our implementation does not yet benefit from the techniques of [CL22]. These techniques improved the decapsulation performance by a factor 15 (for 128 bits of security) with respect to the existing (and simpler to adapt) implementations we used as a basis for our benchmarking.

3.2 LowMS: Loidreau's encryption scheme with multiple syndromes

3.2.1 Improving Gabidulin codes decoding using multiple syndromes

Interleaved Gabidulin codes can be corrected with high probability beyond the $\lfloor \frac{d-1}{2} \rfloor$ bound. More precisely, efficient decoders are known that are able to correct $t \leq \lfloor \frac{\ell}{\ell+1}(n-k) \rfloor$ errors with high probability. We recall below the result of [SJB11] regarding the decoding probability of an interleaved Gabidulin code.

Proposition 3.2.1 ([SJB11], Equations (43) and (44)). *Let $\mathcal{G}$ be a Gabidulin code of parity check matrix $\boldsymbol{H}$ and $\mathcal{IG}(\ell;\mathcal{G})$ the corresponding interleaved code of order ℓ.*

Let $E \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, t)$ be an error support of dimension t with $\ell \leq t \leq \lfloor \frac{\ell}{\ell+1}(n-k) \rfloor$. Let an error $\boldsymbol{e} = (\boldsymbol{e}_1 \dots \boldsymbol{e}_\ell) \in E^{\ell n}$ where for each i, $\boldsymbol{e}_i \xleftarrow{\$} E^n$. Let $\boldsymbol{y} \in \mathbb{F}_{q^m}^{\ell(n-k)}$ be the corresponding syndrome of the interleaved code $\mathcal{IG}(\ell;\mathcal{G})$:

$$\boldsymbol{y} = (\boldsymbol{e}_1 \boldsymbol{H}^T \dots \boldsymbol{e}_\ell \boldsymbol{H}^T).$$

The decoding Algorithm 4 from [SJB11], on input $\boldsymbol{y}$, fails to output correctly the error $\boldsymbol{e}$ with a probability upper bounded by

$$3.5q^{-m\left\{(\ell+1)\left(\frac{\ell}{\ell+1}(n-k)-t\right)+1\right\}}.$$

We can then build a decoding algorithm for Interleaved Gabidulin codes that takes as input an $\ell \times (n-k)$ syndrome matrix and returns the error vector.

Algorithm 3.2 InterleavedGab.Decode

Input: Received syndrome matrix $\boldsymbol{Y} \in \mathbb{F}_{q^m}^{\ell \times (n-k)}$

Output: Error matrix $\boldsymbol{E} \in \mathbb{F}_{q^m}^{\ell \times n}$ or decoding failure $\perp$

1: Flatten $\boldsymbol{Y}$ into $\boldsymbol{y} = (\boldsymbol{y}_1 \dots \boldsymbol{y}_\ell) \in \mathbb{F}_{q^m}^{\ell(n-k)}$ where $\boldsymbol{y}_i$ denotes the i-th row of $\boldsymbol{Y}$.
2: Apply Algorithm 4 from [SJB11] to $\boldsymbol{y}$.
3: If it fails, return $\perp$.
4: Else, we get an error vector $\boldsymbol{e} = (\boldsymbol{e}_1 \dots \boldsymbol{e}_\ell)$ where each suberror $\boldsymbol{e}_i \in \mathbb{F}_{q^m}^n$.
5: **return** the matrix $\boldsymbol{E}$ whose rows are $\boldsymbol{e}_1, \dots, \boldsymbol{e}_\ell$.

Proposition 3.2.1 turns immediately into the following corollary which is adapted to InterleavedGab.Decode algorithm.

Corollary 3.2.1. *Let $\mathcal{G}$ be a Gabidulin code of parity-check matrix $\boldsymbol{H}$. Let $E \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, t)$ an error support of dimension t with $\ell \leq t \leq \lfloor \frac{\ell}{\ell+1}(n-k) \rfloor$. Let $\boldsymbol{Y} \in \mathbb{F}_{q^m}^{\ell \times n}$ be defined by $\boldsymbol{Y} = \boldsymbol{E}\boldsymbol{H}^T$ where the error is a matrix $\boldsymbol{E} \xleftarrow{\$} E^{\ell \times n}$ whose coefficients are picked uniformly at random in the error support.*

Algorithm InterleavedGab.Decode (3.2), on input $\boldsymbol{Y}$, fails to output correctly the error matrix $\boldsymbol{E}$ with a probability upper bounded by

$$3.5q^{-m\left\{(\ell+1)\left(\frac{\ell}{\ell+1}(n-k)-t\right)+1\right\}}.$$

3.2.2 Description of the LowMS scheme

The LowMS KEM scheme is given by three algorithms (LowMS.KeyGen, LowMS.Encaps, LowMS.Decaps) defined in Algorithms 3.3, 3.4, 3.5. LowMS KEM is parametrized by the following parameters:

- q the size of the base field $\mathbb{F}_q$
- m the degree of the field $\mathbb{F}_{q^m}$ used in rank metric
- (k, n) the dimension and length of a Gabidulin code
- r the rank weight of the error[1]
- λ the rank weight of the perturbation matrix
- ℓ the number of syndromes sent in the ciphertext (interleaving order)
- $\mathcal{H}$ is a hash function which outputs values $\in \mathbb{F}_2^{512}$, such as SHA-512

We use the Niederreiter framework [Nie86] instead of the McEliece one to define our scheme, i.e., we perform all operations using parity-check matrices instead of generator matrices. This allows to divide the size of the ciphertext by 2 (if $k = n/2$). Similarly to ROLLO [ABD+19] and other rank metric KEMs, using a Niederreiter system implies to compute the shared secret as a hashed value of the error support E.

Algorithm 3.3 LowMS.KeyGen

Input: None

Output: Keypair $(pk, sk) \in \left(\mathbb{F}_{q^m}^{(n-k)\times n}, \mathbb{F}_{q^m}^{(n-k)\times(n-k)} \times \mathbb{F}_{q^m}^{(n-k)\times n} \times \mathbb{F}_{q^m}^{n\times n}\right)$

1: Choose a parity-check matrix of an $[n, k]$ Gabidulin code $\boldsymbol{H} \xleftarrow{\$} \mathcal{G}_{(n,k)}^T \in \mathbb{F}_{q^m}^{(n-k)\times n}$.
2: Choose an $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$, $F \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, \lambda)$.
3: Choose uniformly at random an $n \times n$ perturbation matrix with entries in F, $\boldsymbol{P} \xleftarrow{\$} F^{n\times n}$.
4: Compute $\boldsymbol{S} \in \mathbb{F}_{q^m}^{(n-k)\times(n-k)}$ such that $\boldsymbol{H'} = \boldsymbol{S}^T \boldsymbol{H}\, \boldsymbol{P}^T$ is in systematic form.
5: Define $pk := \boldsymbol{H'}$ and $sk := (\boldsymbol{S}, \boldsymbol{H}, \boldsymbol{P})$.
6: **return** (pk, sk).

Algorithm 3.4 LowMS.Encaps

Input: Public key $pk = \boldsymbol{H'} \in \mathbb{F}_{q^m}^{(n-k)\times n}$.

Output: Ciphertext $c \in \mathbb{F}_{q^m}^{\ell\times(n-k)}$, session key $K \in \mathbb{F}_2^{512}$.

1: Sample the error support $E \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, r)$.
2: Sample the error matrix $\boldsymbol{E} \xleftarrow{\$} E^{\ell\times n}$, such that $\mathbf{Supp}(\boldsymbol{E}) = E$.
3: Compute $\boldsymbol{C} = \boldsymbol{E}\, \boldsymbol{H'}^T$.
4: Compute $K = \mathcal{H}(E)$.
5: **return** $c = \boldsymbol{C}, K$.

Algorithm 3.5 LowMS.Decaps

Input: Ciphertext $c = \boldsymbol{C} \in \mathbb{F}_{q^m}^{\ell\times(n-k)}$ and secret key $sk = (\boldsymbol{S}, \boldsymbol{H}, \boldsymbol{P}) \in \mathbb{F}_{q^m}^{(n-k)\times(n-k)} \times \mathbb{F}_{q^m}^{(n-k)\times n} \times \mathbb{F}_{q^m}^{n\times n}$

Output: Session key $K \in \mathbb{F}_2^{512}$

1: Compute $\boldsymbol{C'} = \boldsymbol{C}\boldsymbol{S}^{-1}$.
2: Recover $\boldsymbol{E'} = \boldsymbol{E}\boldsymbol{P}$ = InterleavedGab.Decode($\boldsymbol{C'}$).
3: Compute $\boldsymbol{E} = \boldsymbol{E'}\boldsymbol{P}^{-1}$ and $E = \mathbf{Supp}(\boldsymbol{E})$
4: **return** $K = \mathcal{H}(E)$.

[1] In the comparison paper [RPWZ19], r is noted t_{pub}.

The decoding algorithm recovers the support E of the error matrix as long as conditions of Corollary 3.2.1 apply, i.e. $\ell \leqslant r\lambda \leqslant \lfloor \frac{\ell}{\ell+1}(n-k) \rfloor$.

Remark. In order to hash E and obtain the same value during encryption and decryption, we need a canonical representation for a subspace E of $\mathbb{F}_{q^m}$ of dimension r. We choose the unique matrix $\in \mathbb{F}_q^{r \times m}$ in reduced row echelon form whose rows form a basis of E.

3.2.3 Decoding Failure Rate

We prove simultaneously the correctness of our KEM and its decoding failure rate.

Proposition 3.2.2 (DFR). *Under the assumption that the entries of $\boldsymbol{EP}$ are randomly independent elements of EF, the decoding failure rate (DFR) of our scheme is upper bounded by*

$$3.5q^{-m((\ell+1)(\frac{\ell}{\ell+1}(n-k)-r\lambda)+1)}.$$

Justification for the assumption. The behaviour of a product matrix $\boldsymbol{EP}$ was previously studied in the context of LRPC decoding. In [ABD+19, Proposition 2.4.3], the decoding failure rate calculation, validated by simulations, relies on the fact that a product of a vector with entries in E by a matrix with entries in F is a random vector with entries in EF. In Theorem 3.1.1, it is shown that the support of a product matrix $\boldsymbol{EP}$ has the same probability, up to a constant factor, of being equal to EF as a random matrix with entries in EF. Therefore we can reasonably make the assumption that every entry of $\boldsymbol{EP}$ is a random element of EF.

Proof. We have

$$\begin{aligned}\boldsymbol{C}' &= \boldsymbol{C}\boldsymbol{S}^{-1} \\ &= \boldsymbol{E}\boldsymbol{H}'^{T}\boldsymbol{S}^{-1} \\ &= \boldsymbol{E}\boldsymbol{P}\boldsymbol{H}^{T} \\ &= \boldsymbol{E}'\boldsymbol{H}^{T},\end{aligned}$$

with $\boldsymbol{E}' = \boldsymbol{EP}$ being the error matrix decoded by InterleavedGab.Decode (Algorithm 3.2). Each of its entries E'_{ij} is such that $E'_{ij} \in EF$, where E is the support of the entries of $\boldsymbol{E}$ and F is the support of the entries of $\boldsymbol{P}$.

According to the assumption, every entry of $\boldsymbol{E}'$ is a random element of EF, thus we can apply Corollary 3.2.1, the dimension of the error support being $t = r\lambda$. In our parameter sets, we were careful enough to fulfill inequalities $\ell \leq r\lambda \leq \lfloor \frac{\ell}{\ell+1}(n-k) \rfloor$, so that the conditions of Corollary 3.2.1 are met. We thus obtain the upper bound on the decryption failure rate. □

3.2.4 Difference with previous work

In this subsection, we try to present in the most understandable manner the difference with the approach of [RPWZ19] which also suggests to interleave Loidreau's cryptosystem. The fine comprehension of this difference led us to build this new system with much more efficient parameters. The two main differing points concern the DFR and the error model.

Decoding failure rate. In [RPWZ19], Theorem 6, the DFR is given by a complex formula which can be approximated by

$$\frac{4}{q^m}.$$

To ensure a negligible DFR, the value $q = 16$ has been chosen in [RPWZ19]. Our formula seems more natural because it takes the value of ℓ into account, and therefore we are able to choose $q = 2$. This results in significantly more competitive parameters and also takes into account that for implementation reasons, cryptographic systems are usually preferred to work over binary fields.

Error model. Another key difference between this scheme and the one from [RPWZ19] stems from the error model. In [RPWZ19], the error matrix $\boldsymbol{E}$ is chosen of small *vertical* rank norm (Definition 2.2.3) where in our solution $\boldsymbol{E}$ is chosen of small *horizontal* rank norm (Definition 2.2.4).

This being said, the difference between Interleaved RSD and RSL is only a matter of norm, as shown in the following table.

Interleaved RSD	RSL
Given $(\boldsymbol{H}, \boldsymbol{Y}) \in \mathbb{F}_{q^m}^{(n-k)\times n} \times \mathbb{F}_{q^m}^{\ell\times(n-k)}$, find $\boldsymbol{E} \in \mathbb{F}_{q^m}^{\ell\times n}$ such that $\boldsymbol{H}\boldsymbol{E}^\top = \boldsymbol{Y}^\top$ and $\underline{\|\boldsymbol{E}\|_{vert} = r}$.	Given $(\boldsymbol{H}, \boldsymbol{Y}) \in \mathbb{F}_{q^m}^{(n-k)\times n} \times \mathbb{F}_{q^m}^{\ell\times(n-k)}$, find $\boldsymbol{E} \in \mathbb{F}_{q^m}^{\ell\times n}$ such that $\boldsymbol{H}\boldsymbol{E}^\top = \boldsymbol{Y}^\top$ and $\underline{\|\boldsymbol{E}\|_{horiz} = r}$.

We state that using the horizontal rank norm (and thus RSL) for the error instead of the vertical rank norm is a better choice for cryptographic applications. We state below a few elements in support for this claim:

- The technique presented in [RPWZ21] allows to decode an interleaved code of interleaving order $\ell \geq t$ where t is the *vertical* rank norm of the error. It penalizes the parameters of [RPWZ19] by forcing to choose $\ell < t$. The decoding algorithm in [RPWZ21] succeeds with negligible probability in the horizontal rank norm (see next subsection for details), hence opening the possibility of a higher interleaving order.

- The RSL problem has been studied for several years, and the recent algebraic attacks from [BB21] allow us to precisely compute the complexity of the RSL instances resulting from the chosen parameters.

- Because of the vertical rank norm, the error matrix $\boldsymbol{E}$ must be chosen in [RPWZ19] as a product matrix $\boldsymbol{AB}$, which in turn implies high constraints, such as

$$\frac{n-k}{2\lambda} < d_E \leq t - \ell + 1,$$

  where d_E is the minimum rank distance of the code spanned by the rows of $\boldsymbol{E}$. These constraints are lifted when using the horizontal rank norm.

Choosing the horizontal rank norm therefore allows a higher interleaving order ℓ and leads to better parameter sets (see Section 3.2.6).

3.2.5 Security

3.2.5.1 Public key indistinguishability

Problem 3.2.1 (Distorted Gabidulin codes indistinguishability Gab-Ind). *Given a matrix $\boldsymbol{H}' \in \mathbb{F}_{q^m}^{(n-k)\times n}$, the problem Gab-Ind$_{q,m,n,k,\lambda}$ distinguish whether $\boldsymbol{H}'$ is random or the parity-check matrix of a distorted Gabidulin code, i.e. $\boldsymbol{H}' = \boldsymbol{SHP}$ with $\boldsymbol{S}$ an $(n-k)\times(n-k)$ matrix with entries in $\mathbb{F}_{q^m}$, $\boldsymbol{H}$ the parity-check matrix of an $[n,k]$ Gabidulin code, and $\boldsymbol{P}$ an $n\times n$ homogeneous matrix of weight λ.*

This problem was studied in [Loi22] and we give the complexity of the best known attack to solve this problem below.

3.2.5.2 IND-CPA proof

Theorem 3.2.1. *Under the hardness of the distorted Gabidulin codes indistinguishability (Problem 3.2.1) and Rank Support Learning (Problem 2.2.1), the KEM presented in Section 3.2.2 is IND-CPA secure in the Random Oracle Model (ROM).*

Proof. We proceed in a sequence of games. The simulator starts from the real scheme. First we replace the public key by a random code instead of a distorted Gabidulin code, and then we use the ROM to solve the Rank Support Learning problem.

- We start with the game G_0: in this game we generate $\boldsymbol{H}$, $\boldsymbol{H}'$, $\boldsymbol{E}$ and $\boldsymbol{C}$ honestly.
- In game G_1 we replace $\boldsymbol{H}'$ by a parity-check matrix of a random $[n, k]$ code. From an adversary point of view, everything is identical, except the distribution on $\boldsymbol{H}'$ which is either generated at random or from a distorted Gabidulin code. Distinguishing between the two is an instance of Gab-Ind$_{q,m,n,k,\lambda}$ (see Problem 3.2.1), hence

$$\mathsf{Adv}_{\mathcal{A}}^{G_0} \leq \mathsf{Adv}_{\mathcal{A}}^{G_1} + \mathsf{Adv}_{\mathcal{A}}^{\text{Gab-Ind}}.$$

- In game G_2 we now replace $\mathcal{H}(E)$ by a random value r. By monitoring the calls the adversary makes to the random oracle, we can prove that the difference between G_1 and G_2 is solving the DRSL problem:

$$\mathsf{Adv}_{\mathcal{A}}^{G_1} \leq \mathsf{Adv}_{\mathcal{A}}^{G_2} + \mathsf{Adv}_{\mathcal{A}}^{\text{DRSL}}.$$

  In game G_2 everything is sampled independently from the secret values, which leads to the conclusion.

□

3.2.5.3 Known attacks

Attacks against RSD and RSL. The reader is referred to Chapter 2.

Attacks against the masking of Gabidulin codes. One of the key-points in the security reduction presented in Section 3.2.5 is the complexity of distinguishing the public-key pk, a.k.a $\boldsymbol{G}'$ in Algorithm 3.3 from a randomly generated $[n, k]$ matrix over $\mathbb{F}_{q^m}$. This precise problem was addressed in the paper [Loi22].

To sum up the results, there are two ways to investigate the problem:

- If $\lambda(n-k) < n$, there exists a polynomial-time distinguisher, see [CC20]. Moreover, a decryption algorithm can be recovered in polynomial-time for $\lambda = 2, 3$, see [CC20, Gha22] and exponential time for $\lambda > 4$, but with a complexity much less than expected to be suitable for encryption purposes [LP21]. Since in our parameter sets, the rate k/n is $1/2$ and $\lambda \geq 3$, we are not in that case.
- If $\lambda(n-k) \geq n$, then the best distinguisher to date is the one published in [BL23]. The exponential part corresponds to the enumeration of some constrained vector spaces and the polynomial term consists of the use of Wiedemann's algorithm. This gives

$$\mathcal{W}_{\text{Mask}} \geq m^3 n^5 R^3 (1+R) q^{m(\lambda-1)-\lambda n R(1-R)},$$

  where $R = k/n$ is the rate of the code.

Avoiding the Metzner-Kapturowski approach. The algorithm in [RPWZ21] is an adaptation to the rank metric of the Metzner-Kapturowski approach [MK90] and constitutes a polynomial-time algorithm for decoding arbitrary linear interleaved codes of high-interleaving order.

As said earlier, the decoding algorithm works when the interleaved order satistifies $\ell \geq t$ where t is the *vertical* rank norm of the error. We thus need to study the *vertical* rank norm of our error matrix $\boldsymbol{E}$ (which is of *horizontal* norm r) and show that it is larger than ℓ with great probability.

Proposition 3.2.3. *Let $E \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, r)$ and $\boldsymbol{E} \xleftarrow{\$} E^{\ell \times n}$. Let t be the vertical rank norm of $\boldsymbol{E}$. We have*

$$\mathbb{P}(t \leq \ell) < q^{\ell^2 r + n\ell(1-r)}.$$

Proof of Proposition 3.2.3. Let $(e_1, ..., e_r)$ be a basis of E. We can complete it into a basis $\gamma := (e_1, ..., e_r, x_1, ..., x_{m-r})$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. We will use γ to calculate the vertical rank norm, since it does not depend on the choice of the basis.

It is clear that when one picks at random a matrix $\boldsymbol{E} \xleftarrow{\$} E^{\ell \times n}$, then $ext_\gamma(\boldsymbol{E})$ writes as follows:

$$ext_\gamma(\boldsymbol{E}) = \begin{pmatrix} \boldsymbol{A}_1 \\ \hline \mathbf{0} \\ \hline \dots \\ \hline \boldsymbol{A}_\ell \\ \hline \mathbf{0} \end{pmatrix},$$

with $\boldsymbol{A}_i \xleftarrow{\$} \mathbb{F}_q^{r \times n}$ being the unfoldings of the ℓ rows of $\boldsymbol{E}$ in the basis of E and the $\mathbf{0}$ blocks being of size $(m-r) \times n$.

The probability distribution of the rank of $ext_\gamma(\boldsymbol{E})$ is therefore identical to the distribution of the rank of a matrix $\boldsymbol{A} \xleftarrow{\$} \mathbb{F}_q^{\ell r \times n}$.

Finally we conclude with Lemma B.1.1 applied with parameters $i = \ell$ and $m = \ell r$. □

For all parameter sets presented in Section 3.2.6, the probability obtained with Proposition 3.2.3 is less than 2^{-1000}. We can consider that the threat of the Metzner-Kapturowski approach is avoided by design, and we do not need to take additional precautions when sampling the error matrix $\boldsymbol{E}$.

3.2.6 Parameters

We give six sets of parameters (see Table 3.4): two sets for each security level in $\eta \in \{128, 192, 256\}$ bits. For each security level, we give an efficient parameter set with a smaller value of λ and a conservative parameter set with a higher value of λ.

The parameters are chosen following these steps in order:

- q is always equal to 2;
- the parameter r is chosen in a way to avoid RSD and RSL attacks. We need $r = 7$ for 128-bit security, $r = 8$ for 192-bit security and $r = 9$ for 256-bit security;
- the parameters n and k are chosen such that $k = n/2$ and $n - k$ is slightly larger than $r\lambda$, so as to respect the condition $r\lambda \leqslant \lfloor \frac{\ell}{\ell+1}(n-k) \rfloor$ with a reasonably small ℓ;
- m is set as the next prime after n;
- if needed, m and n are increased in order to have a complexity large enough for MaxMinors (algebraic attack from [BBC+20]) and $\mathcal{W}_{\text{Mask}}$. We always keep $k = n/2$ and m prime[2] larger than n;
- finally, parameter ℓ is chosen large enough so that the DFR is at most $2^{-\eta}$.

The sizes of the proposed parameters are expressed in kilobytes. The public key is an $(n-k) \times n$ parity-check matrix with entries in $\mathbb{F}_{q^m}$ given in systematic form, therefore

$$pk \text{ size} = \log_2(q) mk(n-k) \text{ bits}.$$

The ciphertext consists of ℓ syndromes of $n - k$ entries in $\mathbb{F}_{q^m}$ each, therefore

$$ct \text{ size} = \log_2(q) m\ell(n-k) \text{ bits}.$$

For the DFR, MaxMinors and $\mathcal{W}_{\text{Mask}}$ columns, we chose to put the base 2 logarithm.

[2] We traditionally choose m prime to avoid any potential attacks.

Security level 128											
q	m	n	k	λ	r	ℓ	pk size	ct size	DFR	MaxMinors	$\mathcal{W}_{\text{Mask}}$
2	61	50	25	3	7	6	4.77KB	1.14KB	-242	139	131
2	67	66	33	4	7	6	9.12KB	1.66KB	-199	155	183
Security level 192											
q	m	n	k	λ	r	ℓ	pk size	ct size	DFR	MaxMinors	$\mathcal{W}_{\text{Mask}}$
2	101	74	37	3	8	2	17.28KB	0.93KB	-301	193	197
2	79	78	39	4	8	5	15.02KB	1.93KB	-314	218	209
Security level 256											
q	m	n	k	λ	r	ℓ	pk size	ct size	DFR	MaxMinors	$\mathcal{W}_{\text{Mask}}$
2	101	88	44	4	9	5	24.44KB	2.78KB	-503	278	267
2	107	106	53	5	9	6	37.57KB	4.25KB	-426	313	349

Table 3.4: Parameters for LowMS

Comparison with other KEMs We compare our cryptosystem to other GPT-based KEMs, as well as to unstructured proposals, either lattice-based or code-based. Our comparison metric is the usual TLS-oriented communication size (public key + ciphertext). Although our scheme is only proven IND-CPA at this stage, we believe that, since our DFR is negligible, it can be turned to an IND-CCA scheme using the Fujisaki-Osamoto transform [FO99]. Indeed, when applying the HHK framework [HHK17], similarly to [ABD+19, §5.3.2], the difference of advantages between CPA and CCA adversaries is explained by a term being equal to the product of the number of queries to the random oracle, by the probability of generating an decipherable ciphertext in an honest execution. With a negligible DFR, the advantages are thus similar. This comes at the cost of adding only two 64-byte hashes to the ciphertext and would only be a negligible increase, hence we took the liberty to compare our work with other IND-CCA parameters.

For the original Loidreau cryptosystem, we consider the parameters presented in the conclusion of [Pha21] which take into account the recent improvements on algebraic attacks. For this cryptosystem, parameters were not available (N/A) for 192 bits of security.

Instance	128 bits	192 bits
LowMS (λ = 3)	5.76KB	14.97KB
LowMS (λ = 4)	10.78KB	16.95KB
DRANKULA [AAB+18]	28.8KB	N/A
Interleaved Loidreau [RPWZ19]	33.35KB	N/A
Original Loidreau [Loi17]	36.30KB	N/A

Table 3.5: Comparison of sizes of other GPT-based KEMs. The sizes represent the sum of the public key and the ciphertext expressed in bytes.

3.3 Improving key generation in LRPC code-based cryptosystems

Notation In this section, we work with $\mathbb{F}_{2^m}$ a field of order $q = 2^m$. Let us denote $\mathbb{F}$ the field $\mathbb{F}_{2^m}[X]/(Q)$. Recall that Q a monic irreducible polynomial in $\mathbb{F}_{2^m}[X]$ of degree n.

Instance	128 bits	192 bits
LowMS (λ = 3)	5.76KB	14.97KB
NH-Multi-UR-AG [BBBG22]	7.12KB	12.60KB
LRPC-MS (Section 3.1)	7.21KB	14.27KB
LowMS (λ = 4)	10.78KB	16.95KB
Multi-UR-AG [BBBG22]	11.03KB	21.08KB
FrodoKEM [NAB+20]	19.34KB	31.38KB
Classic McEliece [ABC+22]	261KB	524KB

Table 3.6: Comparison of sizes of unstructured post-quantum KEMs. The sizes represent the sum of public key and ciphertext expressed in bytes.

3.3.1 The bottleneck in LRPC schemes key generation

The key generation phase of an LRPC-based cryptosystem requires a polynomial inversion which is the most time-consuming step.

As an example, the key generation phase in ROLLO can be written as follows:

Algorithm 3.6 Key generation in ROLLO-I and ROLLO-II

1: $(x, y) \xleftarrow{\$} S_d^{2n}(\mathbb{F}_{2^m})$
2: $h \leftarrow x^{-1}y \bmod Q$
3: $\begin{cases} pk &= h \\ sk &= (x, y) \end{cases}$

Some existing implementations, such as `rbc-lib` [ABB+21], use Euclid's algorithm to perform the inversion, which is not constant time and may leak secret information when exposed to side-channel attacks. Even if the inversion is calculated only once, the existence of attacks, such as Big Mac [Wal01], where a single trace is necessary, stresses the importance of doing constant time algoirthms even for key generation. Most cryptographic libraries implement constant-time algorithms as a countermeasure against such attacks.

The use of the Itoh-Tsuiji algorithm [IT88] as a constant-time alternative to Euclid's algorithm for polynomial inversion was recently suggested and implemented [AAB+21]. However, this leads to computational costs for key generation ten times greater than for the non-constant-time implementation. A variant of the Itoh-Tsuiji algorithm was adapted for the case of BIKE to invert a polynomial and lead to very efficient performance results [DGK20]. Unfortunately, such an adaptation is not possible for LRPC codes, as the quotient ring in which the inversion is performed is different. Indeed, in the case of BIKE, the inversion is done in a polynomial ring $\mathbb{F}_2[X]$ quotiented by $X^n - 1$. For LRPC, the polynomial ring is $\mathbb{F}_{2^m}[X]$ quotiented by an irreducible polynomial of degree n. The identity $X^n = 1$, leveraged by the authors of [DGK20], is thus not possible anymore.

3.3.2 Optimal normal basis

Our improvement involves the use of an optimal normal basis representation instead of a polynomial basis for the Itoh-Tsuiji algorithm. x will be represented as a vector over an optimal normal basis (ONB). This section defines the concept of optimal normal basis and shows how to build such a basis for field $\mathbb{F}$. We first recall results from prior papers (Proposition 3.3.1 and Theorem 3.3.1) that serve as building blocks for Proposition 3.3.2, which is an original adaptation to the specific case of $\mathbb{F}$.

3.3.2.1 Definition

Definition 3.3.1 (Normal basis). *An element $\alpha \in \mathbb{F}$ is a normal element over $\mathbb{F}_{2^m}$ when $(\alpha, \alpha^q, \alpha^{q^2}, ..., \alpha^{q^{n-1}})$ is a basis of $\mathbb{F}$ seen as a $\mathbb{F}_{2^m}$-vector space. The family $(\alpha, \alpha^q, \alpha^{q^2}, ..., \alpha^{q^{n-1}})$ is then*

called the normal basis generated by α.

Definition 3.3.2 (Complexity of a normal basis)**.** *For a normal basis* $N = (\alpha, \alpha^q, \alpha^{q^2}, ..., \alpha^{q^{n-1}})$ *there exists a matrix* $T = (t_{i,j})$ *of size* $n \times n$ *with coefficients in* $\mathbb{F}_q$ *such that for all* i *in* $[0..n-1]$*:*

$$\alpha\alpha^{q^i} = \sum_{j=0}^{n-1} t_{i,j}\alpha^{q^j}. \tag{3.1}$$

The number c_N *of non-zero entries in* T *is called the complexity of the normal basis* N.

The matrix T is called the multiplication table of the normal basis N as it is used to implement the multiplication of two elements of $\mathbb{F}$ represented as linear combinations of elements in the normal basis N. The complexity c_N thus represents the complexity of a circuit implementing such a multiplication.

Proposition 3.3.1 (Theorem 1.2.1 in [Gao93])**.** *For a normal basis* N*, the complexity* $c_N \geq 2n-1$.

Definition 3.3.3 (Optimal normal basis)**.** *An optimal normal basis* N *is a normal basis whose complexity* c_N *is exactly* $2n-1$.

Lemma 3.3.1. *The multiplication matrix* T *of an optimal normal basis* N *has exactly one non-zero entry in the first column and exactly two non-zero entries for each other column.*

Proof : Let $b = \sum_{i=0}^{n-1} \alpha^{q^i}$. Note that $b^q = b$ hence $b \in \mathbb{F}_q$. By summing up the equations (3.1) and comparing the coefficient of α^{q^j} we find for all $j \neq 0$

$$\sum_{i=0}^{n-1} t_{i,j} = 0.$$

Since α is non-zero and $\{\alpha\alpha^{q^i} : 0 \leq i \leq n-1\}$ is also a basis of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$, the matrix T is invertible. Thus for each j there is at least one non-zero $t_{i,j}$. The result of the previous paragraph implies that for each $j \neq 0$ there is at least two non-zero $t_{i,j}$. Because the total number of non-zero entries in T is $2n-1$, it has exactly one non-zero entry in the first column and exactly two non-zero entries for each other column. □

3.3.2.2 Constructing optimal normal basis

For our algorithm to work, we need to build an optimal normal basis of $\mathbb{F}$ over $\mathbb{F}_{2^m}$. We first study a simpler problem: the following theorem shows how to build an optimal normal basis of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$ when some conditions on n are met.

Theorem 3.3.1 (Theorem 4.2.1 in [Gao93])**.** *Let* $2n+1$ *be a prime and assume that either*

1. 2 *is primitive in* $\mathbb{F}_{2n+1}$*, or*
2. $2n+1 = 3 \pmod 4$ *and* 2 *generates the quadratic residues in* $\mathbb{F}_{2n+1}$.

Then $\alpha = \gamma + \gamma^{-1}$ *generates an optimal normal basis of* $\mathbb{F}_{2^n}$ *over* $\mathbb{F}_2$*, where* γ *is a primitive* $(2n+1)$*-th root of unity.*

The following proposition shows that it is easy to build an optimal normal basis of $\mathbb{F}$ when an optimal normal basis of $\mathbb{F}_{2^n}$ is known.

Proposition 3.3.2. *Let α an optimal normal element in $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$ and suppose m is prime to n. Then α is also an optimal normal element of $\mathbb{F}$ over $\mathbb{F}_{2^m}$.*

Proof: Let $N = (\alpha, \alpha^2, \alpha^{2^2}, ..., \alpha^{2^{n-1}})$ the normal basis generated by α. According to Lemma 2.3.2 in [Gao93], N is also a basis of $\mathbb{F}$ over $\mathbb{F}_{2^m}$.

If we prove that $N' = (\alpha, \alpha^{2^m}, \alpha^{2^{2m}}, ..., \alpha^{2^{(n-1)m}})$ is a permutation of N, then N' will be a normal basis of $\mathbb{F}$ over $\mathbb{F}_{2^m}$.

To prove this, note that since m is prime to n, the application

$$i \mapsto mi \bmod n$$

is obviously a permutation of $\mathbb{Z}/n\mathbb{Z}$. As a result, the sets $\{\alpha^{2^i}, i \in 0..n-1\}$ and $\{\alpha^{2^{mi \bmod n}}, i \in 0..n-1\}$ are equal.

Since $\alpha \in \mathbb{F}_{2^n}$, $\alpha^{2^n} = 1$. Then for all i we have $\alpha^{2^{mi \bmod n}} = \alpha^{2^{mi}}$.

This proves that the sets $\{\alpha^{2^i}, i \in 0..n-1\}$ and $\{\alpha^{2^{mi}}, i \in 0..n-1\}$ are equal hence N' is a permutation of N and N' is a normal basis.

We now need to prove the optimality of the normal basis N'. Looking at Definition 2, it is obvious that since N' is a permutation of N then its multiplication table is a permutation of the multiplication table of N. Hence N' and N have the same complexity. N' is thus an ONB of $\mathbb{F}$ over $\mathbb{F}_{2^m}$. □

We now know that it is possible to build an ONB of $\mathbb{F}$ when the conditions on n of Theorem 1 are met. It is also proven in [Gao93] that these conditions are necessary for the existence of an ONB. Therefore, it is possible to get an ONB of $\mathbb{F}$ only for some values of n. Table 3.7 shows which ROLLO schemes have a value for n that allows to build an ONB.

Scheme	n	ONB?
ROLLO-I-128	83	✓
ROLLO-I-192	97	✗
ROLLO-I-256	113	✓
ROLLO-II-128	189	✓
ROLLO-II-192	193	✗
ROLLO-II-256	211	✗

Table 3.7: Existence of an optimal normal basis depending on the value n for each ROLLO set of parameters

Below is presented a Sagemath algorithm to find an optimal normal element in $\mathbb{F}$.

```
from sage.coding.relative_finite_field_extension \
import *
import numpy

q = 83 #for the example
K.<a> = GF(2^q)
L.<b> = GF(2^(2*q))
FE = RelativeFiniteFieldExtension(L, K)
R.<X> = PolynomialRing(L)
P = X^(2*q+1) - 1
for x in P.roots():
if x[0].multiplicative_order() == 2*q+1:
y = x[0]
break
z = y + y^(-1)
s = FE.relative_field_representation(z)[0]
t = s.polynomial().list()
u = numpy.nonzero(t)[0].tolist()
print(u)
```

3.3.3 Operations in an optimal normal basis representation

In this section we present two operations in $\mathbb{F}$ using a normal basis representation. We suppose that n meets the conditions of Theorem 1 and that we have found an optimal normal basis of $\mathbb{F}$ over $\mathbb{F}_{2^m}$ generated by α.

3.3.3.1 Polynomial and normal basis representation

An element $x \in \mathbb{F}$ can be represented two usual ways. The first and natural way is the polynomial representation

$$x = \sum_{i=0}^{n-1} x_i X^i$$

with x_i in $\mathbb{F}_{2^m}$ We write $[x]_P = (x_0, ..., x_{n-1})$.

The second way is the normal representation, which is the entries of x according to the normal basis.

$$x = \sum_{i=0}^{n-1} \xi_i \alpha^{q^i}$$

We write $[x]_N = (\xi_0, ..., \xi_{n-1})$.

x can be converted from polynomial to normal representation, and vice-versa, using an invertible matrix $\boldsymbol{M}$.

$$[x]_N = [x]_P \, \boldsymbol{M}$$

3.3.3.2 q-powers

In a normal basis, raising an element of $\mathbb{F}$ to a power q^k is easy. The following result holds for any normal basis, regardless of its optimality.

Proposition 3.3.3. *Forall $k \in \mathbb{N}$, $[x^{q^k}]_N = rot_k([x]_N)$ where rot_k is k circular right shifts.*

Proof : Writing in a normal representation $x = \sum_{i=0}^{n-1} \xi_i \alpha^{q^i}$, we have according to Froebinius endomorphism proprety:

$$x^{q^k} = \sum_{i=0}^{n-1} \xi_i^{q^k} \alpha^{q^{i+k}}$$

$\xi_i \in \mathbb{F}_q$ implies $\xi_i^{q^k} = \xi_i$ and $\alpha \in \mathbb{F}$ implies $\alpha^{q^n} = 1$, hence:

$$x^{q^k} = \sum_{i=0}^{n-1} \xi_{i+k \bmod n} \, \alpha^{q^i}$$

which proves the result. □

We thus see that the q-power operation in a normal basis is almost free ($O(n)$ memory writes).

Algorithm 3.7 **`q_power`**(x, k)

Input: $x = (x_0, ..., x_{n-1}) \in \mathbb{F}; k \in \mathbb{N}$
Output: $y = x^{q^k}$

1: **for** i in $0..n-1$ **do**
2: $\quad y_i = x_{i+k \, mod \, n}$

3.3.3.3 Multiplication

The optimality of the normal basis generated by α is necessary to reduce the complexity of the multiplication operation.

Proposition 3.3.4. *Let $x, y \in \mathbb{F}$ and $z = xy$. We denote*

$$[x]_N = (x_0, ..., x_{n-1})$$
$$[y]_N = (y_0, ..., y_{n-1})$$
$$[z]_N = (z_0, ..., z_{n-1})$$

There exist two permutations σ_1 and σ_2 of $[0..n-1]$ such that for all k:

$$z_k = x_k y_{\sigma_1(0)+k} + \sum_{i=1}^{n-1} x_{i+k}(y_{\sigma_1(i)+k} + y_{\sigma_2(i)+k})$$

where all subscripts are taken modulo n. Permutations σ_1 and σ_2 are independent of the choice of x and y.

Proof : This proof is adapted from Section 2 in [NY01].
First observe that for all i, j:

$$\begin{aligned}\alpha^{q^i}\alpha^{q^j} &= \left(\alpha\alpha^{q^{j-i}}\right)^{q^i} \\ &= \sum_{k=0}^{n-1} t_{j-i,k}\,\alpha^{q^{i+k}} \\ &= \sum_{k=0}^{n-1} t_{j-i,k-i}\,\alpha^{q^k}\end{aligned}$$

where all subscripts and all exponents of q are taken modulo n.
As a result,

$$z = xy = \sum_{i,j=0}^{n-1} x_i\, y_j\, \alpha^{q^i}\alpha^{q^j}$$

rewrites for all k:

$$z_k = \sum_{i,j=0}^{n-1} x_i\, y_j\, t_{j-i,k-i}$$

and by changing variables:

$$z_k = \sum_{i,j=0}^{n-1} x_{i+k}\, y_{j+k}\, t_{j-i,-i}$$

By Lemma 1, for each $i \neq 0$, there are only two values $j = \sigma_1(i)$ and $j = \sigma_2(i)$ for which $t_{j-i,-i} = 1$. We also note $\sigma_1(0)$ the index of the single non-zero entry of the first column of T.
This yields the final formula:

$$z_k = x_k y_{\sigma_1(0)+k} + \sum_{i=1}^{n-1} x_{i+k}(y_{\sigma_1(i)+k} + y_{\sigma_2(i)+k})$$

with σ_1 and σ_2 depending only of T and not of x or y. □

We thus have a quadratic multiplication in a normal basis.

Algorithm 3.8 **mul_norm**(x, y)

Input: $x = (x_0, ..., x_{n-1}) \in \mathbb{F}; y = (y_0, ..., y_{n-1}) \in \mathbb{F}; k \in \mathbb{N}$
Output: $z = xy$

1: **for** i in $0..n-1$ **do**
2: $\quad z_i = x_i y_{i+\sigma_1(0) \bmod n}$
3: **for** j in $1..n-1$ **do**
4: $\quad$ **for** i in $0..n-1$ **do**
5: $\quad\quad z_i = z_i + x_{i+k \bmod n}(y_{k+\sigma_1(i) \bmod n} + y_{k+\sigma_2(i) \bmod n})$

3.3.4 Itoh-Tsuiji algorithm for polynomial inversion

Recall that $q = 2^m$. We can now present the algorithm for inversion in $\mathbb{F}$. The method is inspired from Itoh-Tsuiji [IT88]. The idea is to compute $x^{-1} = (x^r)^{-1}x^{r-1}$ with $r = \frac{q^n-1}{q-1}$. It is easy to prove that $x^r \in \mathbb{F}_q$. This reduces the inversion in $\mathbb{F}$ to the computation of x^{r-1} , plus one inversion in the base field $\mathbb{F}_q$ and n multiplications in $\mathbb{F}_q$.

