Analysis and design of post-quantum cryptographic algorithms : PKP-based signature scheme and ultra-short multivariate signatures

Eliane Koussa

université PARIS-SACLAY

Analysis and design of some post-quantum cryptographic algorithms

PKP-based signature scheme and ultra-short multivariate signatures

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Informatique
Unité de recherche: Université Paris-Saclay, UVSQ, CNRS, Laboratoire de mathématiques de Versailles, 78000, Versailles, France.
Référent: Université de Versailles -Saint-Quentin-en-Yveline

Thèse présentée et soutenue à Versailles, le 18/12/2020, par

ELIANE KOUSSA

Composition du Jury

Louis GOUBIN Professeur, HDR, Université de Versailles Saint-Quentin en Yvelines (LMV/UVSQ)	Président
Pierre-Alain FOUQUE Professeur, HDR, Université de Rennes 1 (DI/ENS)	Rapporteur & Examinateur
David NACCACHE Professeur, HDR, École normale supérieure (DI/ENS)	Rapporteur & Examinateur
Christina BOURA Maître de Conférences, Université de Versailles Saint-Quentin en Yvelines (LMV/UVSQ)	Examinatrice
Jean-Charles FAUGÈRE Professeur, HDR, INRIA, Sorbonne Université (LIP6/Uni Paris 6), Co-fondateur et CTO de CRYPTONEXT	Examinateur
Aline GOUGET MORIN Experte sénior en cryptographie, Thales DIS / Gemalto Meudon	Examinatrice
Gilles MACARIO-RAT Ingénieur, Orange Labs	Examinateur
Jacques PATARIN Professeur, HDR, Université de Versailles Saint-Quentin en Yvelines (LMV/UVSQ)	Directeur de thèse

Thèse de doctorat

NNT : 2020UPASG027

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Analyse et conception d'algorithmes de cryptographie post-quantique: Schéma de signature basé sur PKP et signatures ultra-courtes de type multivariées

Mots clés: Cryptographie post-quantique, Cryptographie à clé publique, Cryptographie multivariée, Schéma d'identification zero-knowledge, Schéma de signature digitale, Signatures ultra-courtes

Résumé: La construction d'un ordinateur quantique remettrait en cause la plupart des schémas à clef publique utilisés aujourd'hui. Par conséquent, il existe actuellement un effort de recherche important pour développer de nouveaux schémas cryptographiques post-quantique. En particulier, nous nous intéressons aux schémas post-quantiques dont la sécurité repose sur la dureté de la résolution de certains problèmes mathématiques tels que le problème PKP et le problème HFE. Ce travail étudie d'abord la complexité de PKP. Et après une analyse approfondie des attaques connus sur PKP, nous avons pu mettre à jour certains résultats qui n'étaient pas précis, et fournir une formule de complexité explicite qui nous permet d'identifier les instances difficiles de ce problème et de donner des ensembles de paramètres sécurisés. PKP a été utilisé en 1989 pour développer le premier schéma d'identification à divulgation nulle de connaissance (ZK-IDS) qui a une implémentation efficace sur les cartes à puce. Dans un deuxième temps, nous optimisons le ZK-IDS basé sur PKP, puis nous introduisons PKP-DSS: un schéma de signature digitale basé sur PKP. Nous construisons PKP-DSS à partir du ZK-IDS basé sur PKP en utilisant la transformation Fiat-Shamir (FS) traditionnelle qui convertit les schémas d'identification en schémas de signature. Nous développons une implémentation à temps constant de PKP-DSS. Il semble que notre schéma soit très compétitif par rapport aux autres schémas de signature FS post-quantiques. Étant donné que PKP est un problème NP-dur et qu'il n'y a pas d'attaques quantiques connues pour résoudre PKP nettement mieux que les attaques classiques, nous pensons que notre schéma est post-quantique.
D'autre part, nous étudions les schémas de signature à clé publique de type multivariés qui fournissent des signatures ultra-courtes. Nous analysons d'abord les attaques les plus connues contre les signatures multivariées, puis nous définissons les paramètres minimaux permettant une signature ultra-courte. Nous présentons également de nouveaux modes d'opérations spécifiques afin d'éviter des attaques particulières. Deuxièmement, nous fournissons divers exemples explicites de schémas de signature ultra-courts, pour plusieurs niveaux de sécurité classique, qui sont basés sur des variantes de HFE sur différents corps finis.

Title: Analysis and design of some post-quantum cryptographic algorithms: PKP-based signature scheme and ultra-short multivariate signatures

Keywords: Post-quantum cryptography, Public-key cryptography, Multivariate cryptography, Zero-knowledge identification scheme , Digital signature scheme, Ultra-short signatures.

Abstract: The construction of large quantum computers would endanger most of the public-key cryptographic schemes in use today. Therefore, there is currently a large research effort to develop new post-quantum secure schemes. In particular, we are interested in post-quantum cryptographic schemes whose security relies on the hardness of solving some mathematical problems such as the Permuted Kernel Problem (PKP) and the Hidden Field Equations (HFE). This work investigates first the complexity of PKP. And after a thorough analysis of the State-of-the-art attacks of PKP, we have been able to update some results that were not accurate, and to provide an explicit complexity formula which allows us to identify hard instances and secure sets of parameters of this problem. PKP was used in 1989 to develop the first Zero-Knowledge Identification Scheme (ZK-IDS) that has an efficient implementation on low-cost smart cards. In a second step, we optimize the PKP-based ZK-IDS and then we introduce PKP-DSS: a Digital Signature Scheme based on PKP. We construct PKP-DSS from the ZK-IDS based on PKP by using the traditional Fiat-Shamir (FS) transform that converts Identification schemes into Signature schemes. We develop a constant time implementation of PKP-DSS. It appears that our scheme is very competitive with other post-quantum FS signature schemes. Since that PKP is an NP-hard problem and since there are no known quantum attacks for solving PKP significantly better than classical attacks, we believe that our scheme is post-quantum secure.
On the other hand, we study multivariate public-key signature schemes that provide"ultra"-short signatures. We first analyze the most known attacks against multivariate signatures, and then define the minimal parameters that allow ultra-short signature. We also design some specific new modes of operations in order to avoid particular attacks. Second, we provide various explicit examples of ultra-short signature schemes that are based on variants of HFE. We present parameters for several level of classical security: 80, 90, 100 bits in addition to 128, 192, and 256 bits; for each level, we propose different choices of finite fields.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

"To my beloved country Lebanon!"

Acknowledgments

My sincere gratitude and appreciation go to my PhD Director, Pr. Jacques Patarin, for his professional and aspiring guidance that pushed me to widen my work from different perspectives. Our continuous discussion, his immense knowledge and illuminating views on all the issues related to this work helped me to get results of better quality.

Also, i would especially like to show my deepest thanks for Gilles Macario-rat, R.D Jean-Charles Faugere, and A.P Ludovic Perret for sharing their enormous experiences, valuable comments and suggestions.
Their involvement was a major contributor to the completion of my PhD. I am really grateful for their precious advice and help.

I take this opportunity to acknowledge and thank the Sorbonne University - Pierre and Marie Curie campus (UPMC), and in particular, to the LIP6 laboratory and the PolSys team for allowing me to conduct my research and providing me with all the assistance requested and the essential facilities required for my work.

Also, I extend my thanks to the University of Versailles Saint-Quentin en Yvelines, in particular to the members of the LMV Laboratory: the CRYPTO team, the PhD students, researchers, trainees and the administrative staff for creating a good working atmosphere.
I would like to thank each one of my colleagues for their support and encouragement, for the discussions we had and the good times we spent together when things were really complicated.

I am also grateful to the members of the jury who were more than generous with their expertise and precious time. A special thanks to Prof. Pierre-Alain FOUQUE and Prof. David NACCACHE for their countless hours of reflecting, reading, and most of all patience throughout the entire process. Thank you Dr. Christina BOURA, Prof. Jean-Charles FAUGÈRE, Prof. Louis GOUBIN, Mrs. Aline GOUGET MORIN, and Mr. Gilles MACARIO-RAT for agreeing to invest time for reading this work and providing interesting and constructive feedback which was much appreciated.

I am very fortunate and grateful to all my friends outside work whom I consider my family in France. Their undying support made the completion of this research an enjoyable experience.
Sylvain, my warmest appreciation to the positive influence that you've had on my life. I thank you from the bottom of my heart for always being there for me. I am beyond lucky to have you in my life.

Of course, last but not least, I wholeheartedly would like to thank my whole family in Lebanon especially my loving parents Joseph and Nazira, my irreplaceable brothers Elie and Nakhlie, my friends that become family Melina and Tamara.Your spiritual support, intense caring, endless love, and unceasing encouragement in every aspect of my life have made me the person i am today. Your prayers were what kept me moving in this arduous journey. I am sincerely grateful for having you by my side.
Jennifer, a special thanks for you my wonderful niece for putting a smile on my face. We are so lucky to have you in our lives.

Contents

List of Tables

List of Figures

CHAPTER 1

General introduction

In modern society, electronic communication (also known as e-communication) is becoming more and more essential for most human activities, where individuals are extensively using technology in order to communicate, or to easily share information at anytime and any place on the planet. Electronic communication is fundamental and vital in numerous fields such as medicine, business, finance, market, social networking, etc.
Given the growing need for such type of information exchange, it is therefore, important to establish efficient methods to highly manage and protect the shared data in order to guarantee its correctness, integrity and durability.

Several security aspects exist in order to keep information and data from intentional or accidental alteration, modification or exposure. Using various technologies and standards, data security has a large number of applications as key and passwords management, private communications, secure transactions and financial data, etc.
A large number of technical and legal methodologies is required in order to accomplish information and data security. Cryptography is one of the main resources that provides these technical and legal means.

Cryptography grants privacy for information and protection for stored data or even for exchanged data through untrusted electronic lines. It ensures data sensitivity and information security by employing mathematical techniques that are associated with aspects of data security such as secrecy, integrity, authenticity and non-repudiation.
Cryptography is mainly a combination of three scientific fields: mathematical theory, computer science and engineering. It can be classified into three principal categories based on the number of employed keys for encryption and decryption in a cryptographic algorithm: Secret Key Cryptography (known as SKC), Public Key Cryptography (known as PKC), and Hash functions.
The main type of algorithms that will be used and discussed throughout this work will be the public key based cryptosystems.

Public Key Cryptography (PKC) is considered to be one of the most significant improvements in cryptography for the last three centuries. PKC strongly relies on a fundamental tool, the so-called one-way functions which are mathematical functions known to be easy to compute on every input, but difficult to invert given the image of a random input.
The most known and used candidates for one-way functions with PKC schemes are:

- Integer Multiplication and factorization: While computing the product of two integers is easy, it is relatively difficult to express a composite number as a product of smaller integers.

For significantly large composite numbers, there is no known classical polynomial algorithm to solve its decomposition (factorization) problem that is the FACT problem.

- Discrete exponentiation and logarithms: While exponentiation over a modulus can be easily performed, it is relatively hard to invert this process that requires computing a discrete logarithm.
  There is no efficient classical algorithm known to solve in polynomial time the discrete logarithm problem that is known also as the DLOG problem.

The security of various important public-key cryptographic protocols frequently used nowadays (such as RSA public key encryption, $\mathrm{Diffie\text{-}Hellman}$ key exchange, etc.) is essentially based on the mathematical hardness of solving the integer factorization FACT and the discrete logarithm DLOG problem.

Despite the fact that there are no efficient algorithms known to solve these problems on classical computers but, since Shor's quantum factoring algorithm [64], it is well known that a technological advance such as the development of a quantum computer will question the difficulty of both of these problems [64]; therefore, it might make completely obsolete such primitives as RSA scheme and $\mathrm{Diffie\text{-}Hellman}$ key exchange.

Quantum computers are believed to solve problems that are conjectured to be impossible to crack using a classical computer. All electronic communication channels will become more vulnerable to being hacked or even to being completely paralyzed. The development of quantum computers puts at risk the integrity of encrypted data.
Thus, it is essential to anticipate such technological breakthrough, therefore the need for post-quantum cryptography and quantum-secure cryptographic schemes.

In the coming decades, quantum computers would constitute a significant threat essentially to public key based cryptographic schemes. Consequently, the race to build new algorithms believed to withstand both of classical and future quantum attacks is basically focused on asymmetric schemes.
Post-quantum cryptography is the initiation of new type of cryptographic schemes called "quantum-safe" algorithms (also known as post-quantum cryptographic algorithms). It is possible to run these algorithms on today's classical computers but, they will be resistant to attacks carried out on tomorrow's quantum computers.

Post-quantum cryptography is based, in particular, on the fundamental assumption that there is no quantum polynomial algorithm to solve the "NP-Hard" problems [8].
A wide range of families has been suggested for post-quantum cryptography. The most trustworthy families include: multivariate cryptography, code-based cryptography, lattice-based cryptography, hash-based signatures, and supersingular elliptic curve isogeny cryptography.