3.3.4.1 Helper function

We use the following helper function which computes x^e with $e = \frac{q^t-1}{q-1}$ for any odd integer t. Our helper function is inspired from [DGK20].

Algorithm 3.9 **helper_exp**(x, t)

Input: $x = (x_0, ..., x_{n-1}) \in \mathbb{F}; t$ an odd integer
Output: $z = x^e$ with $e = \frac{q^t-1}{q-1}$

1: $y = z = x$
2: **for** i in $1..log(t)$ **do**
3: $\quad tmp =$**q_power**$(y, 2^{i-1})$
4: $\quad y =$**mul_norm**(tmp, y)
5: $\quad$ **if** $t[i] == 1$ **then**
6: $\quad\quad z =$**mul_norm**$(z,$ **q_power**$(y, 2^{t \bmod 2^i}))$

3.3.4.2 Computing x^{r-1}

To finalize the inversion we need to compute x^{r-1} with $r - 1 = \frac{q^n-1}{q-1} - 1 = q(\frac{q^{n-1}-1}{q-1})$.

If $n-1$ is odd, it is trivial: we use the helper function and raise the result to the power q. However for ROLLO parameters, $n-1$ is not odd, thus we must use the following technique. First we apply the helper function to $t = \frac{n-1}{2^k}$ (k being the smallest integer s.t. $\frac{n-1}{2^k}$ is odd). Then we use q-powers and multiplication to compute a^{r-1}.

Algorithm 3.10 Compute x^{r-1}

Input: $x = (x_0, ..., x_{n-1}) \in \mathbb{F}$
Output: $z = x^{r-1}$ with $r = \frac{q^n-1}{q-1}$

1: $k = min\{i$ integer s.t. $\frac{n-1}{2^i}$ odd$\}$
2: $z =$**helper_exp**$(x, \frac{n-1}{2^k})$
3: **for** i in $0..k-1$ **do**
4: $\quad y =$**q_power**$(z, 1)$
5: $\quad z =$**mul_norm**(y, z)

3.3.4.3 Final algorithm

The following and last algorithm (Algorithm 3.11) describes how to compute x^{-1}. The first three lines are dedicated to computing x^{r-1} through an intermediate representation in optimal normal form. Lines 4 and 5 ensure $t = (x^r)^{-1}$. Line 6 returns the final result.

Algorithm 3.11 Compute x^{-1}

Input: $x = (x_0, ..., x_{n-1}) \in \mathbb{F}$ in a polynomial representation
Output: $z = x^{-1}$ in a polynomial representation

1: $\xi = x\mathcal{P}$ [transform to normal representation]
2: $\gamma = \xi^{r-1}$ [using algorithm 4]
3: $y = \gamma\mathcal{P}^{-1}$ [transform to polynomial representation]
4: $t = xy$
5: $t = t^{-1}$ [inversion in $\mathbb{F}_q$]
6: $z = yt$

3.3.4.4 Constant-time property

In `helper_exp`, the conditional branches are on the value of t only. The only usage of that function is in Algorithm 5, where it is called with a value t depending only on n, which is a public parameter. Other conditional branchings in Algorithm 5 depend only on n, therefore computing x^{-1} with Itoh-Tsuiji algorithm is constant-time.

3.3.5 Experimental results

This section provides performance results and compares them to other polynomial inversion algorithms.

Measurements were carried out on a MacBook Pro laptop equipped with 2.7GHz Intel Core i5-5257U CPU and 8GB memory for which Hyper-Threading and Turbo Boost were disabled.

To benchmark our implementation of ROLLO, we used the SUPERCOP [BL] platform version `210114`. All primitives were compiled using `clang` with parameters `-march=native -mtune=native -O3 -fomit-frame-pointer -fwrapv -fPIC -fPIE`. We could test our algorithm only for parameters that allow the existence of an ONB (see table 3.7).

On the same machine, we compared our algorithms with the original version in `rbc-lib` (which is not constant time since it is using Euclid's algorithm) using the same SUPERCOP configuration.

On the same machine again, we also compared with the constant time algorithm in [AAB+21], using the testing platform provided by the authors[3]. For a fair comparison, we changed the compilation options to match those used in SUPERCOP. The results are really close to those reported in Table 7 in [AAB+21].

In all measurements, AVX instructions are used for binary field arithmetic.

Our results are summarized in table 3.8, where CT stands for "constant time" and NCT for "non constant time". The overhead with the NCT algorithm remains limited to a maximum of 5 whereas the speedup with the existing CT algorithm is almost 4.

[3] available at https://github.com/peacker/constant_time_rollo.

Keygen	NCT algorithm [ABB+21]	CT algorithm [AAB+21]	CT algorithm (our work)
ROLLO-I-128	1,030,500	12,959,125	3,514,016
ROLLO-I-256	1,702,620	N/A	5,785,700
ROLLO-II-128	4,295,704	N/A	22,859,614

Keygen	Speedup [AAB+21] / (our)	Overhead (our) / [ABB+21]
ROLLO-I-128	3.69	3.41
ROLLO-I-256	N/A	3.40
ROLLO-II-128	N/A	5.32

Table 3.8: Performance measurements

To understand which are the most time-consuming steps of our algorithm, we analyzed the weights of each line of pseudo-code. Measurements were made with the profiling tool `Callgrind` from the `Valgrind` suite.

	Description	**Cost**
Line 1	Transform from polynomial to normal representation	1.2%
Line 2	Elevate to power $r-1$	94.7%
Line 3	Transform from normal to polynomial representation	1.2%
Line 4	Multiplication in $\mathbb{F}$	2.8%
Line 5	Inversion in $\mathbb{F}_q$	$< 0.1\%$
Line 6	Scalar multiplication	$< 0.1\%$

Table 3.9: Relative computation cost of each line of Algorithm 6 for polynomial inversion in $\mathbb{F}$

Chapter 4

Design and cryptanalysis of signature schemes

Contents

4.1 Cryptanalysis of Durandal

We present a new attack against the PSSI problem, one of the three problems at the root of security of Durandal [ABG+19], a rank metric code-based signature scheme. Our attack recovers the private key using a leakage of information coming from several signatures produced with the same key. Section 4.1.1 contains preliminary lemmas on subspaces of $\mathbb{F}_{q^m}$. Background on the attacked scheme is given in Sections 4.1.2 and 4.1.3, which present Durandal and PSSI problem. For understanding the gist and the main steps of the attack, the reader should read Sections 4.1.4 and 4.1.5.1. The rest of Section 4.1.5 provides full details on the correctness and complexity of the attack. Finally, experimental results supporting our attack are shown in Section 4.1.6.

4.1.1 Preliminaries

4.1.1.1 Dimension of an intersection of subspaces

In this subsection, we prove some lemmas on the probability distribution of the dimension of an intersection of two or more random subspaces of $\mathbb{F}_{q^m}$. These lemmas will be useful for a fine analysis of our attack.

Lemma 4.1.1. *Let $x \in \mathbb{F}_{q^m} \backslash \{0\}$. Let $B \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, b)$ be a random subspace of dimension b. Then*

$$\mathbb{P}(x \in B) = \frac{q^b - 1}{q^m - 1}.$$

Proof. The set of subspaces of $\mathbb{F}_{q^m}$ of dimension b containing x is in bijection with the set of subspaces of the projective hyperplane $\mathbb{F}_{q^m}/\langle x \rangle$ of dimension $b-1$. $\mathbb{F}_{q^m}/\langle x \rangle$ is an $\mathbb{F}_q$-vector space of dimension $m-1$, hence the number of subspaces of $\mathbb{F}_{q^m}$ of dimension b containing x is $\binom{m-1}{b-1}_q$.

Then we divide this number by the total number $\binom{m}{b}_q$ of subspaces of $\mathbb{F}_{q^m}$ of dimension b, to get the desired probability:

$$\begin{aligned}
\mathbb{P}(x \in B) &= \frac{\binom{m-1}{b-1}_q}{\binom{m}{b}_q} \\
&= \prod_{i=0}^{b-2} \frac{q^{m-1} - q^i}{q^{b-1} - q^i} \prod_{i=0}^{b-1} \frac{q^b - q^i}{q^m - q^i} \\
&= \frac{q^b - 1}{q^m - 1}.
\end{aligned}$$

□

Lemma 4.1.2. *Let $A \in \mathbf{Gr}(\mathbb{F}_{q^m}, a)$ and $B \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, b)$ be subspaces of $\mathbb{F}_{q^m}$. Then*

$$\mathbb{P}(\dim(A \cap B) > 0) \leq q^{a+b-m}.$$

Proof. Notice that

$$\dim(A \cap B) > 0 \Leftrightarrow \exists x \in A \backslash \{0\}, x \in B,$$

hence:

$$\begin{aligned}
\mathbb{P}(\dim(A \cap B) > 0) &= \mathbb{P}\Big(\bigvee_{x \in A \setminus \{0\}} x \in B \Big) \\
&\leq \sum_{x \in A \setminus \{0\}} \mathbb{P}(x \in B) \\
&= \sum_{x \in A \setminus \{0\}} \frac{q^b - 1}{q^m - 1} \qquad \text{(Lemma 4.1.1)} \\
&\leq \sum_{x \in A \setminus \{0\}} q^{b-m} \\
&= (q^a - 1) q^{b-m} \\
&\leq q^{a+b-m}.
\end{aligned}$$

□

Remark. When $a+b > m$, $\dim(A \cap B)$ is always greater than 0 according to Grassmann's formula on dimensions. Note that the above lemma still holds in that case, since the right-hand side of the equality is larger than 1.

We can generalize this lemma to an arbitrary family of independent random subspaces.

Lemma 4.1.3. *For $1 \leq i \leq n$, let $A_i \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, a_i)$ be uniformly random independent (in the sense of probability) subspaces of $\mathbb{F}_{q^m}$. Then*

$$\mathbb{P}(\dim(\cap_i A_i) > 0) \leq q^{\sum_i a_i - (n-1)m}.$$

Proof. As before, $\dim(\cap_i A_i) > 0$ if and only if there exists $x \neq 0$ such that $x \in \cap_i A_i$, hence:

$$\begin{aligned}
\mathbb{P}(\dim(\cap_i A_i) > 0) &= \mathbb{P}\Big(\bigvee_{x \in \mathbb{F}_{q^m} \setminus \{0\}} x \in \cap_i A_i \Big) \\
&= \mathbb{P}\Big(\bigvee_{x \in \mathbb{F}_{q^m} \setminus \{0\}} \big(\bigwedge_{i=1}^{n} x \in A_i \big) \Big) \\
&\leq \sum_{x \in \mathbb{F}_{q^m} \setminus \{0\}} \mathbb{P}\big(\bigwedge_{i=1}^{n} x \in A_i \big) \\
&= \sum_{x \in \mathbb{F}_{q^m} \setminus \{0\}} \prod_{i=1}^{n} \mathbb{P}(x \in A_i) \qquad \text{(by independency of spaces } A_i\text{)} \\
&= \sum_{x \in \mathbb{F}_{q^m} \setminus \{0\}} \prod_{i=1}^{n} \frac{q^{a_i} - 1}{q^m - 1} \\
&\leq \sum_{x \in \mathbb{F}_{q^m} \setminus \{0\}} \prod_{i=1}^{n} q^{a_i - m} \\
&= \sum_{x \in \mathbb{F}_{q^m} \setminus \{0\}} q^{\sum_i a_i - nm} \\
&= (q^m - 1) q^{\sum_i a_i - nm} \\
&\leq q^{\sum_i a_i - (n-1)m}.
\end{aligned}$$

□

Remark. Similarly to the previous remark, when $\sum_i a_i > (n-1)m$, $\dim(\cap_i A_i) > 0$ and the lemma is still valid.

We now present a slight variation of the above lemma when the random subspaces A_i all share a common element x. Let us introduce the following notation:

Definition 4.1.1. *Let $U \in \mathbf{Gr}(\mathbb{F}_{q^m}, u)$ be a subspace of dimension u. For $a \geq u$, we define*

$$\mathbf{Gr}(\mathbb{F}_{q^m}, U, a) := \{A \in \mathbf{Gr}(\mathbb{F}_{q^m}, a) \,|\, U \subset A\},$$

the set of all subspaces of $\mathbb{F}_{q^m}$ of dimension a containing U.

$\mathbf{Gr}(\mathbb{F}_{q^m}, U, a)$ is in bijection with $\mathbf{Gr}(\mathbb{F}_{q^m}/U, a-u)$, hence is of cardinality $\binom{m-u}{a-u}_q$.

Lemma 4.1.4. *Let $x \in \mathbb{F}_{q^m} \backslash \{0\}$. For $1 \leq i \leq n$, let $A_i \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, \langle x \rangle, a_i)$ be random independent subspaces of $\mathbb{F}_{q^m}$ all containing x. Then, $\dim(\cap_i A_i) \geq 1$ and*

$$\mathbb{P}(\dim(\cap_i A_i) > 1) \leq q^{\sum_i a_i - (n-1)m - 1}.$$

Proof. Since the subspaces A_i all contain x, we have immediately $\dim(\cap_i A_i) \geq 1$. Next, we note that

$$\dim\left(\bigcap_i A_i\right) > 1 \Leftrightarrow \dim\left(\bigcap_i A_i/\langle x \rangle\right) > 0.$$

Therefore, we can apply Lemma 4.1.3 with the space $\mathbb{F}_{q^m}/\langle x \rangle$ (of dimension $m-1$) and subspaces $A_i' \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}/\langle x \rangle, a_i - 1)$, to get

$$\begin{aligned}\mathbb{P}(\dim(\cap_i A_i) > 1) &\leq q^{\sum_i (a_i - 1) - (n-1)(m-1)} \\ &= q^{\sum_i a_i - (n-1)m - 1}.\end{aligned}$$

□

4.1.1.2 Some results on product spaces

The following proposition states that it is easy to compute E from F and the product space (see Definition 0.0.2) EF when $\dim(EF) \ll m$. It is analogue to the division $\frac{ef}{f} = e$, but in a vector space setting. It will be necessary for a fine undersanding of the PSSI problem, and is also used extensively for the decoding of LRPC codes, a family of rank-metric codes not used in Durandal but found in other rank-metric cryptographic algorithms.

Proposition 4.1.1 ([AGH+19], Proposition 3.5)**.** *Suppose m is prime. Let $E \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, r)$ and $F \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, d)$. Let (f_i) be a basis of F. Then*

$$E = \bigcap_i f_i^{-1} EF$$

with probability at least

$$1 - rq^{r\frac{d(d+1)}{2} - m}.$$

Remark. The above result requires m to be prime, which is always the case for parameters of rank-based cryptographic primitives, including Durandal.

Remark. This proposition shows that it is possible to recover E with high probability when $rd \ll m$. In the other extreme case where $rd \geq m$ (i.e. $EF = \mathbb{F}_{q^m}$), we get $f_i^{-1} EF = \mathbb{F}_{q^m}$ so the chain of intersections will always be $\mathbb{F}_{q^m}$ and no information on E can be retrieved.

Definition 4.1.2 (Filtered subspace)**.** *Let E and F be two $\mathbb{F}_q$-subspaces of $\mathbb{F}_{q^m}$. A strict subspace $U \subsetneq EF$ of the product space EF is said to be filtered when it contains no non-zero product elements of the form ef with $e \in E$ and $f \in F$:*

$$\{ef, e \in E, f \in F\} \cap U = \{0\}.$$

Key generation: $E \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, r)$
Secret key: $\boldsymbol{S} \in E^{lk \times n}$, $\boldsymbol{S}' \in E^{l'k \times n}$
Public key: $\boldsymbol{H} \xleftarrow{\$}$ ideal matrix $\in \mathbb{F}_{q^m}^{(n-k)\times n}$, $\boldsymbol{T} = \boldsymbol{H}\boldsymbol{S}^T, \boldsymbol{T}' = \boldsymbol{H}\boldsymbol{S}'^T$

Sign($\mu, \boldsymbol{S}, \boldsymbol{S}'$):

1. $W \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, w)$, $F \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, d)$
2. $\boldsymbol{y} \xleftarrow{\$} (W + EF)^n$, $\boldsymbol{x} = \boldsymbol{H}\boldsymbol{y}^T$
3. $\boldsymbol{c} = \mathcal{H}(\boldsymbol{x}, F, \mu)$, $\boldsymbol{c} \in F^{l'k}$
4. $U \xleftarrow{\$}$ filtered subspace of EF of dimension $rd - \lambda$
5. $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{c}\boldsymbol{S}' + \boldsymbol{p}\boldsymbol{S}$, $\boldsymbol{z} \in (W + U)^n$
6. Output $(\boldsymbol{z}, F, \boldsymbol{c}, \boldsymbol{p})$

Verify($\mu, \boldsymbol{z}, F, \boldsymbol{c}, \boldsymbol{p}, \boldsymbol{H}, \boldsymbol{T}, \boldsymbol{T}'$):

1. Accept if and only if : $\|\boldsymbol{z}\| \leqslant w + rd - \lambda$ and $\mathcal{H}(\boldsymbol{H}\boldsymbol{z}^T - \boldsymbol{T}'\boldsymbol{c}^T + \boldsymbol{T}\boldsymbol{p}^T, F, \mu) = \boldsymbol{c}$

Figure 4.1: The Durandal Signature scheme

4.1.2 Durandal signature scheme

4.1.2.1 Description of the scheme

We briefly recap in Figure 4.1 and in the below paragraphs the Durandal signature scheme, although no deep understanding of the scheme is required for the rest of the article, since our attack targets more specifically the PSSI problem defined in the next section. The reader is referred to [ABG+19] for more details on Durandal. The scheme is parametrized with variables m, n, k, l, l', r, d, and λ. In Durandal, only half-rate codes are considered, therefore $n = 2k$.

Key generation. The secret key consists of two matrices $\boldsymbol{S} \in E^{lk \times n}$ and $\boldsymbol{S}' \in E^{l'k \times n}$. $\boldsymbol{S}$ and $\boldsymbol{S}'$ are composed of ideal blocks of size $k \times k$ and their entries belong to the same secret support $E \subset \mathbb{F}_{q^m}$ of dimension r.

The public key consists of a random $(n - k) \times n$ ideal matrix $\boldsymbol{H}$, together with the matrices $\boldsymbol{T} = \boldsymbol{H}\boldsymbol{S}^\top$ and $\boldsymbol{T}' = \boldsymbol{H}\boldsymbol{S}'^\top$.

Signature of a message μ. Similar to the Lyubashevsky approach, the signer first computes to a vector $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{c}\boldsymbol{S}'$, where $\boldsymbol{y}$ is a vector whose entries are sampled in a space $W + EF$ depending on the secret key and $\boldsymbol{c}$ is a challenge depending on the message μ.

However, in order to avoid an attack, the vector $\boldsymbol{z}$ must be corrected with a corrective term $\boldsymbol{p}\boldsymbol{S}$ such that $\mathbf{Supp}(\boldsymbol{z}) = W + U$, where U is a filtered subspace of the product space EF of dimension $rd - \lambda$. $\boldsymbol{p}$ is a vector with entries in F and is computed through linear algebra during the signing process.

The signature is the tuple $(\boldsymbol{z}, F, \boldsymbol{c}, \boldsymbol{p})$. The signature consists therefore of the challenge $\boldsymbol{c}$, computed through a hash function, together with the answer to this challenge.

Verification of a signature $(\mu, \boldsymbol{z}, F, \boldsymbol{c}, \boldsymbol{p})$. To verify the signature, we have to check the rank weight of $\boldsymbol{z}$ and that $\mathcal{H}(\boldsymbol{x}, F, \mu) = \boldsymbol{c}$. The vector $\boldsymbol{x}$ is recomputed using $\boldsymbol{z}, \boldsymbol{c}, \boldsymbol{p}$ and the public key.

4.1.2.2 Parameters

The parameters of Durandal, as presented in [ABG+19], are shown in Table 4.1.

	q	m	n	k	l	l'	d	r	w	λ	pk size	σ size	Security
Durandal-I	2	241	202	101	4	1	6	6	57	12	15,245	4,064	128
Durandal-II	2	263	226	113	4	1	7	7	56	14	18,606	5,019	128

Table 4.1: Parameters for Durandal. The sizes of public key (pk) and signature (σ) are in bytes.

4.1.3 PSSI problem

The security of the Durandal signature scheme relies on the hardness of several problems: I-RSL, ARSD and PSSI. (see Theorem 20 in [ABG+19]).

While the first two problems are slight variants of the well-known *syndrome decoding problem in the rank metric* (RSD) and are widely used among rank-based cryptographic primitives, the PSSI is an ad-hoc problem that was also introduced in Durandal paper [ABG+19]. This latter problem will be our main focus for the rest of the article.

The PSSI problem appears naturally when trying to prove the indistinguishability of the signatures. Remember that we wrote in the previous section that the first two components of a Durandal signature are a subspace $F \in \mathbf{Gr}(\mathbb{F}_{q^m}, d)$ and a vector $\boldsymbol{z}$ whose entries belong to the subspace $Z = W + U$, where U is a filtered subspace of EF (see Definition 4.1.2). When a signer signs N times with the same key, it produces several subspaces $(F_i, Z_i)_{1 \leq i \leq N}$, the space E being fixed since it is linked to the private key. It is natural to require that pairs of such subspaces (F_i, Z_i) are indistinguishable from random subspaces of the same dimension. This is captured by the following definition:

Problem 4.1.1 (Product Spaces Subspaces Indistinguishability). *Let E be a fixed $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$ of dimension r. Let F_i, U_i and W_i be subspaces defined as follows:*

- $F_i \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, d)$;
- $U_i \xleftarrow{\$} \mathbf{Gr}(EF_i, rd - \lambda)$ *such that* $\{ef, e \in E, f \in F_i\} \cap U_i = \{0\}$ *(i.e. U_i is a filtered subspace of EF_i);*
- $W_i \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, w)$.

The $\text{PSSI}_{r,d,\lambda,w,m,N}$ problem consists in distinguishing N samples of the form (Z_i, F_i) where $Z_i = W_i + U_i$, from N samples of the form (Z'_i, F_i) where Z'_i is a random subspace of $\mathbb{F}_{q^m}$ of dimension $w + rd - \lambda$.[1]

Remark. An easy distinguisher could be to guess randomly unless $\dim(Z_i) < w + rd - \lambda$, in which case Z_i is bound to be of the first form $W_i + U_i$ described above. However, this can occur only if spaces U_i and W_i have a non-zero intersection, which happens with a probability dominated by $q^{w+rd-\lambda-m}$ (cf. Lemma 4.1.2). As a result, with practical parameters of Durandal presented in Table 4.1, this easy distinguisher gets a negligible advantage of less than 2^{-128}. Therefore, in the rest of this document, we consider the intersection $W_i \cap U_i$ to be trivial.

We define more precisely the two distributions between which a PSSI attacker must discriminate.

Definition 4.1.3 (PSSI distribution $\mathcal{D}_{PSSI}$). *Let E be a $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$ of dimension r. Let $\mathcal{D}_{PSSI}(E)$ be the distribution that outputs samples (F_i, Z_i) defined as follows:*

- $F_i \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, d)$;

[1] In the original paper of Durandal, the first component of the samples are vectors $\boldsymbol{z}_i$ of length n and support Z_i but this has been proven equivalent to the version defined in this paper (see the beginning of Section 4.1 in [ABG+19]).

- $U_i \xleftarrow{\$} \mathbf{Gr}(EF_i, rd - \lambda)$ *such that* $\{ef, e \in E, f \in F\} \cap U_i = \{0\}$;
- $W_i \xleftarrow{\$} \mathbf{Gr}(\mathbb{F}_{q^m}, w)$;
- $Z_i = W_i + U_i$.

Definition 4.1.4 (Random distribution $\mathcal{D}_{Random}$)**.** *Let $\mathcal{D}_{Random}$ the distribution that outputs samples (F_i, Z_i) where F_i and Z_i are independent random variables picked uniformly in, respecively, $\mathbf{Gr}(\mathbb{F}_{q^m}, d)$ and $\mathbf{Gr}(\mathbb{F}_{q^m}, w + rd - \lambda)$.*

The problem PSSI now simply consists in distinguishing N independent samples from the PSSI distribution or from the random distribution.

We can now define the search version of this problem which will be attacked in the next sections. It is obviously harder than PSSI.

Problem 4.1.2 (Search-PSSI)**.** *Given N independent samples (F_i, Z_i) from $\mathcal{D}_{PSSI}(E)$ with $\dim(E) = r$, the Search-PSSI$_{r,d,\lambda,w,m,N}$ problem consists in finding the vector space E.*

Why filtering U?

There exists a simple attack on Search-PSSI in the case where U is equal to the entire space EF and is not a strict subspace of it. In such a problematic setting, we can use similar arguments to Proposition 4.1.1 to recover E from the knowledge of $W + EF$ and F.

The filtration condition is a stronger constraint than having U being a strict subspace of EF. The objective is to avoid an attacker gaining information from intersections I of the form $f^{-1}Z \cap f'^{-1}Z$ with $(f, f') \in F^2$. If Z contains some product elements ef then the probability that $\dim I \neq 0$ is much higher than if Z was truly random. With the filtration of the space U, such techniques would not be useful.

Recovering the private key from Search-PSSI

Assume you can solve Search-PSSI. Then you can compute E, and from the public key $(\boldsymbol{H}, \boldsymbol{T}, \boldsymbol{T}')$ and the space E, it is easy to recover the private key $(\boldsymbol{S}, \boldsymbol{S}')$. Indeed, the equation $\boldsymbol{T} = \boldsymbol{H}\boldsymbol{S}^{\top}$ with coefficients in $\mathbb{F}_{q^m}$ can be rewritten as linear systems in $\mathbb{F}_q$. The number of equations $m(n-k)lk$ is way larger than the number of unknowns $rnlk$, so with overwhelming probability the private key will be the unique solution.

Existing attacks on PSSI

A security analysis of PSSI was presented in Durandal paper (see Section 4.1 in [ABG+19]). The analysis relied on a distinguisher, whose idea is to consider product spaces of the form Z_iG_i where G_i is a subspace of F_i of dimension 2. The probability that $\dim Z_iG_i = 2(w + rd - \lambda)$ depends on whether Z_i is random or from the PSSI distribution. The claimed work factor of this distinguisher was

$$2^{m-2(rd-\lambda)}$$

and the authors of Durandal chose their parameters such that this work factor is above the security level. Up to the present work, the above distinguisher was the state-of-the-art attack on PSSI and it seemed that a large value for m was enough to prevent an attacker from breaking PSSI. As we will see next, that is not the case, and on the contrary, the larger m is, the more attackable the parameters are.

4.1.4 An observation when m is high

Before unveiling a practical attack against PSSI, we first make an interesting observation which reveals the secret space E to an attacker who has no constraint on m. **Therefore, in this subsection, we place ourselves in the simplified situation where $2d(w+rd-\lambda) \ll m$.** Even though it is unrealistic as compared to practical parameters of the Durandal signature scheme, it gives a first glimpse of the ideas that will be used for a practical attack against PSSI in the next section.

The idea is the following: suppose an attacker has two instances from the PSSI distribution (F_1, Z_1) and (F_2, Z_2). They can compute a "cross-product" of these instances

$$A := F_1 Z_2 + F_2 Z_1.$$

Even though for $i \in \{1, 2\}$, Z_i contains a subspace U_i that is filtered and does not contain any product element ef_i with $e \in E$ and $f_i \in F_i$, nothing guarantees that A is not filtered, meaning it can contain product elements of the form eg with $e \in E$ and $g \in F_1F_2$. Indeed, we observed empirically with great probability that the entire product space $E(F_1F_2)$ is contained in A:

$$E(F_1F_2) \subset A.$$

The dimension of A is upper bounded by $2d(w+rd-\lambda)$ (which is by hypothesis greatly less than m) and since an attacker can compute very easily a basis of the vector space F_1F_2, one can use a chain of intersections, similar to Proposition 4.1.1, in order to recover E by computing

$$\bigcap_{g \in F_1F_2} g^{-1}A.$$

An informal explanation for why A contains some product elements lies in the fact that, even though Z_i contains no product elements, it contains "2-sums" of product elements of the form $ef_i + e'f_i'$ for $(e, e') \in E^2$ and $(f_i, f_i') \in F_i^2$.

More problematically, we will see in the next section that one can find 2-sums in both Z_1 and Z_2 for the *same pair* (e, e'), meaning that there exists $(e, e') \in E^2$, $(f_1, f_1') \in F_1^2$ and $(f_2, f_2') \in F_2^2$ such that

$$\begin{aligned} ef_1 + e'f_1' &= z_1 \in Z_1, \\ ef_2 + e'f_2' &= z_2 \in Z_2. \end{aligned}$$

Notice that, in that case, the cross product $f_1'z_2 - f_2'z_1$, which is an element of A, is also a product element because:

$$\begin{aligned} f_1'z_2 - f_2'z_1 &= ef_2f_1' + e'f_2'f_1' - ef_1f_2' - e'f_1'f_2' \\ &= e(f_2f_1' - f_1f_2'). \end{aligned}$$

This explains why A contains product elements. As said earlier, we observed furthermore that A contains all of them.

As said at the beginning of the section, computing A is only useful when m is high enough. With practical parameters of PSSI, m is much less than $2d(w+rd-\lambda)$, and the computation of $F_1Z_2 + F_2Z_1$ would only lead to $A = \mathbb{F}_{q^m}$. This does not give any information on E.

The next section overcomes this limitation on parameters; we refine the observation to give a practical attack against PSSI.

4.1.5 An attack against PSSI

Since the vector space $F_1Z_2 + F_2Z_1$ is too large for a practical attack, we turn our initial observation into a combinatorial attack where the attacker picks individual elements $f_1 \in F_1$ and $f_2 \in F_2$ and

computes spaces $f_1Z_2 + f_2Z_1$. If the attacker is lucky enough, they can obtain a product element eg with $e \in E$ and $g \in F_1F_2$. Since the vector spaces F_1 and F_2 are of small dimension d, the combinatorial factor in the attack is manageable.

4.1.5.1 General overview of the attack

Our combinatorial attack against PSSI consists in repeatedly applying Algorithm 4.1. The algorithm returns an element of $\mathbb{F}_{q^m}$, which is most of the time 0. We will show later on that, with a non negligible probability, it returns a non-zero element of $\mathbb{F}_{q^m}$ and in that case, this element belongs to the secret space E with overwhelming probability.

Algorithm 4.1 Attack against PSSI

Input: Four PSSI samples $(F_1, Z_1), (F_2, Z_2), (F_3, Z_3), (F_4, Z_4)$
Output: An element $x \in \mathbb{F}_{q^m}$

1: Choose $(f_1, f_1') \xleftarrow{\$} F_1^2$
2: Choose $(f_2, f_2') \xleftarrow{\$} F_2^2$
3: Choose $(f_3, f_3') \xleftarrow{\$} F_3^2$
4: Choose $(f_4, f_4') \xleftarrow{\$} F_4^2$
5: **for** $(i, j) \in [\![1, 4]\!]^2$ with $i < j$ **do**
6: **if** $\begin{vmatrix} f_i & f_i' \\ f_j & f_j' \end{vmatrix} = 0$ **then** go back to step 1
7: **else**
8: Compute

$$A_{i,j} := \frac{f_j' Z_i + f_i' Z_j}{\begin{vmatrix} f_i & f_i' \\ f_j & f_j' \end{vmatrix}}$$

9: Compute

$$B := \bigcap_{\substack{(i,j) \in [\![1,4]\!]^2 \\ i<j}} A_{i,j}$$

10: **if** $\dim(B) = 1$ **then**
11: **return** a non-zero element of B
12: **else**
13: **return** 0

The attacker starts by drawing random pairs in the subspaces F_i. If they are lucky, there exists a pair $(e, e') \in E^2$, such that a system $(\mathcal{S})$ of four conditions is verified:

$$(\mathcal{S}) : \begin{cases} ef_1 + e'f_1' = z_1 \in Z_1 \\ ef_2 + e'f_2' = z_2 \in Z_2 \\ ef_3 + e'f_3' = z_3 \in Z_3 \\ ef_4 + e'f_4' = z_4 \in Z_4 \end{cases}$$

Because the matrices $\begin{pmatrix} f_i & f_i' \\ f_j & f_j' \end{pmatrix}$ are chosen invertible (if not, the attacker retries with new random pairs), the element e can be recovered using Cramer's formula

$$e = \frac{\begin{vmatrix} z_i & f_i' \\ z_j & f_j' \end{vmatrix}}{\begin{vmatrix} f_i & f_i' \\ f_j & f_j' \end{vmatrix}}.$$

However, only the space Z_i is known to the attacker, not the exact element z_i. The attack thus consists in computing **a Cramer-like formula with vector spaces**, in order to get vector spaces containing e:

$$e \in A_{i,j} := \frac{\begin{vmatrix} Z_i & f_i' \\ Z_j & f_j' \end{vmatrix}}{\begin{vmatrix} f_i & f_i' \\ f_j & f_j' \end{vmatrix}}.$$

Finally, the attacker intersects all the spaces $A_{i,j}$. Two cases can then happen:

- If the attacker was lucky in the random sampling of (f_i, f_i'), there exists $(e, e') \in E^2$ such that conditions $(\mathcal{S})$ are verified, and then the intersection will be almost surely $\langle e \rangle$.
- In the other case, the intersection will be almost surely of dimension 0 and the attacker can retry with other samples.

The following subsections will be dedicated to proving our main result on the probability of success of the attack. It relies on an equality on parameters, which is verified for Durandal, as well as on an assumption which is discussed in the next subsection and is supported by simulations.

Theorem 4.1.1. *Under the equality $\lambda = 2r$ and under Assumption 1, the attack presented in Algorithm 4.1 outputs:*

- *0 with a probability $\geq 1 - \alpha$;*
- *an element $x \in E \backslash \{0\}$ with a probability $\geq \beta$;*
- *an element $x \in \mathbb{F}_{q^m} \backslash E$ with a probability $\leq \alpha - \beta$,*

with

$$\begin{cases} \alpha \approx q^{-6r} + q^{12(w+rd-\lambda)-5m} - q^{12(w+rd-\lambda)-5m-6r} \\ \beta \approx q^{-6r} - q^{12(w+rd-\lambda)-5m-6r-1} \end{cases}$$

Note that $\alpha - \beta$ is always greater than 0 and that for existing parameters of Durandal, both α and β weigh approximately q^{-6r}, therefore the probability that the third case happens is negligible in front of the chances of being in one of the two first cases.

Before delving into the details of the proof, we need technical results about the existence of 2-sums in a product space.

4.1.5.2 Technical results about 2-sums

For this subsection, let E be a subspace of $\mathbb{F}_{q^m}$ of dimension r and let $(F_1, Z_1), (F_2, Z_2), (F_3, Z_3), (F_4, Z_4)$ be four PSSI samples.

Definition 4.1.5. *For a fixed pair (e, e') of linearly independent elements in E, we define $X_{e,e',F,Z}$ the Bernouilli random variable*

$$\begin{aligned} X_{e,e',F,Z} : F^2 &\longrightarrow \{0,1\} \\ (f, f') &\longmapsto \mathbf{1}_Z(ef + e'f') \end{aligned}$$

where $\mathbf{1}_Z$ refers to the indicator function of the set Z.

The element $ef + e'f'$ belongs to the product space EF and it is natural to think that its statistical distribution is close to uniformly random inside the space EF, therefore the probability that it falls in the space Z is expected to be $q^{-\lambda}$, since $Z \cap EF$ is of codimension λ in EF. We formalize it in the following assumption, alongside with an additional hypothesis on the independence of the random variables defined above.

Assumption 1. *The family of random variables (X_{e,e',F_i,Z_i}), parametrized by all the pairs (e,e') of linearly independent elements in E and the four PSSI samples, form a family of independent Bernoulli variables of parameter $q^{-\lambda}$.*

This assumption was validated by numerous simulations. Using the above assumption, we can prove the following lemma which explicitly gives the probability of fulfilling conditions from $(\mathcal{S})$. We present in Section 4.1.6 an experimental validation of Lemma 4.1.5.

Lemma 4.1.5. *Let $(f_i, f_i') \xleftarrow{\$} F_i$ for $i \in [\![1,4]\!]$. Under the condition $\lambda = 2r$ and under Assumption 1, the probability ε that there exists a pair $(e,e') \in E^2$, such that the system $(\mathcal{S})$ of four conditions is verified:*

$$(\mathcal{S}) : \begin{cases} ef_1 + e'f_1' = z_1 \in Z_1 \\ ef_2 + e'f_2' = z_2 \in Z_2 \\ ef_3 + e'f_3' = z_3 \in Z_3 \\ ef_4 + e'f_4' = z_4 \in Z_4 \end{cases}$$

admits an asymptotic development

$$\varepsilon = q^{-6r} + o_{r\to\infty}(q^{-10r}).$$

Proof. The system $(\mathcal{S})$ is verified for a pair (e,e') when the four boolean variables X_{e,e',F_i,Z_i} (defined above) are all true for $i \in [\![1,4]\!]$. Therefore,

$$\begin{aligned}
\varepsilon &= \mathbb{P}\Big(\bigvee_{(e,e')\in E^2} \Big(\bigwedge_{i=1}^{4} X_{e,e',F_i,Z_i}\Big)\Big) \\
&= 1 - \mathbb{P}\Big(\bigwedge_{(e,e')\in E^2} \Big(\bigvee_{i=1}^{4} \overline{X}_{e,e',F_i,Z_i}\Big)\Big) \\
&= 1 - \prod_{(e,e')\in E^2} \mathbb{P}\Big(\bigvee_{i=1}^{4} \overline{X}_{e,e',F_i,Z_i}\Big) \\
&= 1 - \prod_{(e,e')\in E^2} \Big(1 - \mathbb{P}\Big(\bigwedge_{i=1}^{4} X_{e,e',F_i,Z_i}\Big)\Big) \\
&= 1 - \prod_{(e,e')\in E^2} (1 - q^{-4\lambda}) \\
&= 1 - (1 - q^{-4\lambda})^{q^{2r} - o_{r\to\infty}(q^{r+2})} \\
&= 1 - (1 - q^{-8r})^{q^{2r} - o_{r\to\infty}(q^{r+2})} \\
&= 1 - (1 - q^{-6r} + o_{r\to\infty}(q^{-12r})) \\
&= q^{-6r} + o_{r\to\infty}(q^{-12r}).
\end{aligned}$$

□

In the rest of the paper we will omit the residue in $o_{r\to\infty}(q^{-12r})$.

4.1.5.3 Proof of the probability of success of the attack

We can now finalize the proof of success of the attack.

Proof of Theorem 4.1.1. The three cases of Theorem 4.1.1 form a partition of the possible outputs of Algorithm 4.1, hence we only need to prove the first two inequalities on the probabilities of the theorem and the third inequality will follow immediately.

For $i \in [\![1,4]\!]$, let $(f_i, f_i') \in F_i^2$ be the pairs sampled at random during the first four steps of the attack.

We will consider two separate cases depending on whether conditions from $(\mathcal{S})$ are fulfilled. Each case will yield one of the equalities to be proven.

First case. Suppose that there exists $(e, e') \in E^2$ such that the conditions from $(\mathcal{S})$ are verified. According to Lemma 4.1.5, this happens with probability q^{-6r}.

In that case, we can assume the vector spaces $A_{i,j}$ are independent (as random variables) subspaces of $\mathbb{F}_{q^m}$, all containing e, of dimension $a_{i,j} \leq 2(w + rd - \lambda)$, hence $\sum_{i,j} a_{i,j} \leq 12(w + rd - \lambda)$. By using Lemma 4.1.4, $\langle e \rangle \subset B$ and the probability that B is exactly $\langle e \rangle$ is greater than $1 - q^{12(w+rd-\lambda)-5m-1}$. As a result, Algorithm 4.1 outputs an element of E with a probability greater or equal to

$$q^{-6r}(1 - q^{12(w+rd-\lambda)-5m-1}) = \beta.$$

Second case. If there does not exist a pair $(e, e') \in E$ such that the conditions from $(\mathcal{S})$ are verified (it happens with probability $1 - q^{-6r}$), then the vector spaces $A_{i,j}$ can be seen as random independent subspaces of $\mathbb{F}_{q^m}$ of dimension $a_{i,j} \leq 2(w+rd-\lambda)$, so this time we use Lemma 4.1.3.

It proves that Algorithm 4.1 returns 0 with a probability of at least

$$(1 - q^{-6r})(1 - q^{12(w+rd-\lambda)-5m}) = 1 - \alpha.$$

□

4.1.5.4 Complexity of the attack

Algorithm 4.1 returns only one element of E with a small probability of success. In order to fully solve the Search-PSSI problem, the attacker has to recover the whole space E, i.e. at least r elements of the secret space. In this subsection we study the complexity of the full attack, which recovers E totally.

Let us first study the complexity of one call to Algorithm 4.1. The most costly operation is Step 8, which consists in five intersections of subspaces of $\mathbb{F}_{q^m}$, each of dimension less than $2(w+rd-\lambda)$. An intersection of two subspaces is usually computed through the Zassenhaus algorithm, and is essentially a Gaussian elimination of a binary matrix of size $4(w + rd - \lambda) \times 2m$, which costs $2m \times (4(w+rd-\lambda))^2 = 32m(w+rd-\lambda)^2$ operations in $\mathbb{F}_q$. Repeating the operation five times yields a total complexity of

$$160m(w + rd - \lambda)^2$$

operations in $\mathbb{F}_q$.

It remains to evaluate the number of calls to Algorithm 4.1. To simplify, because the probability that Algorithm 4.1 returns an element outside the space E is negligible, we will consider that the algorithm either

- returns a random element of E with probability q^{-6r}, or
- returns 0 with probability $1 - q^{-6r}$.

On average, the number of times Algorithm 4.1 must be run is q^{6r} multiplied by the expectancy of the number of elements needed to recover E, which is given by the following lemma.