There is another variety of schemes that might be potentially quantum-safe and it does not make part of the above cited families. It is particularly based on specific "NP-Hard" problems.
This work aims to develop new public key based cryptographic schemes that provide different levels of security against both classical and quantum threats.
These schemes will belong to the multivariate cryptographic algorithms family or more generally based on

specific NP-Hard problems such as the Permuted Kernel Problem PKP [63].

Multivariate cryptography involves asymmetric cryptographic schemes whose security relies on the hardness of solving a system of multivariate polynomials, of small degree over a finite field. Many multivariate schemes have been proposed presenting different types of functionalities. Several multivariate schemes have been broken, while there still exist some interesting and secure schemes: typically variants of the Hidden Field Equations HFE [25, 55].
The main advantage of multivariate cryptography is that it is historically known for its efficiency to construct short signature schemes. Moreover, on top of being flexible in terms of designing distinct schemes and variants, multivariate cryptography allows the development of successful schemes over small binary fields.

Public key cryptography, not only provides data security, but can also be used for user authentication and more precisely for digital signatures.
Digital signatures, the analogue form of handwritten signatures, binds the identity of an entity to a certain document and information which leads to protect the integrity of data in addition to the entity authentication.
The major goals of this thesis can be summarized as follows:

- To build a post-quantum digital signature scheme whose security is based on the difficulty of robust mathematical problems such as PKP.
- To develop multivariate signature schemes, based on variants of the HFE problem, which provide ultra-short signatures.

Post-quantum signatures

One of the principal goals is to study the design of post-quantum signatures constructed from identification schemes based on “NP-Hard" problems.
One of these problems is the Permuted Kernel Problem: the problem of finding a permutation of a known vector such that the resulting vector is in the kernel of a given matrix. This is a classical NP-Hard combinatorial problem which requires only simple operations such as basic linear algebra and permuting the entries of a vector.
For quite some time, no new attacks have been reported on PKP, thus it is possible to estimate the concrete complexity of the problem and then utilize it to introduce new cryptographic schemes.

In 1989, Shamir [63] proposed a five-move ZK-Identification scheme, based on PKP.
The so-called Fiat-Shamir paradigm [35] is a technique to convert a Zero-Knowledge (ZK) identification scheme into a signature scheme.
The main idea of Fiat-Shamir transform is to use the interactions exchanged during a ZK identification protocol as a signature. This leads to a signature scheme whose security is directly based on the difficulty of solving an algorithmic problem.

Typically, this technique was used in [22] to construct the MQDSS multivariate signature scheme. MQDSS is built from a ZK scheme whose security relies on the difficulty of solving a random set of quadratic equa-

tions (a specific case of the PoSSo), in order to obtain an efficient post-quantum signature scheme.
Chen et al. [22] proved that, in the random oracle model, applying the Fiat-Shamir transform to this five-move identification scheme provides a secure signature scheme.
A different line of work resulted in the Picnic signature scheme. Chase et al. [20] constructed this digital signature scheme by applying the Fiat-Shamir transform to an identification scheme whose security relies purely on symmetric primitives.

Before the NIST post-quantum standardization process, a cryptosytem, like MQDSS or Picnic, that produces signatures of significant length was considered completely impractical.
The situation now seems completely different with a greater emphasis on the security.
Thus, it is important to examine the possibility of having other post-quantum ZK schemes based on NP-Hard problems, in particular as PKP.

Ultra-short signatures

Another principal goal is to design secure signature schemes that provide ultra-short signatures. It is known that multivariate cryptography successfully provides such types of signatures.

Nowadays, the most widely used signature schemes, such as RSA, DSA, Elliptic Curve schemes, produce signatures of considerable lengths. Therefore, and for some specific application such as watermarking, QR codes, etc., it is useful to define new standards that provide ultra-short signatures. These will mainly belong to the multivariate family of cryptosystem.

The so-called C* algorithm is the first public-key multivariate-based cryptographic algorithm. It was proposed by T. Matsumoto and H. Imai in 1988 [51]. C* was later broken in [54] by J. Patarin, who suggested a way to fix it with the Hidden Field Equations (HFE) scheme in 1999 [56].
Various multivariate-based schemes have failed. However, some multivariate schemes as UOV [45], variants of HFE [25, 55], Rainbow [28], and Gemss [19] could make the basis for robust schemes.

The HFE family cryptosystems, proposed by J. PATARIN [55], is one of the best known and most studied schemes among all Multivariate Public Key Cryptosystems MPKCs. It can be used for public-key encryption and also for signatures, but it is generally more efficient for signatures schemes.
The HFE family consists of many variants due to the fact that it is always possible to apply different modifications (namely perturbations), and even to use different finite field ($\mathbb{F}_2$ is mostly used).
Considering the existence of some quasi-polynomial time algorithms to attack the "nude HFE", it is strongly recommended to always use suitable perturbations.

Therefore, the main motivation is to propose some sets of parameters for ultra-short HFE-based signature schemes over a finite field that can be signed and verified in less than one minute on a standard modern computer, despite the public-key size and by considering distinct levels of security.

Cryptanalysis

Cryptanalysis is a complementary aspect of the design that allows studying the hidden information of a cryptographic system in order to eliminate weak primitives, and thus to choose suitable parameters that guarantee security and efficiency.
Most of the time, cyptnalysis involves studying and analysing pure mathematical problems.

Since this work aims to design a post-quantum signature scheme based on the difficulty of solving hard problems (as the Permuted Kernel Problem PKP for example), then the first thing to look at is the complexity analysis of the underlying problem.
To do so, one must review all the well-known mathematical methods to solve this problem to determine and develop the most effective tool to solve the problem.
After the examination of all the possible approaches to solve the mathematical problem, an explicit complexity bound can be provided to better estimate the efficiency of the best attacks. Hence, it is possible to determine optimal parameters for hard instances of the underlying problem.

For multivariate cryptography, algebraic cryptanalysis is a general method for evaluating the security of a wide variety of cryptosystems.
This relatively recent technique is now widely considered as a fundamental tool in cryptanalysis. The main idea of this technique is to model a multivariate cryptographic primitive by a set of non-linear equations.
The system of equations is constructed to link the solutions of this system with secret information of the considered cryptographic primitive (for example, the secret key of an encryption scheme).

Considering the goal of developing ultra-short multivariate-based signature schemes, algebraic cryptanalysis can offer a powerful framework for studying the security of designed multivariate signature schemes.
The starting point here is to use a well-known tool proposed in the literature: the Gröbner bases. This tool seems well suited to systems that appear in algebraic cryptanalysis which often have naturally a specific structure.

Therefore, in this thesis, we provide the security analysis of different problems (such as the Permuted Kernel Problem (PKP), and the Hidden Field Equations (HFE)) via classical and algebraic cryptanalysis.
On the other hand, we propose new post-quantum and classical signature schemes.

CHAPTER 2

Introduction générale

Dans la société moderne, la communication électronique (également connue sous le nom de e-communication) devient de plus en plus essentielle pour la plupart des activités humaines, où les individus utilisent largement la technologie pour communiquer ou pour partager facilement des informations à tout moment et en tout lieu sur la planète. La communication électronique est fondamentale dans de nombreux domaines tels que la médecine, les affaires, la finance, le marché, les réseaux sociaux, etc.
Compte tenu du besoin croissant d'un tel type d'échange d'informations, il est donc important de mettre en place des méthodes efficaces pour gérer et protéger de manière optimale les données partagées afin de garantir leur exactitude, leur intégrité et leur durabilité.

Il existe plusieurs aspects de la sécurité qui empêchent l'altération, la modification ou l'exposition (intentionnelle ou accidentelleles) des informations et des données. Utilisant diverses technologies et normes, la sécurité des données possède un grand nombre d'applications comme la gestion des clés et des mots de passe, les communications privées, les transactions sécurisées et les données financières, etc.
Un grand nombre de méthodologies techniques et juridiques est nécessaire pour assurer la sécurité des informations et des données. La cryptographie est l'une des principales ressources qui fournit ces moyens techniques et juridiques.

La cryptographie garantit la confidentialité des informations et la protection des données stockées ou même des données échangées via des lignes électroniques non fiables. Elle garantit la sécurité des données et des informations sensibles en utilisant des techniques mathématiques associées à des aspects de la sécurité des données tels que le secret, l'intégrité, l'authenticité et la non-répudiation.
La cryptographie est principalement une combinaison de trois domaines scientifiques: la théorie mathématique, l'informatique et l'ingénierie. Elle peut être classée en trois catégories principales en fonction du nombre de clés utilisées pour le cryptage et le décryptage dans un algorithme cryptographique: la cryptographie à clé secrète, connue sous le nom de SKC (Secret Key Cryptography), la cryptographie à clé publique, connue sous le nom de PKC (Public Key Cryptography), et les fonctions de hachage.
Les principaux types d'algorithmes qui seront utilisés et discutés tout au long de ce travail seront les cryptosystèmes à clé publique.

La cryptographie à clé publique (PKC) est considérée comme l'une des améliorations les plus significatives de la cryptographie au cours des trois derniers siècles. La cryptographie s'appuie fortement sur un outil fondamental, les fonctions dites à sens unique qui sont des fonctions mathématiques connues pour être faciles à calculer sur toutes les entrées, mais difficiles à inverser étant donné l'image d'une entrée aléatoire.

Les candidats les plus connus et utilisés pour les fonctions à sens unique avec des schémas PKC sont:

- Multiplication et factorisation d'entiers: bien que le calcul du produit de deux entiers soit facile, il est relativement difficile d'exprimer un nombre composé comme un produit d'entiers plus petits.
  Pour les nombres composés significativement grands, il n'y a pas d'algorithme polynomial non quantique connu pour résoudre son problème de décomposition (factorisation) qui est le problème FACT.
- Exponentiation discrète et logarithmes: alors que l'exponentiation sur un module peut être facilement réalisée, il est en général relativement difficile d'inverser ce processus qui nécessite le calcul d'un logarithme discret.
  De même que pour la factorisation, il n'y a pas d'algorithme efficace connu non quantique pour résoudre en temps polynomial le problème du logarithme discret connu aussi sous le nom de problème DLOG.

La sécurité de divers protocoles cryptographiques à clé publique importants fréquemment utilisés de nos jours (tels que le chiffrement à clé publique RSA, l'échange de clés $\mathrm{Diffie\text{-}Hellman}$, etc.) est essentiellement basée sur la dureté mathématique de la résolution de la factorisation d'entiers FACT ou sur le problème du logarithme discret DLOG.

Bien qu'il n'existe pas d'algorithmes efficaces connus pour résoudre ces problèmes sur les ordinateurs classiques. Le développement d'un ordinateur quantique puissant remettra en question la difficulté de ces deux problèmes [64]; et par conséquent, cela pourrait rendre complètement obsolètes des primitives telles que le schéma RSA et l'échange de clés $\mathrm{Diffie\text{-}Hellman}$. En effet, l'algorithme de factorisation quantique de Shor [64] résout les deux problèmes (FACT et DLOG) avec une complexité polynomial sur un ordinateur quantique.

On pense que les ordinateurs quantiques résolvent des problèmes supposés impossibles à résoudre avec un ordinateur classique. Tous les canaux de communication électroniques deviendront plus vulnérables au piratage ou même à la paralysie complète. Le développement des ordinateurs quantiques mettera donc en péril l'intégrité des données chiffrées par les algorithmes habituels.
Ainsi, il est essentiel d'anticiper cette avancée technologique, et d'où le besoin de la cryptographie post-quantique et de schémas cryptographiques sécurisée contre les attaques quantiques.

Dans les décennies à venir, les ordinateurs quantiques constitueraient une menace importante essentiellement pour les schémas cryptographiques à clé publique. Par conséquent, la course à la construction de nouveaux algorithmes censés résister à la fois aux attaques quantiques comme classiques est essentiellement dirigée vers les schémas asymétriques.
La cryptographie post-quantique est l'initiation d'un nouveau type de schémas cryptographiques appelés algorithmes de sécurité quantique (également connus sous le nom d'algorithmes cryptographiques post-quantiques). Il est possible d'exécuter ces algorithmes sur les ordinateurs classiques d'aujourd'hui, mais ils seront résistants aux attaques menées sur les ordinateurs quantiques de demain.

La cryptographie post-quantique est basée, en particulier, sur l'hypothèse fondamentale qu'il n'y a pas d'algorithme polynomial quantique pour résoudre les problèmes " NP-dur " [8].

Un large éventail de familles a été suggéré pour la cryptographie post-quantique. Les familles les plus étudiées sont: la cryptographie multivariée, la cryptographie à base de code, la cryptographie sur les réseaux, les signatures à base de hachage et la cryptographie par isogénie sur les courbes elliptiques supersingulières.

Il existe une autre variété de systèmes qui pourraient être sûrs face aux attaques quantiques et qui ne font pas partie des familles citées ci-dessus. Elle est notamment basée sur des problèmes particuliers connus sous le nom des problèmes " NP-hard ". Ce sont souvent des problèmes combinatoires qui permettent de développer des schémas d'authentifications ou de signatures, mais pas de chiffrement.
Ce travail vise à développer de nouveaux schémas cryptographiques à clé publique qui offrent différents niveaux de sécurité contre les menaces classiques et quantiques.
Ceux-ci appartiendront à deux familles: les algorithmes cryptographiques multivariés, et les algorithmes basés sur des problèmes NP-dur spécifiques tels que le problème du noyau permuté PKP (Permuted Kernel Problem) [63].