Lemma 4.1.6. *Let E be a subspace of $\mathbb{F}_{q^m}$ of dimension r. Let $\mathcal{O}$ be an oracle which, on each call i, returns an independent $x_i \xleftarrow{\$} E$. The average number n of calls to the oracle such that $\langle x_1, ..., x_n \rangle = E$ is upper bounded as follows:*

$$n \leq r + \frac{1}{q-1}.$$

Proof. Let X be the integer-value random variable defined as the number of calls to the oracle until it generates E, i.e. $\langle x_1, ..., x_{X-1} \rangle \subsetneq E$ and $\langle x_1, ..., x_X \rangle = E$.

It is clear that $X \geq r$ with probability 1. For $i > r$, $X \geq i$ if and only if $\langle x_1, ..., x_i \rangle \subsetneq E$, which is equivalent to having a uniformly random $r \times i$ matrix with entries in $\mathbb{F}_q$ not of full rank. This happens with a probability upper bounded by q^{r-i} (see Lemma B.1.1).

To finish the proof, we calculate the expectancy n of X:

$$\begin{aligned} n &= \mathbb{E}(X) \\ &= r + \sum_{i=r+1}^{\infty} \mathbb{P}(X \geq i) \\ &\leq r + \sum_{i=r+1}^{\infty} q^{r-i} \\ &\leq r + \frac{1}{q-1}. \end{aligned}$$

□

Therefore, we can formulate the following result.

Proposition 4.1.2 (Complexity of the attack)**.** *Under the same conditions of validity than Theorem 4.1.1, the average complexity of the attack is given by*

$$160m(w + rd - \lambda)^2(r + \frac{1}{q-1})q^{6r}$$

operations in $\mathbb{F}_q$.

Applying the above formula to parameters of Durandal, it gives the following table:

	Theoretical complexity	Security
Durandal-I	66	128
Durandal-II	73	128

Table 4.2: Theoretical base-2 logarithm of the average number of bit operations necessary to run our attack against Search-PSSI.

4.1.5.5 Number of signatures

In the previous subsection, we saw that Algorithm 4.1 must be run on average $(r + \frac{1}{q-1})q^{6r}$ to finalize the attack. Since 4 PSSI samples are used in Algorithm 4.1, it could seem that an average number of $4(r + \frac{1}{q-1})q^{6r}$ of signatures would be necessary to recover the private key. This would be a very large number of signatures with the considered parameters.

Fortunately, the same signatures can be reused by running Algorithm 4.1 several times with the same 4 PSSI samples. Indeed, this algorithm starts by choosing at random 8 elements in vector spaces of $\mathbb{F}_q$-dimension d, which makes q^{8d} possibilities.

We can assume that if the algorithm is run with the same set of 4 signatures a number of times greatly less than q^{8d}, the event that one run outputs an element of E remains probabilistically independent from the other runs with the same samples.

Empirically, we set to q^{5d} the number of reuses of the same signatures in Algorithm 4.1, which makes an average number of signatures necessary to finalize the attack of:

$$4(r + \frac{1}{q-1})\frac{q^{6r}}{q^{5d}}.$$

Applying the above formula to parameters of Durandal, it gives the following table:

	Expected signatures
Durandal-I	1,792
Durandal-II	4,096

Table 4.3: Excepted number of signatures to perform our attack on Search-PSSI.

4.1.6 Experimental results

We implemented the attack in C language, using the RBC library [ABB+21] which provides useful functions when working with finite field subspaces. Our implementation is publicly available in the following Github repository:

https://github.com/victordyseryn/pssi-security-implementation

All of our experiments were performed on a laptop equipped with an Intel Core i5-7440HQ CPU and 16GB RAM.

Since we didn't have a sufficient computing power at our disposal to run the 2^{66} attack on the actual parameters of Durandal in reasonable time, we ran experiments with lower parameter sets, which are represented in the following table:

Experiment number	q	m	d	r	λ	w
A2	2	83	2	2	3	19
A3	2	127	3	3	6	28
A4	2	163	4	4	8	38
A5	2	199	5	5	10	47

Table 4.4: Reduced parameter sets for experiments on PSSI attack

For each experiment, we ran the attack a number of times depending on the complexity of the attack, and we recorded the average number of cycles to recover the entire secret space E, as well as the average number of PSSI samples needed. We were able to complete the attack up to the parameter set A4. Experiment A5 was out of reach in a reasonable time. We computed the experimental complexity of our experiments as the average number of cycles required to recover the secret key, and then multiplying this cycle count by 64 to obtain an approximation of the number of bit operations performed by our 64-bit processor. Our experimental results are presented in Table 4.5.

Experiment	q	m	d	r	λ	w	Number of tests	Number of signatures (avg)	Experimental complexity	Theoretical complexity
A2	2	83	2	2	3	19	1,000	10	$2^{32.4}$	$2^{35.9}$
A3	2	127	3	3	6	28	100	301	$2^{44.9}$	2^{44}
A4	2	163	4	4	8	38	1	502	$2^{51.2}$	$2^{51.7}$

Table 4.5: Experimental results on PSSI attack

Figure 4.2 shows the comparison between the experimental and theoretical complexities, as well as the expected complexities for parameter sets A5, Durandal-I and Durandal-II.

Finally, we also validated the result from Lemma 4.1.5 by running the following experiment: we randomly generated PSSI samples and checked whether there exists a pair $(e, e') \in E^2$ such

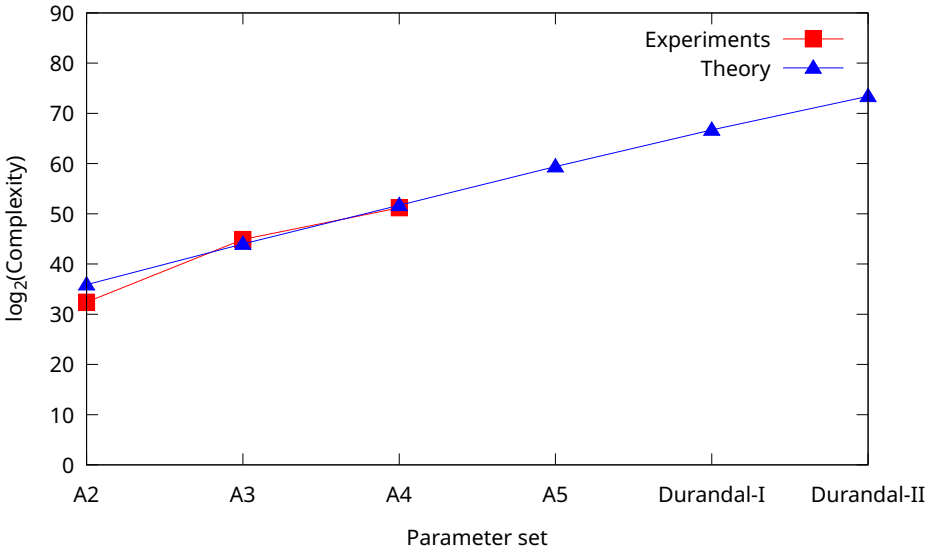

Figure 4.2: Comparison between experimental and theoretical complexities for different values of r.

that the system $(\mathcal{S})$ described in Lemma 4.1.5 is verified, by enumerating every possible pair (e, e'). Results are presented in Table 4.6.

q	m	d	r	λ	w	Number of tests	Experimental probability	Theoretical probability
2	83	3	3	6	19	2^{24}	$2^{-18.6}$	2^{-18}
2	127	3	3	6	28	2^{24}	$2^{-18.9}$	2^{-18}

Table 4.6: Experimental results validating Lemma 4.1.5

4.2 PERK: a signature scheme based on a variant of PKP

In this section, we present PERK, a new compact digital signature scheme based on a new multi-dimensional version of PKP: the relaxed IPKP problem (r-IPKP). In Section 4.2.1 we cover basic notations and definitions, recall ZKPoK constructions as well as the MPCitH paradigm. In the subsequent Section 4.2.2 we then present our new ZKPoK protocol based on the hardness of the r-IPKP, convert it into a non-interactive signature scheme via the Fiat-Shamir transform and provide comprehensive security proofs. Section 4.2.3 covers a detailed analysis of the complexity of r-IPKP, where Section 4.2.3.1 covers previous approaches and Section 4.2.3.2 introduces our new algorithm. Parameters of our scheme are presented in Section 4.2.4 which also provides a comparison to the state-of-the-art. Eventually, in Section 4.2.5 performance benchmarks of our optimized AVX2 consant-time implementation are given.

I mainly contributed during my doctoral work to the definition of the r-IPKP problem and its security analysis, as well as the parameter selection for our algorithms. The construction and implementation of the new protocol and signature scheme was mostly handled by other members of the PERK team.

4.2.1 Preliminaries

4.2.1.1 Proofs of Knowledge

Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a binary relation. If $(x;w) \in R$, we say x is a *statement* and w is a *witness* for x. The set of valid witnesses for x is denoted by $R(x) = \{w \,|\, (x;w) \in R\}$. A statement that admits a witness is called a *true* or *valid* statement. The set of true statements is denoted by $L_R := \{x : \exists\, w \text{ such that } (x;w) \in R\}$. A binary relation is said to be an NP relation if the validity of a witness w can be verified in time polynomial in the size $|x|$ of the statement x. From now on we assume all relations to be NP relations.

An interactive proof for relation R aims for a prover P to convince a verifier V that a statement x admits a witness, or even that the prover *knows* a witness $w \in R(x)$.

Definition 4.2.1 (Interactive proof (cf. [AF22]))**.** *An interactive proof $\Pi = (\mathsf{P}, \mathsf{V})$ for relation R is an interactive protocol between two probabilistic machines, a prover P and a polynomial time verifier V. Both P and V take as public input a statement x and, additionally, P takes as private input a witness $w \in R(x)$, which is denoted as $(\mathsf{P}(w), \mathsf{V})(x)$. The verifier V either accepts or rejects the prover's claim of knowing a witness for x, the output of the protocol is the verifier's decision. The set of all messages exchanged in the protocol execution is called a transcript and is denoted $\langle \mathsf{P}(x,w), \mathsf{V}(x) \rangle$. We call the transcript* accepting *(or resp.* rejecting*) based on whether the verifier accepts (or rejects) the prover's claim.*

We assume that the prover sends the first and the last message in any interactive proof. Hence, the number of messages is always an odd number $2\mu+1$. We also say Π is a $(2\mu+1)$-round proof. It is represented in the following figure.

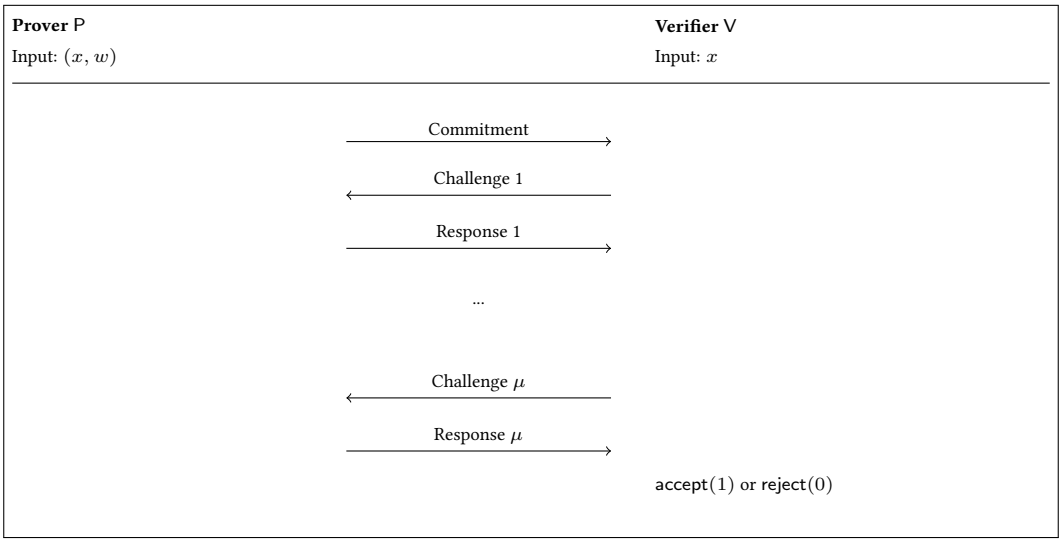

Figure 4.3: $(2\mu + 1)$-round interactive proof

An interactive proof Π is *complete* if the verifier V accepts honest executions with a public-private input pair $(x;w) \in R$ with high probability. It is *sound* if the verifier rejects the false statements $x \notin L_R$ with high probability. In this work, we follow the presentation of [AF22] and do not require these properties as part of definition of interactive proofs, but consider them as desirable additional security properties.

Definition 4.2.2 (Completeness (cf. [AF22]))**.** *An interactive proof $\Pi = (\mathsf{P}, \mathsf{V})$ for relation R is* complete *with completeness error $\rho : \{0,1\}^* \to [0,1]$ if for every $(x;w) \in R$,*

$$\mathbb{P}\left[(\mathsf{P}(w), \mathsf{V})(x) = \textit{reject}\right] \leq \rho(x).$$

If $\rho(x) = 0$ for all $x \in L_R$, then Π is said to be perfectly *complete.*

Definition 4.2.3 (Soundness (cf. [AF22])). *An interactive protocol* $\Pi = (\mathsf{P}, \mathsf{V})$ *for relation* R *is* sound *with soundness error* $\sigma : \{0,1\}^* \to [0,1]$ *if for every* $x \notin L_R$ *and every prover* P^*,

$$\mathbb{P}\left[(\mathsf{P}^*, \mathsf{V})(x) = \textit{accept}\right] \leq \sigma(x).$$

An interactive proof which is complete and sound allows a prover to convince a verifier that the statement x is true, i.e., $x \in L_R$. However, this does not necessarily convince a verifier that the prover actually "knows" the witness $w \in R(x)$. This stronger property is captured by the notion of *knowledge soundness*. Informally, knowledge soundness guarantees that if a prover convinces a verifier about the validity of some statement x with sufficiently high probability, then the prover can actually compute a witness $w \in R(x)$ with high probability. [2]

Definition 4.2.4 (Knowledge Soundness (cf. [AF22])). *An interactive protocol* $\Pi = (\mathsf{P}, \mathsf{V})$ *for relation* R *is* knowledge sound *with knowledge error* $\varepsilon_{\mathsf{KS}} : \{0,1\}^* \to [0,1]$ *if there exists a positive polynomial* q *and an algorithm* Ext*, called* knowledge extractor*, with the following properties: The extractor* Ext*, given input* x *and rewindable oracle access to a (potentially dishonest) prover* P^**, runs in an expected number of steps that is polynomial in* $|x|$ *and outputs a witness* $w \in R(x)$ *with probability*

$$\mathbb{P}\left[\left(\mathsf{Ext}^{\mathsf{P}^*}(x)\right) \in R\right] \geq \frac{\epsilon(x, \mathsf{P}^*) - \varepsilon_{\mathsf{KS}}(x)}{\mathsf{q}(|x|)},$$

where $\epsilon(x, \mathsf{P}^*) := \mathbb{P}\left[(\mathsf{P}^*, \mathsf{V})(x) = \textit{accept}\right]$.

If $\epsilon(x, \mathsf{P}^*) = \mathbb{P}\left[(\mathsf{P}^*, \mathsf{V})(x) = \text{accept}\right] > \varepsilon_{\mathsf{KS}}(x)$, then the success probability of the knowledge extractor Ext in Definition 4.2.4 is positive. Therefore, $\epsilon(x, \mathsf{P}^*) > \varepsilon_{\mathsf{KS}}(x)$ implies that x admits a witness, i.e., $x \in L_R$. Hence, knowledge soundness implies soundness.

Definition 4.2.5 (Proof of Knowledge (cf. [AF22])). *An interactive proof that is both complete with completeness error* $\rho(\cdot)$ *and knowledge sound with knowledge error* $\varepsilon_{\mathsf{KS}}(\cdot)$ *is a* Proof of Knowledge *(PoK) if there exists a polynomial* q *such that* $1 - \rho(x) \geq \varepsilon_{\mathsf{KS}}(x) + 1/\mathsf{q}(|x|)$ *for all* x.

It is desirable to have simple verifiers which can send uniform random challenges to the prover, and efficiently verify the transcript.

Definition 4.2.6 (Public-Coin (cf. [AFK22])). *An interactive proof* $\Pi = (\mathsf{P}, \mathsf{V})$ *is* public-coin *if all of* V*'s random choices are made public, i.e. are part of the transcript. The message* $\mathsf{ch}_i \xleftarrow{\$} \mathcal{CH}_i$ *of* V *in the* $2i$*-th round is called the* i*-th* challenge*, and* $\mathcal{CH}_i$ *is the* challenge set.

Public-coin protocols can be turned into non-interactive protocols by using the Fiat-Shamir transformation [FS87]. In this work, we consider only public-coin protocols.

Next, we discuss the notion of *special-soundness*. Special-soundness property is easier to check than knowledge soundness and for many protocols, knowledge soundness follows from special-soundness. Note that this requires special-sound protocols to be public-coin.

Definition 4.2.7 (k-out-of-N Special Soundness (cf. [AF22])). *Let* $k, N \in \mathbb{N}$. *A* 3*-round public-coin protocol* $\Pi = (\mathsf{P}, \mathsf{V})$ *for relation* R*, with challenge set of cardinality* $N \geq k$*, is* k-out-of-N special sound *if there exists a polynomial time algorithm that, on input a statement* x *and* k *accepting transcripts* $(\mathsf{cmt}, \mathsf{ch}_1, \mathsf{rsp}_1), \ldots, (\mathsf{cmt}, \mathsf{ch}_k, \mathsf{rsp}_k)$ *with common first message* cmt *and pairwise distinct challenges* $\mathsf{ch}_1, \ldots, \mathsf{ch}_k$*, outputs a witness* $w \in R(x)$*. We also say* Π *is* k-special-sound *and, if* $k = 2$*, it is simply called special-sound.*

In order to generalize k-special-soundness to multi-round protocols we will introduce the notion of a tree of transcripts following the definitions given in [ACK21].

Definition 4.2.8 (Tree of Transcripts (cf. [AF22])). *Let* $k_1, \ldots, k_\mu \in \mathbb{N}$. *A* $(k_1, \ldots, k_\mu)$*-tree of transcripts for a* $(2\mu + 1)$*-round public-coin protocol* $\Pi = (\mathsf{P}, \mathsf{V})$ *is a set of* $K = \prod_{i=1}^{\mu} k_i$ *transcripts*

[2] Since the protocol presented in this work only achieves computational soundness, and is secure when the prover runs in *polynomial time*, technically our protocol is an *argument of knowledge*. However, we avoid this distinction for simplicity.

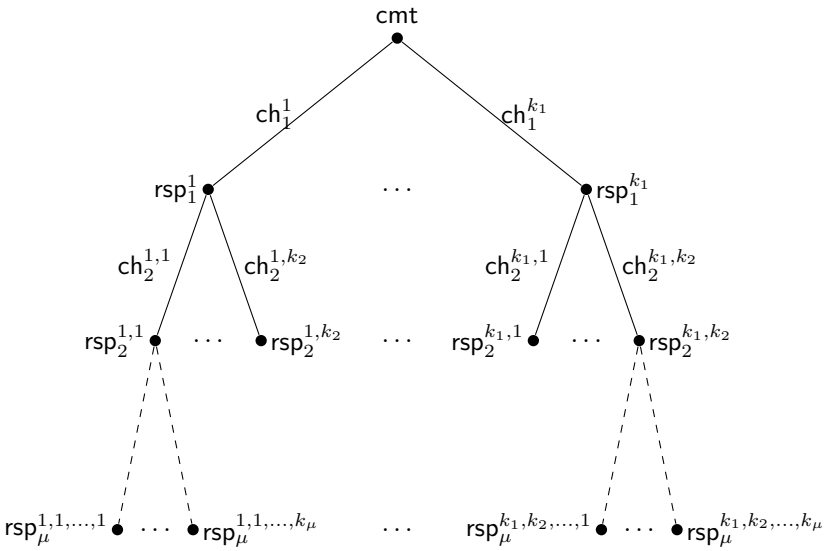

Figure 4.4: $(k_1, k_2, \ldots, k_\mu)$ tree of transcripts of a $(2\mu+1)$-round public-coin protocol

arranged in the following tree structure. The nodes in this tree correspond to the prover's messages and the edges to the verifier's challenges. Every node at depth i has precisely k_i children corresponding to k_i pairwise distinct challenges. Every transcript corresponds to exactly one path from the root to a leaf node. For a graphical representation we refer to Figure 4.4. We refer to the corresponding tree of challenges as a $(k_1, \ldots, k_\mu)$-tree of challenges.

We will also write $\boldsymbol{k} = (k_1, \ldots, k_\mu) \in \mathbb{N}^\mu$ and refer to a $\boldsymbol{k}$-tree of transcripts.

Definition 4.2.9 ($(k_1, \ldots, k_\mu)$-out-of-$(N_1, \ldots, N_\mu)$ Special Soundness (cf. [AF22])). *Let $k_1, \ldots, k_\mu$, $N_1, \ldots, N_\mu \in \mathbb{N}$. A $(2\mu+1)$-round public-coin protocol $\Pi = (\mathsf{P}, \mathsf{V})$ for a relation R, where V samples the i-th challenge from a set of cardinality $N_i \geq k_i$ for $1 \leq i \leq \mu$, is $(k_1, \ldots, k_\mu)$-out-of-$(N_1, \ldots, N_\mu)$ special-sound if there exists a polynomial time algorithm that, on an input statement x and a $(k_1, \ldots, k_\mu)$-tree of accepting transcripts outputs a witness $w \in R(x)$. We also say Π is $(k_1, \ldots, k_\mu)$ special-sound.*

The following theorem proved in [ACK21] states that special soundness implies knowledge soundness.

Theorem 4.2.1 ($(k_1, \ldots, k_\mu)$ Special Soundness implies Knowledge Soundness [ACK21, Theorem 1]). *Let $\mu, k_1, \ldots, k_\mu \in \mathbb{N}$ be such that $K = \Pi_{i=1}^{\mu} k_i$. Let (P, V) be a $(k_1, \ldots, k_\mu)$ special sound $(2\mu+1)$-round interactive protocol for relation R, where V samples each challenge uniformly at random from a set of cardinality N_i for $1 \leq i \leq \mu$. Then (P, V) is knowledge sound with knowledge error*

$$\varepsilon_{\mathsf{KS}} = \frac{\prod_{i=1}^{\mu} N_i - \prod_{i=1}^{\mu}(N_i - k_i + 1)}{\prod_{i=1}^{\mu} N_i} \leq \sum_{i=1}^{\mu} \frac{k_i - 1}{N_i}. \tag{4.1}$$

We write $\Pi^\tau := (\mathsf{P}^\tau, \mathsf{V}^\tau)$ for the τ-fold parallel repetition of Π, which runs τ instances of Π in parallel and the verifier V^τ accepts if *all* the parallel instances are accepted.

The following theorem proved in [AF22] states that the knowledge soundness is retained (and knowledge error is reduced) via parallel repetition.

Theorem 4.2.2 (Parallel Repetition for Multi-Round Protocols [AF22, Theorem 4]). *Let (P, V) be a $(k_1, \ldots, k_\mu)$-out-of-$(N_1, \ldots, N_\mu)$ special sound protocol. Let $(\mathsf{P}^\tau, \mathsf{V}^\tau)$ be the τ-fold repetition of protocol (P, V) . Then $(\mathsf{P}^\tau, \mathsf{V}^\tau)$ is knowledge sound with knowledge error $\varepsilon_{\mathsf{KS}}^\tau$, where*

$$\varepsilon_{\mathsf{KS}} = 1 - \prod_{i=1}^{\mu} \frac{(N_i - k_i + 1)}{N_i} \tag{4.2}$$

is the knowledge error of (P, V).

Definition 4.2.10 (Special Honest-Verifier Zero Knowledge (SHVZK) (adapted from [ACK21])). *An interactive proof* $\Pi = (\mathsf{P}, \mathsf{V})$ *is called* { *perfectly, statistically, computationally* } honest-verifier zero knowledge (HVZK) *if there exists a polynomial time simulator that on input* $x \in L_R$ *outputs an accepting transcript which is distributed* { *perfectly, statistically, computationally* } *close to the transcripts generated by honest executions of* Π. *If the simulator proceeds by first sampling the verifier's messages uniformly at random, then* Π *is called* special honest-verifier zero knowledge (SHVZK).

4.2.1.2 Merkle Trees

Merkle trees can be used in our context to compress randomness seeds as suggested in [KKW18]. Suppose a party needs to generate N seeds and then to send only $N-1$ of those seeds (without knowing in advance which seed should not be sent). The principle is to build a binary tree of depth $\lceil \log_2(N) \rceil$. The root of the tree is labeled with a master seed θ. The rest of the tree is labeled inductively by using a PRG of double extension on each parent node and splitting the output on the left and right children.

To reveal all seeds except seed number $i \in [N]$, the principle is to reveal the labels on the siblings of the paths from the root of the tree to leave i. It allows to reconstruct all seeds but seed number i at the cost of communicating $\lfloor \log_2(N) \rfloor$ labels, which is more effective than communicating $N-1$ seeds.

4.2.1.3 MPC-in-the-Head and PoK

Our construction relies on the *MPC-in-the-Head* (MPCitH) paradigm introduced by Ishai, Kushilevitz, Ostrovsky, and Sahai in [IKOS07]. This paradigm builds a zero-knowledge proof based on a secure multiparty computation (MPC) protocol. Informally, the MPC protocol is used to compute the verification of an NP relation, where the privacy guarantee of the protocol is used to achieve the zero-knowledge property.

The main steps of the proof of knowledge resulting from the MPCitH technique are the following:

1. The prover splits its witness into N parties by secret sharing the witness;
2. The prover then simulates locally ("in her head") all the parties of the MPC protocol which evaluates a Boolean function that is expected to be 1 whenever the witness is correct (this is supposed to correspond to the verification of desired NP relation);
3. The prover commits to the views of all the parties in the MPC protocol;
4. The verifier chooses a random subset of $N' < N$ parties and asks to reveal their corresponding views;
5. The verifier finally checks that the views of the revealed parties are consistent with each other and with an honest execution of the MPC protocol that yields output 1.

This transformation achieves the zero-knowledge property as long as the views of any N' parties do not leak any information about the secret witness.

Since our proof of knowledge is an instantiation of the MPCitH technique for the specific case of r-IPKP, it benefits from an extensive literature of optimizations generic to any MPCitH construction, such as:

- The preprocessing extension, introduced in [KKW18], allows the MPC protocol – used in the MPCitH technique – to rely on a preprocessing phase (under certain conditions) thus drastically reducing the proof size;
- the challenge space amplification technique, introduced in [BG23], that is itself an optimization of the PoK with Helper paradigm introduced in [Beu20b];
- Merkle trees to reveal a partial number of random seeds, as explained in Section 4.2.1.2.

4.2.1.4 Fiat-Shamir Transformation

In this section, we explain the random oracle model and Fiat-Shamir transformation used for transforming interactive protocols into non-interactive ones. We closely follow the presentation of [AFK22, Section 2.3] in the following exposition.

In the *random oracle model (ROM)*, algorithms have black-box (or input-output) access to an oracle $\mathsf{RO} : \{0,1\}^* \to \mathcal{Z}$, called as *random oracle*, which is instantiated with a uniform random function with domain $\{0,1\}^*$ and codomain $\mathcal{Z}$. Generally, $\mathcal{Z} = \{0,1\}^\eta$ for some $\eta \in \mathbb{N}$ related to the security parameter. In practice, RO can be implemented by lazy sampling, which means for each input string $x \in \{0,1\}^*$, $\mathsf{RO}(x)$ is sampled uniform randomly from $\mathcal{Z}$ and then fixed. To avoid technical difficulties, we limit the domain from $\{0,1\}^*$ to $\{0,1\}^{\leq \ell}$, the finite set of all bitstrings of length at most ℓ, for a sufficiently large $\ell \in \mathbb{N}$.

An algorithm $\mathcal{A}^{\mathsf{RO}}$ that is given black-box access to a random oracle is called a *random oracle algorithm*. We say $\mathcal{A}$ is a *Q-query random-oracle algorithm*, if it makes at most Q queries to RO (independent of RO).

A natural extension of the ROM is when $\mathcal{A}$ is given access to *multiple independent* random oracles $\mathsf{RO}_1, \mathsf{RO}_2, \ldots, \mathsf{RO}_\mu$, possibly with different codomains. In practice, these random oracles can be instantiated by a single random oracle $\mathsf{RO} : \{0,1\}^* \to \{0,1\}^\eta$ using the standard techniques for domain separation (refer to [BDG20] for more details) and for sampling random elements from non-binary sets.

The Fiat-Shamir transform [FS87], turns a public-coin interactive proof into a non-interactive proof in the random oracle model. The general idea of this transformation is to compute the i-th challenge message ch_i as a hash of the i-th prover message a_i along with (partial) communication transcript generated till that point. For a Σ-protocol, the challenge ch is computed as $\mathsf{ch} := \mathsf{H}(\mathsf{cmt})$ or as $\mathsf{ch} := \mathsf{H}(x, \mathsf{cmt})$, where the former is sufficient for *static* security, where the statement x is given as input to the dishonest prover, and the latter is necessary for *adaptive* security, where the dishonest prover can choose the statement x for which it wants to forge a proof.

For multi-round public-coin interactive proofs, there is some degree of freedom in the computation of the i-th challenge. For concreteness we consider a particular version where all previous messages are hashed along with the current message.

Let $\Pi = (\mathsf{P}, \mathsf{V})$ be a $(2\mu+1)$-round public-coin interactive proof, where the challenge for the i-th round is sampled from set $\mathcal{CH}_i$. For simplicity, we consider μ random oracles $\mathsf{RO}_i : \{0,1\}^{\leq \ell} \to \mathcal{CH}_i$ that map into the respective challenge spaces.

Definition 4.2.11 (Fiat-Shamir Transformation (cf. [AFK22])). *The static Fiat-Shamir transformation $\mathsf{FS}[\Pi] = (\mathsf{P}_{\mathsf{fs}}, \mathsf{V}_{\mathsf{fs}})$ is a non-interactive proof in the ROM, where $\mathsf{P}_{\mathsf{fs}}^{\mathsf{RO}_1.\mathsf{RO}_2\ldots,\mathsf{RO}_\mu}(x; w)$ runs $\mathsf{P}(x; w)$ but instead of asking the verifier for the challenge ch_i on message a_i, the challenges are computed as*

$$\mathsf{ch}_i = \mathsf{RO}_i\left(a_1, a_2, \ldots, a_{i-1}, a_i\right); \tag{4.3}$$

the output is then the proof $\pi = (a_1, \ldots, a_{\mu+1})$. On input a statement x and a proof $\pi = (a_1, \ldots, a_{\mu+1})$, $\mathsf{P}_{\mathsf{fs}}^{\mathsf{RO}_1.\mathsf{RO}_2\ldots,\mathsf{RO}_\mu}(x, \pi)$ accepts if, for ch_i as above V accepts the transcript $(a_1, \mathsf{ch}_1, \ldots a_\mu, \mathsf{ch}_\mu, a_{\mu+1})$ on input x.

If the challenges are computed as

$$\mathsf{ch}_i = \mathsf{RO}_i\left(x, a_1, \mathsf{ch}_1, \ldots a_{i-1}, \mathsf{ch}_{i-1}, a_i\right); \tag{4.4}$$

the resulting non-interactive proof in ROM is called as the adaptive *Fiat-Shamir transformation.*

4.2.2 PoK and Signature based on r-IPKP

4.2.2.1 Relaxed-IPKP: our variant of the Permuted Kernel Problem

The classical IPKP problem, in its generic form with an inhomogeneous syndrome $\boldsymbol{y}$ and a dimension parameter t, was defined in Definition 1.3.1.

Instead of directly relying on the hardness of IPKP, we consider a relaxed version r-IPKP which allows for more efficient constructions. In this relaxed variant the searched permutation does not necessarily have to satisfy the identity for all given pairs but only for an arbitrary (non-zero) linear combination of those pairs.

Definition 4.2.12 (r-IPKP). *Let (q, m, n, t) be positive integers such that $m < n$, $\boldsymbol{H} \in \mathbb{F}_q^{m\times n}$, $(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathbb{F}_q^n \times \mathbb{F}_q^m$ and $\pi \in \mathcal{S}_n$ be a permutation such that $\boldsymbol{H}(\pi[\boldsymbol{x}_i]) = \boldsymbol{y}_i$ for all $i \in [t]$. Furthermore, the matrix whose columns are the $\boldsymbol{x}_i$ has full rank. Given $\left(\boldsymbol{H}, (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i\in[t]}\right)$, the Relaxed Inhomogeneous Permuted Kernel Problem r-IPKP(q, m, n, t) asks to find any $\tilde{\pi} \in \mathcal{S}_n$ such that $\boldsymbol{H}\left(\tilde{\pi}\left[\sum_{i\in[t]} \kappa_i \cdot \boldsymbol{x}_i\right]\right) = \sum_{i\in[t]} \kappa_i \cdot \boldsymbol{y}_i$ for some $\kappa \in \mathbb{F}_q^t \setminus \mathbf{0}$, where $\kappa := (\kappa_1, \ldots, \kappa_t)$ and $\mathbf{0} \in \mathbb{F}_q^t$ is the all zero vector.*

The hardness of the above problem is discussed in Section 4.2.3.

4.2.2.2 Proof of Knowledge based on r-IPKP

Recall that our goal is to construct a post-quantum digital signature scheme based on the ZKPoK protocol by using the Fiat-Shamir transformation [FS87]. However, Kales and Zaverucha in [KZ20] showed that 5-round PoK which use parallel repetition to achieve a negligible soundness error can be attacked when they are converted to their non-interactive version with the Fiat-Shamir transform. The attack strategy optimally guesses the challenges in each round allowing to convince the verifier by crafting the responses based on the guessed challenge values, without actually knowing the secret. While this attack can be thwarted by increasing the number of parallel repetitions appropriately, it was shown in [BG23] that one can achieve a similar result by *amplifying the challenge space* queried by the verifier, which can lead to more efficient constructions.

Our construction leverages the r-IPKP problem introduced in Section 4.2.2.1 as a way to perform challenge space amplification. Informally, the problem requires, given a matrix $\boldsymbol{H}$ and t pairs of vectors $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, to find a permutation π that sends $\boldsymbol{H}\pi(\boldsymbol{x})$ to $\boldsymbol{y}$ where $\boldsymbol{x} := \sum_i \kappa_i \boldsymbol{x}_i$ (resp. $\boldsymbol{y} := \sum_i \kappa_i \boldsymbol{y}_i$) and $\kappa_1, \ldots, \kappa_t$ are the coefficients of an arbitrary adversarially chosen (non-zero) linear combination. Let $x = \left(\boldsymbol{H}, (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i\in[t]}\right)$ and let $w = \pi \in \mathcal{S}_n$ as defined in Definition 4.2.12. Let $\mathcal{R}_{t-\text{r-IPKP}}$ be a relation for r-IPKP problem defined as,

$$\mathcal{R}_{t-\text{r-IPKP}} := \left\{ \left(\left(\boldsymbol{H}, (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i\in[t]}\right); \tilde{\pi} \right) \;:\; \begin{array}{l} \boldsymbol{H}\left(\tilde{\pi}\left[\sum_{i\in[t]} \kappa_i \cdot \boldsymbol{x}_i\right]\right) = \sum_{i\in[t]} \kappa_i \cdot \boldsymbol{y}_i \\ \text{for some } \kappa \in {\mathbb{F}_q}^t \setminus \mathbf{0} \end{array} \right\}$$

We now present our protocol in Figure 4.5 that is inspired from [BG23] and [FJR23]. Informally, it consists of three main steps, following the MPCitH paradigm:

1. In the commitment step, the witness π is split into N *compositional* shares $\pi_1, \ldots, \pi_N$ such that $\pi = \pi_N \circ \pi_{N-1} \circ \cdots \circ \pi_1$. The prover also generates N (pseudo) random vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ in $\mathbb{F}_q^n$. The compositional and vector shares are then combined to construct a syndrome $\boldsymbol{Hv}$ (the vector $\boldsymbol{v}$ is generated by combining the shares π_is and $\boldsymbol{v}_i$s, refer Figure 4.5 for the details), which is committed together with the generated shares (π_i and $\boldsymbol{v}_i$).

2. The verifier then sends coefficients κ_i of an $\mathbb{F}_q$-linear combination as a first challenge. The prover then computes values $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N$ with the help of the π_i and $\boldsymbol{v}_i$ values committed earlier and the public statement $x = \left(\boldsymbol{H}, (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i\in[t]}\right)$, such that $\boldsymbol{Hs}_N = \boldsymbol{Hv} + \sum_{i\in[t]} \kappa_i \boldsymbol{y}_i$. The prover then sends $\boldsymbol{s}_i$ values as its response. In the actual protocol we use a collision-resistant hash function to compress the information sent to the verifier.

3. Finally, the verifier sends an index $\alpha \in [N]$ as the second challenge. The prover reveals all shares π_i and $\boldsymbol{v}_i$ except the ones with index α. Additionally, the prover reveals the share $\boldsymbol{s}_\alpha$. This allows the verifier to verify the consistency of the views of all the shares except the ones with index α by recomputing the commitments. The verifier can also recompute all the $\boldsymbol{s}_i$ values for $i \neq \alpha$ and together with $\boldsymbol{s}_\alpha$ sent by the prover, the verifier can then reconstruct $\boldsymbol{s}_N$. Finally the verifier computes $\boldsymbol{Hv} = \boldsymbol{Hs}_N - \sum \kappa_i \boldsymbol{y}_i$ and checks if this value is consistent with the commitment received in first message (Step 1 above).

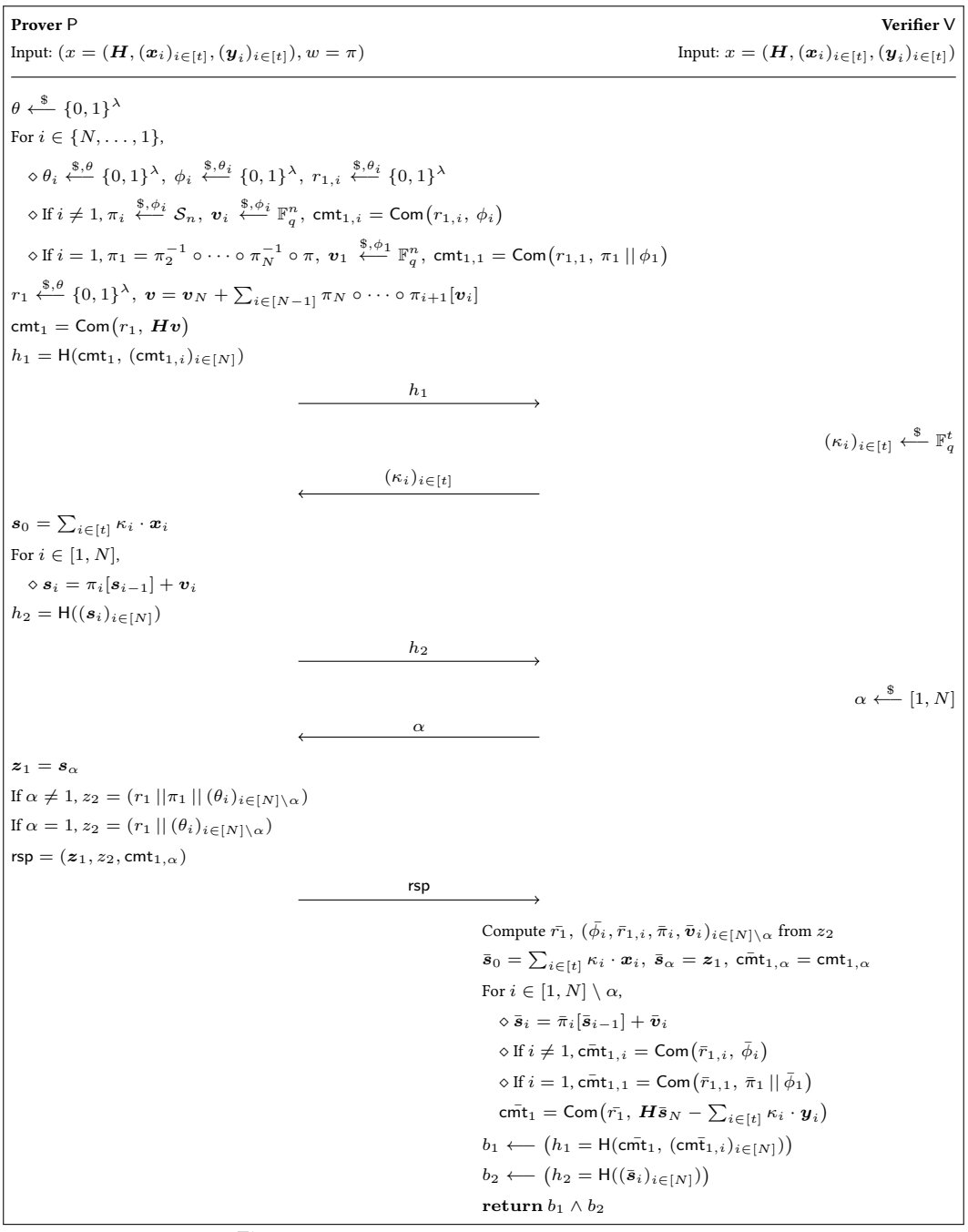

Figure 4.5: PoK leveraging structure for the r-IPKP problem

Theorem 4.2.3 (Completeness). *The protocol presented in Figure 4.5 is* perfectly *complete.*

Proof. The completeness follows from the protocol description once it is observed that $\boldsymbol{s}_N = \pi\left[\sum_{i\in[t]} \kappa_i \cdot \boldsymbol{x}_i\right] + \boldsymbol{v}$ which implies that

$$\boldsymbol{H}\boldsymbol{s}_N - \sum_{i\in[t]} \kappa_i \cdot \boldsymbol{y}_i = \boldsymbol{H}\left(\pi\left[\sum_{i\in[t]} \kappa_i \cdot \boldsymbol{x}_i\right]\right) + \boldsymbol{H}\boldsymbol{v} - \sum_{i\in[t]} \kappa_i \cdot \boldsymbol{y}_i = \boldsymbol{H}\boldsymbol{v}.$$

Therefore for every true statement $(\boldsymbol{H}, (\boldsymbol{x}_i)_{i\in[t]}, (\boldsymbol{y}_i)_{i\in[t]})$ with witness π if the protocol described in Figure 4.5 is executed honestly then the verifier V accepts with probability 1 for all possible random choices of P and V. □

Theorem 4.2.4 (Knowledge Soundness). *The protocol presented in Figure 4.5 is knowledge sound with knowledge error*

$$\varepsilon_{\mathsf{KS}} = \frac{1}{N} + \frac{N-1}{N\cdot(q^t-1)}.$$

Theorem 4.2.5 (Special Honest-Verifier Zero Knowledge). *Assume that there exists a* $(\mathsf{t}, \epsilon_{\mathsf{PRG}})$*-secure* PRG*, and the commitment scheme* Com *is* $(\mathsf{t}, \epsilon_{\mathsf{Com}})$*-hiding. Then there exists an efficient simulator* Sim *which, outputs a transcript such that no distinguisher running in time at most* $\mathsf{t}(\lambda)$ *can distinguish between the transcript produced by* Sim *and a real transcript obtained by honest execution of the protocol in Figure 4.5 with probability better than* $(\epsilon_{\mathsf{PRG}}(\lambda) + \epsilon_{\mathsf{Com}}(\lambda))$.