La sécurité de la cryptographie multivariée repose sur la dureté de la résolution d'un système de polynômes multivariés, de petit degré sur un corps fini. De nombreux schémas multivariés ont été proposés présentant différents types de fonctionnalités. Plusieurs schémas multivariés ont été cassés, alors qu'il existe encore des schémas intéressants et sécurisés: en particulier des variantes de HFE (Hidden Field Equations) [25, 55].
Le principal avantage de la cryptographie multivariée est qu'elle est historiquement connue pour son efficacité à construire des schémas de signature courtes. En plus d'être flexible en termes de conception de schémas et de variantes distinctes, la cryptographie multivariée permet le développement de schémas efficaces sur de petits corps.

La cryptographie à clé publique assure non seulement la sécurité des données, mais peut également être utilisée pour l'authentification des utilisateurs et plus précisément pour les signatures digitales.
Les signatures digitales, l'analogue des signatures manuscrites, associent l'identité d'une entité à certains documents et informations, ce qui permet de protéger l'intégrité des données en plus de l'authentification de l'entité.
Les principaux objectifs de cette thèse peuvent être résumés comme suit:

- La construction d'un schéma de signature digitale post-quantique dont la sécurité est basée sur la difficulté du problèmes mathématique robustes tels que PKP.
- Le développement de schémas de signature multivariés, basés sur des variantes du problème HFE, qui fournissent des signatures ultra-courtes.

Signatures post-quantiques

L'un des principaux objectifs est d'étudier la conception de signatures post-quantiques construites à partir de schémas d'identification basés sur des problèmes " NP-durs ".
Un de ces problèmes est le problème du noyau permuté PKP: le problème de trouver une permutation d'un vecteur connu de telle sorte que le vecteur résultant soit dans le noyau d'une matrice donnée. Il

s'agit d'un problème combinatoire NP-dur classique qui ne nécessite que des opérations simples telles que l'algèbre linéaire de base et la permutation des entrées d'un vecteur.
Depuis un certain temps, aucune nouvelle attaque n'a été signalée sur PKP, il est donc possible d'estimer la complexité concrète du problème puis de l'utiliser pour introduire de nouveaux schémas cryptographiques.

En 1989, Shamir [63] a proposé un schéma d'identification ZK (Zero-Knowledge) en cinq passes, basé sur PKP.
Le paradigme Fiat-Shamir [35] est une technique permettant de convertir un schéma d'identification Zero-Knowledge (ZK) en schéma de signature.
L'idée principale de la transformation Fiat-Shamir est d'utiliser comme signature les interactions échangés lors d'un protocole d'identification ZK. Cela conduit à un schéma de signature dont la sécurité est directement basée sur la difficulté de résoudre un problème algorithmique.

Généralement, les auteurs de [22] ont utilisé cette technique pour construire le schéma de signature multivariée MQDSS. MQDSS est construit à partir d'un schéma ZK dont la sécurité repose sur la difficulté de résoudre un ensemble aléatoire d'équations quadratiques (un cas particulier du PoSSo: Polynomial System Solving), afin d'obtenir un schéma de signature post-quantique efficace.
Chen et al. ont prouvé que, dans le modèle d'oracle aléatoire, l'application de la transformation Fiat-Shamir à ce schéma d'identification en cinq passes fournit un schéma de signature sécurisé.
Une ligne de travail différente a abouti au schéma de signature Picnic. Chase et al. [20] ont construit ce schéma de signature digitale en appliquant la transformation Fiat-Shamir à un schéma d'identification dont la sécurité repose uniquement sur des primitives symétriques.

Avant le processus de standardisation post-quantique du NIST, un cryptosytème, comme MQDSS ou Picnic, qui produit des signatures de longueur significative était complètement considéré irréaliste.
La situation semble cependant maintenant différente avec un plus grand accent sur la sécurité.
Ainsi, il est important d'examiner la possibilité d'avoir d'autres schémas ZK post-quantiques basés sur des problèmes NP-durs, en particulier comme PKP.

Signatures ultra-courtes

Un autre objectif principal est de concevoir des schémas de signature sécurisés qui fournissent des signatures ultra-courtes. On sait que la cryptographie multivariée fournit avec succès de tels types de signatures.

De nos jours, les schémas de signature les plus largement utilisés, tels que RSA, DSA, schémas de courbe elliptique, produisent des signatures de longueurs d'environ 3000 bits (pour RSA) ce qui est relativement grand pour certains applications; DSA et les courbes elliptiques ont des signatures plus courtes. Par conséquent, et pour certaines applications spécifiques comme les codes QR par exemple, il est utile de définir de nouvelles normes qui fournissent des signatures ultra-courtes. Celles-ci appartiendront à la famille de la cryptographie multivariée .

L'algorithme dit C* est le premier algorithme cryptographique multivarié à clé publique. Il a été proposé

par T. Matsumoto et H. Imai en 1988 [51]. C* a ensuite été cassé dans [54] par J. Patarin, qui a suggéré un moyen de le corriger avec le schéma HFE (Hidden Field Equations) en 1999 [56].
Divers schémas multivariés ont échoué. Cependant, certains schémas multivariés comme UOV [45], variantes de HFE [25, 55], $\mathrm{Rainbow}$ [28], et Gemss [19] pourraient constituer la base de schémas robustes.

La famille de cryptosystèmes basés sur HFE, proposée par J. Patarin [55], fait partie des schémas les plus connus et les plus étudiés parmi tous les cryptosystèmes à clé publique multivariés MPKC. HFE et ses variantes peuvent être utilisés pour le chiffrement à clé publique et également pour les signatures, mais il est généralement plus efficace pour les schémas de signatures.
La famille HFE se compose de nombreuses variantes du fait qu'il est toujours possible d'appliquer différentes modifications (notamment des perturbations), et même d'utiliser des corps finis différents ($\mathbb{F}_2$ est le plus souvent utilisé).
Compte tenu de l'existence d'algorithmes en temps quasi-polynomial pour attaquer le " HFE nu", il est fortement recommandé de toujours utiliser des perturbations appropriées.

Par conséquent, la motivation principale est de proposer quelques ensembles de paramètres pour des schémas de signature ultra-courts basés sur des variantes de HFE sur un corps fini qui peuvent être signés et vérifiés en moins d'une minute sur un ordinateur moderne standard, malgré la taille de la clé publique et en considérant des niveaux de sécurité distincts.