We prove Theorem 4.2.4 and Theorem 4.2.5 in Appendix C.1.1 and Appendix C.1.2 respectively.

4.2.2.3 PERK: Signature Scheme based on r-IPKP

By applying the Fiat-Shamir transformation on the protocol shown in Figure 4.5, one gets the digital signature scheme described in Figure 4.6, Figure 4.7 and Figure 4.8.

Theorem 4.2.6. *Suppose* PRG *is* $(\mathsf{t}, \epsilon_{\mathsf{PRG}})$*-secure and any adversary running in time* $\mathsf{t}(\lambda)$ *can solve the the underlying r-IPKP instance with probability at most* $\epsilon_{\text{r-IPKP}}$*. Model* H_0*,* H_1*, and* H_2 *as random oracles where* H_0*,* H_1*, and* H_2 *have* 2λ*-bit output length. Then a chosen-message attacker against the signature scheme (PERK) presented in Figure 4.7, running in time* $\mathsf{t}(\lambda)$*, making* q_s *signing queries, and making* q_0*,* q_1*,* q_2 *queries, respectively, to the random oracles, succeeds in outputting a valid forgery with probability*

$$\begin{aligned}\mathbb{P}[\mathsf{Forge}] \leq \frac{(q_0 + \tau\cdot(N+1)\cdot q_s)^2}{2\cdot 2^{2\lambda}} + \frac{q_s\cdot(q_0+q_1+q_2+q_s)}{2^{2\lambda}} \\ + \tau\cdot q_s\cdot\epsilon_{\mathsf{PRG}}(\lambda) + \epsilon_{\text{r-IPKP}} + q_2\cdot\varepsilon_{\mathsf{KS}}^{\tau},\end{aligned} \tag{4.5}$$

where $\varepsilon_{\mathsf{KS}} = \frac{1}{N} + \frac{N-1}{N\cdot(q^t-1)}$.

We prove Theorem 4.2.6 in Appendix C.2.1.

4.2.3 On the Hardness of r-IPKP

In this section we study the hardness of the r-IPKP from Definition 4.2.12. Even though, we introduce the r-IPKP together with our scheme its hardness is still tied to the hardness of the multi-dimensional and mono-dimensional versions of IPKP as we outline in the following.

Recall that the difference to IPKP is that for r-IPKP the solution does not necessarily have to be the permutation that works for all the given pairs, but it has to satisfy the PKP identity only for an arbitrary (non-zero) linear combination of those pairs, i.e. any permutation π such that

$$\boldsymbol{H}\left(\pi\left[\sum_i \kappa_i \boldsymbol{x}_i\right]\right) = \sum_i \kappa_i \boldsymbol{y}_i,$$

1. Sample $\mathsf{sk\_seed} \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{pk\_seed} \xleftarrow{\$} \{0,1\}^\lambda$
2. Sample $\pi \longleftarrow \mathsf{PRG}(\mathsf{sk\_seed})$ from $\mathcal{S}_n$
3. Sample $(\boldsymbol{H}, (\boldsymbol{x}_i)_{i\in[t]}) \longleftarrow \mathsf{PRG}(\mathsf{pk\_seed})$ from $\mathbb{F}_q^{m\times n} \times (\mathbb{F}_q^n)^t$
3. For $j \in [t]$,
   $\diamond$ Compute $\boldsymbol{y}_j = \boldsymbol{H}\pi[\boldsymbol{x}_j]$
4. Output $(\mathsf{sk}, \mathsf{pk}) = (\mathsf{sk\_seed}, (\mathsf{pk\_seed}, (\boldsymbol{y}_j)_{j\in[t]}))$

Figure 4.6: PERK - KeyGen algorithm

Inputs

- Secret key $\mathsf{sk} = \pi$
- Public key $\mathsf{pk} = (\boldsymbol{H}, (\boldsymbol{x}_j, \boldsymbol{y}_j)_{j\in[t]})$
- Message $m \in \{0,1\}^*$

Step 1: Commitment

1. Sample salt and master seed $(\mathsf{salt}, \mathsf{mseed}) \xleftarrow{\$} \{0,1\}^{2\lambda} \times \{0,1\}^\lambda$
2. Sample seeds $(\theta^{(e)})_{e\in[\tau]} \longleftarrow \mathsf{PRG}(\mathsf{salt}, \mathsf{mseed})$ from $(\{0,1\}^\lambda)^\tau$
3. For each iteration $e \in [\tau]$,
   $\diamond$ Compute $(\theta_i^{(e)})_{i\in[N]} \longleftarrow \mathsf{TreePRG}(\mathsf{salt}, \theta^{(e)})$
   $\diamond$ For each party $i \in \{N, \dots, 1\}$,
      - If $i \neq 1$, sample $(\pi_i^{(e)}, \boldsymbol{v}_i^{(e)}) \longleftarrow \mathsf{PRG}(\mathsf{salt}, \theta_i^{(e)})$ from $\mathcal{S}_n \times \mathbb{F}_q^n$
      - If $i = 1$, sample $\boldsymbol{v}_1^{(e)} \longleftarrow \mathsf{PRG}(\mathsf{salt}, \theta_1^{(e)})$ from $\mathbb{F}_q^n$
      - If $i \neq 1$, compute $\mathsf{cmt}_{1,i}^{(e)} = \mathsf{H}_0(\mathsf{salt}, e, i, \theta_i^{(e)})$
      - If $i = 1$, compute $\pi_1^{(e)} = (\pi_2^{(e)})^{-1} \circ \cdots \circ (\pi_N^{(e)})^{-1} \circ \pi$ and $\mathsf{cmt}_{1,1}^{(e)} = \mathsf{H}_0(\mathsf{salt}, e, 1, \pi_1^{(e)}, \theta_1^{(e)})$
   $\diamond$ Compute $\boldsymbol{v}^{(e)} = \boldsymbol{v}_N^{(e)} + \sum_{i\in[N-1]} \pi_N^{(e)} \circ \cdots \circ \pi_{i+1}^{(e)}[\boldsymbol{v}_i^{(e)}]$ and $\mathsf{cmt}_1^{(e)} = \mathsf{H}_0(\mathsf{salt}, e, \boldsymbol{H}\boldsymbol{v}^{(e)})$

Step 2: First Challenge

4. Compute $h_1 = \mathsf{H}_1(\mathsf{salt}, m, \mathsf{pk}, (\mathsf{cmt}_1^{(e)}, \mathsf{cmt}_{1,i}^{(e)})_{e\in[\tau], i\in[N]})$
5. Sample $(\kappa_j^{(e)})_{e\in[\tau], j\in[t]} \longleftarrow \mathsf{PRG}(h_1)$ from $(\mathbb{F}_q^t)^\tau$

Step 3: First Response

6. For each iteration $e \in [\tau]$,
   $\diamond$ Compute $\boldsymbol{s}_0^{(e)} = \sum_{j\in[t]} \kappa_j^{(e)} \cdot \boldsymbol{x}_j$
   $\diamond$ For each party $i \in [N]$,
      - Compute $\boldsymbol{s}_i^{(e)} = \pi_i^{(e)}[\boldsymbol{s}_{i-1}^{(e)}] + \boldsymbol{v}_i^{(e)}$

Step 4: Second Challenge

7. Compute $h_2 = \mathsf{H}_2(\mathsf{salt}, m, \mathsf{pk}, h_1, (\boldsymbol{s}_i^{(e)})_{e\in[\tau], i\in[N]})$
8. Sample $(\alpha^{(e)})_{e\in[\tau]} \longleftarrow \mathsf{PRG}(h_2)$ from $([1, N])^\tau$

Step 5: Second Response

9. For each iteration $e \in [\tau]$,
   $\diamond$ Compute $\boldsymbol{z}_1^{(e)} = \boldsymbol{s}_\alpha^{(e)}$
   $\diamond$ If $\alpha^{(e)} \neq 1$, $z_2^{(e)} = (\pi_1^{(e)} \,||\, (\theta_i^{(e)})_{i\in[N]\setminus\alpha^{(e)}})$
   $\diamond$ If $\alpha^{(e)} = 1$, $z_2^{(e)} = (\theta_i^{(e)})_{i\in[N]\setminus\alpha^{(e)}}$
   $\diamond$ Compute $\mathsf{rsp}^{(e)} = (\boldsymbol{z}_1^{(e)}, z_2^{(e)}, \mathsf{cmt}_{1,\alpha^{(e)}}^{(e)})$
10. Compute $\sigma = (\mathsf{salt}, h_1, h_2, (\mathsf{rsp}^{(e)})_{e\in[\tau]})$

Figure 4.7: PERK - Sign algorithm

Inputs

- Public key $\mathsf{pk} = (\boldsymbol{H}, (\boldsymbol{x}_j, \boldsymbol{y}_j)_{j\in[t]})$
- Signature σ
- Message $m \in \{0,1\}^*$

Step 1: Parse signature

1. Parse signature as $\sigma = (\mathsf{salt}, h_1, h_2, (\boldsymbol{z}_1^{(e)}, z_2^{(e)}, \mathsf{cmt}^{(e)}_{1,\alpha^{(e)}})_{e\in[\tau]})$
2. Recompute $(\kappa_j^{(e)})_{e\in[\tau],j\in[t]} \longleftarrow \mathsf{PRG}(h_1)$ from $(\mathbb{F}_q^t)^\tau$
3. Recompute $(\alpha^{(e)})_{e\in[\tau]} \longleftarrow \mathsf{PRG}(h_2)$ from $([1,N])^\tau$

Step 2: Verification

4. For each iteration $e \in [\tau]$,
   - ◇ Compute $\boldsymbol{s}_0^{(e)} = \sum_{j\in[t]} \kappa_j^{(e)} \cdot \boldsymbol{x}_j$ and $\boldsymbol{s}_\alpha^{(e)} = \boldsymbol{z}_1^{(e)}$
   - ◇ Compute $(\pi_i^{(e)}, \boldsymbol{v}_i^{(e)})_{i\in[N]\setminus\alpha}$ from $\boldsymbol{z}_2^{(e)}$
   - ◇ For each party $i \in [N] \setminus \alpha^{(e)}$,
     - If $i \neq 1$, compute $\mathsf{cmt}_{1,i}^{(e)} = \mathsf{H}_0(\mathsf{salt}, e, i, \theta_i^{(e)})$
     - If $i = 1$, compute $\mathsf{cmt}_{1,1}^{(e)} = \mathsf{H}_0(\mathsf{salt}, e, 1, \pi_1^{(e)}, \theta_1^{(e)})$
   - ◇ For each party $i \in [N] \setminus \alpha^{(e)}$,
     - Compute $\boldsymbol{s}_i^{(e)} = \pi_i^{(e)}[\boldsymbol{s}_{i-1}^{(e)}] + \boldsymbol{v}_i^{(e)}$
   - ◇ Compute $\mathsf{cmt}_1^{(e)} = \mathsf{H}_0(\mathsf{salt}, e, \boldsymbol{H}\boldsymbol{s}_N^{(e)} - \sum_{i\in[t]} \kappa_i^{(e)} \cdot \boldsymbol{y}_i)$
5. Compute $\bar{h}_1 = \mathsf{H}_1(\mathsf{salt}, m, \mathsf{pk}, (\mathsf{cmt}_1^{(e)}, \mathsf{cmt}_{1,i}^{(e)})_{e\in[\tau],i\in[N]})$.
6. Compute $\bar{h}_2 = \mathsf{H}_2(\mathsf{salt}, m, \mathsf{pk}, h_1, (\boldsymbol{s}_i^{(e)})_{e\in[\tau],i\in[N]})$.
7. Output accept if and only if $\bar{h}_1 = h_1$ and $\bar{h}_2 = h_2$.

Figure 4.8: PERK - Verify algorithm

for a choice of the $\kappa_i's \in \mathbb{F}_q$, is a solution. Clearly any algorithm applicable to IPKP can also be applied to find a solution to the r-IPKP problem. However, a solution to r-IPKP does not necessarily have to be a solution to IPKP. Therefore algorithms to solve r-IPKP can be split into those initially proposed for IPKP and those specifically designed to solve r-IPKP. In the following section we first describe known approaches for solving IPKP, after which we present a new Algorithm 4.2 specifically designed to solve r-IPKP.

4.2.3.1 Attacks on IPKP

The IPKP problem was introduced by Shamir in 1990 [Sha90]. Still, the best attack on mono-dimensional IPKP is a meet-in-the-middle adaptation known as the KMP algorithm by Koussa, Macario-Rat and Patarin [KMP19]. This algorithm extends easily to $t > 1$, which was recently formalized in [SBC22]. The multi-dimensional IPKP first appeared in the literature in 2011 [LP11]. However, until recently cryptanalysis only resulted in better algorithms for the particular case of binary fields [PT21]. Recently, Santini, Baldi and Chiraluce [SBC22] proposed the SBC algorithm which extends the KMP algorithm by a pre-processing step. For $t > 1$, i.e., for the multi-dimensional case, this results in improvements over the KMP approach. In the following we give a brief overview of those attacks. For fully-fledged descriptions, analysis and estimation scripts the reader is referred to [KMP19, SBC22, EVZB23].

The KMP Algorithm. The algorithm by Koussa, Macario-Rat and Patarin [KMP19] is a slight variant of previously known combinatorial techniques [Geo92, BCCG93, PC94, JJ01]. Here we outline first the initial proposal for the mono-dimensional IPKP [KMP19].

Initially, the matrix $\boldsymbol{H}$ is transformed into semi-systematic form by applying a change of basis

(modelled by the invertible matrix $\boldsymbol{Q}$)

$$\boldsymbol{QH} = \begin{pmatrix} \boldsymbol{I}_{m-u} & \boldsymbol{H}_1 \\ \boldsymbol{0} & \boldsymbol{H}_2 \end{pmatrix},$$

where $\boldsymbol{H}_1 \in \mathbb{F}_q^{(m-u)\times(n-m+u)}$, $\boldsymbol{H}_2 \in \mathbb{F}_q^{u\times(n-m+u)}$ and u is an optimization parameter of the algorithm. By multiplying the syndrome $\boldsymbol{y}$ by the same matrix $\boldsymbol{Q}$ one maintains the validity of the PKP identity

$$\begin{aligned}\boldsymbol{QH}\pi(\boldsymbol{x}) &= \begin{pmatrix} \boldsymbol{I}_{m-u} & \boldsymbol{H}_1 \\ \boldsymbol{0} & \boldsymbol{H}_2 \end{pmatrix} \pi(\boldsymbol{x}) = \begin{pmatrix} \boldsymbol{I}_{m-u} & \boldsymbol{H}_1 \\ \boldsymbol{0} & \boldsymbol{H}_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix} = (\boldsymbol{x}_1 + \boldsymbol{H}_1\boldsymbol{x}_2, \boldsymbol{H}_2\boldsymbol{x}_2)^\top \\ &= (\boldsymbol{y}_1, \boldsymbol{y}_2)^\top = \boldsymbol{Qy},\end{aligned}$$

where $\boldsymbol{Qy} = (\boldsymbol{y}_1, \boldsymbol{y}_2) \in \mathbb{F}_q^{m-u} \times \mathbb{F}_q^u$ and $\pi(\boldsymbol{x}) = (\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathbb{F}_q^{m-u} \times \mathbb{F}_q^u$. The algorithm now focuses on solving the identity $\boldsymbol{H}_2\boldsymbol{x}_2 = \boldsymbol{y}_2$. For any found $\boldsymbol{x}_2$ satisfying the identity it is then checked if $\boldsymbol{x}_1 = \boldsymbol{y}_1 - \boldsymbol{H}_1\boldsymbol{x}_2$ and $\boldsymbol{x}_2$ together form a permutation of $\boldsymbol{x}$.

Candidates for $\boldsymbol{x}_2$ are obtained by a meet-in-the-middle enumeration strategy. Therefore $\boldsymbol{x}_2$ is further split as $\boldsymbol{x}_2 = (\boldsymbol{x}_{21}, \boldsymbol{x}_{22})$, with $\boldsymbol{x}_{21}, \boldsymbol{x}_{22} \in \mathbb{F}_q^{u\times((n-m+u)/2)}$ to obtain the meet-in-the-middle identity

$$\boldsymbol{H}_2(\boldsymbol{x}_{21}, \boldsymbol{0}) = \boldsymbol{y}_2 - \boldsymbol{H}_2(\boldsymbol{0}, \boldsymbol{x}_{22}). \tag{4.6}$$

Then the algorithm enumerates all candidates for $\boldsymbol{x}_{21}$ and $\boldsymbol{x}_{22}$, that is all permutations of any selection of $(n-m+u)/2$ entries of $\boldsymbol{x}$. For each such vector the left (resp. right) side of Equation (4.6) is stored in a list L_1 (resp. L_2). In a final step the algorithm searches for matches between the lists L_1 and L_2 yielding the candidates for $\boldsymbol{x}_2$. From there $\boldsymbol{x}_1$ can be computed as $\boldsymbol{x}_1 = \boldsymbol{y}_1 - \boldsymbol{H}_1\boldsymbol{x}_2$. If $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ forms a permutation of $\boldsymbol{x}$ this yields the solution π.

The complexity of the algorithm is (up to polynomial factors) linear in the sizes of the lists L_1, L_2 and L, where L is the list of matches. The expected sizes are

$$|L_1| = |L_2| = \binom{n}{(n-m+u)/2}((n-m+u)/2)! \quad \text{and} \quad |L| = \frac{|L_1 \times L_2|}{q^u}$$

Extension to multi-dimensional IPKP. The algorithm can easily be extended to solve IPKP for arbitrary t, as it was recently made explicit in [SBC22]. Therefore let $\boldsymbol{X}$ be the matrix containing the $\boldsymbol{x}_i$ as rows and $\boldsymbol{Y}$ containing the $\boldsymbol{y}_i$ as columns. Substituting the occurrences of $\boldsymbol{x}$ and $\boldsymbol{y}$ by $\boldsymbol{X}$ and $\boldsymbol{Y}$ resp., where the permutation now operates as a column permutation on matrices, one obtains this generalization. Then of course the definition of $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ and $(\boldsymbol{y}_1, \boldsymbol{y}_2)$ analogously extends to matrices.

In terms of complexity, the enumeration effort stays (up to polynomial factors) exactly the same, as the possible number of permutations remains unchanged. The only difference is that the expected size of the list L of matches reduces to $\frac{|L_1\times L_2|}{q^{u\cdot t}}$.

The SBC algorithm. The algorithmic improvement by Santini-Baldi-Chiraluce (SBC) extends the KMP algorithm by a preprocessing step.

Therefore, assume that the matrix $\boldsymbol{H}_2$ constructed in the KMP algorithm would contain zero columns. Clearly, those columns could be removed without affecting the validity of the identity $\boldsymbol{H}_2\boldsymbol{x}_2 = \boldsymbol{y}_2$. But in turn this would reduce the enumeration effort to find candidates for $\boldsymbol{x}_2$.

The preprocessing step of the SBC algorithm now consists in finding a u-dimensional subcode of $\boldsymbol{H}$ that has small support $w < n - m + u$, i.e., an $\boldsymbol{H}_2$ that contains some zero columns. This can be accomplished by adaptations of Information Set Decoding (ISD) algorithms [Beu20a]. Subsequently, the SBC algorithm continues as the KMP algorithm by finding candidates for $\boldsymbol{x}_2$ in $\boldsymbol{H}_2\boldsymbol{x}_2 = \boldsymbol{y}_2$, now with reduced enumeration complexity. This resulting list of candidates is now treated as the first list, L_1, in the KMP algorithm.

Note that the KMP algorithm creates two lists each giving candidates for $(n-m+u)/2$ entries of the permutation. Now, as there are already candidates for w entries in L_1, the second list enumerates the permutation for further $n - m + u - w$ positions. Eventually both lists are matched

on $u \cdot t$ coordinates as in the KMP algorithm to obtain a list of final candidates. Note that as in the KMP algorithm now each candidate of the final list reveals $n - m + u$ potential positions of the permutation which can be checked in polynomial time for extending to a full solution.

4.2.3.2 A new Algorithm Solving r-IPKP

We introduce the r-IPKP problem together with our scheme. However, it is still very related to the multi-dimensional and mono-dimensional versions of IPKP and their corresponding hardness. We already discussed the relation to the *multi-dimensional* case of IPKP. Let us now focus on the relation between r-IPKP and the *mono-dimensional* version of IPKP. In this context r-IPKP can be seen as a multi-instance version of IPKP.

Therefore disregard the (most likely) unique solution to r-IPKP which simultaneously solves IPKP for the same t, i.e. the permutation that works for all pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$. Further, assume that for any of the given pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ there exist a permutation π_i solving the corresponding mono-dimensional IPKP instance, that is a π_i that satisfies $\boldsymbol{H}\pi_i[\boldsymbol{x}_i] = \boldsymbol{y}_i$. If we would be forced to recover one of the π_i, this would exactly be a multi-instance version of IPKP. However, the r-IPKP allows to recover not only those but also any permutation that works for an arbitrary linear combination of the given pairs. Clearly, this gives a total of q^t different pairs, but unlike the multi-instance case those pairs are related.

In fact, from a coding theory perspective the r-IPKP asks to recover a permutation that works for some codeword and the corresponding syndrome where the code is defined by the generator matrix containing the $\boldsymbol{x}_i$s as rows. In the following we give a new algorithm for solving r-IPKP that exploits this view on the problem. The algorithm is based on a preprocessing of the given pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, a subsequent instance permutation and an adapted KMP-style enumeration technique. Moreover, our algorithm contains the KMP algorithm as a special case.

The algorithm starts by finding a low Hamming-weight codeword in the code whose generator matrix has the $\boldsymbol{x}_i$ as rows. This task is accomplished by application of an ISD algorithm. Let $\boldsymbol{x}' = \sum \kappa_i \boldsymbol{x}_i$ be this codeword of weight w and $\boldsymbol{y}' = \sum \kappa_i \boldsymbol{y}_i$ the corresponding syndrome.

We now focus on finding a permutation π that satisfies $\boldsymbol{H}(\pi[\boldsymbol{x}']) = \boldsymbol{y}'$. Therefore, we apply a KMP-style enumeration with some modifications. Again we derive the identity $\boldsymbol{H}_2\boldsymbol{x}_2 = \boldsymbol{y}_2 \in \mathbb{F}_q^u$, with $\boldsymbol{x}_2 = (\boldsymbol{x}_{21}, \boldsymbol{x}_{22})$ as in the usual KMP algorithm. Now, recall that $\pi[\boldsymbol{x}'] = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ contains $n - w$ zeros. For the enumeration of $\boldsymbol{x}_{21}$ and $\boldsymbol{x}_{22}$ we now assume that z of those zeros are mapped into $\boldsymbol{x}_2$ by the permutation. Moreover, we assume that $z/2$ of those zeros are mapped to $\boldsymbol{x}_{21}$ and $z/2$ to $\boldsymbol{x}_{22}$. This leads to a reduced amount of candidates for $\boldsymbol{x}_{21}, \boldsymbol{x}_{22}$ that has to be enumerated.

Of course we do not know a priori if the permutation indeed distributes $z/2$ zeros onto $\boldsymbol{x}_{21}$ and $z/2$ zeros onto $\boldsymbol{x}_{22}$. Therefore prior to the enumeration we apply a random permutation to the columns of $\boldsymbol{H}$ to redistribute the weight (and zeros) of $\pi[\boldsymbol{x}']$. If the enumeration does not lead to a solution we repeat with a different column permutation of $\boldsymbol{H}$.

A pseudocode description is given in Algorithm 4.2. Note that for $w = n$, i.e., a maximum-weight codeword, we resemble the standard KMP algorithm for solving mono-dimensional IPKP.

Analysis of Algorithm 4.2. Let us start with the correctness of the algorithm.

Correctness. Note that the permuted instance $(\boldsymbol{H}^*, \boldsymbol{x}', \boldsymbol{y}')$ with $\boldsymbol{H}^* = \pi'[\boldsymbol{H}]$ has solution $\pi' \circ \pi$ if π solves the original instance $(\boldsymbol{H}, \boldsymbol{x}', \boldsymbol{y}')$. Therefore the algorithm correctly returns $(\pi')^{-1} \circ \tilde{\pi}$ as the solution to the original instance, where $\tilde{\pi}$ solves the permuted instance.

Accordingly the solution to the permuted instance is $(\boldsymbol{x}_1, \boldsymbol{x}_2) = \pi'[\pi[\boldsymbol{x}']]$. Line 9 to 11 enumerate all candidates for $\boldsymbol{x}_2$ satisfying $\boldsymbol{H}_2\boldsymbol{x}_2 = \boldsymbol{y}_2$, where $\boldsymbol{x}_2 = (\boldsymbol{z}_1, \boldsymbol{z}_2)$ with each of the $\boldsymbol{z}_i$ containing $z/2$ zeros. Note that by construction $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ contains $n - w$ zeros. As the permutation π' redistributes the zeros the algorithm can recover the permutation.

Complexity. The complexity of the algorithm splits into the cost of finding the short codeword $\boldsymbol{x}'$ and the cost of the `repeat` loop. The codeword is found by application of an ISD algorithm. Let us denote this cost by T_{ISD}.

The cost of the loop is equal to the amount of repetitions times the cost of one iteration. The

Algorithm 4.2 Algorithm Solving r-IPKP

Input: r-IPKP instance $(\boldsymbol{H}, (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i\in[t]})$
Output: solution π, $\kappa_i \in \mathbb{F}_q$, $i \in [t]$

For a vector $\boldsymbol{x}$ and an integer k let $\mathcal{P}_{\boldsymbol{x},k}$ be the set of vectors of length k with entries from $\boldsymbol{x}$ (with their maximum occurence as in $\boldsymbol{x}$).
Choose optimal positive $u \leq n-m$, $w \leq n$, and $z \leq n-w$, let $k := (n-m+u)/2$
Find weight-w codeword $\boldsymbol{x}' = \sum_i \kappa_i \boldsymbol{x}_i$ in the code defined by the $\boldsymbol{x}_i$
Let $\boldsymbol{y}' = \sum_i \kappa_i \boldsymbol{y}_i$
repeat
    choose random permutation π'
    $\boldsymbol{H}^* = \pi'[\boldsymbol{H}]$
    $\boldsymbol{H}' = \boldsymbol{Q}\boldsymbol{H}^* = \begin{pmatrix} \boldsymbol{I}_{m-u} & \boldsymbol{H}_1 \\ \boldsymbol{0} & \boldsymbol{H}_2 \end{pmatrix}$, $(\boldsymbol{y}_1, \boldsymbol{y}_2) = \boldsymbol{Q}\boldsymbol{y}$
    $L_1 = \{(\boldsymbol{H}_2(\boldsymbol{z}_1, 0^k), \boldsymbol{z}_1) \mid \boldsymbol{z}_1 \in \mathcal{P}_{\boldsymbol{x},k} \wedge \|\boldsymbol{z}_1\| = k - z/2\}$
    $L_2 = \{(\boldsymbol{y}_2 - \boldsymbol{H}_2(0^k, \boldsymbol{z}_2), \boldsymbol{z}_2) \mid \boldsymbol{z}_2 \in \mathcal{P}_{\boldsymbol{x},k} \wedge \|\boldsymbol{z}_2\| = k - z/2\}$
    Compute $L = \{(\boldsymbol{z}_1, \boldsymbol{z}_2) \in L_1 \times L_2 \mid \boldsymbol{H}_2(\boldsymbol{z}_1, \boldsymbol{z}_2) = \boldsymbol{y}_2\}$ from L_1, L_2
    for Each $\boldsymbol{x}_2 \in L$ **do**
        $\boldsymbol{x}_1 = \boldsymbol{y}_1 - \boldsymbol{H}_1\boldsymbol{x}_2$

        if $\exists \tilde{\pi}\colon \tilde{\pi}[\boldsymbol{x}_1, \boldsymbol{x}_2] = \boldsymbol{x}'$ **then return** $(\pi')^{-1} \circ \tilde{\pi}$
until false

amount of different permutations until the zeros are distributed as desired is

$$P = \frac{\binom{n}{n-w}}{\binom{n-2k}{n-w-z}\binom{k}{z/2}^2},$$

where $k = (n-m+u)/2$. The cost for one iteration is (up to polynomial factors) linear in the involved lists' sizes. Note that we have

$$|L_i| = \binom{k}{z/2}\binom{n-z}{k-z/2}(k-z/2)! \quad \text{and} \quad |L| = \frac{|L_1 \times L_2|}{q^u}.$$

The total time complexity therefore amounts to

$$T = \tilde{\mathcal{O}}\left(T_{\text{ISD}} + (|L_1| + |L|) \cdot P\right),$$

while the memory complexity is equal to $M = \tilde{\mathcal{O}}\left(|L_1| + |L|\right)$.

In our numerical optimization we use for T_{ISD} the basic ISD procedure by Prange [Pra62] which gives

$$T_{\text{ISD}} = \tilde{\mathcal{O}}\left(\frac{\binom{n}{w}}{\binom{n-m}{w}}\right).$$

There are more sophisticated ISD procedures with lower cost, but as T_{ISD} does not dominate the running time, we refrain from further optimizations.

Note that this running time assumes a single existing solution for the considered mono-dimensional instance solved. In case of multiple solutions the running time can be lower, which we discuss in the next section.

Further improvement of Algorithm 4.2. For completeness, we point out that the algorithm can be (slightly) improved by considering in L_1 and L_2 vectors of weight $k - z \leq \|\boldsymbol{z}_i\| \leq k - z/2$. This would only insignificantly increase the list sizes, while giving a slightly higher probability of the permutation distributing the weight as desired. However, the overall improvement is a factor

of order roughly 2. For small values of $t \leq 5$ as considered later, the factor turns out to be smaller than $\sqrt{2}$. Therefore for the sake of clarity we omit this improvement.

Another intuitive strategy seems the consideration of small support subcodes instead of small codewords in the code defined by the $\boldsymbol{x}_i$. This would lead to a reduced amount of matches, i.e., reduced list size L at the expense of larger w and correspondingly smaller z. However, the nature of instances considered in PERK renders this strategy ineffective. As detailed later, instances are chosen, such that for any single codeword there exist exponentially many solutions, leading to an exponential speedup. On the other hand already when considering a subcode of dimension two, the only existing solution is the permutation solving the IPKP defined by all pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$. We find that the reduced amount of matches from considering subcodes does not compensate for the speedup from the amount of solutions. We therefore again omit further details for the sake of clarity.

4.2.3.3 Concrete Complexity of Solving r-IPKP

In this section we give details on how the variation of different parameters affects the hardness of r-IPKP. A solid understanding of those effects is crucial for secure parameter selection. As outlined previously, for solving the r-IPKP one can either directly apply an IPKP algorithm (see Section 4.2.3.1) or solve one of the single IPKP instances defined by any linear combination of input pairs (see Section 4.2.3.2). Which of the two attack strategies is more efficient depends on the particular choice of parameters.

Effect of the number of solutions. Multiple existing solutions can lead to a maximum speedup that is linear in this amount of solutions. Whether this maximum speedup can be realized depends on the particular algorithm. However, for our parameter selection we conservatively assume that any algorithm can leverage this maximum speedup.

Note that the expected number of solutions differs for the considered sub-problems. The expected number of solutions for any random IPKP instance is about $\mathtt{Sol}_{\text{IPKP}} = \frac{n!}{q^{m \cdot t}}$. Note that the *mono*-dimensional IPKP instance solved in the context of our new Algorithm 4.2 is not random but contains z zeros. In this case the amount of expected solutions is only $\mathtt{Sol}^{\text{mono}}_{\text{IPKP},z} = \frac{n!}{q^m z!}$.

Effect of t on the running time. Santini et al. [SBC22] observed that the running time of the KMP algorithm as well as the running time of their SBC algorithm for IPKP is asymptotically independent of t.[3] For concrete parameters, these algorithms still yield speedups for increasing t but the running time quickly converges to a stable minimum. Therefore, based on known algorithms the hardness of IPKP does not seem to deteriorate for high values of t. Contrary, the complexity of our new Algorithm 4.2 is monotonically decreasing for increasing t. This shows that high choices of t in the r-IPKP context are vulnerable.

To visualize this we consider in Table 4.7 a fixed instance for increasing t. The SBC algorithm reaches its minimum running time already for $t = 10$, while Algorithm 4.2 constantly improves. However, the SBC algorithm has a lower complexity for small choices of t and obtains a larger gain for early increases.

Note that for the chosen parameters in Table 4.7 already a random mono-dimensional IPKP instance has at most one solution in expectation, i.e. $\mathtt{Sol}^{\text{mono}}_{\text{IPKP}} \leq 1$.

Effect of m on the running time. Generally the hardness of IPKP is increasing with decreasing m. This holds up to the point where there exist multiple solutions. Previous parameter selection for PKP-based schemes therefore chooses m minimal such that there exists no more than one random solution in expectation. However, for the specific case of the r-IPKP problem, the two sub-problems, i.e., the multi- and mono-dimensional IPKP instances, have a different amount of expected solutions. Here, decreasing m leads, generally, only to an increase of the problem complexity as long as the solution to both sub-problems is still unique.

[3] Informally, this can be seen by observing that t only affects the amount of matches, i.e. the size of L (compare to Section 4.2.3.1). However, asymptotically the size of the initial lists L_i and L are balanced, therefore a decrease of L does not lead to runtime improvements.

t	T_{new}	T_{SBC}
1	139.35	139.35
2	139.01	116.25
3	137.46	110.70
10	115.98	88.18
15	100.27	88.18
19	89.14	88.18
20	85.63	88.18
25	71.77	88.18
30	63.47	88.18

Table 4.7: Complexity of SBC algorithm (T_{SBC}) and Algorithm 4.2 (T_{new}) for increasing t on r-IPKP(n, m, q, t) instance with $(n, m, q) = (66, 31, 1021)$.

4.2.4 Parameters

In this section we present the parameters of our scheme. Generally the parameters divide into r-IPKP specific parameters, i.e., (q, n, m, t), and MPC parameters, i.e., the number of parties N and the number of parallel repetitions τ. The rationale for their selection are as follows.

Selection of MPC parameters. The number of parties and iterations is governed by the knowledge soundness of the protocol. Following common practice we propose two different parameter sets, a *short* variant using $N = 256$ and a fast variant using $N = 32$. The number of protocol repetitions τ is then chosen to guarantee a soundness probability of $2^{-\lambda}$ for $\lambda \in \{128, 192, 256\}$ for category I, III and V respectively. For deriving the soundness we take into account the attack by Kales-Zavurecha, see Appendix C.3.

Selection of problem parameters. For parameter selection we ensure that the complexity of the SBC algorithm as well as the complexity of our new Algorithm 4.2 are above the security threshold, when assuming a linear speedup from the existing amount of solutions. Note that it is important to consider both strategies as the IPKP suggests to decrease m to increase the difficulty of the problem. This is related to the low amount of expected solutions $\texttt{Sol}_{\text{IPKP}}$, which allows to decrease m significantly without introducing multiple solutions. Contrary, for any mono-dimensional instance given by the possible linear combinations, there exist several solutions $\texttt{Sol}^{\text{mono}}_{\text{IPKP}}$ for such small choices of m, which decrease the complexity of Algorithm 4.2.

Note that we, conservatively, restrict in our parameter selection to small choices of $t \in \{3, 5\}$ to guard against attacks that exploit the specifics of r-IPKP over IPKP. For such small values of t, the SBC algorithm has generally a lower complexity than Algorithm 4.2 (compare to Table 4.7). In those cases, the parameter selection process leads to a choice of m which implies a unique solution to the multi-dimensional IPKP instance, while there exist multiple solutions to the mono-dimensional instance solved in the context of Algorithm 4.2. This leads to a balancing of both complexities via the amount of solutions.

In our complexity estimations we ignore polynomial factors and ensure that the exponential factors of the complexity formulas already match the NIST security level definitions of 143, 207 and 272 bits of category I, III and V respectively. For the complexity estimation of the SBC algorithm we rely on the *CryptographicEstimators* library[4] incorporating a more efficient version of the script from [SBC22]. For our algorithm we rely on a separate script.

Overall this leads to the choices of parameters specified in Table 4.8.

Note that even though we are quite conservative in parameter selection by restricting to small choices of t, disregarding polynomial factors and assuming a maximum speedup from multiple so-

[4] https://github.com/Crypto-TII/cryptographic_estimators

		PKP parameters				MPC parameters			
Parameter Set	λ	q	n	m	t	N	τ	pk size	σ size
PERK-I-fast3	128	1021	79	35	3	32	30	0.15 kB	8.35 kB
PERK-I-fast5	128	1021	83	36	5	32	28	0.24 kB	8.03 kB
PERK-I-short3	128	1021	79	35	3	256	20	0.15 kB	6.56 kB
PERK-I-short5	128	1021	83	36	5	256	18	0.24 kB	6.06 kB
PERK-III-fast3	192	1021	112	54	3	32	46	0.23 kB	18.8 kB
PERK-III-fast5	192	1021	116	55	5	32	43	0.37 kB	18.0 kB
PERK-III-short3	192	1021	112	54	3	256	31	0.23 kB	15.0 kB
PERK-III-short5	192	1021	116	55	5	256	28	0.37 kB	13.8 kB
PERK-V-fast3	256	1021	146	75	3	32	61	0.31 kB	33.3 kB
PERK-V-fast5	256	1021	150	76	5	32	57	0.51 kB	31.7 kB
PERK-V-short3	256	1021	146	75	3	256	41	0.31 kB	26.4 kB
PERK-V-short5	256	1021	150	76	5	256	37	0.51 kB	24.2 kB

Table 4.8: Parameters of PERK signature scheme

lutions, we obtain competitive signature sizes. Also all considered algorithms use as much memory as they consume time, which in a more realistic estimate that accounts for memory access leads to an even higher security margin.

In Table 4.9 we provide the corresponding time complexities in logarithmic scale of the SBC algorithm and Algorithm 4.2 (accounting for a linear speedup from multiple existing solutions) on the suggested parameter sets. As outlined, the parameter sets are chosen to balance both time complexities. Additionally the tables provide the internal parameters of the algorithms. In the case of Algorithm 4.2 we find that for small values of t as considered here, the choice $w = n - z$ is optimal, meaning all contained zeros should be contained in the part of the vector which is enumerated. For the SBC algorithm, recall that u is the dimension of the subcode and we denote by z the amount of zero columns in $\boldsymbol{H}_2$ (compare to Section 4.2.3.1), i.e., the subcode has support $w = n - m + u - z$.

	SBC [SBC22]			Algorithm 4.2		
Parameter Set	time	u	z	time	u	z
PERK-I-fast3	145.7	5	1	147.5	17	2
PERK-I-fast5	147.7	3	2	147.2	19	0
PERK-III-fast3	210.7	7	1	210.1	26	2
PERK-III-fast5	212.5	4	2	210.8	27	4
PERK-V-fast3	274.8	9	1	275.5	35	4
PERK-V-fast5	274.1	6	2	275.5	37	3

Table 4.9: Parameters of PERK signature scheme

4.2.4.1 Key and Signature Sizes

Table 4.8 already states the public key and signature sizes for the different parameter sets. Let us give an overview how those numbers are composed.

Key size. The private key as well as most of the components of the public key can be derived from a seed. The only elements not generated from a seed in the public key are the t syndromes $(\boldsymbol{y}_i)$.

Public key size (bits)
$\lambda + t \cdot m\lceil \log_2(q) \rceil$

Signature size (bits)
$6\lambda + \tau \cdot \big(\underbrace{n\lceil \log_2(q) \rceil}_{\text{vector in } \mathbb{F}_q^n} + \underbrace{n\lceil \log_2(n) \rceil}_{\text{permutation}} + \underbrace{\lambda\lceil \log_2(N) \rceil}_{\text{seeds}} + \underbrace{2\lambda}_{\text{commitment}} \big)$

Table 4.10: Public key and signature sizes in bits

Signature size. The signature consists of a salt and two hashes (h_1, h_2), making a subtotal of 6λ bits, and then τ repetitions of the following:

- A vector $\boldsymbol{z}_1^{(e)} \in \mathbb{F}_q^n$;
- A permutation in $\mathcal{S}_n$;
- $N - 1$ seeds (of size λ) arranged in a PRG tree, hence of size only $\lambda \cdot \lceil \log_2(N) \rceil$;
- A commitment $\mathsf{cmt}_{1,\alpha^{(e)}}^{(e)}$ of size 2λ.

Overall, for a security level λ, the key and signature sizes for our signature scheme are captured by the following formulas:

Signature compression. Our implementation features an optimization that further reduces the aforementioned signature theoretical size. The idea is to pack the permutation two by two. Instead of representing a permutation $\pi \in \mathcal{S}_n$ as a sequence of n elements in $[0, n-1]$ it is represented as a sequence of $\lceil n/2 \rceil$ elements in $[0, n^2 - 1]$. When the following inequality holds

$$\lceil \log_2(n^2) \rceil < 2\lceil \log_2(n) \rceil,$$

the packed permutation takes less memory size. Numbers given in Table 4.8 take into account this compression technique.

4.2.4.2 Comparison

We compare our scheme with other signature schemes, either based on PKP or based on other security assumptions. The results are presented in Table 4.11 and Table 4.12.

4.2.5 Performances

This section provides performance measures of PERK signature. Our constant-time implementation is written in C, and uses AVX2 vector instructions. The benchmarks have been performed on a machine that has 64 GB of memory and an Intel® Core™ i9-13900K @ 3.00 GHz for which the Hyper-Threading, Turbo Boost and SpeedStep features were disabled. For each parameter set, the results have been obtained by computing the average from 1000 random instances. The following optimization flags have been used during compilation: `-O3 -std=c99 -pedantic -funroll-all-loops -march=native -mavx2 -mpclmul -msse4.2 -maes`.

Name	Variant	pk	σ	Security assumption
PKP-DSS [BFK+19]	-	0.1 kB	21.0 kB	PKP
SUSHYFISH [Beu20b]	fast	0.1 kB	18.4 kB	IPKP
	short	0.1 kB	12.1 kB	IPKP
BG22 [BG23]	fast	0.1 kB	10.0 kB	IPKP
	short	0.1 kB	8.9 kB	IPKP
Fen22 [Fen22]	fast	0.1 kB	16.4 kB	IPKP
	short	0.1 kB	12.8 kB	IPKP
PERK-I-fast3	fast	0.15 kB	8.35 kB	r-IPKP
PERK-I-short5	**short**	**0.24 kB**	**6.06 kB**	r-IPKP

Table 4.11: Comparison of our scheme with other digital signature schemes based on PKP assumptions

Name	Variant	pk	σ	Security assumption
SPHINCS+ [HBD+22]	fast	0.03 kB	17.1 kB	Hash Collisions
	short	0.03 kB	7.9 kB	Hash Collisions
FJR22 [FJR22]	fast	0.1 kB	9.7 kB	SD over $\mathbb{F}_{256}$
	short	0.1 kB	6.9 kB	SD over $\mathbb{F}_{256}$
Fen22 [Fen22]	fast	0.1 kB	7.4 kB	RSD over $\mathbb{F}_2$
	short	0.1 kB	5.9 kB	RSD over $\mathbb{F}_2$
Fen22 [Fen22]	fast	0.1 kB	7.2 kB	MinRank over $\mathbb{F}_{16}$
	short	**0.1 kB**	**5.5 kB**	MinRank over $\mathbb{F}_{16}$
Fen22 [Fen22]	fast	0.1 kB	8.5 kB	$\mathcal{MQ}$ over $\mathbb{F}_{256}$
	short	0.1 kB	7.1 kB	$\mathcal{MQ}$ over $\mathbb{F}_{256}$
R-BG [BBP+23]	fast	0.1 kB	7.7 kB	Restricted-SD
	short	0.1 kB	7.2 kB	Restricted-SD
PERK-I-fast3	fast	0.15 kB	8.35 kB	r-IPKP
PERK-I-short5	short	0.24 kB	6.06 kB	r-IPKP

Table 4.12: Comparison of our scheme with other digital signature schemes not based on PKP assumptions

Parameter Set	Keygen	Sign	Verify
PERK-I-fast3	77 k	7.6 M	5.3 M
PERK-I-fast5	88 k	7.2 M	5.1 M
PERK-I-short3	80 k	39 M	27 M
PERK-I-short5	92 k	36 M	25 M
PERK-III-fast3	167 k	16 M	13 M
PERK-III-fast5	184 k	15 M	12 M
PERK-III-short3	174 k	82 M	65 M
PERK-III-short5	194 k	77 M	60 M
PERK-V-fast3	297 k	36 M	28 M
PERK-V-fast5	322 k	34 M	27 M
PERK-V-short3	299 k	184 M	142 M
PERK-V-short5	329 k	170 M	131 M

Table 4.13: Performances of our implementation for different instances of PERK. The key generation numbers are in kilo CPU cycles, while the signing and verification numbers are in million CPU cycles.

4.3 NIST submissions: PERK, MIRA and RYDE

During our doctoral studies, we participed to NIST onramp call for additional digital signature proposals [NISa]. We were involved in three candidates:

- MIRA [ABB+23c]
- PERK [ABB+23a]
- RYDE [ABB+23b]

They are signature schemes built from the MPC-in-the-Head paradigm, each of them relying on a different difficult problem: MinRank for MIRA; a variant of PKP for PERK; and RSD for RYDE. All of them achieve very competitive sizes with a public key never exceeding a hundred bytes and signature sizes ranging from 5.6 kB to 8.0 kB.

Chapter 5

Code-based homomorphic encryption

Contents

In this chapter we present a secret-key encryption scheme based on random rank metric ideal linear codes with a simple decryption circuit. It supports unlimited homomorphic additions and plaintext absorptions as well as a fixed arbitrary number of homomorphic multiplications.

Our scheme is an Aleknovich-inspired [Ale03] construction. It can be seen as a secret key version of the NIST Round 2 candidate RQC [AAB+19], with important differences (see below). The ciphertext is a pair $(\boldsymbol{u}, \boldsymbol{v})$ of vectors in $\mathbb{F}_{q^m}^n$ where $\boldsymbol{v}$ is the noisy version of the multiplication $\boldsymbol{u} \cdot \boldsymbol{s}$ of the first component of the ciphertext times the secret key $\boldsymbol{s}$. The noise is taken in a secret space and contains an encoding of the message $\boldsymbol{m}$ being a vector in $\mathbb{F}_q^n$. The ciphertext space has an $\mathbb{F}_q^n$-module structure which makes addition and plaintext multiplication completely straightforward. The additive scheme is presented in Section 5.1 and is augmented with multiplication in Section 5.2.

Contrary to RQC in which the encoded message is a codeword of a public Gabidulin code that can be recovered from the noise using a decoding algorithm, in our construction the message is encoded into a vector space orthogonal to the error vector. The decryption algorithm is thus quite simple since it consists in a secret orthogonal projection of the noise term (i.e. a scalar product with a secret basis). Consequently, a natural homomorphic decryption algorithm (Section 5.3) can be designed. The key switching material consists in encrypted coordinates of the previous key and projection vector, split against a public basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. By splitting the ciphertext onto the same basis and plaintext multplying each component to the key switching material, one

obtains a fresh ciphertext under a new key with no multiplication, only with additions and plaintext multiplications.

The security of our scheme can be reduced to the well-studied ideal rank syndrome decoding problem (IRSD). The reduction is presented in Section 5.4. The rate of the code tends towards 0 as the number of independent ciphertexts increases, giving an upper bound of $2w$ (where w is the rank weight of the error term in the ciphertext) to the number of ciphertexts that can be safely published. This prevents from using safely the homomorphic decryption algorithm which requires $2m$ ciphertexts: one for each of the m components against the $\mathbb{F}_{q^m}$-basis of the secret key and the projection vector.

Aiming at reducing the number of ciphertexts necessary to a homomorphic decryption, we finally present in Section 5.5 a way to pack several plaintexts in a single ciphertext. Instead of having the message encoded in a single dimension orthogonal to the error, the idea is to increase the dimension of the encoded message, which is now a matrix with components in $\mathbb{F}_q$. Rows of the matrix can be rotated using a public operation that allows to perform homomorphic linear combinations on the rows of the plaintext matrix. Because the ciphertext now contains more information, the size of the bootstrapping material is reduced to $2(w+1)$. However, this is still higher than the secure upper bound $2w$.

Finally, concrete parameters for our scheme are presented in Section 5.6.

5.1 Additive scheme

In this section we present an additive secret key encryption scheme. It can also multiplicatively absorb a plaintext.

5.1.1 Fundamental Algorithms

The three polynomial-time algorithms constituting our additive scheme AHE (for Additively Homomorphic Encryption) are depicted in Figure 5.1. The reader is referred to the Notation section for a reminder of the meanings of the symbols $\cdot, \star$ and $\boldsymbol{b}^{(i)}$.

The scheme is parametrized by:

- q, the base field cardinality;
- m, the dimension of the field extension;
- n, the length of the vectors;
- w, the rank weight of the error; it must be that $w < m$.

Remark. $\mathsf{Encrypt}_{\mathsf{AHE}}$ and $\mathsf{Decrypt}_{\mathsf{AHE}}$ are functions, not randomized algorithms. In general we will hide the randomness in the encryption function: $\mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}, \boldsymbol{m})$ being the randomized algorithm that samples $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q)$ and returns $\mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}, \boldsymbol{m}, \boldsymbol{r})$.

Remark. In $\mathsf{Encrypt}_{\mathsf{AHE}}$, having $\boldsymbol{e} = \boldsymbol{f}\boldsymbol{R}_2$ with $\boldsymbol{R}_2 \xleftarrow{\$} \mathcal{M}_{w,n}(\mathbb{F}_q)$ is equivalent to having $\boldsymbol{e} \xleftarrow{\$} F^n$. As for $\hat{\boldsymbol{m}}$, it belongs to $(\langle \boldsymbol{g}^{(1)} \rangle_{\mathbb{F}_q})^n$.

Proposition 5.1.1 (Fresh Ciphertext Decryption Correctness)**.** *For* $\mathsf{sk} \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{AHE}}()$, $\boldsymbol{m} \in \mathbb{F}_q^n$ *and* $\boldsymbol{r} \in \mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q)$ *it holds that*

$$\mathsf{Decrypt}_{\mathsf{AHE}}\left(\mathsf{sk}, \mathsf{Encrypt}_{\mathsf{AHE}}\left(\mathsf{sk}, \boldsymbol{m}, \boldsymbol{r}\right)\right) = \boldsymbol{m}.$$

Proof. We suppose in this proof that the $\mathsf{KeyGen}_{\mathsf{AHE}}$ and $\mathsf{Encrypt}_{\mathsf{AHE}}$ protocols are well defined and can be executed properly. We just note that $\boldsymbol{B}^{-1}$ exists as $\boldsymbol{b}$ is a basis of $\mathbb{F}_{q^m}$ and thus $\boldsymbol{B}$ is of full rank.

As $\boldsymbol{d}$ is the $(w+1)$-th column vector of $(\boldsymbol{B}^{-1})^T$, we thus have that $\boldsymbol{d}^T \mathbf{vec}(\boldsymbol{g}^{(1)}) = 1 \in \mathbb{F}_q$ and $\boldsymbol{d}^T \mathbf{vec}(x) = 0$ for any $x \in \boldsymbol{b} \setminus \{\boldsymbol{g}^{(1)}\}$.

- $\mathsf{KeyGen}_{\mathsf{AHE}}()$:
  - samples $\boldsymbol{f} = (f_1, \ldots, f_w) \xleftarrow{\$} \mathcal{S}_w^w(\mathbb{F}_{q^m})$
  - extends $\boldsymbol{f}$ into a basis $\boldsymbol{b} = (f_1, \ldots, f_w, g_1, \ldots, g_{m-w}) \in \mathcal{S}_m^m(\mathbb{F}_{q^m})$
  - defines $\boldsymbol{g} = (g_1, \ldots, g_{m-w})$
  - computes the matrix $\boldsymbol{B} = \mathbf{Mat}(\boldsymbol{b})$
  - defines $\boldsymbol{D}$ as the last $m - w$ columns of its transposed inverse $(\boldsymbol{B}^{-1})^T$
  - samples $\boldsymbol{s} \xleftarrow{\$} F^n$ with $F = \mathbf{Supp}(\boldsymbol{f})$
  - returns $\mathsf{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s})$.
- $\mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s}), \boldsymbol{m} \in \mathbb{F}_q^n, \boldsymbol{r} \in \mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q))$: notes $(\boldsymbol{r}_1, \boldsymbol{R}_2) = \boldsymbol{r}$, defines $\boldsymbol{u} = \boldsymbol{r}_1$, $\boldsymbol{e} = \boldsymbol{f}\boldsymbol{R}_2$ and sets $\boldsymbol{v} = \boldsymbol{s} \cdot \boldsymbol{u} + \boldsymbol{e} + \hat{\boldsymbol{m}}$ with $\hat{\boldsymbol{m}} = \boldsymbol{g}^{(1)} \star \boldsymbol{m} \in \mathbb{F}_{q^m}^n$. Returns $\mathsf{ct} = (\boldsymbol{u}, \boldsymbol{v})$.
- $\mathsf{Decrypt}_{\mathsf{AHE}}(\mathsf{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s}), \mathsf{ct} = (\boldsymbol{u}, \boldsymbol{v}))$: returns $\boldsymbol{d}^T\mathbf{Mat}(\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u})$ with $\boldsymbol{d} = \boldsymbol{D}^{(1)}$.

Figure 5.1: Description of the additive scheme.

The correctness is thus straightforward as, noting $(\boldsymbol{u}, \boldsymbol{v}) = \mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}, \boldsymbol{m}, \boldsymbol{r})$, from $\boldsymbol{v} = \boldsymbol{s} \cdot \boldsymbol{u} + \boldsymbol{e} + \hat{\boldsymbol{m}}$ we get that $\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u} = \boldsymbol{e} + \hat{\boldsymbol{m}}$ and therefore

$$\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u} = \sum_{1 \leq i \leq w} f_i \star \boldsymbol{e}_i + \boldsymbol{g}^{(1)} \star \boldsymbol{m}.$$

with $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_w, \boldsymbol{m} \in \mathbb{F}_q^n$. We can thus write,

$$\begin{aligned} \boldsymbol{d}^T\mathbf{Mat}(\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u}) &= \sum_{1 \leq i \leq w} \boldsymbol{d}^T\mathbf{vec}(f_i) \star \boldsymbol{e}_i + \boldsymbol{d}^T\mathbf{vec}(\boldsymbol{g}^{(1)}) \star \boldsymbol{m} \\ &= \boldsymbol{m} \end{aligned}$$

using the fact that, as noted above, $\boldsymbol{d}^T\mathbf{vec}(\boldsymbol{g}^{(1)}) = 1 \in \mathbb{F}_q$ and $\boldsymbol{d}^T\mathbf{vec}(x) = 0$ for any $x \in \boldsymbol{b} \setminus \{\boldsymbol{g}^{(1)}\}$. □

Proposition 5.1.2 (Ciphertext Distribution). *For* $\mathsf{sk} \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{AHE}}()$*, the set of ciphertexts of zero* $\{\mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}, \mathbf{0})\}$ *is a subgroup of* $\mathbb{F}_{q^m}^n \times \mathbb{F}_{q^m}^n$*, and more generally the set of ciphertexts of a message* $\boldsymbol{m} \in \mathbb{F}_q^n$ *are the cosets* $\{\mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}, \mathbf{0})\} + \boldsymbol{g}^{(1)} \star \boldsymbol{m}$*. The output probability distribution is uniform in the associated coset.*

Proof. The set $\{\mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}, \mathbf{0})\}$ is a subgroup as for any $(\boldsymbol{u}, \boldsymbol{s} \cdot \boldsymbol{u} + \boldsymbol{e}), (\boldsymbol{u}', \boldsymbol{s} \cdot \boldsymbol{u}' + \boldsymbol{e}') \in \{\mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}, \mathbf{0})\}$ we have that $\boldsymbol{u} - \boldsymbol{u}' \in \mathbb{F}_{q^m}^n$ and $\boldsymbol{e} - \boldsymbol{e}' \in F^n$ and thus $(\boldsymbol{u} - \boldsymbol{u}', \boldsymbol{s} \cdot (\boldsymbol{u} - \boldsymbol{u}') + (\boldsymbol{e} - \boldsymbol{e}'))$ is in $\{\mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}, \mathbf{0})\}$. Moreover the output distribution of $\mathsf{Encrypt}_{\mathsf{AHE}}$ on this set is uniform as for a given $\boldsymbol{s}$ there is a one to one mapping between the pairs $(\boldsymbol{u}, \boldsymbol{e})$ and $(\boldsymbol{u}, \boldsymbol{s} \cdot \boldsymbol{u} + \boldsymbol{e})$ and the pairs $(\boldsymbol{u}, \boldsymbol{e})$ are chosen uniformly on $\mathbb{F}_{q^m}^n \times F^n$. Proving the rest of the proposition is trivial as the encryption process consists exactly on generating a ciphertext of zero and adding $\boldsymbol{g}^{(1)} \star \boldsymbol{m}$ to it. □

The security of this encryption scheme is reduced to the IRSD problem in Section 5.4; the problem of decrypting ℓ independent ciphertexts is equivalent to solving the syndrome decoding in an $(\ell + 1)$-ideal code. For small values of ℓ, the IRSD is known to be hard and is at the core of the security of other encryption schemes such as ROLLO or RQC.

5.1.2 Additively Homomorphic Algorithms

Figure 5.2 presents the homomorphic algorithms of our additive scheme.

- $\mathsf{Add}_{\mathsf{AHE}}(\mathsf{ct} = (\boldsymbol{u}, \boldsymbol{v}), \mathsf{ct}' = (\boldsymbol{u}', \boldsymbol{v}'))$: returns $(\boldsymbol{u} + \boldsymbol{u}', \boldsymbol{v} + \boldsymbol{v}') \in \mathbb{F}_{q^m}^n \times \mathbb{F}_{q^m}^n$.
- $\mathsf{PtxtMul}_{\mathsf{AHE}}(\boldsymbol{m}' \in \mathbb{F}_q^n, \mathsf{ct} = (\boldsymbol{u}, \boldsymbol{v}))$: returns $(\boldsymbol{m}' \cdot \boldsymbol{u}, \boldsymbol{m}' \cdot \boldsymbol{v}) \in \mathbb{F}_{q^m}^n \times \mathbb{F}_{q^m}^n$.

Figure 5.2: Additively Homomorphic Algorithms.

In the following proposition we consider $\mathbb{F}_q^n$ as the ring $(\mathbb{F}_q^n, +, \cdot)$, $+$ the natural addition in $\mathbb{F}_q^n$ and $\cdot$ the multiplication in $\mathbb{F}_q[X]/\langle Q \rangle$ (see Product of vectors).

Proposition 5.1.3 (Encryption is an $\mathbb{F}_q^n$-module isomorphism)**.** *For any properly generated key* sk*, the function* $f = \mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}, \cdot, (\cdot, \cdot))$ *is an* $\mathbb{F}_q^n$*-module isomorphism between* $(Dom(f), +)$ *and* $(Im(f), +)$ *with* $Dom(f) = \mathbb{F}_q^n \times \mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q)$ *and* $Im(f) \subset \mathbb{F}_{q^m}^n \times \mathbb{F}_{q^m}^n$.

Proof. First note that $(\mathrm{Im}(f), +)$ is an $\mathbb{F}_q^n$-module. It is a subgroup of $(\mathbb{F}_{q^m}^n \times \mathbb{F}_{q^m}^n, +)$ as for any two ciphertexts $\mathsf{ct} = (\boldsymbol{u}, \boldsymbol{s} \cdot \boldsymbol{u} + \boldsymbol{e} + \boldsymbol{g}^{(1)} \star \boldsymbol{m})$, $\mathsf{ct}' = (\boldsymbol{u}', \boldsymbol{s} \cdot \boldsymbol{u}' + \boldsymbol{e}' + \boldsymbol{g}^{(1)} \star \boldsymbol{m}')$, we have $\mathsf{ct} - \mathsf{ct}' = (\boldsymbol{u} - \boldsymbol{u}', \boldsymbol{s} \cdot (\boldsymbol{u} - \boldsymbol{u}') + (\boldsymbol{e} - \boldsymbol{e}') + \boldsymbol{g}^{(1)} \star (\boldsymbol{m} - \boldsymbol{m}')) \in \{\mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}, \boldsymbol{m} - \boldsymbol{m}')\} \subset \mathrm{Im}(f)$. Moreover for any $\boldsymbol{m}' \in \mathbb{F}_q^n$ we have $\boldsymbol{m}' \cdot \mathsf{ct} = (\boldsymbol{m}' \cdot \boldsymbol{u}, \boldsymbol{s} \cdot (\boldsymbol{m}' \cdot \boldsymbol{u}) + \boldsymbol{m}' \cdot \boldsymbol{e} + \boldsymbol{g}^{(1)} \star (\boldsymbol{m}' \cdot \boldsymbol{m})) \in \{\mathsf{Encrypt}(\mathsf{sk}, \boldsymbol{m}' \cdot \boldsymbol{m})\} \subset \mathrm{Im}(f)$ as $\boldsymbol{m} \cdot \boldsymbol{u} \in \mathbb{F}_{q^m}^n$ and $\boldsymbol{m}' \cdot \boldsymbol{e} \in F^n$ (F being an $\mathbb{F}_q$-linear span).

To prove that $(\mathrm{Dom}(f), +)$ with $\mathrm{Dom}(f) = \mathbb{F}_q^n \times \mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q)$ is an $\mathbb{F}_q^n$-module we must define the external multiplication with an element in $\mathbb{F}_q^n$. We define it by multiplying coordinate-wise. The first two coordinates correspond to multiplications in $\mathbb{F}_q[X]/\langle Q \rangle$ and $\mathbb{F}_{q^m}[X]/\langle Q \rangle$. For the last one we consider that the external multiplication is done with each of the w rows of the matrix over $\mathbb{F}_q[X]/\langle Q \rangle$. With this operation it is trivial to verify that we obtain an $\mathbb{F}_q^n$-module.

The identity element of $(\mathbb{F}_q^n \times \mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q), +)$ is $(\boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0})$ and $f(\boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0}) = \mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{0}, (\boldsymbol{0}, \boldsymbol{0})) = (\boldsymbol{0}, \boldsymbol{0})$ the identity element of $\mathrm{Im}(f)$.

For $\boldsymbol{m}' \in \mathbb{F}_q^n$ and $(\boldsymbol{m}, \boldsymbol{r}_1, \boldsymbol{R}_2) \in \mathrm{Dom}(f)$, we have $f(\boldsymbol{m}' \cdot \boldsymbol{m}, \boldsymbol{m}' \cdot \boldsymbol{r}_1, \boldsymbol{m}' \cdot \boldsymbol{R}_2) = (\boldsymbol{m}' \cdot \boldsymbol{r}_1, \boldsymbol{s} \cdot (\boldsymbol{m}' \cdot \boldsymbol{r}_1) + \boldsymbol{f} \cdot \boldsymbol{m}' \cdot \boldsymbol{R}_2 + \boldsymbol{g}^{(1)} \star \boldsymbol{m}' \cdot \boldsymbol{m}) = \boldsymbol{m}' \cdot f(\boldsymbol{m}, \boldsymbol{r}_1, \boldsymbol{R}_2)$ using the commutative and associative properties of the involved polynomial operations, which concludes the proof. □

Corollary 5.1.1 (Homomorphic Addition Distribution)**.** *For any* $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{F}_q^n$*, properly generated key* sk*, and* $\mathsf{ct} \in \{\mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}, \boldsymbol{m})\}$*, let* ct'' *be obtained by*

$$\begin{aligned} \mathsf{ct}' &\xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}, \boldsymbol{m}') \\ \mathsf{ct}'' &= \mathsf{Add}_{\mathsf{AHE}}(\mathsf{ct}, \mathsf{ct}') \end{aligned}$$

and let ct''' *be obtained by*

$$\mathsf{ct}''' \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}, \boldsymbol{m} + \boldsymbol{m}').$$

Then the distributions of ct'' *and* ct''' *are identical.*

Proof. The proof is immediately derived from Proposition 5.1.3 as it proves that $\mathsf{ct}'' = \mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}, \boldsymbol{m} + \boldsymbol{m}', \boldsymbol{r} + \boldsymbol{r}')$ with $\boldsymbol{r}'$ uniformly sampled and $\boldsymbol{r}$ independent from $\boldsymbol{r}'$, and thus $\boldsymbol{r} + \boldsymbol{r}'$ is uniform in $\mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q)$. □

Note that obtaining the same distribution as a fresh ciphertext of the sum is a much stronger property than decrypting to the sum. In practice it implies (among other things) that no information about a computation, besides the result, can leak from the output ciphertext if one of the input ciphertexts was generated with $\mathsf{Encrypt}_{\mathsf{AHE}}$ and unknown to the decrypter (which is definitely not naturally true with lattice-based schemes). It also implies that there is no bound on the amount of ciphertexts that can be added (which again is not naturally true for lattice-based schemes), but we delay the formalization of these properties into an associated corollary to make it more general so that it takes into account arbitrary linear combinations of ciphertexts. We prove thus first that multiplications of ciphertexts by plaintexts also lead to the same distribution as fresh ciphertexts.

Corollary 5.1.2 (Homorphic Plaintext Multiplication Distribution). *For any* $\boldsymbol{m} \in \mathbb{F}_q^n, \boldsymbol{m}' \in \mathbb{F}_q^n\{\mathbf{0}\}$ *and properly generated key* sk*, let* ct' *be obtained by* $\mathsf{ct} \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}, \boldsymbol{m})$ *and* $\mathsf{ct}' = \boldsymbol{m}' \cdot \mathsf{ct}$*. Let* ct'' *be obtained by* $\mathsf{ct}'' \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}, \boldsymbol{m} \cdot \boldsymbol{m}')$*. The distributions of* ct' *and* ct'' *are identical.*

Proof. Again, the proof is immediately derived from 5.1.3 as it proves that $\mathsf{ct}' = \mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}, \boldsymbol{m}' \cdot \boldsymbol{m}, \boldsymbol{m}' \cdot \boldsymbol{r})$ with $\boldsymbol{r}$ uniformly sampled and $\boldsymbol{m}'$ independent from $\boldsymbol{r}$, and thus $\boldsymbol{m}' \cdot \boldsymbol{r}$ is uniform in $\mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q)$ as $\boldsymbol{m}'$ is invertible ($\mathbb{F}_q[X]/\langle Q \rangle$ being a field) and it therefore does not alter the uniform distribution. □

We are now ready to prove the corollary summarizing the results of this section.

Corollary 5.1.3. *Any non-null linear combination, with coefficients in* $\mathbb{F}_q^n$*, of independent ciphertexts follows the same distribution as a fresh encryption of the same linear combination over the associated plaintexts. The resulting ciphertext decrypts correctly.*

Proof. The first result is obtained by removing first the null coefficients (not all are null as it is a non-null linear combination). Then we apply corollary 5.1.2 to each plaintext multiplication and corollary 5.1.1 iteratively. The second result is obtained using the first result and proposition 5.1.1. □

The case in which some ciphertexts are inter-dependent, even maliciously, is more complex and beyond the scope of this chapter. However it is important to note that with the properties described in this section it is quite manageable, unlike for lattice-based homomorphic encryption schemes for which this issue is quickly very complex.

Remark. Functions $\mathsf{Add}_{\mathsf{AHE}}$ and $\mathsf{PtxtMul}_{\mathsf{AHE}}$ corresponding to natural operations, from now on we will in general not call these functions explicitly replacing directly $\mathsf{Add}_{\mathsf{AHE}}(\mathsf{ct}, \mathsf{ct}')$ with operation $\mathsf{ct} + \mathsf{ct}'$ and $\mathsf{PtxtMul}_{\mathsf{AHE}}(\boldsymbol{m}, \mathsf{ct})$ with $\boldsymbol{m} \cdot \mathsf{ct}$ (where $\boldsymbol{m}$ multiplies each of the two coordinates of the vector ct).

5.2 Somewhat Homomorphic Scheme

In this section we extend our additive scheme to a somewhat homomorphic scheme that can perform unlimited additions and one multiplication. The homomorphic multiplication operation transforms a two-component ciphertext into a three-component ciphertext, which can be decrypted with an alternative decryption algorithm.

5.2.1 Fundamental and Additively Homomorphic Algorithms

The fundamental algorithms constituting our scheme SHE, and the additively homomorphic algorithms directly inherited from AHE are depicted in Figure 5.3. The scheme is parametrized by the same variables than the previous scheme, with a different condition on the weight w:

- q, the base field cardinality;
- m, the dimension of the field extension;
- n, the length of the vectors;
- w, the rank weight of the error; it must be that $\frac{w(w+3)}{2} + 1 < m$.

As for AHE we define a randomized algorithm $\mathsf{Encrypt}_{\mathsf{SHE}}(\mathsf{sk}, \boldsymbol{m})$ that samples $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q)$ and returns $\mathsf{Encrypt}_{\mathsf{SHE}}(\mathsf{sk}, \boldsymbol{m}, \boldsymbol{r})$. The SHE scheme does not change the encryption, decryption, addition and plaintext multiplication algorithms, it only gives a stronger constraint on w when defining the parameters and has a more complex key generation algorithm to ensure that noise can be separated from the message space even after a homomorphic multiplication.

- $\mathsf{KeyGen}_{\mathsf{SHE}}()$:
  - samples $g_1 \xleftarrow{\$} \mathbb{F}_{q^m}$, $\boldsymbol{f} = (f_1, \ldots, f_w) \xleftarrow{\$} \mathcal{S}_w^w(\mathbb{F}_{q^m})$
  - defines $F = \mathbf{Supp}(f_1, \ldots, f_w)$ and $\tilde{F} = \mathbf{Supp}(\boldsymbol{f}, g_1 \star \boldsymbol{f}, (f_i f_j)_{1 \leq i,j \leq w})$
  - computes $\tilde{\boldsymbol{f}} = (f_1, \ldots, f_{d_{\tilde{F}}}) \in \mathcal{S}_{d_{\tilde{F}}}^{d_{\tilde{F}}}(\mathbb{F}_{q^m})$ a basis of $\tilde{F}$ with $d_{\tilde{F}} = \dim(\tilde{F})$
  - defines $g_2 = g_1 g_1$
  - checks that $\left\|(f_1, \ldots, f_{d_{\tilde{F}}}, g_1, g_2)\right\|_r = d_{\tilde{F}} + 2$ (if not it restarts)
  - extends this vector into a basis $\boldsymbol{b} = (f_1, \ldots, f_{d_{\tilde{F}}}, g_1, \ldots, g_{m-d_{\tilde{F}}}) \in \mathcal{S}_m^m(\mathbb{F}_{q^m})$
  - defines $\boldsymbol{g} = (g_1, \ldots, g_{m-d_{\tilde{F}}})$
  - computes the matrix $\boldsymbol{B} = \mathbf{Mat}(\boldsymbol{b})$
  - defines $\boldsymbol{D}$ as the last $m - d_{\tilde{F}}$ columns of $(\boldsymbol{B}^{-1})^T$
  - samples $\boldsymbol{s} \xleftarrow{\$} F^n$ with $F = \mathbf{Supp}(\boldsymbol{f})$
  - returns $\mathsf{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s})$.
- $\mathsf{Encrypt}_{\mathsf{SHE}} = \mathsf{Encrypt}_{\mathsf{AHE}}$.
- $\mathsf{Decrypt}_{\mathsf{SHE}} = \mathsf{Decrypt}_{\mathsf{AHE}}$.
- $\mathsf{Add}_{\mathsf{SHE}} = \mathsf{Add}_{\mathsf{AHE}}$.
- $\mathsf{PtxtMul}_{\mathsf{SHE}} = \mathsf{PtxtMul}_{\mathsf{AHE}}$.

Figure 5.3: Description of the fully homomorphic scheme.

Proposition 5.2.1 (Extension of AHE properties to SHE). *Propositions 5.1.2, 5.1.1, and 5.1.3 and Corollaries 5.1.1, 5.1.2 and 5.1.3 remain true when replacing* AHE *with* SHE.

Proof. The associated proofs only use properties of $\boldsymbol{D}$ and the definitions of $\mathsf{Encrypt}_{\mathsf{AHE}}$, $\mathsf{Decrypt}_{\mathsf{AHE}}$, $\mathsf{Add}_{\mathsf{AHE}}$ and $\mathsf{PtxtMul}_{\mathsf{AHE}}$. It can be easily checked that the used properties of $\boldsymbol{D}$ are maintained and that the definitions of $\mathsf{Encrypt}_{\mathsf{AHE}}$, $\mathsf{Decrypt}_{\mathsf{AHE}}$, $\mathsf{Add}_{\mathsf{AHE}}$ and $\mathsf{PtxtMul}_{\mathsf{AHE}}$ are unchanged. □

5.2.2 Multiplicative Homomorphic Algorithms

- $\mathsf{Mul}(\mathsf{ct} = (\boldsymbol{u}, \boldsymbol{v}), \mathsf{ct}' = (\boldsymbol{u}', \boldsymbol{v}'))$: returns $(\boldsymbol{v} \cdot \boldsymbol{v}', -(\boldsymbol{u} \cdot \boldsymbol{v}' + \boldsymbol{u}' \cdot \boldsymbol{v}), \boldsymbol{u} \cdot \boldsymbol{u}') \in \mathbb{F}_{q^m}^n \times \mathbb{F}_{q^m}^n \times \mathbb{F}_{q^m}^n$
- $\mathsf{DecryptMul}(\mathsf{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s}), (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}))$: computes $\boldsymbol{tmp} = \boldsymbol{a} + \boldsymbol{s} \cdot \boldsymbol{b} + \boldsymbol{s} \cdot \boldsymbol{s} \cdot \boldsymbol{c}$, and returns $\boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{tmp})$ with $\boldsymbol{d} = \boldsymbol{D}^{(2)}$.

Figure 5.4: Multiplicative Homomorphic Algorithms.

Proposition 5.2.2 (Homomorphic Multiplication Decryption Correctness). *For any $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{F}_q^n$, and properly generated key* sk, *let* $\mathsf{ct} \in \{\mathsf{Encrypt}(\mathsf{sk}, \boldsymbol{m})\}$ *and* $\mathsf{ct}' \in \{\mathsf{Encrypt}(\mathsf{sk}, \boldsymbol{m}')\}$. *We have* $\mathsf{DecryptMul}(\mathsf{sk}, \mathsf{Mul}(\mathsf{ct}, \mathsf{ct}')) = \boldsymbol{m} \cdot \boldsymbol{m}'$.

Proof. Let's note $\mathsf{ct} = (\boldsymbol{u}, \boldsymbol{v}), \mathsf{ct}' = (\boldsymbol{u}', \boldsymbol{v}')$ and $\mathsf{Mul}(\mathsf{ct}, \mathsf{ct}') = (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$. We then have

$$\begin{aligned}
\boldsymbol{tmp} &= \boldsymbol{a} + \boldsymbol{s} \cdot \boldsymbol{b} + \boldsymbol{s} \cdot \boldsymbol{s} \cdot \boldsymbol{c} \\
&= \boldsymbol{v} \cdot \boldsymbol{v}' - \boldsymbol{s} \cdot (\boldsymbol{u} \cdot \boldsymbol{v}' + \boldsymbol{u}' \cdot \boldsymbol{v}) + \boldsymbol{s} \cdot \boldsymbol{s} \cdot \boldsymbol{u} \cdot \boldsymbol{u}' \\
&= (\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u}) \cdot (\boldsymbol{v}' - \boldsymbol{s} \cdot \boldsymbol{u}') \\
&= (\hat{\boldsymbol{m}} + \boldsymbol{e}) \cdot (\hat{\boldsymbol{m}}' + \boldsymbol{e}') \\
&= \hat{\boldsymbol{m}} \cdot \hat{\boldsymbol{m}}' + \hat{\boldsymbol{m}}' \cdot \boldsymbol{e} + \hat{\boldsymbol{m}} \cdot \boldsymbol{e}' + \boldsymbol{e} \cdot \boldsymbol{e}'
\end{aligned}$$

with $\hat{\boldsymbol{m}} \cdot \hat{\boldsymbol{m}}' = \boldsymbol{g}^{(1)}\boldsymbol{g}^{(1)} \star \boldsymbol{m} \cdot \boldsymbol{m}'$ and $\hat{\boldsymbol{m}}' \cdot \boldsymbol{e} + \hat{\boldsymbol{m}} \cdot \boldsymbol{e}' + \boldsymbol{e} \cdot \boldsymbol{e}' \in \tilde{F}^n$. We thus can write

$$\boldsymbol{tmp} = \boldsymbol{g}^{(1)}\boldsymbol{g}^{(1)} \star \boldsymbol{m} \cdot \boldsymbol{m}' + \sum_{1 \leq i \leq d_{\tilde{F}}} f_i \star \boldsymbol{e}_i \qquad \text{with } \boldsymbol{e}_i \in \mathbb{F}_q^n.$$

As $\boldsymbol{g}^{(1)}\boldsymbol{g}^{(1)} = \boldsymbol{g}^{(2)}$ and $\boldsymbol{d} = \boldsymbol{D}^{(2)}$, $\boldsymbol{d}^T \mathbf{vec}(\boldsymbol{g}^{(2)}) = 1$ and $\boldsymbol{d}^T \mathbf{vec}(f_i) = 0$ for $1 \leq i \leq d_{\tilde{F}}$, we thus have $\boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{tmp}) = \boldsymbol{m} \cdot \boldsymbol{m}'$. □

It is possible to define an encryption function that directly creates ciphertexts of the form $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ (with noise drawn from $\tilde{F}$) and show that the result of the multiplication of two fresh ciphertexts of the form $(\boldsymbol{u}, \boldsymbol{v})$ follows the same distribution as a fresh three-coordinate ciphertext associated with the product of plaintexts. It is also possible to show that non-null linear combinations of these three-coordinate ciphertexts have the same distributions as fresh three-coordinate ciphertexts. We do not delve into these proofs as in practice three-coordinate ciphertexts will be transformed back to two coordinate ciphertexts. However if for some reason (e.g. computing a degree two polynomial with a simple scheme) one wants to handle three-coordinate ciphertexts it is important to understand that the nice distributional properties of two-coordinate ciphertexts are maintained.

The Somewhat Homomorphic Encryption scheme described above can be adapted so as to perform the evaluation of a arbitrary polynomial of degree d. However this has two implications that we state informally. First, the ciphertext is expanded to $d + 1$ coordinates. Second, multiplications are expanding the noise space, meaning that the parameters must satisfy $w^d = \mathcal{O}(m)$. These two conditions require the choice of large and impractical parameters for high values of d.

5.3 (Insecure) Bootstrapping and Fully Homomorphic Encryption

In this section we present bootstrapping algorithms that homomorphically applies either $\mathsf{Decrypt}_{\mathsf{SHE}}$ or $\mathsf{DecryptMul}_{\mathsf{SHE}}$ on a ciphertext. Our bootstrapping algorithm has no multiplicative depth so it produces a two-component $(\boldsymbol{u}, \boldsymbol{v})$ fresh ciphertext with a new key. The simplicity of our construction gives a glimpse of a practical Fully Homomorphic Encryption scheme that would allow to compute arbitrary circuits.

However, this first bootstrapping construction is unsecure as the number of bootstrapping keys ($2m$ for the case of a homomorphic evaluation of $\mathsf{Decrypt}_{\mathsf{SHE}}$) is higher than the upper bound on the number of independent ciphertexts allowed ($2w$, cf. Section 5.4). Therefore, at the moment, it cannot be used for a secure FHE.

We first present in Figure 5.5 a homomorphic decryption algorithm that works on two-component $(\boldsymbol{u}, \boldsymbol{v})$ ciphertexts, then present in Figure 5.6 a bootstrapping relinearization algorithm that is a homomorphic decryption on three-component $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ ciphertexts.

5.3.1 Homomorphic Decryption Algorithms

In this section we explicitly note $(\gamma_1, \ldots, \gamma_m)$ the public basis in which an element of $\mathbb{F}_q^m$ represents an element of $\mathbb{F}_{q^m}$.

- $\mathsf{GenKeySwitch}_{\mathsf{SHE}}(\mathsf{sk}_1 = (\boldsymbol{f}_1, \boldsymbol{g}_1, \boldsymbol{D}_1, \boldsymbol{s}_1))$:
  - generates a new basis $\boldsymbol{b}_2$ of $\mathbb{F}_{q^m}$, and $\mathsf{sk}_2 = (\boldsymbol{f}_2, \boldsymbol{g}_2, \boldsymbol{D}_2, \boldsymbol{s}_2)$ as in $\mathsf{KeyGen}_{\mathsf{SHE}}$
  - for $1 \leq i \leq m$ defines $\boldsymbol{s}_{1,i} = \boldsymbol{d}^T \mathbf{Mat}(\gamma_i \star \boldsymbol{s}_1)$ with $\boldsymbol{d} = \boldsymbol{D}_1^{(1)}$
  - defines $\mathsf{ksk} = (\mathsf{ksk}_1, \ldots, \mathsf{ksk}_m)$, with $\mathsf{ksk_i} \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{SHE}}(\mathsf{sk}_2, \boldsymbol{s}_{1,i})$
  - for $1 \leq i \leq m$ defines $\boldsymbol{p}_i = \boldsymbol{d}^{(i)} \star (1, 0, \ldots, 0) \in \mathbb{F}_{q^m}^n$
  - defines $\mathsf{projk} = (\mathsf{projk}_1, \ldots, \mathsf{projk}_m)$ with $\mathsf{projk}_i \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{SHE}}(\mathsf{sk}_2, \boldsymbol{p}_i)$
  - returns $(\mathsf{sk}_2, \mathsf{ksk}, \mathsf{projk})$
- $\mathsf{HomDecrypt}_{\mathsf{SHE}}(\mathsf{ksk}, \mathsf{projk}, \mathsf{ct} = (\boldsymbol{u}, \boldsymbol{v}))$:
  - defines $\boldsymbol{u}_i \in \mathbb{F}_q^n$ such that $\boldsymbol{u} = \sum_{1 \leq i \leq m} \gamma_i \star \boldsymbol{u}_i$
  - defines $\boldsymbol{v}_i \in \mathbb{F}_q^n$ such that $\boldsymbol{v} = \sum_{1 \leq i \leq m} \gamma_i \star \boldsymbol{v}_i$
  - computes $\mathsf{ct}_1 = \sum_{1 \leq i \leq m} \boldsymbol{v}_i \cdot \mathsf{projk}_i$
  - computes $\mathsf{ct}_2 = \sum_{1 \leq i \leq m} \boldsymbol{u}_i \cdot \mathsf{ksk}_i$
  - returns $\mathsf{ct}_1 - \mathsf{ct}_2$

Figure 5.5: Homomorphic Decryption Algorithms.