Cryptanalyse

La cryptanalyse est un aspect complémentaire de la conception qui permet d'étudier les informations cachées d'un système cryptographique afin d'éliminer les primitives faibles, et ainsi de choisir des paramètres adaptés qui garantissent la sécurité et l'efficacité.
La cryptanalyse implique la plupart du temps d'étudier et d'analyser des problèmes mathématiques purs.

Puisque ce travail vise à concevoir un schéma de signature post-quantique basé sur la difficulté de résoudre des problèmes difficiles (comme le problème du noyau permuté PKP par exemple), alors la première chose à regarder est l'analyse de complexité de ce problème.
Pour ce faire, il faut revoir toutes les méthodes mathématiques bien connues pour résoudre ce problème afin de déterminer et de développer l'outil le plus efficace pour résoudre le problème.
Après l'étude de toutes les approches possibles pour résoudre le problème mathématique, une borne de complexité explicite peut être fournie pour mieux estimer l'efficacité des meilleures attaques. Par conséquent, il est possible de déterminer les paramètres optimaux pour l'état de l'art actuel pour les instances difficiles du problème.

Pour la cryptographie multivariée, la cryptanalyse algébrique est une méthode générale pour évaluer la sécurité d'une grande variété de cryptosystèmes.
Cette technique relativement récente est aujourd'hui largement considérée comme un outil fondamental de la cryptanalyse. L'idée principale de cette technique est de modéliser une primitive cryptographique multivariée par un ensemble d'équations non linéaires.
Le système d'équations est construit pour relier les solutions de ce système aux informations secrètes de

la primitive cryptographique considérée (par exemple, la clé secrète d'un schéma de chiffrement).

Compte tenu de l'objectif de développement de schémas de signature multivariés ultra-courts, la cryptanalyse algébrique peut offrir un cadre puissant pour étudier la sécurité des schémas de signature multivariées conçus.
Le point de départ ici est d'utiliser un outil bien connu proposé dans la littérature: les bases de Gröbner. Cet outil semble bien adapté aux systèmes qui apparaissent en cryptanalyse algébrique qui ont souvent naturellement une structure spécifique.

Par conséquent, dans cette thèse, nous fournissons l'analyse de sécurité de différents problèmes (tels que le problème du noyau permuté (PKP) et le problème (HFE)) via la cryptanalyse classique et algébrique. D'autre part, nous proposons de nouveaux schémas de signature post-quantiques et classiques.

PART I:

PRELIMINARIES

CHAPTER 3

An overview of cryptography

Contents

3.1 Introduction

Most of modern cryptographic schemes are heavily based upon mathematical theories, specifically on Number theory and algebra. This chapter focuses on some essential mathematical definitions required for the construction and the security analysis of a variety of cryptosystems.
Basic techniques from algebra, number theory, and computational complexity are briefly described in next sections in addition to essential cryptographic concepts and some related terms.

3.2 Mathematical background

This section provides some basic mathematical tools used to develop secure cryptographic schemes.

3.2.1 Number theory and Algebra

Number theory is a branch of mathematics dedicated mainly for the study of integers and number-theoretic functions, while algebra is devoted to the study of mathematical symbols and the rules for using them. Some of the major concepts of number theory and algebra that frequently occur in cryptology are defined below.
Let $\mathbb{Z}$ be the set of all integers $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$.

Definition 1 (Prime number) *An integer $p > 1$ is a prime number if its only divisors are 1 and p. An integer that is not prime is said to be composite.*

In digital electronic, it is important to represent each number by a sequence of binary digits similarly to the most common representation, namely the decimal representation.

Theorem 1 (Representation of integers) *A positive integer x can be expressed, using an integer $b > 1$ as a base, uniquely in the form:*

$$x = a_n b^n + a_{n-1} b^{n-1} + \ldots + a_1 b + a_0,$$

where the a_is are integers such that $0 \leq a_i < b$ and $a_n > 0$ for $i \in \{0, \ldots, n-1\}$.

By taking the base b, the representation length of any integer x can be expressed as

$$\lfloor \log_b x \rfloor + 1 = n + 1,$$

where $\log_b x$ is the logarithm of x to base b.
The representations are named according to the value of b. The most used representations are binary ($b = 2$), decimal ($b = 10$) and hexadecimal ($b = 16$).

Definition 2 (Congruence modulo n) *Let n be a positive integer.*
The integers x and y are said to be congruent modulo n, written $x \equiv y$ (mod n), if x and y have the same remainder when divided by n. The integer n is called the modulus of the congruence.

The congruence modulo n satisfies the following:

- for any $x, y \in \mathbb{Z}$, $x \equiv y$ (mod n) if and only if n divides $x - y$.
- Reflexivity: for any $x \in \mathbb{Z}$, $x \equiv x$ (mod n).
- Symmetry: for any $x, y \in \mathbb{Z}$, if $x \equiv y$ (mod n) then $y \equiv x$ (mod n).
- Transitivity: for any $x, y, z \in \mathbb{Z}$, if $x \equiv y$ (mod n) and $y \equiv z$ (mod n), then $x \equiv z$ (mod n).
- For any $x, y, z, t \in \mathbb{Z}$, if $x \equiv y$ (mod n) and $z \equiv t$ (mod n), then $x + z \equiv y + t$ (mod n) and $xz \equiv yt$ (mod n).

Rings and finite fields

An algebraic structure is formed of a set A, a certain number of operations, and a finite collection of conditions (also known as axiomes), that these operations must follow. Algebraic structures also involve special roles for some elements of the set A.
Most common examples of algebraic structures include groups, rings, field... .

Definition 3 (Ring) *A ring $R = (A, +, \cdot)$ consists of an addition operation (denoted arbitrarily $+$) and a multiplication operation (denoted arbitrarily $\cdot$), fulfilling the following conditions:*

- *Closure law: for any $a, b \in A$, $a + b \in A$ and $a \cdot b \in A$.*
- *Associative law: for any $a, b, c \in A$, $(a + b) + c = a + (b + c)$ and $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.*

- *Identity law: there are an additive zero element denoted 0_R, and a multiplicative identity element denoted 1_R, such that for any $a \in A$, the following holds:*

$$a + 0 = 0 + a = a, \qquad \textit{and} \qquad 1 \cdot a = a \cdot 1 = a.$$

- *Commutative law: for any $a, b \in A$, $a + b = b + a$.*
- *Distributive laws: for any $a, b, c \in A$, $(a + b) \cdot c = a \cdot c + b \cdot c$ and $c \cdot (a + b) = c \cdot a + c \cdot b$.*
- *Inverse law: for any $a \in A$, there is an element $b \in A$ such that*

$$a + b = b + a = 0, \textit{ where } b \textit{ is the additive inverse of } a \textit{ and denoted by } -a.$$

A ring is called a commutative ring if $a \cdot b = b \cdot a$ for any $a, b \in A$.