Proposition 5.3.1 (Homomorphic Decryption Distribution). *For any properly generated key* sk_1, *any* $\mathsf{ct} \in (\mathbb{F}_{q^m}^n)^* \times (\mathbb{F}_{q^m}^n)^*$ *such that* $\mathsf{Decrypt}_{\mathsf{SHE}}(\mathsf{sk}_1, \mathsf{ct}) = \boldsymbol{m} \in \mathbb{F}_q^n$, *and* $(\mathsf{sk}_2, \mathsf{ksk}, \mathsf{projk}) \xleftarrow{\$} \mathsf{GenKeySwitch}_{\mathsf{SHE}}(\mathsf{sk}_1)$, *let* ct' *be obtained by* $\mathsf{ct}' = \mathsf{HomDecrypt}_{\mathsf{SHE}}(\mathsf{ksk}, \mathsf{projk}, \mathsf{ct})$ *and let* ct'' *be obtained by* $\mathsf{ct}'' \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{SHE}}(\mathsf{sk}_2, \boldsymbol{m})$. *The distributions of* ct' *and* ct'' *are identical.*

Proof. As ct is non-null, $\mathsf{ct}_1 - \mathsf{ct}_2$ is a non-null linear combination of the ciphertexts projk_i and ksk_i, that have been generated independently. Thus, using Corollary 5.1.3 and Proposition 5.2.1, $\mathsf{ct}_1 - \mathsf{ct}_2$ follows the same distribution as $\mathsf{ct}''' \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{SHE}}(\mathsf{sk}_2, \sum_{1 \leq i \leq m} \boldsymbol{v}_i \cdot \boldsymbol{p}_i - \sum_{1 \leq i \leq m} \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,i})$.

We thus only need to prove that $\sum_{1 \leq i \leq m} \boldsymbol{v}_i \cdot \boldsymbol{p}_i - \sum_{1 \leq i \leq m} \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,i} = \boldsymbol{m}$. We have

$$\begin{aligned}
\sum_{1 \leq i \leq m} \boldsymbol{v}_i \cdot \boldsymbol{p}_i - \sum_{1 \leq i \leq m} \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,i} &= \sum_{1 \leq i \leq m} \boldsymbol{v}_i \cdot \boldsymbol{d}^{(i)} \star (1, 0, \ldots, 0) - \sum_{1 \leq i \leq m} \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,i} \\
&= \sum_{1 \leq i \leq m} \boldsymbol{d}^{(i)} \star \boldsymbol{v}_i - \sum_{1 \leq i \leq m} \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,i} \\
&= \boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{v}) - \sum_{1 \leq i \leq m} \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,i} \\
&= \boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{v}) - \sum_{1 \leq i \leq m} \boldsymbol{u}_i \cdot \boldsymbol{d}^T \mathbf{Mat}(\gamma_i \star \boldsymbol{s}_1)
\end{aligned}$$

noting that $\boldsymbol{d}^T \mathbf{Mat}(\gamma_i \star \boldsymbol{s}_1) = \sum_j \boldsymbol{d}^{(j)} \boldsymbol{\ell}_j$ with $\boldsymbol{\ell}_j$ the rows of $\mathbf{Mat}(\gamma_i \star \boldsymbol{s}_1)$ we can use the distributivity of the polynomial multiplication over the addition to get $\boldsymbol{u}_i \cdot \boldsymbol{d}^T \mathbf{Mat}(\gamma_i \star \boldsymbol{s}_1) = \boldsymbol{d}^T \mathbf{Mat}(\gamma_i \star \boldsymbol{u}_i \cdot \boldsymbol{s}_1)$. We thus obtain

$$\begin{aligned}
\sum_{1 \leq i \leq m} \boldsymbol{v}_i \cdot \boldsymbol{p}_i - \sum_{1 \leq i \leq m} \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,i} &= \boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{v}) - \boldsymbol{d}^T \mathbf{Mat}\Big(\sum_{1 \leq i \leq m} \gamma_i \star \boldsymbol{u}_i \cdot \boldsymbol{s}_1\Big) \\
&= \boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{v} - \boldsymbol{u} \cdot \boldsymbol{s}_1) \\
&= \boldsymbol{m}
\end{aligned}$$

which concludes the proof. □

It is very important to note that Proposition 5.3.1 ensures that the output of $\mathsf{HomDecrypt}_{\mathsf{SHE}}$ is a well formed and well distributed ciphertext even if ct is not, as long as it decrypts to $\boldsymbol{m}$. There are many implications to Proposition 5.3.1. Among them we can highlight that it opens the path to fast and simple algorithms for: bootstrapping relinearization, and trans-encryption. It also allows to rerandomize a ciphertext (assuming that the key switching key is well-formed).

5.3.2 Bootstrapping relinearization

- $\mathsf{GenRelinKey}_{\mathsf{SHE}}(\mathsf{sk}_1 = (\boldsymbol{f}_1, \boldsymbol{g}_1, \boldsymbol{D}_1, \boldsymbol{s}_1, P))$:
  - generates a new basis $\boldsymbol{b}_2$ of $\mathbb{F}_{q^m}$, and $\mathsf{sk}_2 = (\boldsymbol{f}_2, \boldsymbol{g}_2, \boldsymbol{D}_2, \boldsymbol{s}_2, P)$ as in $\mathsf{KeyGen}_{\mathsf{SHE}}$
  - for $1 \leq i \leq m$ defines $\boldsymbol{s}_{1,i} = \boldsymbol{d}^T \mathbf{Mat}(\gamma_i \star \boldsymbol{s}_1)$ with $\boldsymbol{d} = \boldsymbol{D}_1^{(2)}$
  - for $1 \leq i \leq m$ defines $\boldsymbol{s}_{1sq,i} = \boldsymbol{d}^T \mathbf{Mat}(\gamma_i \star \boldsymbol{s}_1 \cdot \boldsymbol{s}_1)$
  - defines $\mathsf{ksk} = (\mathsf{ksk}_1, \ldots, \mathsf{ksk}_m)$, with $\mathsf{ksk_i} \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{SHE}}(\mathsf{sk}_2, \boldsymbol{s}_{1,i})$
  - defines $\mathsf{ksksq} = (\mathsf{ksksq}_1, \ldots, \mathsf{ksksq}_m)$, with $\mathsf{ksksq_i} \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{SHE}}(\mathsf{sk}_2, \boldsymbol{s}_{1sq,i})$
  - for $1 \leq i \leq m$ defines $\boldsymbol{p}_i = \boldsymbol{d}^{(i)} \star (1, 0, \ldots, 0) \in \mathbb{F}_{q^m}^n$
  - defines $\mathsf{projk} = (\mathsf{projk}_1, \ldots, \mathsf{projk}_m)$ with $\mathsf{projk}_i \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{SHE}}(\mathsf{sk}_2, \boldsymbol{p}_i)$
  - returns $(\mathsf{sk}_2, \mathsf{ksk}, \mathsf{ksksq}, \mathsf{projk})$
- $\mathsf{Relinearize}_{\mathsf{SHE}}(\mathsf{ksk}, \mathsf{ksksq}, \mathsf{projk}, (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \in (\mathbb{F}_{q^m}^n)^* \times (\mathbb{F}_{q^m}^n)^* \times (\mathbb{F}_{q^m}^n)^*)$:
  - defines $\boldsymbol{a}_i \in \mathbb{F}_q^n$ such that $\boldsymbol{a} = \sum_{1 \leq i \leq m} \gamma_i \star \boldsymbol{a}_i$
  - defines $\boldsymbol{b}_i \in \mathbb{F}_q^n$ such that $\boldsymbol{b} = \sum_{1 \leq i \leq m} \gamma_i \star \boldsymbol{b}_i$
  - defines $\boldsymbol{c}_i \in \mathbb{F}_q^n$ such that $\boldsymbol{c} = \sum_{1 \leq i \leq m} \gamma_i \star \boldsymbol{c}_i$
  - computes $\mathsf{ct}_a = \sum_{1 \leq i \leq m} \boldsymbol{a}_i \cdot \mathsf{projk}_i$
  - computes $\mathsf{ct}_b = \sum_{1 \leq i \leq m} \boldsymbol{b}_i \cdot \mathsf{ksk}_i$
  - computes $\mathsf{ct}_c = \sum_{1 \leq i \leq m} \boldsymbol{c}_i \cdot \mathsf{ksksq}_i$
  - returns $\mathsf{ct}_a + \mathsf{ct}_b + \mathsf{ct}_c$

Figure 5.6: Relinearization Algorithms.

Proposition 5.3.2 (Relinearization Distribution)**.** *For any properly generated key* sk_1*, non-null* $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \in \mathbb{F}_{q^m}^n \times \mathbb{F}_{q^m}^n \times \mathbb{F}_{q^m}^n$ *such that* $\mathsf{DecryptMul}_{\mathsf{SHE}}(\mathsf{sk}_1, (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})) = \boldsymbol{m} \in \mathbb{F}_q^n$*, and* $(\mathsf{sk}_2, \mathsf{ksk}, \mathsf{ksksq}, \mathsf{projk}) \xleftarrow{\$} \mathsf{GenRelinKey}_{\mathsf{SHE}}(\mathsf{sk}_1)$*, let* ct' *be obtained by* $\mathsf{ct}' = \mathsf{Relinearize}_{\mathsf{SHE}}(\mathsf{ksk}, \mathsf{ksksq}, \mathsf{projk}, (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}))$ *and let* ct'' *be obtained by* $\mathsf{ct}'' \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{SHE}}(\mathsf{sk}_2, \boldsymbol{m})$*. The distributions of* ct' *and* ct''' *are identical.*

Proof. The proof is very similar to the one of $\mathsf{HomDecrypt}_{\mathsf{SHE}}$. We use the fact that $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ is not null and the ciphertexts generated in $\mathsf{GenRelinKey}_{\mathsf{SHE}}$ are independently generated to show that the result is uniformly distributed among the encryptions of a given plaintext. We then show that this plaintext is $\boldsymbol{m}$ by evaluating the correctness.

For the correctness we show that ct_a is in $\mathsf{Encrypt}_{\mathsf{SHE}}(\mathsf{sk}_2, \boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{a}))$, then that ct_b is in $\mathsf{Encrypt}_{\mathsf{SHE}}(\mathsf{sk}_2, \boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{s}_1 \cdot \boldsymbol{b}))$, and finally that ct_c is in $\mathsf{Encrypt}_{\mathsf{SHE}}(\mathsf{sk}_2, \boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{s}_1 \cdot \boldsymbol{s}_1 \cdot \boldsymbol{c}))$. As a consequence we obtain that ct is in $\mathsf{Encrypt}_{\mathsf{SHE}}(\mathsf{sk}_2, \mathsf{DecryptMul}_{\mathsf{SHE}}(\mathsf{sk}_1, (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}))$ and thus in $\mathsf{Encrypt}_{\mathsf{SHE}}(\mathsf{sk}_2, \boldsymbol{m})$. □

Note that the relinearization bootstraps the ciphertext. Unlike in previous FHE schemes, the resulting ciphertext is fresh. It has, exactly the same amount of noise as the fresh ciphertexts used

for the relinearization. Note that these are direct results of Corollary 5.1.3 (doing linear combinations of ciphertexts with coefficients in $\mathbb{F}_q$ does not change the distribution). This seems to be a very powerful property.

Another very interesting property that makes these algorithms possible is the structure of the coefficients of the polynomials forming the ciphertexts. As these coefficients have structure it is much easier to project and reconstruct inside a ciphertext than it would be with bits inside an integer (as in lattice cryptosystems). This structure can also lead to many interesting applications such as directly encoding structured plaintexts (e.g. AES states if we take $\mathbb{F}_q = \mathbb{F}_{2^{8\times 4\times 4}}$).

5.4 Problem: Limitation on the number of independent ciphertexts

Even though it is beautiful, this FHE is not secure because, as we will demonstrate in this section, the number of ciphertexts that allows an attacker to retrieve the secret key in polynomial time ($2w$) is lower than the number of ciphertexts of the bootstrapping material ($2m$).

Definition 5.4.1 (Rank-SHE Ciphertext Learning Problem (RCL))**.** *Let* $\mathsf{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{SHE}}()$*. Let* $\mathcal{O}$ *be an oracle which samples randomly independent encryptions of* $\mathbf{0}$ *under secret key* sk*. The problem* $RCL_{n,\ell,w}$ *is to recover* $F = \mathbf{Supp}(\boldsymbol{f})$ *given* ℓ *accesses to the oracle.*

We will prove below the following result which upper bounds to $2w$ the number of independent ciphertexts that can be crafted with the same key. For the sake of the security reduction, we will now assume that w is below the rank Gilbert-Varshamov bound for parameters $(q, \ell n, n, m)$ for every $\ell > 1$, in order to guarantee the unicity of a solution to the considered problems.

Proposition 5.4.1. *$RCL_{n,\ell,w}$ can be solved in polynomial time when $\ell \geq 2w$.*

The proof requires the following lemma that connects RCL and IRSD.

Lemma 5.4.1. *$IRSD_{n,\ell+1,w}$ is polynomially equivalent to $RCL_{n,\ell,w}$.*

Proof. Suppose we have a solver for $\mathrm{RCL}_{n,\ell,w}$. Let $(\boldsymbol{H}, \boldsymbol{y} = \boldsymbol{H}\boldsymbol{e}^\top) \in \mathbb{F}_{q^m}^{\ell n\times(\ell+1)n} \times \mathbb{F}_{q^m}^{n\ell}$ be an instance of $\mathrm{IRSD}_{n,\ell+1,w}$.

By applying Gaussian elimination on the ideal blocks of $\boldsymbol{H}$, we can reduce $\boldsymbol{H}$ to its systematic form. Namely there exists an invertible matrix $\boldsymbol{M} \in \mathcal{M}_{\ell n}(\mathbb{F}_{q^m})$ such that:

$$\boldsymbol{H} = \boldsymbol{M} \begin{pmatrix} \boldsymbol{I}_n & & & & \mathcal{IM}_Q(\boldsymbol{u}_1) \\ & \boldsymbol{I}_n & & & \mathcal{IM}_Q(\boldsymbol{u}_2) \\ & & \ddots & & \vdots \\ & & & \boldsymbol{I}_n & \mathcal{IM}_Q(\boldsymbol{u}_{\ell-1}) \\ & & & \boldsymbol{I}_n & \mathcal{IM}_Q(\boldsymbol{u}_\ell) \end{pmatrix} \tag{5.1}$$

By rewriting $\boldsymbol{M}^{-1}\boldsymbol{y}$ as $(\boldsymbol{v}_1, \cdots, \boldsymbol{v}_\ell)$ where the $\boldsymbol{v}_i$ are in $\mathbb{F}_{q^m}^n$, and $\boldsymbol{e}$ as $(\boldsymbol{e}_1, \cdots, \boldsymbol{e}_\ell, \boldsymbol{s})$ we get the following equalities :

$$\begin{aligned} \boldsymbol{v}_1 &= \boldsymbol{u}_1 \cdot \boldsymbol{s} + \boldsymbol{e}_1 \\ &\vdots \\ \boldsymbol{v}_\ell &= \boldsymbol{u}_\ell \cdot \boldsymbol{s} + \boldsymbol{e}_\ell \end{aligned}$$

Therefore, as the distributions of $\boldsymbol{e}$ in IRSD on one hand, and $(\boldsymbol{e}_1, \cdots, \boldsymbol{e}_\ell, \boldsymbol{s})$ in RCL on the other hand, are the same, $(\boldsymbol{u}_i, \boldsymbol{v}_i)_{i\leq\ell}$ is a (random) instance of $\mathrm{RCL}_{n,\ell,w}$ (for a secret key $\mathsf{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s})$ such that $\mathbf{Supp}(\boldsymbol{f}) = \mathbf{Supp}(\boldsymbol{e})$) so a solver can recover the support of $\boldsymbol{e}$. It is only a matter of linear algebra to compute the exact entries of $\boldsymbol{e}$ and thus to solve the instance of $\mathrm{IRSD}_{n,\ell+1,w}$.

Conversely, suppose we have a solver for $\mathrm{IRSD}_{n,\ell+1,w}$. Let $\mathsf{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s})$ and let $(\boldsymbol{u}_i, \boldsymbol{v}_i = \boldsymbol{u}_i \cdot \boldsymbol{s} + \boldsymbol{e}_i)_{i \leq \ell}$ be an instance of $\mathrm{RCL}_{n,\ell,w}$. We then draw a random invertible matrix $\boldsymbol{M}$ and apply the same transformation as before: define a matrix $\boldsymbol{H}$ as in Equation (5.1) and $\boldsymbol{y} = (\boldsymbol{M}\boldsymbol{v}_1, \ldots, \boldsymbol{M}\boldsymbol{v}_\ell)$. We obtain a random instance $(\boldsymbol{H}, \boldsymbol{y})$ of $\mathrm{IRSD}_{n,\ell+1,w}$,we can use our solver, and thanks to the unicity of a solution, the support of the recovered error will precisely be the secret space F.

All transformations in this proof are obviously polynomial so we get that $\mathrm{IRSD}_{n,\ell+1,w}$ and $\mathrm{RCL}_{n,\ell,w}$ are polynomially equivalent. □

Proof of Proposition 5.4.1. We use the linearization attack against RSD presented in [GRS16, Proposition 2] that is effective against small rate codes. This attacks solves the decoding problem with weight w in an $[n,k]_{q^m}$ code under the following condition:

$$n \geq (k+1)(w+1) - 1.$$

As seen in the above lemma, the ideal code constructed from the $\mathrm{RCL}_{n,\ell,w}$ instance is an $[n(\ell+1), n]_{q^m}$ code. In that setting, the linearization attack works when

$$n(\ell+1) \geq (n+1)(w+1) - 1,$$

i.e.

$$\ell + 1 \geq w + \frac{w}{n}.$$

Because $n \geq 1$, this clearly shows that $\ell \geq 2w$ is a sufficient condition to break RCL in polynomial time. □

Remark. The RCL problem is defined for encryptions of $\mathbf{0}$, so it is not obvious whether the polynomial attack would work for encryptions of random messages $\boldsymbol{m}$. However, we find this attack sufficiently dangerous to claim that $2w$ is the maximal number of ciphertexts that can be safely published, even for non-zero messages.

5.5 Reducing the number of bootstrapping ciphertexts

In order to reduce the number of ciphertexts, we pack the plaintext into several components which are linked publicly so that the server can select which component of the packing they want. It allows to reduce the number of bootstrapping ciphertexts from $2m$ to $2(w+1)$, which is a major improvement but is unfortunately still unsecure.

5.5.1 Packing plaintexts

In order to simplify and since our bootstrapping procedure does not need any multiplication, we only present packing for the additive homomorphic scheme. It could be easily extended to the somewhat homomorphic scheme.

In this section we present a variant of AHE which packs as many plaintexts in $\mathbb{F}_q^n$ as possible in a single ciphertext. We call this variant PAHE (for Packed Additively Homomorphic Encryption).

The fundamental algorithms constituting our scheme PAHE, and the additively homomorphic algorithms directly inherited from AHE are depicted in Figure 5.7. The scheme has now a public key pk that consists of a field element $\rho \in \mathbb{F}_{q^m}$ that will be used for manipulating packed ciphertexts.

The scheme is parametrized by an additional parameter t that accounts for the size of the packing:

- q, the base field cardinality;
- m, the dimension of the field extension;
- n, the length of the vectors;

- w, the rank weight of the error;
- t, the maximal number of plaintexts that can be packed in a single ciphertext; it must be that $t(w+1) \leq m$ and that m is divisible by t.

Actually, in the following we will consider that the equality is met for the above condition, i.e. $t(w+1) = m$, because it is the optimal setup to have the least number of bootstrapping ciphertexts.

- $\mathsf{KeyGen}_{\mathsf{PAHE}}()$:
  - samples $\rho \xleftarrow{\$} \mathbb{F}_{q^m}$ such that $\rho^t = 1$.
  - samples $g_0 \xleftarrow{\$} \mathbb{F}_{q^m}$, $\boldsymbol{f} = (f_1, \ldots, f_w) \xleftarrow{\$} \mathcal{S}_w^w(\mathbb{F}_{q^m})$
  - defines $\tilde{F} = Vect_{\mathbb{F}_q}((\rho^j f_i)_{1 \leq i \leq w, 0 \leq j \leq t-1})$
  - computes $\tilde{\boldsymbol{f}} = (f_1, \ldots, f_{d_{\tilde{F}}}) \in \mathcal{S}_{d_{\tilde{F}}}^{d_{\tilde{F}}}(\mathbb{F}_{q^m})$ a basis of $\tilde{F}$ with $d_{\tilde{F}} = \dim(\tilde{F})$
  - defines $g_i = \rho^{-i} g_0$
  - checks that $\left\|(f_1, \ldots, f_{d_{\tilde{F}}}, g_0, ..., g_{t-1})\right\|_r = d_{\tilde{F}} + t$ (if not it restarts)
  - extends this vector into a basis $\boldsymbol{b} = (f_1, \ldots, f_{d_{\tilde{F}}}, g_0, \ldots, g_{m-d_{\tilde{F}}-1}) \in \mathcal{S}_m^m(\mathbb{F}_{q^m})$
  - defines $\boldsymbol{g} = (g_0, \ldots, g_{m-d_{\tilde{F}}-1})$
  - computes the matrix $\boldsymbol{B} = \mathbf{Mat}(\boldsymbol{b})$ and its transposed inverse $(\boldsymbol{B}^{-1})^T$
  - defines $\boldsymbol{D}$ as the last $m - d_{\tilde{F}}$ columns of $(\boldsymbol{B}^{-1})^T$
  - samples $\boldsymbol{s} \xleftarrow{\$} F^n$ with $F = \mathbf{Supp}(\boldsymbol{f})$
  - returns $(\mathsf{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s}), \mathsf{pk} = \rho)$.
- $\mathsf{Encrypt}_{\mathsf{PAHE}}(\mathsf{sk}, (\boldsymbol{m}_0, \ldots, \boldsymbol{m}_{t-1}), \boldsymbol{r} \in \mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q))$ with $\boldsymbol{m}_i \in \mathbb{F}_q^n$): notes $\boldsymbol{r} = (\boldsymbol{r}_1, \boldsymbol{R}_2)$, defines $\boldsymbol{u} = \boldsymbol{r}_1$ and $\boldsymbol{e} = \boldsymbol{f}\boldsymbol{R}_2$ and sets $\boldsymbol{v} = \boldsymbol{s} \cdot \boldsymbol{u} + \boldsymbol{e} + \hat{\boldsymbol{m}}$ with $\hat{\boldsymbol{m}} = \sum_{0 \leq i < t} g_i \star \boldsymbol{m}_i \in \mathbb{F}_{q^m}^n$. Returns $\mathsf{ct} = (\boldsymbol{u}, \boldsymbol{v})$.
- $\mathsf{Decrypt}_{\mathsf{PAHE}}(\mathsf{sk}, \mathsf{ct} = (\boldsymbol{u}, \boldsymbol{v}))$: returns $(\boldsymbol{m}_0, \ldots, \boldsymbol{m}_{t-1})$ with $\boldsymbol{m}_i = (\boldsymbol{D}^{(i+1)})^T \mathbf{Mat}(\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u})$.
- $\mathsf{Add}_{\mathsf{PAHE}} = \mathsf{Add}_{\mathsf{AHE}}$.
- $\mathsf{PtxtMul}_{\mathsf{PAHE}} = \mathsf{PtxtMul}_{\mathsf{AHE}}$.

Figure 5.7: Description of the packed additive homomorphic scheme.

Proposition 5.5.1 (Extension of AHE properties to PAHE). *Proposition 5.1.1, and Corollaries 5.1.1 to 5.1.3 remain true when replacing* AHE *with* PAHE. *Proposition 5.1.2 remains true except for the definition of the cosets which are now* $\{\mathsf{Encrypt}_{\mathsf{PAHE}}(\mathsf{sk}, \mathbf{0})\} + \sum_{0 \leq i < t} g_i \star \boldsymbol{m}_i$.

Proof. For Proposition 5.1.1 we can follow the same proof noting that

$$\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u} = \sum_{1 \leq i \leq w} f_i \star \boldsymbol{e}_i + \sum_{0 \leq i < t} g_i \star \boldsymbol{m}_i.$$

We can thus write,

$$\begin{aligned}(\boldsymbol{D}^{(j+1)})^T \mathbf{Mat}(\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u}) &= \sum_{1 \leq i \leq w} (\boldsymbol{D}^{(j+1)})^T \mathbf{vec}(f_i) \star \boldsymbol{e}_i + \sum_{0 \leq i < t} (\boldsymbol{D}^{(j+1)})^T \mathbf{vec}(g_i) \star \boldsymbol{m}_i \\ &= \boldsymbol{m}_j\end{aligned}$$

using the same arguments as in the proof of Proposition 5.1.1.

The proofs of Proposition 5.1.2 and Corollaries 5.1.1 to 5.1.3 do not depend on the specific encoding we have on PAHE but only on how the noise vectors are chosen and used. As this is unchanged from AHE, the proofs are immediately valid for PAHE. □

5.5.2 Plaintext rotation

In this section we define a rotation operation that is publicly computable and rotates plaintexts inside a packed ciphertext. It is described in Figure 5.8.

- $Rotate_{\mathsf{PAHE}}(\mathsf{ct} = (\boldsymbol{u}, \boldsymbol{v}), \mathsf{pk} = \rho, j \in \mathbb{N})$ returns $(\rho^j \boldsymbol{u}, \rho^j \boldsymbol{v})$.

Figure 5.8: Description of plaintext rotation.

Proposition 5.5.2 (Rotation correctness). *For any* $(\boldsymbol{m}_0, ..., \boldsymbol{m}_{t-1}) \in \mathbb{F}_q^{n \times t}$*, and a properly generated key* $(\mathsf{sk}, \mathsf{pk})$*, let* $\mathsf{ct} = (\boldsymbol{u}, \boldsymbol{v})$ *be obtained by* $\mathsf{ct} \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{PAHE}}(\mathsf{sk}, \boldsymbol{m}_0, ..., \boldsymbol{m}_{t+1})$, $\mathsf{ct}' \xleftarrow{\$} Rotate_{\mathsf{PAHE}}(\mathsf{ct}, \mathsf{pk}, j)$*. We have* $\mathsf{Decrypt}_{\mathsf{PAHE}}(\mathsf{sk}, \mathsf{ct}') = (\boldsymbol{m}_j, ..., \boldsymbol{m}_{j+t-1})$[1].

Proof. Like in the proof of Proposition 5.5.1, we have

$$\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u} = \sum_{1 \leq i \leq w} f_i \star \boldsymbol{e}_i + \sum_{0 \leq i < t} g_i \star \boldsymbol{m}_i.$$

By noting $\mathsf{ct}' = (\boldsymbol{u}', \boldsymbol{v}')$ and $\mathsf{pk} = \rho$ we get

$$\boldsymbol{v}' - \boldsymbol{s} \cdot \boldsymbol{u}' = \sum_{1 \leq i \leq w} (\rho^j f_i) \star \boldsymbol{e}_i + \sum_{0 \leq i < t} (\rho^j g_i) \star \boldsymbol{m}_i.$$

Changing variables $i' = i - j$ in the second sum gives

$$\begin{aligned} \boldsymbol{v}' - \boldsymbol{s} \cdot \boldsymbol{u}' &= \sum_{1 \leq i \leq w} (\rho^j f_i) \star \boldsymbol{e}_i + \sum_{0 \leq i' < t} (\rho^j g_{i'+j}) \star \boldsymbol{m}_{i+j} \\ &= \sum_{1 \leq i \leq w} (\rho^j f_i) \star \boldsymbol{e}_i + \sum_{0 \leq i' < t} g_{i'} \star \boldsymbol{m}_{i+j} \end{aligned}$$

Now just like in the proof of Proposition 5.5.1, we can write for any $0 \leq k < t$,

$$\begin{aligned} (\boldsymbol{D}^{(k+1)})^T \mathbf{Mat}(\boldsymbol{v}' - \boldsymbol{s} \cdot \boldsymbol{u}') &= \sum_{1 \leq i \leq w} (\boldsymbol{D}^{(k+1)})^T \mathbf{vec}(\rho^j f_i) \star \boldsymbol{e}_i + \sum_{0 \leq i' < t} (\boldsymbol{D}^{(k+1)})^T \mathbf{vec}(g_{i'}) \star \boldsymbol{m}_{i+j} \\ &= \boldsymbol{m}_{k+j} \end{aligned}$$

because thanks to the definition of $\boldsymbol{D} = (\boldsymbol{B}^{-1})^T$, for any $1 \leq i \leq w$, $(\boldsymbol{D}^{(k+1)})^T \mathbf{vec}(\rho^j f_i) = 0$, $(\boldsymbol{D}^{(k+1)})^T \mathbf{vec}(g_k) = 1$ and for any $0 \leq i' < t, i' \neq k$, $(\boldsymbol{D}^{(k+1)})^T \mathbf{vec}(g_{i'}) = 0$. □

Remark. In particular, $Rotate_{\mathsf{PAHE}}(\cdot, \mathsf{pk}, t) = id$.

5.5.3 Homomorphic decrytion with packing

The new relinearization with packing is presented in Figure 5.9. The key switching material (ksk and projk) is now composed of $2(w+1)$ packed ciphertexts (instead of $2m$ simple ciphertexts in Figure 5.5).

The following proposition establishes that a ciphertext ct encrypting a **single** plaintext $\boldsymbol{m}$ (without packing), after homomorphic decryption with a PAHE key switching material, decrypts correctly.

[1] The indexes are taken modulo t

- $\mathsf{GenKeySwitch}_{\mathsf{PAHE}}(\mathsf{sk}_1 = (\boldsymbol{f}_1, \boldsymbol{g}_1, \boldsymbol{D}_1, \boldsymbol{s}_1))$:
  - generates a new basis $\boldsymbol{b}_2$ of $\mathbb{F}_{q^m}$, $\mathsf{sk}_2 = (\boldsymbol{f}_2, \boldsymbol{g}_2, \boldsymbol{D}_2, \boldsymbol{s}_2)$ and pk_2 as in $\mathsf{KeyGen}_{\mathsf{PAHE}}$
  - for $1 \leq i \leq m$ defines $\boldsymbol{s}_{1,i} = \boldsymbol{d}^T \mathbf{Mat}(\gamma_i \star \boldsymbol{s}_1)$ with $\boldsymbol{d} = \boldsymbol{D}_1^{(1)}$
  - defines $\mathsf{ksk} = (\mathsf{ksk}_0, \ldots, \mathsf{ksk}_w)$, with $\mathsf{ksk_i} \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{PAHE}}(\mathsf{sk}_2, (\boldsymbol{s}_{1,it+j})_{1\leq j\leq t})$
  - for $1 \leq i \leq m$ defines $\boldsymbol{p}_i = \boldsymbol{d}^{(i)} \star (1, 0, \ldots, 0) \in \mathbb{F}_{q^m}^n$
  - defines $\mathsf{projk} = (\mathsf{projk}_0, \ldots, \mathsf{projk}_w)$ with $\mathsf{projk}_i \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{PAHE}}(\mathsf{sk}_2, (\boldsymbol{p}_{it+j})_{1\leq j\leq t}))$
  - returns $(\mathsf{sk}_2, \mathsf{pk}_2, \mathsf{ksk}, \mathsf{projk})$.

- $\mathsf{HomDecrypt}_{\mathsf{PAHE}}(\mathsf{ksk}, \mathsf{projk}, \mathsf{ct} = (\boldsymbol{u}, \boldsymbol{v}), \mathsf{pk}_2)$:
  - defines $\boldsymbol{u}_i \in \mathbb{F}_q^n$ such that $\boldsymbol{u} = \sum_{1\leq i\leq m} \gamma_i \star \boldsymbol{u}_i$
  - defines $\boldsymbol{v}_i \in \mathbb{F}_q^n$ such that $\boldsymbol{v} = \sum_{1\leq i\leq m} \gamma_i \star \boldsymbol{v}_i$
  - computes $\mathsf{ct}_1 = \sum_{1\leq i\leq m} \boldsymbol{v}_i \cdot \mathit{Rotate}_{\mathsf{PAHE}}(\mathsf{projk}_{\lfloor (i-1)/t \rfloor}, \mathsf{pk}_2, (i-1))$
  - computes $\mathsf{ct}_2 = \sum_{1\leq i\leq m} \boldsymbol{u}_i \cdot \mathit{Rotate}_{\mathsf{PAHE}}(\mathsf{ksk}_{\lfloor (i-1)/t \rfloor}, \mathsf{pk}_2, (i-1))$
  - returns $\mathsf{ct}_1 - \mathsf{ct}_2$

Figure 5.9: Homomorphic Decryption Packing Algorithms.

Proposition 5.5.3 (Homomorphic Decryption with packing Correctness)**.** *For any properly generated key* sk_1 *and* $\boldsymbol{m} \in \mathbb{F}_q^n$, $\mathsf{ct} \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}_1, \boldsymbol{m})$, $(\mathsf{sk}_2, \mathsf{pk}_2, \mathsf{ksk}, \mathsf{projk}) \xleftarrow{\$} \mathsf{GenKeySwitch}_{\mathsf{PAHE}}(\mathsf{sk}_1)$, $\mathsf{Decrypt}_{\mathsf{PAHE}}(\mathsf{sk}_2, \mathsf{HomDecrypt}_{\mathsf{PAHE}}(\mathsf{ksk}, \mathsf{projk}, \mathsf{ct}, \mathsf{pk}_2)) = \boldsymbol{m}$.

Proof. Let $1 \leq i \leq m$. By Proposition 5.5.2,

$$\mathit{Rotate}_{\mathsf{PAHE}}(\mathsf{projk}_{\lfloor (i-1)/t \rfloor}, \mathsf{pk}_2, (i-1)) \in \{\mathsf{Encrypt}_{\mathsf{PAHE}}(\mathsf{sk}_2, (\boldsymbol{p}_{\alpha(i,j)})_{1\leq j\leq t})\}.$$

where

$$\begin{aligned} \alpha : [1, m] \times [1, t] &\longrightarrow [1, m] \\ (i, j) &\longmapsto \lfloor \frac{i-1}{t} \rfloor t + 1 + ((i + j - 2) \mod t). \end{aligned}$$

Similarly,

$$\mathit{Rotate}_{\mathsf{PAHE}}(\mathsf{ksk}_{\lfloor (i-1)/t \rfloor}, \mathsf{pk}_2, (i-1)) \in \{\mathsf{Encrypt}_{\mathsf{PAHE}}(\mathsf{sk}_2, (\boldsymbol{s}_{1,\alpha(i,j)})_{1\leq j\leq t})\}.$$

Using 5.5.1,

$$(\mathsf{ct}_1 - \mathsf{ct}_2) \in \{\mathsf{Encrypt}_{\mathsf{PAHE}}(\mathsf{sk}_2, \sum_{1\leq i\leq m} (\boldsymbol{v}_i \cdot \boldsymbol{p}_{\alpha(i,j)} - \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,\alpha(i,j)})_{1\leq j\leq t})\}.$$

Using $\mathsf{Decrypt}_{\mathsf{AHE}}$ on ciphertext $(\mathsf{ct}_1 - \mathsf{ct}_2)$ corresponds to retrieving the first component of the packed plaintext hence

$$\mathsf{Decrypt}_{\mathsf{AHE}}(\mathsf{sk}_2, \mathsf{HomDecrypt}_{\mathsf{PAHE}}(\mathsf{ksk}, \mathsf{projk}, \mathsf{ct}, \mathsf{pk}_2)) = \sum_{1\leq i\leq m} \boldsymbol{v}_i \cdot \boldsymbol{p}_{\alpha(i,1)} - \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,\alpha(i,1)}.$$

Noting that for all $1 \leq i \leq m$, we have the identity $\alpha(i, 1) = i$,

$$\begin{aligned}\mathsf{Decrypt}_{\mathsf{AHE}}(\mathsf{sk}_2, \mathsf{HomDecrypt}_{\mathsf{PAHE}}(\mathsf{ksk}, \mathsf{projk}, \mathsf{ct}, \mathsf{pk}_2)) &= \sum_{1\le i\le m} \boldsymbol{v}_i \cdot \boldsymbol{p}_i - \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,i} \\ &= \boldsymbol{m}\end{aligned}$$

using the same argument as in the proof of Proposition 5.3.1. □

Remark. After homomorphic decryption, the resulting ciphertext $\mathsf{ct}' = \mathsf{HomDecrypt}_{\mathsf{PAHE}}(\mathsf{ksk}, \mathsf{projk}, \mathsf{ct}, \mathsf{pk}_2)$ **does not** belong to the distribution $\mathsf{Encrypt}_{\mathsf{AHE}}(\mathsf{sk}_2, \boldsymbol{m})$ nor $\mathsf{Encrypt}_{\mathsf{PAHE}}(\mathsf{sk}_2, (\boldsymbol{m}, \cdot, \dots))$. Indeed, the noise component of ct' lives in the bigger space $\tilde{F}$ (as defined in $\mathsf{KeyGen}_{\mathsf{PAHE}}$), whereas the noise component of the freshly encrypted ciphertexts belong to F. It does not impact security since ct' results from public operations only with fresh ciphertexts ct, ksk and projk.

Remark. Because the $\mathsf{HomDecrypt}_{\mathsf{PAHE}}$ operation consists essentially in a homomorphic evaluation of the orthogonal projection of $\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u}$ on $\langle g_0 \rangle_{\mathbb{F}_q}$, note that the initial ciphertext ct does not need to be fresh (i.e. with its noise in F). In particular, Proposition 5.5.3 is still valid when ct results from a previous homomorphic decryption with another set of keys.

Remark. If the initial ciphertext ct had been taken with packed plaintexts (i.e. drawn from $\mathsf{Encrypt}_{\mathsf{PAHE}}$ instead of $\mathsf{Encrypt}_{\mathsf{AHE}}$), the homomorphic decryption would be correct only for the first component of the packed plaintext. Other components would be lost.

5.6 Parameters

In this section we give example parameters for our scheme. Several sets are proposed for different values of d, the number of possible multiplications in the SHE. The parameter selection ran through the following steps. For given m and n, the weight w is set to the minimum between the half-rate rank Gilbert-Varshamov bound d_{rGV} and w^{d-1} (to prevent an overflow on the noise after d multiplications). We search for the lowest n such that a sufficient number of ciphertexts $\ell = 3w/4$ can be published, i.e. the best attacks against IRSD in an s-ideal $[sn, n]_{q^m}$ random code for every $2 \le s \le \ell$ are above the security level. Two attacks against IRSD were taken into account:

- the combinatorial attack from [AGHT18] whose complexity is given by

$$(sn-n)^{\omega} m^{\omega} q^{w\lceil \frac{m(n+1)}{sn} \rceil - m}$$

  for ω the linear algebra exponent;

- the algebraic attack from [BBC+20] whose complexity is given by $q^{aw} m \binom{sn-n-1}{w} \binom{sn-a}{w}^{\omega-1}$ where a is defined as the smallest integer such that the condition $m\binom{sn-n-1}{w} \ge \binom{sn-a}{w} - 1$ is fulfilled.

If no such n can be found, the process restarts with an increased m. The SageMath script of our parameter selection is available at:

https://www.github.com/victordyseryn/rank-fhe-parameter-selection

The key and ciphertext sizes in bits are given by the following formulas:

$$\begin{aligned}|\mathsf{sk}| &= \log_2(q)(m^2 + nw) \\ |\mathsf{ct}| &= 2\log_2(q)mn.\end{aligned}$$

The approximate timings for addition, multiplication and bootstrapping operations are estimated as follows:

- The addition consists in the sum of two vectors in $\mathbb{F}_{q^m}^n$, the number of bit operations is then

$$\mathcal{T}_{\text{Add}} = 2mn;$$

- The multplication consists in three multiplications of vectors in $\mathbb{F}_{q^m}^n$. The use of the Karatsuba algorithm gives a number of bits operations of

$$\mathcal{T}_{\text{Mul}} = 3(mn)^{1.6};$$

- The bootstrapping requires $2m$ plaintext absorptions, i.e. a multiplication of a vector in $\mathbb{F}_q^n$ times a vector in $\mathbb{F}_{q^m}^n$. With Karatsuba algorithm, the number of bit operations of a plaintext absorption is $mn^{1.6}$, hence the total cost of bootstrapping is

$$\mathcal{T}_{\text{Bootstrap}} = 2m^2n^{1.6}.$$

  The bootstrapping time is informative only as it is unsecure as proven in Section 5.4.

The timings in milliseconds are then computed by diving the number of bit operations by 3 millions, accounting for a processor running at 3 GHz. Note that this is an extremely conservative estimation, as modern processors run several bit operations in one clock cycle.

Our parameters are presented in the following table:

d	q	m	n	w	ℓ	Security	Key size	ct size	Add	Mul	Bootstrap
1	2	172	20	13	9	128	3.7 kB	0.9 kB	0.002 ms	0.5 ms	2 ms
2	2	367	183	7	5	128	17.0 kB	16.8 kB	0.04 ms	52 ms	374 ms
3	2	1296	314	6	4	128	210 kB	102 kB	0.3 ms	944 ms	11 s
4	2	3125	713	5	3	128	1.22 MB	557 kB	1 ms	14.3 s	239 s

Table 5.1: Example of paramaters for our SHE scheme, with associated sizes and execution timings. d is the number of possible multiplications. q, m and n are parameters of the rank linear code and w is the rank weight of the error. ℓ is the number of independant ciphertexts that can be published.

The sizes and expected performance of our somewhat homomorphic encryption scheme are very positive, and could already be used for practical applications with a small number of multiplications.

These numbers additionally show a strong potential regarding bootstrapping, should it be repaired. With bootstrapping enabled, the only parameters to consider would be those for which $d = 1$, and in such a context bootstrapping would be very efficient with an unoptimized running time of only 2 milliseconds. For example, it is more than 6 times more efficient than the bootstrap in TFHE [CGGI20], one of the most popular and widely used lattice-based FHE framework. Our scheme shows that error correcting codes can lead to very competitive FHE constructions.

Conclusion

Code-based cryptography mainly revolves around three difficult problems: the syndrome decoding problems – and their cyclic variants – either in the Hamming or in the rank metric; and the permuted kernel problem.

We presented in Chapter 2 a recent approach to improve the efficiency of code-based schemes, as an alternative to quasi-cyclic or ideal structure, which can be a potential vulnerability. This multi-dimensional approach consists in sampling several errors sharing the same support, or several vectors sharing the same permutation in the case of PKP. Being intensive by nature, the rank metric seems particularly well suited for this approach. We presented crytanalytic efforts on the multi-dimensional variants of the decoding problems. The complexity of attacks has stabilized over the years and it seems that multiple instances of errors with the same support do not bring any advantage to the attacker, when the number of instances remains small. The multi-dimensional approach was the central theme of this manuscript; we explored new and existing applications of this technique.

For rank metric encryption schemes, two families of codes are particularly well adapted to a multi-dimensional (or interleaved) decoding: LRPC codes and Gabidulin codes. We introduced in Chapter 3 two encryption schemes based on the improvements brought by the decoding of multiple errors with the same support: LRPC-MS and LowMS. We proved an upper bound on their decoding failure rate, and consequently showed their IND-CPA security. They achieve very competitive sizes, especially when compared to other cryptosystems that do not feature a quasi-cyclic or ideal structure.

Chapter 4 began with an attack against a signature scheme using the multi-dimensional approach, Durandal. The attack targeted a very specific problem, PSSI, which was created just for Durandal and is totally unliked to the multi-dimensional approach. In the same chapter, we presented our contributions to the NIST onramp standardization call for digital signature with three submissions: MIRA, PERK and RYDE. We gave a special focus on PERK, by presenting a detailed analysis of the security of r-IPKP, a multi-dimensional variant of the traditional IPKP, which appears in the security proof of PERK.

Chapter 5 concluded the manuscript with the presentation of a homomorphic encryption scheme based on random ideal codes in the rank metric. The novel ideas of using the framework of Alekhnovich for homomorphic encryption, as well as the perpendicular encoding of errors and messages, allowed us to build a simple and efficient somewhat homomorphic scheme. Even though it is currently insecure, we also presented a bootstrapping algorithm that demonstrates the high potential of code-based homomorphic encryption.

Perspectives

The multi-dimensional approach opens a large field of possibilities; we present below a few options for further work.

- A reduction from multi-dimensional problems to their single-dimensional counterparts would strenghen the confidence in the security of the multi-dimensional approach;
- We investigated the benefits of the multi-dimensional approach applied to encryption schemes in the rank metric. Similar results in the Hamming metric, especially on the families of low density parity-check (LRPC) codes, found in BIKE [ABB+22] or Reed-Muller-Reed-Solomon codes, found in HQC [AAB+22] would be very attractive;
- The decoding of LRPC codes can fail for two reasons: either the syndrome space does not generate the full product space EF, or the chain of intersections $\cap f_i^{-1}EF$ is strictly larger than E. Although the first probability is well understood, even for the multi-dimensional case as shown in Chapter 3, an efficient upper bound the second probability was not rigorously proven. An improvement of the upper bound from [BO23] would be a great step towards a better understanding of LRPC codes;

- Finding a hash-and-sign signature scheme in the rank metric with secure parameters is still an open problem. It suffices to find a new family of rank metric codes that have a large density of decodable syndromes and that can be securely masked. As mentioned earlier, it would also be a challenge to take advantage of the multi-dimensional approach to build such a scheme;
- Our attack on the PSSI problem presented in Section 4.1 computes Cramer-like formulas of size 2×2. A generalization to size 3×3 or more could lead to a more efficient attack;
- We would like to find new parameters for Durandal that resist our attack. It possibly involves a slight modification to the scheme itself;
- The attack on r-IPKP we presented in Section 4.2.3 first converts the multi-dimensional instance into a mono-dimensional instance with many zeros, then solves it with a mono-dimensional enumeration algorithm adapted to the low weights. Finding a way to enumerate directly on the multi-dimensional instance and to find partial matches efficiently is a possibility to improve our attack;
- We would like to improve the packing capacity of our homomorphic construction in order to make the bootstrapping possible. A possible research direction could be to add some structure to the error.

Bibliography

[AAB+18] Ameera Salem Al Abdouli, Mohamed Al Ali, Emanuele Bellini, Florian Caullery, Alexandros Hasikos, Marc Manzano, and Victor Mateu. DRANKULA: a McEliece-like rank metric based cryptosystem implementation. Cryptology ePrint Archive, Report 2018/768, 2018. https://eprint.iacr.org/2018/768.

[AAB+19] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillippe Gaborit, Gilles Zémor, Alain Couvreur, and Adrien Hauteville. RQC. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions.

[AAB+21] Carlos Aguilar-Melchor, Nicolas Aragon, Emanuele Bellini, Florian Caullery, Rusydi H Makarim, and Chiara Marcolla. Constant time algorithms for ROLLO-I-128. *SN Computer Science*, 2:1–19, 2021.

[AAB+22] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, Jurjen Bos, Arnaud Dion, Jerome Lacan, Jean-Marc Robert, and Pascal Veron. HQC. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

[AAD+22] Carlos Aguilar-Melchor, Nicolas Aragon, Victor Dyseryn, Philippe Gaborit, and Gilles Zémor. LRPC codes with multiple syndromes: near ideal-size KEMs without ideals. In *International Conference on Post-Quantum Cryptography, PQCrypto*, pages 45–68. Springer, 2022.

[AADG21] Carlos Aguilar-Melchor, Nicolas Aragon, Victor Dyseryn, and Philippe Gaborit. Fast and secure key generation for low rank parity check codes cryptosystems. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 1260–1265. IEEE, 2021.

[AAPS11] Frederik Armknecht, Daniel Augot, Ludovic Perret, and Ahmad-Reza Sadeghi. On constructing homomorphic encryption schemes from coding theory. In Liqun Chen, editor, *13th IMA International Conference on Cryptography and Coding*, volume 7089 of *LNCS*, pages 23–40. Springer, Heidelberg, December 2011.

[AASA+20] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. Status report on the second round of the NIST post-quantum cryptography standardization process. *US Department of Commerce, NIST*, 2, 2020.

[ABB+21] Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Yann Connan, Jérémie Coulaud, Philippe Gaborit, and Anaïs Kominiarz. The rank-based cryptography library. In *Code-Based Cryptography Workshop*, pages 22–41. Springer, 2021.

[ABB+22] Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, Gilles Zémor, Valentin Vasseur, Santosh Ghosh, and Jan Richter-Brokmann. BIKE. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

[ABB+23a] Najwa Aaraj, Slim Bettaieb, Loïc Bidoux, Alessandro Budroni, Victor Dyseryn, Andre Esser, Philippe Gaborit, Mukul Kulkarni, Victor Mateu, Marco Palumbi, Lucas Perin, Jean-Pierre Tillich, and Adrien Vincotte. PERK. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

[ABB+23b] Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Antoine Joux, Matthieu Rivain, Jean-Pierre Tillich, and Adrien Vincotte. RYDE. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

[ABB+23c] Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, Matthieu Rivain, and Jean-Pierre Tillich. MIRA. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

[ABC+22] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic McEliece. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-4-submissions.

[ABD+19] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, Gilles Zémor, Carlos Aguilar Melchor, Slim Bettaieb, Loic Bidoux, Magali Bardet, and Ayoub Otmani. ROLLO. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions.

[ABG+19] Nicolas Aragon, Olivier Blazy, Philippe Gaborit, Adrien Hauteville, and Gilles Zémor. Durandal: A rank metric based signature scheme. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 728–758. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17659-4_25.

[ACK21] Thomas Attema, Ronald Cramer, and Lisa Kohl. A compressed Σ-protocol theory for lattices. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 549–579, Virtual Event, August 2021. Springer, Heidelberg. doi:10.1007/978-3-030-84245-1_19.

[ADG+22] Nicolas Aragon, Victor Dyseryn, Philippe Gaborit, Pierre Loidreau, Julian Renner, and Antonia Wachter-Zeh. LowMS: a new rank metric code-based KEM without ideal structure. *Cryptology ePrint Archive*, 2022.

[ADG23a] Carlos Aguilar-Melchor, Victor Dyseryn, and Philippe Gaborit. Somewhat Homomorphic Encryption based on Random Codes. Cryptology ePrint Archive, Paper 2023/1798, 2023. URL: https://eprint.iacr.org/2023/1798.

[ADG23b] Nicolas Aragon, Victor Dyseryn, and Philippe Gaborit. Analysis of the security of the PSSI problem and cryptanalysis of the durandal signature scheme. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 127–149. Springer, 2023. doi:10.1007/978-3-031-38548-3_5.

[AF22] Thomas Attema and Serge Fehr. Parallel repetition of $(k_1, \ldots, k_\mu)$-special-sound multi-round interactive proofs. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 415–443. Springer, Heidelberg, August 2022. doi:10.1007/978-3-031-15802-5_15.

[AFK22] Thomas Attema, Serge Fehr, and Michael Klooß. Fiat-shamir transformation of multi-round interactive proofs. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 113–142. Springer, Heidelberg, November 2022. doi:10.1007/978-3-031-22318-1_5.

[Age23] Agence Nationale de la Sécurité des Systèmes d'Information. ANSSI views on the Post-Quantum Cryptography transition (2023 follow up). Technical report, 2023.

[AGH+19] Nicolas Aragon, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, and Gilles Zémor. Low rank parity check codes: New decoding algorithms and applications to cryptography. *IEEE Transactions on Information Theory*, 65(12):7697–7717, 2019.

[AGHT18] Nicolas Aragon, Philippe Gaborit, Adrien Hauteville, and Jean-Pierre Tillich. A new algorithm for solving the rank syndrome decoding problem. In *2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018*, pages 2421–2425. IEEE, 2018. doi:10.1109/ISIT.2018.8437464.

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th FOCS*, pages 298–307. IEEE Computer Society Press, October 2003. doi:10.1109/SFCS.2003.1238204.

[BB21] Magali Bardet and Pierre Briaud. An algebraic approach to the rank support learning problem. In Jung Hee Cheon and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021*, pages 442–462. Springer, Heidelberg, 2021. doi:10.1007/978-3-030-81293-5_23.

[BBB+20] Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, Vincent Neiger, Olivier Ruatta, and Jean-Pierre Tillich. An algebraic attack on rank metric code-based cryptosystems. In Canteaut and Ishai [CI20], pages 64–93. doi:10.1007/978-3-030-45727-3_3.

[BBB+23] Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, and Jean-Pierre Tillich. Revisiting algebraic attacks on MinRank and on the rank decoding problem. *Designs, Codes and Cryptography*, pages 1–37, 2023.

[BBBG22] Loïc Bidoux, Pierre Briaud, Maxime Bros, and Philippe Gaborit. RQC revisited and more cryptanalysis for Rank-based Cryptography. *arXiv preprint arXiv:2207.01410*, 2022.

[BBC+20] Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray A. Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier A. Verbel. Improvements of algebraic attacks for solving the rank decoding and MinRank problems. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 507–536. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-64837-4_17.

[BBF+19] Nina Bindel, Jacqueline Brendel, Marc Fischlin, Brian Goncalves, and Douglas Stebila. Hybrid key encapsulation mechanisms and authenticated key exchange. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 206–226. Springer, Heidelberg, 2019. doi:10.1007/978-3-030-25510-7_12.

[BBP+23] 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*, 2023.

[BCCG93] Thierry Baritaud, Mireille Campana, Pascal Chauvaud, and Henri Gilbert. On the security of the permuted kernel identification scheme. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 305–311. Springer, Heidelberg, August 1993. doi:10.1007/3-540-48071-4_21.

[BCMN11] Paulo SLM Barreto, Pierre-Louis Cayrel, Rafael Misoczki, and Robert Niebuhr. Quasi-dyadic CFS signatures. In *Information Security and Cryptology: 6th International Conference, Inscrypt 2010, Shanghai, China, October 20-24, 2010, Revised Selected Papers 6*, pages 336–349. Springer, 2011.

[BDG20] Mihir Bellare, Hannah Davis, and Felix Günther. Separate your domains: NIST PQC KEMs, oracle cloning and read-only indifferentiability. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 3–32. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45724-2_1.

[Beu20a] Ward Beullens. Not enough LESS: An improved algorithm for solving code equivalence problems over $\mathbb{F}_q$. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn, editors, *SAC 2020*, volume 12804 of *LNCS*, pages 387–403. Springer, Heidelberg, October 2020. doi:10.1007/978-3-030-81652-0_15.

[Beu20b] Ward Beullens. Sigma protocols for MQ, PKP and SIS, and Fishy signature schemes. In Canteaut and Ishai [CI20], pages 183–211. doi:10.1007/978-3-030-45727-3_7.

[BFK+19] Ward Beullens, Jean-Charles Faugère, Eliane Koussa, Gilles Macario-Rat, Jacques Patarin, and Ludovic Perret. PKP-based signature scheme. In Feng Hao, Sushmita Ruj, and Sourav Sen Gupta, editors, *INDOCRYPT 2019*, volume 11898 of *LNCS*, pages 3–22. Springer, Heidelberg, December 2019. doi:10.1007/978-3-030-35423-7_1.

[BG23] Loïc Bidoux and Philippe Gaborit. Compact Post-quantum Signatures from Proofs of Knowledge Leveraging Structure for the PKP, SD and RSD Problems. In *International Conference on Codes, Cryptology, and Information Security*, pages 10–42. Springer, 2023.

[BGK17] Thierry P. Berger, Cheikh Thiécoumba Gueye, and Jean Belo Klamti. A NP-complete problem in coding theory with application to code based cryptography. In *Codes, Cryptology and Information Security - Second International Conference, C2SI 2017, Rabat, Morocco, April 10-12, 2017, Proceedings - In Honor of Claude Carlet*, pages 230–237, 2017.

[BGKM23] Loïc Bidoux, Philippe Gaborit, Mukul Kulkarni, and Victor Mateu. Code-based signatures from new proofs of knowledge for the syndrome decoding problem. *Designs, Codes and Cryptography*, 91(2):497–544, 2023.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 325–341. Springer, Heidelberg, February 2005. doi:10.1007/978-3-540-30576-7_18.

[BKY03] Daniel Bleichenbacher, Aggelos Kiayias, and Moti Yung. Decoding of interleaved reed solomon codes over noisy data. In Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *ICALP 2003*, volume 2719 of *LNCS*, pages 97–108. Springer, Heidelberg, June / July 2003. doi:10.1007/3-540-45061-0_9.

[BL] Daniel J Bernstein and Tanja Lange. eBATS: ECRYPT benchmarking of asymmetric systems. URL: https://bench.cr.yp.to/.

[BL11] Andrej Bogdanov and Chin Ho Lee. Homomorphic encryption from codes. Cryptology ePrint Archive, Report 2011/622, 2011. https://eprint.iacr.org/2011/622.

[BL23] Pierre Briaud and Pierre Loidreau. Cryptanalysis of Rank-Metric Schemes Based on Distorted Gabidulin Codes. In Thomas Johansson and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 14th International Workshop, PQCrypto 2023*, volume 14154 of *Lecture Notes in Computer Science*, pages 38–56. Springer, 2023.

[BMvT78] Elwyn Berlekamp, Robert McEliece, and Henk van Tilborg. On the inherent intractability of certain coding problems. *IEEE Trans. Inform. Theory*, 24(3):384–386, May 1978.

[BO23] Étienne Burle and Ayoub Otmani. An Upper-Bound on the Decoding Failure Probability of the LRPC Decoder. *arXiv preprint arXiv:2309.14028*, 2023.

[BOS19] Magali Bardet, Ayoub Otmani, and Mohamed Saeed-Taha. Permutation code equivalence is not harder than graph isomorphism when hulls are trivial. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2019*, pages 2464–2468, July 2019. URL: http://arxiv.org/abs/1905.00073.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011. doi:10.1109/FOCS.2011.12.

[CC20] Daniel Coggia and Alain Couvreur. On the security of a Loidreau rank metric code based encryption scheme. *Designs, Codes and Cryptography*, 88:1941–1957, 2020.

[CD23] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 423–447. Springer, Heidelberg, April 2023. doi:10.1007/978-3-031-30589-4_15.

[CDW21] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Mildly short vectors in cyclotomic ideal lattices in quantum polynomial time. *Journal of the ACM (JACM)*, 68(2):1–26, 2021.

[CFS01] Nicolas Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 157–174. Springer, Heidelberg, December 2001. doi:10.1007/3-540-45682-1_10.

[CGG+14] Alain Couvreur, Philippe Gaborit, Valérie Gauthier-Umaña, Ayoub Otmani, and Jean-Pierre Tillich. Distinguisher-based attacks on public-key cryptosystems using Reed-Solomon codes. *Des. Codes Cryptogr.*, (2):641–666, 2014.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, January 2020. doi:10.1007/s00145-019-09319-x.

[CI20] Anne Canteaut and Yuval Ishai, editors. *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*. Springer, Heidelberg, May 2020.

[CKN20] Jinkyu Cho, Young-Sik Kim, and Jong-Seon No. Homomorphic computation in reed-muller codes. Cryptology ePrint Archive, Report 2020/565, 2020. https://eprint.iacr.org/2020/565.

[CL22] Tung Chou and Jin-Han Liou. A constant-time AVX2 implementation of a variant of ROLLO. *IACR TCHES*, 2022(1):152–174, 2022. doi:10.46586/tches.v2022.i1.152-174.

[Del78] Philippe Delsarte. Bilinear forms over a finite field, with applications to coding theory. *J. Comb. Theory, Ser. A*, 25(3):226–241, 1978.

[DGK20] Nir Drucker, Shay Gueron, and Dusan Kostic. Fast polynomial inversion for post quantum QC-MDPC cryptography. In *International Symposium on Cyber Security Cryptography and Machine Learning*, pages 110–127. Springer, 2020.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46800-5_24.

[DST19] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 21–51. Springer, Heidelberg, December 2019. doi:10.1007/978-3-030-34578-5_2.

[DT18] Thomas Debris-Alazard and Jean-Pierre Tillich. Two attacks on rank metric code-based schemes: RankSign and an IBE scheme. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 62–92. Springer, Heidelberg, December 2018. doi:10.1007/978-3-030-03326-2_3.

[Dum89] Il'ya Dumer. Two decoding algorithms for linear codes. *Probl. Inf. Transm.*, 25(1):17–23, 1989.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985. doi:10.1109/TIT.1985.1057074.

[EVZB23] Andre Esser, Javier Verbel, Floyd Zweydinger, and Emanuele Bellini. `CryptographicEstimators`: a Software Library for Cryptographic Hardness Estimation. *Cryptology ePrint Archive*, 2023.

[EWZZ18] Molka Elleuch, Antonia Wachter-Zeh, and Alexander Zeh. A public-key cryptosystem from interleaved Goppa codes. *arXiv preprint arXiv:1809.03024*, 2018.

[Fen22] Thibauld Feneuil. Building MPCitH-based signatures from MQ, MinRank, rank SD and PKP. Cryptology ePrint Archive, Report 2022/1512, 2022. https://eprint.iacr.org/2022/1512.

[FJR22] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 541–572. Springer, Heidelberg, August 2022. doi:10.1007/978-3-031-15979-4_19.

[FJR23] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Shared permutation for syndrome decoding: new zero-knowledge protocol and code-based signature. *Designs, Codes and Cryptography*, 91(2):563–608, 2023.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Heidelberg, August 1999. doi:10.1007/3-540-48405-1_34.

[FOP+16] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, Frédéric de Portzamparc, and Jean-Pierre Tillich. Structural cryptanalysis of McEliece schemes with compact keys. *Des. Codes Cryptogr.*, 79(1):87–112, 2016.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987. doi:10.1007/3-540-47721-7_12.

[Gab85] Ernest M. Gabidulin. Theory of codes with maximum rank distance. *Problemy Peredachi Informatsii*, 21(1):3–16, 1985.

[Gab05] Philippe Gaborit. Shorter keys for code based cryptography. In *Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005)*, pages 81–91, Bergen, Norway, March 2005.

[Gao93] Shuhong Gao. *Normal bases over finite fields*. PhD thesis, University of Waterloo, 1993.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009. doi:10.1145/1536414.1536440.

[Geo92] Jean Georgiades. Some remarks on the security of the identification scheme based on permuted kernels. *Journal of Cryptology*, 5(2):133–137, January 1992. doi:10.1007/BF00193565.

[Gha22] Anirban Ghatak. Extending Coggia–Couvreur attack on Loidreau's rank-metric cryptosystem. *Designs, Codes and Cryptography*, 90(1):215–238, 2022.

[GHPT17] Philippe Gaborit, Adrien Hauteville, Duong Hieu Phan, and Jean-Pierre Tillich. Identity-based encryption from codes with rank metric. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 194–224. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63697-9_7.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, 1979.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th ACM STOC*, pages 365–377. ACM Press, May 1982. doi:10.1145/800070.802212.

[GMRZ13] Philippe Gaborit, Gaétan Murat, Olivier Ruatta, and Gilles Zémor. Low rank parity check codes and their application to cryptography. In *Proceedings of the Workshop on Coding and Cryptography WCC'2013*, Bergen, Norway, 2013. URL: www.selmer.uib.no/WCC2013/pdfs/Gaborit.pdf.

[GPT91] Ernst M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov. Ideals over a non-commutative ring and thier applications in cryptology. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 482–489. Springer, Heidelberg, April 1991. doi:10.1007/3-540-46416-6_41.

[GRS16] Philippe Gaborit, Olivier Ruatta, and Julien Schrek. On the complexity of the rank syndrome decoding problem. *IEEE Trans. Information Theory*, 62(2):1006–1019, 2016.

[GRSZ14a] Philippe Gaborit, Olivier Ruatta, Julien Schrek, and Gilles Zémor. New results for rank-based cryptography. In *Progress in Cryptology - AFRICACRYPT 2014*, volume 8469 of *LNCS*, pages 1–12, 2014.

[GRSZ14b] Philippe Gaborit, Olivier Ruatta, Julien Schrek, and Gilles Zémor. RankSign: An efficient signature algorithm based on the rank metric. In Michele Mosca, editor, *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014*, pages 88–107. Springer, Heidelberg, October 2014. doi:10.1007/978-3-319-11659-4_6.

[GZ16] Philippe Gaborit and Gilles Zémor. On the hardness of the decoding and the minimum distance problems for rank codes. *IEEE Trans. Inform. Theory*, 62(12):7245–7252, 2016.

[HBD+22] Andreas Hülsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. SPHINCS$^+$. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 341–371. Springer, Heidelberg, November 2017. doi:10.1007/978-3-319-70500-2_12.

[HLPWZ19] Lukas Holzbaur, Hedongliang Liu, Sven Puchinger, and Antonia Wachter-Zeh. On Decoding and Applications of Interleaved Goppa Codes. In *IEEE Int. Symp. Inf. Theory (ISIT)*, July 2019.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007. doi:10.1145/1250790.1250794.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st ACM STOC*, pages 12–24. ACM Press, May 1989. doi:10.1145/73007.73009.

[IT88] Toshiya Itoh and Shigeo Tsujii. A fast algorithm for computing multiplicative inverses in GF(2m) using normal bases. *Information and computation*, 78(3):171–177, 1988.

[JAC+22] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, David Urbanik, Geovandro Pereira, Koray Karabina, and Aaron Hutchinson. SIKE. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

[JJ01] Éliane Jaulmes and Antoine Joux. Cryptanalysis of PKP: A new approach. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 165–172. Springer, Heidelberg, February 2001. doi:10.1007/3-540-44586-2_12.

[KKS97] Gregory Kabatianskii, E. Krouk, and Ben J. M. Smeets. A digital signature scheme based on random error-correcting codes. In Michael Darnell, editor, *6th IMA International Conference on Cryptography and Coding*, volume 1355 of *LNCS*, pages 161–167. Springer, Heidelberg, December 1997.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537. ACM Press, October 2018. doi:10.1145/3243734.3243805.

[KMP19] Eliane Koussa, Gilles Macario-Rat, and Jacques Patarin. On the complexity of the permuted kernel problem. Cryptology ePrint Archive, Report 2019/412, 2019. https://eprint.iacr.org/2019/412.

[KZ20] Daniel Kales and Greg Zaverucha. An attack on some signature schemes constructed from five-pass identification schemes. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *CANS 20*, volume 12579 of *LNCS*, pages 3–22. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-65411-5_1.

[LDK+22] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[LJS+16] Carl Löndahl, Thomas Johansson, Masoumeh Koochak Shooshtari, Mahmoud Ahmadian-Attari, and Mohammad Reza Aref. Squaring attacks on McEliece public-key cryptosystems using quasi-cyclic codes of even dimension. *Des. Codes Cryptogr.*, 80(2):359–377, 2016. URL: http://dx.doi.org/10.1007/s10623-015-0099-x, doi:10.1007/s10623-015-0099-x.

[Loi07] Pierre Loidreau. *Metrique rang et cryptographie*. HDR thesis, Université Pierre et Marie Curie-Paris VI, 2007.

[Loi17] Pierre Loidreau. A new rank metric codes based encryption scheme. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017*, pages 3–17. Springer, Heidelberg, 2017. doi:10.1007/978-3-319-59879-6_1.

[Loi22] Pierre Loidreau. Analysis of a public-key encryption scheme based on distorted Gabidulin codes. In *Proceedings of the twelth international workshop on coding and cryptography WCC*, 2022.

[LP11] Rodolphe LAMPE and Jacques PATARIN. Analysis of some natural variants of the PKP algorithm. Cryptology ePrint Archive, Report 2011/686, 2011. https://eprint.iacr.org/2011/686.

[LP21] Pierre Loidreau and Ba-Duc Pham. An analysis of Coggia-Couvreur attack on Loidreau's rank-metric public key encryption scheme in the general case. *arXiv preprint arXiv:2112.12445*, 2021.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Heidelberg, December 2009. doi:10.1007/978-3-642-10366-7_35.

[McE78] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. The deep space network progress report 42-44, Jet Propulsion Laboratory, California Institute of Technology, January/February 1978. https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF.

[MK90] John J. Metzner and Edward J. Kapturowski. A General Decoding Technique Applicable to Replicated File Disagreement Location and Concatenated Code Decoding. *IEEE Trans. Inf. Theory*, 36(4):911–917, 1990.

[MS77] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error-correcting codes*, volume 16. Elsevier, 1977.

[NAB+20] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[Nie86] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166, 1986.

[NISa] NIST. Post-Quantum Cryptography: Digital Signature Schemes. URL: https://csrc.nist.gov/projects/pqc-dig-sig/standardization/call-for-proposals.

[NISb] NIST. Post-Quantum Cryptography Standardization. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization.

[NY01] Peng Ning and Yiqun Lisa Yin. Efficient software implementation for finite field multiplication in normal basis. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *ICICS 01*, volume 2229 of *LNCS*, pages 177–188. Springer, Heidelberg, November 2001.

[OT11] Ayoub Otmani and Jean-Pierre Tillich. An efficient attack on all concrete KKS proposals. In Yang [Yan11], pages 98–116. doi:10.1007/978-3-642-25405-5_7.

[Ove05] Raphael Overbeck. A new structural attack for GPT and variants. In *Mycrypt*, volume 3715 of *LNCS*, pages 50–63, 2005.

[Ove08] R. Overbeck. Structural attacks for public key cryptosystems based on Gabidulin codes. *Journal of Cryptology*, 21(2):280–301, April 2008. doi:10.1007/s00145-007-9003-9.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999. doi:10.1007/3-540-48910-X_16.

[PC94] Jacques Patarin and Pascal Chauvaud. Improved algorithms for the permuted kernel problem. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 391–402. Springer, Heidelberg, August 1994. doi:10.1007/3-540-48329-2_33.

[PFH+22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[Pha21] Ba Duc Pham. *Étude et conception de nouvelles primitives de chiffrement fondées sur les codes correcteurs d'erreurs en métrique rang*. PhD thesis, Rennes 1, 2021.

[PR97] Erez Petrank and Ron. Roth. Is code equivalence easy to decide? *IEEE Trans. Inform. Theory*, 43(5):1602–1604, 1997.

[Pra62] Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962. URL: http://dx.doi.org/10.1109/TIT.1962.1057777, doi:10.1109/TIT.1962.1057777.

[PT21] Thales Bandiera Paiva and Routo Terada. Cryptanalysis of the binary permuted kernel problem. In Kazue Sako and Nils Ole Tippenhauer, editors, *ACNS 21, Part II*, volume 12727 of *LNCS*, pages 396–423. Springer, Heidelberg, June 2021. doi:10.1007/978-3-030-78375-4_16.

[RAD78] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[RJB19] Julian Renner, Thomas Jerkovits, and Hannes Bartz. Efficient Decoding of Interleaved Low-rank Parity-check Codes. In *2019 XVI International Symposium "Problems of Redundancy in Information and Control Systems"(REDUNDANCY)*, pages 121–126. IEEE, 2019.

[RPWZ19] Julian Renner, Sven Puchinger, and Antonia Wachter-Zeh. Interleaving Loidreau's Rank-Metric Cryptosystem. In *2019 XVI International Symposium "Problems of Redundancy in Information and Control Systems"(REDUNDANCY)*, pages 127–132. IEEE, 2019.

[RPWZ21] Julian Renner, Sven Puchinger, and Antonia Wachter-Zeh. Decoding High-Order Interleaved Rank-Metric Codes. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 19–24. IEEE, 2021.

[SAB+22] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[SBC22] Paolo Santini, Marco Baldi, and Franco Chiaraluce. Computational hardness of the permuted kernel and subcode equivalence problems. Cryptology ePrint Archive, Report 2022/1749, 2022. https://eprint.iacr.org/2022/1749.

[Sen00] Nicolas Sendrier. Finding the permutation between equivalent linear codes: The support splitting algorithm. *IEEE Trans. Inform. Theory*, 46(4):1193–1203, 2000.

[Sen11] Nicolas Sendrier. Decoding one out of many. In Yang [Yan11], pages 51–67. doi:10.1007/978-3-642-25405-5_4.

[Sha90] Adi Shamir. An efficient identification scheme based on permuted kernels (extended abstract) (rump session). In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 606–609. Springer, Heidelberg, August 1990. doi:10.1007/0-387-34805-0_54.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994. doi:10.1109/SFCS.1994.365700.

[Sho09] Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge university press, 2009.

[Sid94] Vladimir Michilovich Sidelnikov. A public-key cryptosystem based on binary Reed-Muller codes. 1994.

[SJB11] Vladimir Sidorenko, Lan Jiang, and Martin Bossert. Skew-feedback shift-register synthesis and decoding interleaved Gabidulin codes. *IEEE Transactions on Information Theory*, 57(2):621–632, 2011.

[SS92] Vladimir Michilovich Sidelnikov and Sergey O Shestakov. On insecurity of cryptosystems based on generalized Reed-Solomon codes. 1992.

[Wal01] Colin D. Walter. Sliding windows succumbs to big mac attack. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *CHES 2001*, volume 2162 of *LNCS*, pages 286–299. Springer, Heidelberg, May 2001. doi:10.1007/3-540-44709-1_24.

[Wan19] Li-Ping Wang. Loong: a new IND-CCA-secure code-based KEM. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 2584–2588. IEEE, 2019.

[Yan11] Bo-Yin Yang, editor. *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*. Springer, Heidelberg, November / December 2011.

[ZCD+20] Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Jonathan Katz, Xiao Wang, Vladmir Kolesnikov, and Daniel Kales. Picnic. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

Appendix A

Proofs for the introducing chapters

A.1 Asymptotic equivalent for the multi-rank Gilbert-Varshamov bound

Proposition A.1.1. *When $m = n$ and $\ell \to \infty$, the multi-rank Gilbert-Varshamov bound d_{mRGV} defined as the smallest integer such that*

$$\frac{\binom{m}{w}_q q^{w\ell n}}{q^{m\ell(n-k)}} > 1,$$

admits an asymptotic equivalent

$$d_{mRGV} \sim n\left(1 - \frac{k}{n}\right).$$

Proof. The inequality defining d_{mRGV} becomes an equality in the asymptotic regime, i.e.

$$\binom{m}{d_{mRGV}}_q q^{d_{mRGV}\ell n} = q^{m\ell(n-k)} \tag{A.1}$$

The asymptotic equivalent of the gaussian binomial coefficient is

$$\binom{m}{d_{mRGV}}_q \sim q^{d_{mRGV}(m-d_{mRGV})}$$

hence by looking only at the exponential part of Equation (A.1), d_{mRGV} satisfies a degree two equation

$$d_{mRGV}(m - d_{mRGV}) + d_{mRGV}\ell n = m\ell(n-k),$$

which, when $m = n$ simplifies into

$$d_{mRGV}^2 - n(\ell+1)d_{mRGV} + n\ell(n-k) = 0.$$

The only solution to the above equation satisfying $0 \leq d_{mRGV} \leq n$ is

$$\begin{aligned} d_{mRGV} &\sim \frac{n(\ell+1) - \sqrt{n^2(\ell+1)^2 - 4n\ell(n-k)}}{2} \\ &\sim \frac{n(\ell+1)}{2}\left[1 - \sqrt{1 - \frac{4n\ell(n-k)}{n^2(\ell+1)^2}}\right] \\ &\sim \frac{n(\ell+1)}{2}\frac{4n\ell(n-k)}{2n^2(\ell+1)^2} \\ &\sim \frac{\ell}{\ell+1}(n-k) \end{aligned}$$

which gives the following equivalent, when $\ell \to \infty$,

$$d_{mRGV} \sim (n-k) = n\left(1 - \frac{k}{n}\right).$$

□

Appendix B

Proofs for LRPC-MS

B.1 Dimension of the support of the product of homogeneous matrices

In this section we prove the following theorem, which is required to prove the correctness of the multi-syndrome approach presented in Section 3.1. We fix E and F subspaces of $\mathbb{F}_{q^m}$ of dimension r and d respectively such that EF is of dimension rd. Remember that we have $q = 2$ all along the proof.

Theorem B.1.1. *For $q = 2$, $n_1 + n_2 \leq n$ and for $\boldsymbol{U}$ and $\boldsymbol{V}$ random variables chosen uniformly in $F^{n_1 \times n}$ and $E^{n \times n_2}$ (respectively), $\mathbb{P}(\mathbf{Supp}(\boldsymbol{UV}) \neq EF) \leq (n_1 + 1)q^{rd - n_1 n_2}$*

A first idea which may come to mind when trying to prove this theorem would be to use the Leftover Hash Lemma [ILL89] (LHL) in order to prove that the statistical distribution of $\boldsymbol{UV}$ is ε-close to the uniform statistical distribution on $EF^{n_1 \times n_2}$. However, the total number of different couples $(\boldsymbol{U}, \boldsymbol{V})$ is equal to $\dim F^{n_1 n} \dim E^{n_2 n} = rd^n r^{n_2} d^{n_1}$ and the number of matrices in $EF^{n_1 \times n_2}$ is $rd^{n_1 n_2}$. In a usual code-based cryptography setting where $n_1 \approx n_2 \approx n/2$ and $r \approx d$, we get that $rd^n r^{n_2} d^{n_1} \ll rd^{n_1 n_2}$ therefore we cannot expect to use the LHL.

At first sight, this is quite an issue, as proving the statement of our theorem without standard statistical arguments can be quite complex, or impossible. The rest of the section presents a five stage proof of the theorem (main body and 4 lemmas), using algebraic arguments. Our approach is to study the distribution of $\phi(\boldsymbol{UV})$ for a linear form ϕ on EF. We show that the distribution of $\phi(\boldsymbol{UV})$ is uniform in a subspace of $\mathbb{F}^{n_1 \times n_2}$ whose dimension is depending on the rank of ϕ viewed as a tensor in $E \otimes F$ and on a simple condition on matrix $\boldsymbol{U}$.

B.1.1 Preliminary results on binary matrices

Lemma B.1.1. *For a uniformly random binary matrix $\boldsymbol{M}$ of size $m \times n$ with $m \leq n$ and for $0 < i \leq m$, $\mathbb{P}(\text{rank}(\boldsymbol{M}) = m - i) \leq 2^{i(m-n)}$.*

Proof. Let S be a subspace of $\{0,1\}^m$ of dimension $m - i$. The number of such possible subspaces is $\binom{m}{i}_2 \leq 2^{im}$.

For a uniformly random binary $m \times n$ matrix $\boldsymbol{M}$, since the n columns of $\boldsymbol{M}$ are independent random variables, $\mathbb{P}(\mathbf{Supp}(\boldsymbol{M}) \subset S) = 2^{-in}$. Then:

$$\begin{aligned}
\mathbb{P}(\text{rank}(\boldsymbol{M}) = m - i) &\leq \mathbb{P}(\text{rank}(\boldsymbol{M}) \leq m - i) \\
&\leq \mathbb{P}(\bigcup_S \mathbf{Supp}(\boldsymbol{M}) \subset S) \\
&\leq \sum_S \mathbb{P}(\mathbf{Supp}(\boldsymbol{M}) \subset S) \\
&\leq 2^{i(m-n)}
\end{aligned}$$

□

Definition B.1.1. *For $s > 0$, let R_s be the random variable defined as the rank of a uniformly random binary matrix of size $n_1 \times ns$.*

Lemma B.1.2. *For $n_2 > 0$, $\mathbb{E}(2^{-n_2 R_1}) \leq (n_1+1)2^{-n_1 n_2}$.*

Proof.

$$\begin{aligned}
\mathbb{E}(2^{-n_2 R_1}) &= \sum_{i=0}^{n_1} 2^{-n_2 i}\mathbb{P}(R_1 = i) \\
&= 2^{-n_1 n_2}\mathbb{P}(R_1 = n_1) + \sum_{i=0}^{n_1-1} 2^{-n_2 i}\mathbb{P}(R_1 = i) \\
&\leq 2^{-n_1 n_2} + \sum_{i=1}^{n_1} 2^{-n_2(n_1-i)}\mathbb{P}(R_1 = n_1 - i) \\
&\leq 2^{-n_1 n_2} + \sum_{i=1}^{n_1} 2^{-n_2(n_1-i)}2^{i(n_1-n)} \qquad \text{(Lemma B.1.1)} \\
&\leq 2^{-n_1 n_2} + \sum_{i=1}^{n_1} 2^{i(n_2+n_1-n)-n_1 n_2} \\
&\leq 2^{-n_1 n_2} + \sum_{i=1}^{n_1} 2^{-n_1 n_2} \qquad (n \geq n_1 + n_2) \\
&\leq (n_1+1)2^{-n_1 n_2}
\end{aligned}$$

□

Since $R_1 \leq R_s$, we get an immediate corollary.

Corollary B.1.1. *For $n_2 > 0$ and for $s > 0$, $\mathbb{E}(2^{-n_2 R_s}) \leq (n_1+1)2^{-n_1 n_2}$.*

B.1.2 Proof of Theorem 3.1.1

We first fix ϕ a non-zero linear form from EF to $\mathbb{F}_q$ and we will study the probabibilty that $\mathbf{Supp}(\boldsymbol{UV}) \subset \ker(\phi)$. For a vector $\boldsymbol{x} = (x_1, ..., x_i) \in (EF)^i$, we will note $\phi(\boldsymbol{x})$ the vector $(\phi(x_1), ..., \phi(x_i))$. We use the similar abuse of notation for $\phi(\boldsymbol{X})$ when $\boldsymbol{X}$ is a matrix.

Let ϕ_b be the non-zero bilinear form

$$\begin{aligned}
\phi_b : E \times F &\to \mathbb{F} \\
(e, f) &\mapsto \phi(ef).
\end{aligned}$$

Let $s = \text{rank}(\phi_b)$ be the rank of this bilinear form. Then there exists a basis $(e_1, \ldots, e_r)$ of E and a basis $(f_1, \ldots, f_d)$ of F in which the matrix representation of ϕ_b is

$$\begin{pmatrix} \boldsymbol{I}_s & 0 \\ 0 & 0 \end{pmatrix}.$$

In the product basis of EF

$$(e_1, \ldots, e_r) \otimes (f_1, \ldots, f_d) = (e_1 f_1, ..., e_1 f_d, e_2 f_1, ..., e_r f_1, ..., e_r f_d)$$

the expression of ϕ is very simple. For $x = \sum_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} x_{ij} e_i f_j$ we have

$$\phi(x) = \sum_{1 \leq i \leq s} x_{ii}.$$

Let $\boldsymbol{u} = (u_1, \ldots, u_n)$ be a vector of F^n and consider the map

$$\begin{array}{rcl} E^n & \to & \mathbb{F} \\ \boldsymbol{v} = (v_1, \ldots v_n)^T & \mapsto & \phi(\boldsymbol{uv}) = \phi(u_1 v_1 + \cdots + u_n v_n). \end{array}$$

For $i = 1 \ldots n$, write $u_i = \sum_{j=1}^{d} u_{ij} f_j$ the decomposition of u_i along the basis of F $(f_1, \ldots, f_d)$. Similarly write $v_i = \sum_{j=1}^{r} v_{ij} e_j$ the decomposition of v_i along the basis of E $(e_1, \ldots, e_r)$. We clearly have:

$$\phi(\boldsymbol{uv}) = \sum_{\substack{1 \le i \le n \\ 1 \le j \le s}} u_{ij} v_{ij}. \tag{B.1}$$

Now let $\boldsymbol{U}$ be an $n_1 \times n$ matrix of elements in F. Define $\boldsymbol{U}^s$ to be the $n_1 \times sn$ *binary* matrix obtained from $\boldsymbol{U}$ by replacing every one of its rows $\boldsymbol{u}$ by its expansion

$$u_{11}, \ldots, u_{1s}, u_{21}, \ldots, u_{2s}, \ldots u_{n1}, \ldots, u_{ns}$$

as defined in (B.1). It follows that we have:

Lemma B.1.3. *Let $s = \mathrm{rank}(\phi_b)$, $\boldsymbol{U}$ be an $n_1 \times n$ matrix of elements in F and let $\varphi_{\boldsymbol{U}}$ be the map*

$$\begin{array}{rcl} \varphi_{\boldsymbol{U}} : E^n & \to & \mathbb{F}^{n_1} \\ \boldsymbol{v} & \mapsto & \phi(\boldsymbol{Uv}). \end{array}$$

The rank of the map $\varphi_{\boldsymbol{U}}$ is equal to the rank of the $n_1 \times sn$ binary matrix $\boldsymbol{U}^s$.