Definition 4 (Invertible element) *An element $a \neq 0$ of a ring R is called an invertible element if there exist an element b of R such that $a \cdot b = b \cdot a = 1$. The element b is the multiplicative inverse of a, and it is denoted by a^{-1}.*

Definition 5 (Field) *A field is a commutative ring in which all non-zero elements are invertible.*

Definition 6 (Finite field) *A finite field $\mathbb{F}$ (also known as Galois field GF) is a field with a finite number of elements. The number of elements is also referred to as the order of the field.*

Existence and uniqueness of finite fields:

- $\mathbb{F}$ is a finite field with q elements if and only if $q = p^m$ for some prime number p and integer $m \geq 1$.
- For every prime power order $q = p^m$, there exists a unique finite field $\mathbb{F}$ of order p^m denoted by $\mathbb{F}_{p^m}$ or also by $GF(p^m)$.

3.2.2 Computational complexity theory

Computational complexity theory is a computer science concept that allows classification of computational problems according to the amount of difficulty and resources needed to solve them. The complexity measures include time, storage space, the amount of communication, the processors number, etc... Mainly, time and space storage are the most important resources to look at.

A computational problem is an abstract task to be solved by an algorithm: a finite collection of mathematical steps and instructions that are computer-implementable.

Definition 7 (Algorithm) *An algorithm is a well-determined process or formula for solving a problem or for performing a computation. It takes an input string, referred to as a problem instance, and outputs a solution corresponding to the input.*

Definition 8 (Deterministic algorithm) *An algorithm is said to be deterministic if it follows the same sequence of operations on the same given input.*

Definition 9 (Probabilistic algorithm) *A probabilistic algorithm is a type of non-deterministic algorithms whereby either the execution path or the output of the algorithm might differ even when given the same input. On each computation step there are two possible next steps. The next move to take is chosen according to some probability distribution.*

The efficiency of an algorithm to solve a computational problem is measured by the time required to find a solution. The running time of an algorithm depends on the problem instance. In fact, large instances require more time to solve than normal instances. Therefore, the time (or storage space, or any other measure of complexity) needed to run an algorithm depends on the length of the problem instance (the input string).

Definition 10 (Input size) *For a given algorithm, the size of an instance is defined as the total number of bits required to binary-represent the instance using a suitable encoding scheme.*

For example:

- the number of bits needed to encode a positive integer x is $\lfloor \log x \rfloor + 1$ which is generally approximated by $\log x$.
- the number of bits needed to encode a matrix with m rows, n columns, and positive integer entries, each at most x, is approximately $mn \log x$.

Definition 11 (Running time) *The running time of a given algorithm is defined as the number of operations or steps executed on a particular input.*

A step can be considered as a bit comparison or assignment.

Complexity measures

There are several distinct ways to measure the running time complexity of an algorithm based on the size of its input:

- **The worst-case complexity** is a function (representing time versus input size) defined by the maximal number of operations (steps) to be executed on any input.
- **The average-case complexity** is a function defined by the average number of operations to be executed on any instance. It can be done by running the algorithm several times on different inputs of the same size, then the average complexity is given as the total running time divided by the number of trials.
- **The best-case complexity** is a function defined by the minimum number of operations to be executed on any instance.

In practice, the complexity of an algorithm is usually measured by its worst-case complexity. However, it is complicated to derive the exact form of the functions cited above. Therefore, an asymptotic analysis is usually used instead in order to determine approximations of the running time complexity. The asymptotic analysis allows to study the growth of the running time of an algorithm when the input size increases

without bound.

The most used asymptotic notations to represent running time complexity of algorithms are listed below. The notations require the definition of two non-negative functions f and g valued for non-negative integers n. Let $f(n)$ be the running time of an algorithm on inputs of size n, and $g(n)$ be an arbitrary time complexity to be related to the algorithm.

- **Asymptotic upper bound,** also known as Big O notation, defines an upper bound on the growth of the running time for sufficiently large input sizes n.
  $f(n) = O(g(n))$ means that there exists some positive constant c such that $0 \leq f(n) \leq cg(n)$ for large enough n.
- **Asymptotic lower bound,** also known as Ω notation, defines a lower bound on the growth of the running time for sufficiently large input sizes n.
  $f(n) = \Omega(g(n))$ means that there exists some positive constant c such that $0 \leq cg(n) \leq f(n)$ for large enough n.
- **Asymptotic tight bound,** also known as θ notation, defines an upper and a lower bound on the growth of the running time for sufficiently large input sizes n.
  $f(n) = \theta(g(n))$ means that there exist two positive constants c_1 and c_2 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for large enough n.

The most commonly used notation to express the running time complexity is the Big O notation.

Definition 12 (Polynomial Time algorithm PT) *An algorithm is said to be of polynomial time if its worst-case running time function is polynomial in the size of the algorithm input.*
The worst-case running time function of a PT algorithm is of the form $O(n^k)$, where n is the input size and k is a constant.

An algorithm is said to be of exponential time if its worst-case running function cannot be bounded.

Definition 13 (Sub-exponential Time algorithm) *An algorithm is said to be of sub-exponential time if its worst-case running time function is of the form $\exp(O(n))$, where n is the input size.*

Complexity classes

The notion of polynomial time algorithms yields different complexity classes that can be summarized as follows:

- *The complexity class $\mathcal{P}$* (deterministic polynomial time problems) is the set of problems that can be solved, using a deterministic algorithm, in polynomial time.
- *The complexity class $\mathcal{NP}$* (non-deterministic polynomial time problems) is the set of problems that can be solved, using a non-deterministic algorithm, in polynomial time.
- *The complexity class* co-$\mathcal{NP}$ (complementary non-deterministic polynomial time problems) is the set of the complement problems (i.e. problems resulting from reversing the yes and no answers) of the $\mathcal{NP}$ problems.

It is possible to convert one problem to another, such process is called a reduction operation. Therefore, another complexity classes can be defined using the reduction operation:

- *The complexity class $\mathcal{NP}$-hard:* a problem A is said to be *$\mathcal{NP}$-hard* if for every problem B in the $\mathcal{NP}$ class, there exists a reduction from B to A in deterministic polynomial time.
- *The complexity class $\mathcal{NP}$-complete:* a problem A is said to be *$\mathcal{NP}$-complete* if A is an $\mathcal{NP}$ problem and if for every problem B in the $\mathcal{NP}$ class, there exists a reduction from B to A in deterministic polynomial time.