Corollary B.1.2. *For $\boldsymbol{U}$ a random variable chosen uniformly in $F^{n_1 \times n}$, $\mathrm{rank}(\varphi_{\boldsymbol{U}})$=$R_s$ where s is the rank of ϕ_b.*

Now that we know the probability distribution of the rank of $\varphi_{\boldsymbol{U}}$, we will give a probability on $\mathbf{Supp}(\boldsymbol{UV})$ depending on this rank.

Lemma B.1.4. *Let $\boldsymbol{U}$ such that the above-defined $\varphi_{\boldsymbol{U}}$ is of rank $0 \le i \le n_1$. Then for $\boldsymbol{V}$ a random variable chosen uniformly in $E^{n \times n_2}$, $\mathbb{P}(\mathbf{Supp}(\boldsymbol{UV}) \subset \ker(\phi)) \le q^{-in_2}$*

Proof. Let $H = Im(\varphi_{\boldsymbol{U}})$ Let $\boldsymbol{V} = (\boldsymbol{v}_1, ..., \boldsymbol{v}_{n_2})$ the columns of $\boldsymbol{V}$.
$\varphi_{\boldsymbol{U}}$ is a surjective homomorphism of finite abelian groups E^n and H, so according to Theorem 8.5 in [Sho09], for all i, $\boldsymbol{U}\boldsymbol{v}_i$ is uniformly distributed. Thus because the columns of $\boldsymbol{V}$ are independent, $\phi(\boldsymbol{UV})$ is uniformly distributed in H^{n_2}.
As a result, because $\mathbf{Supp}(\boldsymbol{UV}) \subset \ker(\phi)$ if and only if $\phi(\boldsymbol{UV}) = \mathbf{0}$, $\mathbb{P}(\mathbf{Supp}(\boldsymbol{UV}) \subset \ker(\phi)) \le 1/|H^{n_2}| = q^{-in_2}$. □

Lemma B.1.5. *For a non-null linear form ϕ of EF, $\mathbb{P}(\mathbf{Supp}(\boldsymbol{UV}) \subset \ker(\phi)) \le \mathbb{E}(2^{-n_2 R_1})$*

Proof. Let $s > 0$ be the rank of ϕ_b.

$$\begin{aligned} \mathbb{P}(\mathbf{Supp}(\boldsymbol{UV}) \subset \ker(\phi)) &= \sum_{i=0}^{n_1} \mathbb{P}(\mathbf{Supp}(\boldsymbol{UV}) \subset \ker(\phi) | \mathrm{rank}(\varphi_{\boldsymbol{U}}) = i)\, \mathbb{P}(\mathrm{rank}(\varphi_{\boldsymbol{U}}) = i) \\ &\le \sum_{i=0}^{n_1} 2^{-in_2} \mathbb{P}(\mathrm{rank}(\varphi_{\boldsymbol{U}}) = i) \qquad \text{(Lemma B.1.4)} \\ &\le \mathbb{E}(2^{-n_2\, \mathrm{rank}(\varphi_{\boldsymbol{U}})}) \\ &\le \mathbb{E}(2^{-n_2 R_s}) \qquad \text{(Corollary B.1.2)} \\ &\le \mathbb{E}(2^{-n_2 R_1}) \qquad \text{(Corollary B.1.1)} \end{aligned}$$

□

Proof of Theorem 3.1.1.

$$\begin{aligned}
\mathbb{P}(\mathbf{Supp}(\boldsymbol{UV}) \neq EF) &= \mathbb{P}\Big(\bigcup_{\phi \in EF^* \setminus \{0\}} \mathbf{Supp}(\boldsymbol{UV}) \subset \ker(\phi)\Big) \\
&\leq \sum_{\phi \in EF^* \setminus \{0\}} \mathbb{P}(\mathbf{Supp}(\boldsymbol{UV}) \subset \ker(\phi)) \\
&\leq \sum_{\phi \in EF^* \setminus \{0\}} \mathbb{E}(2^{-n_2 R_1}) \qquad \text{(Lemma B.1.5)} \\
&\leq 2^{rd}\, \mathbb{E}(2^{-n_2 R_1}) \\
&\leq (n_1+1) 2^{rd - n_1 n_2} \qquad \text{(Lemma B.1.2)}
\end{aligned}$$

□

Appendix C

Proofs for PERK

C.1 Security Proofs for PoK

C.1.1 Proof of Theorem 4.2.4

We restate the Theorem 4.2.4 below and follow it by its proof.

Theorem 4.2.4 (Knowledge Soundness). *The protocol presented in Figure 4.5 is knowledge sound with knowledge error*

$$\varepsilon_{\mathsf{KS}} = \frac{1}{N} + \frac{N-1}{N \cdot (q^t - 1)}\,.$$

Proof of Theorem 4.2.4. Before proving the knowledge soundness of our protocol, we will first prove the following useful lemma.

Lemma C.1.1 ($(2,2)$-special soundness). *The protocol shown in Figure 4.5 is* $(2,2)$-special sound.

Proof of Lemma C.1.1. $(2,2)$**-special soundness.**
Following Definition 4.2.9 the protocol is called $(2,2)$-special sound if there exists an efficient knowledge extractor Ext which on an input statement $(\boldsymbol{H}, (\boldsymbol{x}_i)_{i\in[1,t]}, (\boldsymbol{y}_i)_{i\in[1,t]})$ and a $(2,2)$-accepting tree of transcripts (See Definition 4.2.8) returns a solution of the r-IPKP instance defined by $(\boldsymbol{H}, (\boldsymbol{x}_i)_{i\in[1,t]}, (\boldsymbol{y}_i)_{i\in[1,t]})$. We now show such an extractor which takes 4 accepting transcripts associated with challenges $(\kappa, \alpha_1), (\kappa, \alpha_2), (\kappa', \alpha'_1), (\kappa', \alpha'_2)$ such that $\kappa = (\kappa_i)_{i\in[1,t]}$, $\kappa' = (\kappa'_i)_{i\in[1,t]}$ as well as $\kappa \neq \kappa'$, $\alpha_1 \neq \alpha_2$, and $\alpha'_1 \neq \alpha'_2$, and outputs a solution to the r-IPKP instance defined by $(\boldsymbol{H}, (\boldsymbol{x}_i)_{i\in[1,t]}, (\boldsymbol{y}_i)_{i\in[1,t]})$.

Let $z_2^{(\kappa^*,\alpha^*)}$ denote the response z_2 computed as shown in Figure 4.5 when the first and second challenges are κ^* and α^* respectively. Note that, $z_2^{(\kappa,\alpha_1)}$ contains all the seeds θ_i for $i \in [1,N]$ except $i = \alpha_1$. Therefore, the extractor has access to all the seeds θ_i for $i \in [1,N]$ since it knows both $z_2^{(\kappa,\alpha_1)}$ as well as $z_2^{(\kappa,\alpha_2)}$ and $\alpha_1 \neq \alpha_2$. It can compute $(\bar{\pi}_i^{(\kappa)}, \bar{\boldsymbol{v}}_i^{(\kappa)})_{i\in[1,N]}$ and $(\bar{\pi}_i^{(\kappa')}, \bar{\boldsymbol{v}}_i^{(\kappa')})_{i\in[1,N]}$ from $\left(z_2^{(\kappa,\alpha_i)}\right)_{i\in[1,2]}$ and $\left(z_2^{(\kappa',\alpha'_i)}\right)_{i\in[1,2]}$ respectively.

Also, note that the first message $h_1 = \mathsf{H}(\mathsf{cmt}_1, (\mathsf{cmt}_{1,i})_{i\in[1,N]})$ is common to all the 4 transcripts. Since we assume that H is a collision-resistant hash function, it means that the initial commitments $(\mathsf{cmt}_1, (\mathsf{cmt}_{1,i})_{i\in[1,N]})$ are all same in the 4 transcripts. From the binding property of the commitments $(\mathsf{cmt}_{1,i})_{i\in[1,N]}$, we know that

$$(\bar{\pi}_i, \bar{\boldsymbol{v}}_i)_{i\in[1,N]} = (\bar{\pi}_i^{(\kappa)}, \bar{\boldsymbol{v}}_i^{(\kappa)})_{i\in[1,N]} = (\bar{\pi}_i^{(\kappa')}, \bar{\boldsymbol{v}}_i^{(\kappa')})_{i\in[1,N]}.$$

The knowledge extractor Ext computes the solution as

1. Compute $(\bar{\pi}_i)_{i\in[1,n]}$ from $z_2^{(\kappa,\alpha_1)}$ and $z_2^{(\kappa,\alpha_2)}$
2. Output $\bar{\pi} = \bar{\pi}_N \circ \cdots \circ \bar{\pi}_1$

Let us now check the validity of this solution output by the extractor. By construction, we know that $\bar{\boldsymbol{s}}_0^{(\kappa,\alpha_1)} = \bar{\boldsymbol{s}}_0^{(\kappa,\alpha_2)} = \sum_{i\in[1,t]} \kappa_i \cdot \boldsymbol{x}_i$. Also, for all $i \in [1,N] \setminus \alpha_1$, $\bar{\boldsymbol{s}}_i^{(\kappa,\alpha_1)} = \bar{\pi}_i\left[\bar{\boldsymbol{s}}_{i-1}^{(\kappa,\alpha_1)}\right] + \bar{\boldsymbol{v}}_i$. And for all $i \in [1,N] \setminus \alpha_2$, $\bar{\boldsymbol{s}}_i^{(\kappa,\alpha_2)} = \bar{\pi}_i\left[\bar{\boldsymbol{s}}_{i-1}^{(\kappa,\alpha_2)}\right] + \bar{\boldsymbol{v}}_i$. Since the transcripts are accepting and V checks h_2 computed as $h_2 = \mathsf{H}((\boldsymbol{s}_i)_{i\in[1,N]})$, due to the collision-resistance property of H, it follows that for all $i \in [1,N]$, $\bar{\boldsymbol{s}}_i^{(\kappa)} = \bar{\pi}_i\left[\bar{\boldsymbol{s}}_{i-1}^{(\kappa)}\right] + \bar{\boldsymbol{v}}_i$, this implies $\bar{\boldsymbol{s}}_N^{(\kappa)} = \bar{\pi}\left[\sum_{i\in[1,t]} \kappa_i \cdot \boldsymbol{x}_i\right] + \bar{\boldsymbol{v}}$.

Following a similar argument, we know that $\bar{\boldsymbol{s}}_N^{(\kappa')} = \bar{\pi}\left[\sum_{i\in[1,t]} \kappa'_i \cdot \boldsymbol{x}_i\right] + \bar{\boldsymbol{v}}$. Based on the binding property of commitment cmt_1 and using the fact that the transcripts are accepting, we can write

$$\boldsymbol{H}\bar{\boldsymbol{s}}_N^{(\kappa)} - \sum_{i\in[1,t]} \kappa_i \cdot \boldsymbol{y}_i = \boldsymbol{H}\bar{\boldsymbol{s}}_N^{(\kappa')} - \sum_{i\in[1,t]} \kappa'_i \cdot \boldsymbol{y}_i$$

$$\implies \boldsymbol{H}\left(\bar{\pi}\left[\sum_{i\in[1,t]} \kappa_i \cdot \boldsymbol{x}_i\right] + \bar{\boldsymbol{v}}\right) - \sum_{i\in[1,t]} \kappa_i \cdot \boldsymbol{y}_i = \boldsymbol{H}\left(\bar{\pi}\left[\sum_{i\in[1,t]} \kappa'_i \cdot \boldsymbol{x}_i\right] + \bar{\boldsymbol{v}}\right) - \sum_{i\in[1,t]} \kappa'_i \cdot \boldsymbol{y}_i$$

$$\implies \boldsymbol{H}\left(\bar{\pi}\left[\sum_{i\in[1,t]} (\kappa_i - \kappa'_i) \cdot \boldsymbol{x}_i\right]\right) = \sum_{i\in[1,t]} (\kappa_i - \kappa'_i) \cdot \boldsymbol{y}_i$$

This implies that $\bar{\pi}$ is a solution of the considered r-IPKP problem. □

We can now apply the result of Theorem 4.2.1 to Lemma C.1.1, which concludes the proof. □

C.1.2 Proof of Theorem 4.2.5

The following proof is inspired from the proof of [ZCD+20, Lemma 6.1] and [FJR22, Theorem 3]. We now show that the protocol described in Figure 4.5, Section 4.2.2, satisfies the special honest-verifier zero knowledge property. We assume that the commitment algorithm $\mathsf{Com}(\cdot)$ outputs $\ell(\lambda)$-bit strings as output for some polynomial ℓ. We restate the Theorem 4.2.5 below and follow it by its proof.

Theorem 4.2.5 (Special Honest-Verifier Zero Knowledge). *Assume that there exists a* $(\mathsf{t}, \epsilon_{\mathsf{PRG}})$*-secure* PRG*, and the commitment scheme* Com *is* $(\mathsf{t}, \epsilon_{\mathsf{Com}})$*-hiding. Then there exists an efficient simulator* Sim *which, outputs a transcript such that no distinguisher running in time at most* $\mathsf{t}(\lambda)$ *can distinguish between the transcript produced by* Sim *and a real transcript obtained by honest execution of the protocol in Figure 4.5 with probability better than* $(\epsilon_{\mathsf{PRG}}(\lambda) + \epsilon_{\mathsf{Com}}(\lambda))$.

Proof of Theorem 4.2.5. We begin by describing an efficient simulator Sim which outputs a transcript which is indistinguishable from a real transcript obtained by honest execution of the protocol. Sim on input $x = (\boldsymbol{H}, (\boldsymbol{x}_i)_{i\in[1,t]}, (\boldsymbol{y}_i)_{i\in[1,t]})$ works as follows:

Note that the simulator Sim runs in polynomial-time and the challenges sampled in Step 1 are distributed identically to the real world execution since the verifier also samples the challenges uniformly at random. We now show that the transcript output by Sim and a real transcript obtained by honest execution of the protocol in Figure 4.5 with challenges (κ, α^*) cannot be distinguished with probability better than $(\epsilon_{\mathsf{PRG}}(\lambda) + \epsilon_{\mathsf{Com}}(\lambda))$ by any distinguisher running in time at most $\mathsf{t}(\lambda)$. We consider the following sequence of simulators:

Simulator 0 (real world). This simulator takes the statement $x = (\boldsymbol{H}, (\boldsymbol{x}_i)_{i\in[1,t]}, (\boldsymbol{y}_i)_{i\in[1,t]})$, witness π, and the challenges (κ, α^*) as input. It then runs the protocol in Figure 4.5 honestly and outputs the transcript. This transcript is identically distributed as a real-world transcript.

1. Sample $\kappa \xleftarrow{\$} \mathbb{F}_q^t \setminus \mathbf{0}$ and $\alpha^* \xleftarrow{\$} [N]$
2. Sample $\theta \xleftarrow{\$} \{0,1\}^\lambda$
3. For each $i \in [1, N] \setminus \{\alpha^*\}$
   - $\diamond$ $\theta_i \xleftarrow{\$,\theta} \{0,1\}^\lambda$, $\phi_i \xleftarrow{\$,\theta_i} \{0,1\}^\lambda$, $r_{1,i} \xleftarrow{\$,\theta_i} \{0,1\}^\lambda$
   - $\diamond$ **if** $i \neq 1$
     - $\triangleright$ $\pi_i \xleftarrow{\$,\phi_i} \mathcal{S}_n$, $\boldsymbol{v}_i \xleftarrow{\$,\phi_i} \mathbb{F}_q^n$, $\mathsf{cmt}_{1,i} = \mathsf{Com}\big(r_{1,i},\, \phi_i\big)$
   - $\diamond$ **if** $i = 1$
     - $\triangleright$ $\pi_1 \xleftarrow{\$} S_n$, $\boldsymbol{v}_1 \xleftarrow{\$,\phi_1} \mathbb{F}_q^n$, $\mathsf{cmt}_{1,1} = \mathsf{Com}\big(r_{1,1},\, \pi_1 \,||\, \phi_1\big)$
4. For $i = \alpha^*$
   - $\diamond$ Sample $\pi_{\alpha^*} \xleftarrow{\$} \mathcal{S}_n$, $\boldsymbol{v}_{\alpha^*} \xleftarrow{\$} \mathbb{F}_q^n$, $\mathsf{cmt}_{1,\alpha^*} \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$
5. Compute r_1, $\boldsymbol{v}$, cmt_1, h_1 as in the real protocol using the values computed above.
6. Compute $\tilde{\pi} = \pi_N \circ \cdots \circ \pi_1$
7. Compute $\tilde{\boldsymbol{x}}$ such that $\boldsymbol{H}\tilde{\boldsymbol{x}} = \sum_{i\in[1,t]} \kappa_i \cdot \boldsymbol{y}_i$
8. Compute $\boldsymbol{s}_0 = \sum_{i\in[1,t]} \kappa_i \cdot \boldsymbol{x}_i$
9. For each $i \in [1, \alpha^* - 1]$
   - $\diamond$ Compute $\boldsymbol{s}_i = \pi_i[\boldsymbol{s}_{i-1}] + \boldsymbol{v}_i$
10. Compute $\boldsymbol{s}_{\alpha^*} = \pi_{\alpha^*}[\boldsymbol{s}_{\alpha^*-1}] + \boldsymbol{v}_{\alpha^*} + \pi_{\alpha^*+1}^{-1} \circ \cdots \circ \pi_N^{-1}\left[\tilde{\boldsymbol{x}} - \tilde{\pi}\left[\sum_{i\in[1,t]} \kappa_i \cdot \boldsymbol{x}_i\right]\right]$
11. For each $i \in [\alpha^* + 1, N]$
    - $\diamond$ Compute $\boldsymbol{s}_i = \pi_i[\boldsymbol{s}_{i-1}] + \boldsymbol{v}_i$
12. Compute h_2, z_1, z_2, rsp as in the real protocol using the values computed above.
13. Output $(x, h_1, (\kappa_i)_{i\in[1,t]}, h_2, \alpha^*, \mathsf{rsp})$

Figure C.1: Simulator Sim for generating indistinguishable transcripts without knowledge of secret witness π

Simulator 1. Simulator 1 works exactly same as Simulator 0 except that instead of computing $\mathsf{cmt}_{1,\alpha^*}$ as in the real protocol, it samples a uniform string as $\mathsf{cmt}_{1,\alpha^*} \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$. The probability of distinguishing Simulator 0 from Simulator 1 by any distinguisher running in time at most $\mathsf{t}(\lambda)$ is upper bounded by $\epsilon_{\mathsf{Com}}(\lambda)$.

Simulator 2. The only difference between Simulator 1 and Simulator 2 is that, Simulator 2 samples $\pi_{\alpha^*} \xleftarrow{\$} \mathcal{S}_n$ and $\boldsymbol{v}_{\alpha^*} \xleftarrow{\$} \mathbb{F}_q^n$ uniformly at random instead of using the seed-derived randomness from the seed θ. The probability of distinguishing Simulator 2 from Simulator 1 by any distinguisher running in time at most $\mathsf{t}(\lambda)$ is upper bounded by $\epsilon_{\mathsf{PRG}}(\lambda)$.

Simulator 3 (Sim). Simulator 3 takes the statement $x = (\boldsymbol{H}, (\boldsymbol{x}_i)_{i\in[1,t]}, (\boldsymbol{y}_i)_{i\in[1,t]})$, and works as Sim defined in Figure C.1. Note that, this simulator does not depend on the witness π. Also, Sim first samples the challenges (κ, α^*) uniform randomly (this is identical to honest verifier in real world).

If $\alpha^* = 1$, then Simulator 2 and Sim work exactly same till Step 5 of Sim. Therefore, h_1 is distributed identically in both the transcripts. Also, $\boldsymbol{s}_0$ is computed honestly by Sim and hence matches with that computed by Simulator 2. While computing $\boldsymbol{s}_1$, both Simulator 2 and Sim add $\boldsymbol{v}_1$ to it. However, $\boldsymbol{v}_1$ is sampled uniformly at random by both Simulator 2 and Sim. Hence, $\boldsymbol{s}_1$ is also distributed identically in both the transcripts. Step 11 of Sim works exactly as Simulator 2 which means h_2 is also distributed identically in both the transcripts. The response rsp in this case

is $\mathsf{rsp} = \big(\boldsymbol{s}_1, (r_1 \,||\theta_i)_{i\in[2,N]}, \mathsf{cmt}_{1,1}\big)$. As explained above $\boldsymbol{s}_1$ is uniform random and distributed identically in both transcripts. The seeds $(\theta_i)_{i\in[2,N]}$ and randomness r_1 are computed identically by both the simulators since they work exactly the same way till Step 5 of Sim. Also, $\mathsf{cmt}_{1,1}$ is sampled uniformly at random in both experiments (refer Simulator 1). Therefore, the transcript $(x, h_1, (\kappa_i)_{i\in[1,t]}, h_2, \alpha^*, \mathsf{rsp})$ is distributed identically in Simulator 2 and Sim when $\alpha^* = 1$.

If $\alpha^* \neq 1$, then Simulator 2 and Sim work exactly same till Step 5 of Sim, except for sampling of π_1. Simulator 2 computes π_1 from witness π, whereas Sim samples π_1 uniformly at random. However, the values $r_1, \boldsymbol{v}, \mathsf{cmt}_1$, and h_1 are computed independently of π_1 and those are distributed identically in both the transcripts. Also note Simulator 2 computes π_1 by composing it with $\pi_{\alpha^*}^{-1}$. Since π_{α^*} is sampled uniformly at random by Simulator 2, this implies that π_1 computed by Simulator 2 is also uniform random permutation and hence π_1 is also distributed identically. Also, for $i \in [0, \alpha^* - 1]$, the values $\boldsymbol{s}_i$ are computed honestly by Sim and since π_1 is distributed identically the values $\boldsymbol{s}_i$ for $i \in [0, \alpha^* - 1]$ are also distributed identically. As in the previous case, while computing $\boldsymbol{s}_{\alpha^*}$, both Simulator 2 and Sim add $\boldsymbol{v}_{\alpha^*}$ to it. However, $\boldsymbol{v}_{\alpha^*}$ is sampled uniformly at random by both Simulator 2 and Sim. Hence, $\boldsymbol{s}_{\alpha^*}$ is also distributed identically in both the transcripts. Step 11 of Sim works exactly as Simulator 2 which means h_2 is also distributed identically in both the transcripts. The response rsp in this case is $\mathsf{rsp} = \big(\boldsymbol{s}_{\alpha^*}, (\pi_1 \,||r_1\,||\theta_i)_{i\in[1,N]\setminus\alpha^*}, \mathsf{cmt}_{1,\alpha^*}\big)$. As explained above $\boldsymbol{s}_{\alpha^*}$ is uniform random and distributed identically in both transcripts. The seeds $\theta_i)_{i\in[1,N]\setminus\alpha^*}$ and randomness r_1 are computed identically by both the simulators since they work exactly the same way till Step 5 of Sim. Also, $\mathsf{cmt}_{1,\alpha^*}$ is sampled uniformly at random in both experiments (refer Simulator 1). Therefore, the transcript $(x, h_1, (\kappa_i)_{i\in[1,t]}, h_2, \alpha^*, \mathsf{rsp})$ is distributed identically in Simulator 2 and Sim when $\alpha^* \neq 1$.

Therefore, any distinguisher running in time at most $\mathsf{t}(\lambda)$ cannot distinguish between the real-world transcript and the transcript produced by Sim with probability better than $(\epsilon_{\mathsf{PRG}}(\lambda) + \epsilon_{\mathsf{Com}}(\lambda))$. □

C.2 Security Proof for PERK signature scheme

C.2.1 Proof of Theorem 4.2.6

We restate the Theorem 4.2.6 below and follow it by its proof.

Theorem 4.2.6. *Suppose* PRG *is* $(\mathsf{t}, \epsilon_{\mathsf{PRG}})$*-secure and any adversary running in time* $\mathsf{t}(\lambda)$ *can solve the the underlying r-IPKP instance with probability at most* $\epsilon_{\text{r-IPKP}}$*. Model* H_0*,* H_1*, and* H_2 *as random oracles where* H_0*,* H_1*, and* H_2 *have* 2λ*-bit output length. Then a chosen-message attacker against the signature scheme (PERK) presented in Figure 4.7, running in time* $\mathsf{t}(\lambda)$*, making* q_s *signing queries, and making* q_0*,* q_1*,* q_2 *queries, respectively, to the random oracles, succeeds in outputting a valid forgery with probability*

$$\mathbb{P}[\mathsf{Forge}] \leq \frac{(q_0 + \tau \cdot (N+1) \cdot q_s)^2}{2 \cdot 2^{2\lambda}} + \frac{q_s \cdot (q_0 + q_1 + q_2 + q_s)}{2^{2\lambda}} + \tau \cdot q_s \cdot \epsilon_{\mathsf{PRG}}(\lambda) + \epsilon_{\text{r-IPKP}} + q_2 \cdot \varepsilon_{\mathsf{KS}}^{\tau}, \tag{4.5}$$

where $\varepsilon_{\mathsf{KS}} = \frac{1}{N} + \frac{N-1}{N\cdot(q^t-1)}$.

The following proof is greatly inspired from the proof of the Picnic signature scheme [ZCD+20, Theorem 6.2] and [FJR22, Theorem 5].

Proof of Theorem 4.2.6. Let $\mathcal{A}$ be a EUF-CMA attacker against the signature scheme, which makes q_s queries to the signing oracle. Also, let q_0, q_1, and q_2 respectively denote the number of queries made by $\mathcal{A}$ to the random oracles H_0, H_1, and H_2. To prove security we define a sequence of experiments involving $\mathcal{A}$, starting with an experiment in which $\mathcal{A}$ interacts with the real signature scheme. We let $\Pr_i[\cdot]$ refer to the probability of an event in experiment i. We let $\mathsf{t}(\lambda)$ denote the running time of the entire experiment, *i.e.*, including both $\mathcal{A}$'s running time and the time required to answer signing queries and to verify $\mathcal{A}$'s output.

Experiment 1. This corresponds to the interaction of $\mathcal{A}$ with the real signature scheme. In more detail: first KeyGen is run to obtain, the secret key $\mathsf{sk} = \pi$ along with the public key $\mathsf{pk} = (\boldsymbol{H}, (\boldsymbol{x}_j, \boldsymbol{y}_j)_{j\in[1,t]})$, and $\mathcal{A}$ is given pk. In addition, we assume that the random oracles H_0, H_1, and H_2 are chosen uniformly from the appropriate spaces. $\mathcal{A}$ may make signing queries, which will be answered as in the signature algorithm; $\mathcal{A}$ may also query any of the random oracles. Finally, $\mathcal{A}$ outputs a message-signature pair; we let Forge denote the event that the message was not previously queried by $\mathcal{A}$ to its signing oracle, and the signature is valid. Our goal is to upper-bound $\Pr_1[\mathsf{Forge}]$.

Experiment 2. We abort the experiment if, during the course of the experiment, a collision occurs in H_0. The number of queries to any oracle throughout the experiment (by either the adversary or the signing algorithm) is at most $(q_0 + \tau \cdot (N+1) \cdot q_s)$. Therefore,

$$|\Pr_1[\mathsf{Forge}] - \Pr_2[\mathsf{Forge}]| \leq \frac{(q_0 + \tau \cdot (N+1) \cdot q_s)^2}{2 \cdot 2^{2\lambda}}.$$

Experiment 3. We abort the experiment if during the course of the experiment, while answering to a signature query, the sampled salt collides with the value salt in any previous query to H_0, H_1, or H_2. For each single signature query, the probability to abort is upper bounded by $(q_0 + q_1 + q_2 + q_s)\,/2^{2\lambda}$. Thus,

$$|\Pr_2[\mathsf{Forge}] - \Pr_3[\mathsf{Forge}]| \leq \frac{q_s \cdot (q_0 + q_1 + q_2 + q_s)}{2^{2\lambda}}.$$

Experiment 4. The difference with the previous experiment is that, when signing a message m we begin by choosing h_1 and h_2 uniformly at random and then we expand them as $(\kappa_j^{(e)})_{e\in[1,\tau],j\in[1,t]}$ and $(\alpha^{(e)})_{e\in[1,\tau]}$. Steps 1, 3, and 5 are computed as before, but in Steps 2 and 4 we simply set the output of H_1 to h_1 and the output of H_2 to h_2.

The outcome of this experiment compared to the previous one only changes if, in the course of answering a signing query, the query to H_1 or the query to H_2 was ever made before (by either the adversary or as a part of answering some other signing query). But this cannot happen since in such a case Experiment 3 would abort. Thus,

$$\Pr_3[\mathsf{Forge}] = \Pr_4[\mathsf{Forge}].$$

Experiment 5. The difference with the previous experiment is that, for each $e \in [1, \tau]$, we sample $\mathsf{cmt}^{(e)}_{1,\alpha^{(e)}}$ uniformly at random instead of making a query to H_0.

The only difference between this experiment and the previous experiment occurs if, during the course of answering a signing query, the seed $\theta^{(e)}_{\alpha^{(e)}}$ (for some $e \in [1, \tau]$) was previously queried to H_0. However, such collisions cannot occur within the same signing query (since indices e and i are part of the input of H_0) and if it occurs from a previous query (signing query or query to H_0) then the experiment aborts (according to the difference introduced in Experiment 3). Thus,

$$\Pr_4[\mathsf{Forge}] = \Pr_5[\mathsf{Forge}].$$

Experiment 6. We again modify the experiment. Now, for $e \in [1, \tau]$ the signer uses the SHVZK simulator Sim (see proof of Theorem 4.2.5) to generate the views of the parties during the execution of Step 1 and Step 3. We denote by $\mathsf{Sim}_{\mathsf{salt}}(\cdot)$ a call to this simulator which appends salt to the sampled seed θ as input to PRG. This simulation results in $\left\{\left(\theta_i^{(e)}, \pi_i^{(e)}\right)\right\}_{i\neq\alpha^{(e)}}$ and $(\boldsymbol{s}_j^{(e)})_{j\in[1,N]}$. Thus the signing queries are now answered as shown in C.2.

Note that the secret π is no longer used for generating signatures. Recall that an adversary against Sim has distinguishing advantage $\epsilon_{\mathsf{PRG}}(\lambda)$ (corresponding to execution time $\mathsf{t}(\lambda)$), since

Step 0

1. Sample $h_1 \xleftarrow{\$} \{0,1\}^{2\lambda}$ uniformly at random.
2. Sample $(\kappa_j^{(e)})_{e\in[1,\tau],j\in[1,t]} \longleftarrow \mathsf{PRG}(h_1)$.
3. Sample $h_2 \xleftarrow{\$} \{0,1\}^{2\lambda}$ uniformly at random.
4. Sample $(\alpha^{(e)})_{e\in[1,\tau]} \longleftarrow \mathsf{PRG}(h_2)$.
5. Sample a uniform random salt $\mathsf{salt} \xleftarrow{\$} \{0,1\}^{2\lambda}$.

Steps 1 and 3 For each iteration $e \in [1,\tau]$

1. $\left\{\left(\theta_i^{(e)}, \pi_i^{(e)}\right)\right\}_{i\neq\alpha^{(e)}}, (\boldsymbol{s}_j^{(e)})_{j\in[1,N]} \leftarrow \mathsf{Sim}_{\mathsf{salt}}((\kappa_j^{(e)})_{e\in[1,\tau],j\in[1,t]}, (\alpha^{(e)})_{e\in[1,\tau]})$
2. Choose commitment $\mathsf{cmt}_{1,\alpha^{(e)}}^{(e)} \xleftarrow{\$} \{0,1\}^{2\lambda}$ uniform randomly.
3. For $i \neq \alpha^{(e)}$:
   - ◇ If $i \neq 1$, set $\mathsf{cmt}_{1,i}^{(e)} = \mathsf{H}_0(\mathsf{salt}, e, i, \theta_i^{(e)})$.
   - ◇ If $i = 1$, set $\mathsf{cmt}_{1,1}^{(e)} = \mathsf{H}_0(\mathsf{salt}, e, 1, \pi_1, \theta_1^{(e)})$.
4. Compute $\boldsymbol{Hv}^{(e)}$ as $\boldsymbol{Hv}^{(e)} = (\boldsymbol{H}(\boldsymbol{s}_N^{(e)}) - \sum_{i\in[1,t]} \kappa_i^{(e)} \cdot \boldsymbol{y}_i)$.
5. Set $\mathsf{cmt}_1^{(e)} = \mathsf{H}_0(\mathsf{salt}, e, \boldsymbol{Hv}^{(e)})$.

Steps 2 and 4

1. Set $\mathsf{H}_1(\mathsf{salt}, m, \mathsf{pk}, (\mathsf{cmt}_1^{(e)}, \mathsf{cmt}_{1,i}^{(e)})_{e\in[1,\tau],i\in[1,N]})$ equal to h_1.
2. Set $\mathsf{H}_2(\mathsf{salt}, m, \mathsf{pk}, h_1, (\boldsymbol{s}_i^{(e)})_{e\in[1,\tau],i\in[1,N]})$ equal to h_2.

Step 5 Output signature σ built as

1. For each iteration $e \in [1,\tau]$:
   - ◇ Compute $\boldsymbol{z}_1^{(e)} = \boldsymbol{s}_\alpha^{(e)}$
   - ◇ If $\alpha^{(e)} \neq 1$, $z_2^{(e)} = \pi_1^{(e)} \,||\, (\theta_i^{(e)})_{i\in[1,N]\setminus\alpha^{(e)}}$
   - ◇ If $\alpha^{(e)} = 1$, $z_2^{(e)} = (\theta_i^{(e)})_{i\in[1,N]\setminus\alpha^{(e)}}$
   - ◇ Compute $\mathsf{rsp}^{(e)} = (\boldsymbol{z}_1^{(e)}, z_2^{(e)}, \mathsf{cmt}_{1,\alpha^{(e)}}^{(e)})$
2. Compute $\sigma = (\mathsf{salt}, h_1, h_2, (\mathsf{rsp}^{(e)})_{e\in[1,\tau]})$

Figure C.2: Experiment 6: Answer to a signature query for a message m.

the commitments are built outside of the simulator. Therefore,

$$|\Pr_5[\mathsf{Forge}] - \Pr_6[\mathsf{Forge}]| \leq \tau \cdot q_s \cdot \epsilon_{\mathsf{PRG}}(\lambda).$$

Experiment 7. At any point during the experiment, we say that the execution e^* of a query

$$h_2 = \mathsf{H}_2(\mathsf{salt}, m, \mathsf{pk}, h_1, (\boldsymbol{s}_i^{(e)})_{e\in[1,\tau],i\in[1,N]})$$

defines a *correct witness* if the following four conditions are fulfilled:

1. h_1 was output by a previous query

$$h_1 = \mathsf{H}_1(\mathsf{salt}, m, \mathsf{pk}, (\mathsf{cmt}_1^{(e)}, \mathsf{cmt}_{1,i}^{(e)})_{e\in[1,\tau],i\in[1,N]}),$$

2. each $\mathsf{cmt}_{1,i}^{(e^*)}$ in this H_1-query was output by a previous query

$$\mathsf{cmt}_{1,i}^{(e^*)} = \mathsf{H}_0(\mathsf{salt}, e^*, i, \theta_i^{(e^*)})$$

for $i \in [2,N]$, and

$$\mathsf{cmt}_{1,1}^{(e^*)} = \mathsf{H}_0(\mathsf{salt}, e^*, 1, \pi_1^{(e^*)}, \theta_1^{(e^*)})$$

for $i = 1$.

3. each $\mathsf{cmt}_1^{(e^*)}$ in the above H_1-query was output by a previous query

$$\mathsf{cmt}_1^{(e^*)} = \mathsf{H}_0\left(\mathsf{salt}, e^*, \left(\boldsymbol{H}\boldsymbol{s}_N^{(e^*)} - \sum_{i\in[1,t]} \kappa_i^{(e^*)} \cdot \boldsymbol{y}_i\right)\right)$$

4. the permutation π derived from $\left\{\pi_i^{(e^*)}\right\}_{i\in[1,N]}$ *i.e.* $\pi = \pi_N^{(e^*)} \circ \pi_{N-1}^{(e^*)} \circ \cdots \pi_1^{(e^*)}$ satisfies

$$\boldsymbol{H}\left(\pi\left[\sum_{i\in[1,t]} \kappa_i \cdot \boldsymbol{x}_i\right]\right) = \sum_{i\in[1,t]} \kappa_i \cdot \boldsymbol{y}_i$$

for some $\kappa \in {\mathbb{F}_q}^t \setminus \mathbf{0}$, where $\kappa := \{\kappa_1, \ldots, \kappa_t\}$ and $\mathbf{0} \in {\mathbb{F}_q}^t$ is the all zero vector.

(Note that in all cases the commitments in the relevant prior H_1-query, if it exists, must be unique since the experiment is abortedif there is ever a collision in H_0.)

In Experiment 7, for each query of the above form made by the adversary to H_2 (where m was not previously queried to the signing oracle), check if there exists an execution e^* which defines a correct witness. We let Solve be the event that this occurs for some query to H_2. Note that, if that event occurs, the $\left\{\pi_i^{(e^*)}\right\}_{i\in[1,N]}$ (which can be determined from the oracle queries of $\mathcal{A}$) allow the computation of solution to r-IPKP. Therefore, $\Pr_7[\mathsf{Solve}] \leq \epsilon_{\text{r-IPKP}}$. We claim that

$$\Pr_7\left[\mathsf{Forge} \bigwedge \overline{\mathsf{Solve}}\right] \leq q_2 \cdot \varepsilon_{\mathsf{KS}}^{\tau},$$

where $\varepsilon_{\mathsf{KS}} = \frac{1}{N} + \frac{N-1}{N\cdot(q^t-1)}$ is the knowledge soundness error of one execution. To see this, assume Solve does not occur. Then there is no execution of any H_2-query which defines a correct witness. When considering an arbitrary execution $e \in [1,\tau]$, the attacker can only possibly generate a forgery (using this H_2-query) if

1. $\mathcal{A}$ guesses the first challenge $\kappa^{(e^*)} \in {\mathbb{F}_q}^t \setminus \mathbf{0}$, where $\kappa^{(e^*)} := \left\{\kappa_1^{(e^*)}, \ldots, \kappa_t^{(e^*)}\right\}$ and $\mathbf{0} \in {\mathbb{F}_q}^t$ is the all zero vector.
2. or even if $\mathsf{cmt}_1^{(e^*)} \neq \mathsf{H}_0(\mathsf{salt}, e^*, (\boldsymbol{H}\boldsymbol{s}_N^{(e^*)} - \sum_{i\in[1,t]} \kappa_i^{(e^*)} \cdot \boldsymbol{y}_i))$ the attacker guesses the second challenge α^* such that the views of all remaining $N-1$ parties are consistent.

Thus, the overall probability with which the attacker can generate a forgery using this H_2-query is

$$\varepsilon_{\mathsf{KS}}^{\tau} = \left(\frac{1}{q^t-1} + \left(1 - \frac{1}{q^t-1}\right) \cdot \frac{1}{N}\right)^{\tau}.$$

The final bound is obtained by taking a union bound over all queries to H_2. This concludes the proof of Theorem 4.2.6.

□

C.3 Generic Attacks against Fiat-Shamir Signatures

Kales and Zaverucha in [KZ20], showed a generic attack on the non-interactive version of 5-round PoK schemes. The attack strategy is to guess either one of the challenges (ch_1 or ch_2) correctly which permits the prover to cheat. The attacker then aims to split the work by attempting to guess the first challenge for η^* instances out of τ parallel repetitions, and tries to guess ch_2 for the remaining $(\tau - \eta^*)$ instances.

If the attacker can guess η^* challenges for the first phase correctly, then he can answer all the N possible challenges for the $\alpha^{(e)}$ for those instances. Subsequently, to successfully cheat he has to guess the remaining $(\tau - \eta^*)$ values of $\alpha^{(e)}$ correctly.

The parameter η^* allows the attacker to balance the cost for both guessing phases. The overall cost is minimized for a choice of

$$\eta^* = \underset{0 \leq \eta \leq \tau}{\arg\min} \left\{ \frac{1}{P_1(\eta, \tau, q, t)} + N^{(\tau - \eta)} \right\} \tag{C.1}$$

where,

$$P_1(\eta, \tau, q, t) := \sum_{j=\eta}^{\tau} \left(\frac{1}{q^t - 1} \right)^j \left(\frac{q^t - 2}{q^t - 1} \right)^{\tau - j} \binom{\tau}{j}.$$

Finally, the total cost for the attacker is thus

$$\mathcal{W}_{KZ} = \frac{1}{P_1(\eta^*, \tau, q, t)} + N^{(\tau - \eta^*)}. \tag{C.2}$$

In [KZ20] the authors classify the 5-round protocols based on whether it is possible for the verifier to detect if the tuple $(\mathsf{cmt}, \mathsf{ch}_1, \mathsf{rsp}_1)$ is valid or not. If the verifier can detect the validity of this tuple then the scheme is said to possess *early abort* property. The cost of the attack varies for different schemes based on whether they satisfy the early abort property or not. Our protocol and signature scheme do not possess the early abort property and hence the expected cost of attacking the PERK signature scheme proposed in this work, is given by Equation (C.2).