The following inclusions hold:

$$\mathcal{P} \subseteq \mathcal{NP} \text{ and } \mathcal{P} \subseteq co\text{-}\mathcal{NP},$$

while there are some outstanding unsolved problems:

1. Is $\mathcal{P} = \mathcal{NP}$?
2. Is $\mathcal{NP} = co\text{-}\mathcal{NP}$?
3. Is $P = \mathcal{NP} \cap co\text{-}\mathcal{NP}$?

3.3 Basic concepts of cryptography

Cryptography is the science of protecting private communications and information in the presence of malicious third parties referred to as adversaries. Cryptography relies heavily on mathematical concepts and computer science techniques allowing the conversion of an ordinary text called plain-text into apparent nonsense text called ciphertext (the process of encryption) and back again upon arrival (the process of decryption).

The encryption/decryption processes are controlled each by an algorithm (a set of rule-based calculations) and a key (a string of characters that can be public or remains secret). Cryptography is closely related to the encryption and decryption processes using mathematical algorithms, whereas cryptanalysis is the term used for the analysis of encryption schemes in order to search for algorithm vulnerabilities and breach cryptographic or information security systems to gain access to the content of ciphertexts. Cryptology is the term reffering to the combined study of both cryptography and cryptanalysis.

The main goals of cryptography is to ensure the following information security objectives:

- **Confidentiality** refers to keeping the content of sensitive data from being available or disclosed to illegitimate parties.
- **Data integrity** refers to assuring that the shared information cannot be modified by unauthorized parties without the modification being detected.
- **Authentication (Identification)** allows each party to verify and confirm the identity of the other party participating in the communication (authentication of parties). It provides also the verification of the origin/destination of the delivered information (authentication of data).

- **Non-repudiation** refers to preventing a party from denying the intention in the creation or transmission of previous commitment at later stage.

Cryptographic algorithms are further classified into three distinct categories according to the number of keys utilized for encryption and decryption:

- **Symmetric cryptography** or Private/Secret-key cryptography (SKC)
- **Asymmetric cryptography** or Public-key cryptography (PKC)
- **Hash functions**

3.3.1 Symmetric cryptography

Symmetric-key cryptography (also known as Secret-key cryptography) refers to cryptographic techniques in which a single common key is used for both encryption and decryption processes.

The sender and the receiver must already share the same key or, uncommonly, different keys but linked in a simple computable way. Therefore, symmetric cryptographic algorithms employ a shared key that is known to both sender and receiver and remains secret to other parties. The sender/creator encrypt the plain-text using a shared key and sends the corresponding ciphertext to the receiver. The receiver uses the same shared key to decrypt the ciphertext and obtain the original plain-text.

The significant drawback of symmetric-key cryptography is the secure exchange of the key that is used by both parties for encryption and decryption as well. The key distribution problem yields a new type of cryptography : the asymmetric cryptography.

3.3.2 Asymmetric cryptography

Unlike symmetric-key cryptography, asymmetric-key cryptography (also known as Public-key cryptography) refers to cryptographic techniques in which two different keys are used for encryption and decryption processes.

Each party possesses a key pair (a public key and a private key). The private key must remain secret at all the times, while its paired public key may be freely shared across the network so if any party needs to communicate with an another party can use its corresponding public key. The public key is used for data encryption and the ciphertext is then decrypted withe the paired secret key.

3.3.3 Hash functions

The concept of hash functions depends on a special type of functions, namely one-way functions. There are some type of functions that are essential in cryptography. The definitions of such functions are given below.

Definition 14 *A function f from a set X to a set Y is defined by a rule that assigns to each element $x \in X$ a unique element $y \in Y$. y is the image of x under f and it is denoted by $y = f(x)$.*
The subset of Y consisting of all outputs of f is denoted by $f(X) = Im(f) = \{f(x) \mid x \in X\}$.

Definition 15 (One-way function) *Let f be a map from a set X to a set Y. f is called to be one-way function if it is easy to compute the image $f(x)$ for any $x \in X$, but it is computationally hard to find for essentially all $y \in Im(f)$ a suitable $x \in X$ such that $y = f(x)$.*

Informally, a one way function is a function for which computation in one direction is straightforward, yet difficult in the reverse direction.

Definition 16 (Trapdoor function) *A trapdoor function is a one-way function $f : X \mapsto Y$ coming with the property that given a special information, called the trapdoor, it is then feasible to compute for any $y \in Im(f)$ a suitable $x \in X$ such that $y = f(x)$.*

Cryptographic hash functions refer to mathematical deterministic algorithms that use basically no key. Hash functions are one-way functions that take numeric data of arbitrary length as input and map to a bit string of fixed length. The output of a hash function is called the message digest or the hash value. Practically, it is impossible to recover the original input of a hash function knowing the corresponding hash value. Cryptographic hash functions provide ideally the following properties:

- *Pre-image resistance:* While it is easy to compute the hash value of an input or a message m; knowing a hash value H, it is hard to find any message m such that $H = \text{Hash}(m)$.
- *Second pre-image resistance:* Given a fixed message m_1, it is hard to find a message $m_2 \neq m_1$ such that $\text{Hash}(m_1) = \text{Hash}(m_2)$.
- *Collision resistance:* It is hard to find two different message m_1 and m_2 such that $\text{Hash}(m_1) = \text{Hash}(m_2)$.

3.3.4 The difference between Symmetric, Asymmetric and Hash function cryptography

- Symmetric cryptography requires the use of a single key for both encryption and decryption processes, asymmetric key requires the use of a key pair, whereas hash function require no key for encryption or for decryption.
- Symmetric cryptography is generally faster than asymmetric and hash function cryptography but, it is less secure comparing to the other types of cryptography. Symmetric cryptography is ideally used to encrypt large amount of data.
- The disadvantage of symmetric algorithms is that the key must remain secret, and yet must be transmitted to the second party. Asymmetric cryptography was introduced to overcome this key exchange problem, and the hash function cryptography to provide higher levels of security.

CHAPTER 4

Identification Schemes (IDS)

Contents

4.1 Introduction

In this thesis we study the conversion of an Identification scheme (IDS) into a Digital Signature Scheme (DSS) using the well known technique of Fiat-Shamir (FS) [35]. Hence, we first introduce some of the formal definitions of an Identification scheme (IDS). Also, the structure of an identification protocol and its essential properties will be presented.

Authentication methods

Authentication is the identity verification process of an entity (a person, a user, or a device) that asks for access to the objects, services, data or resources of the system. Validating the identity authorizes or rejects the usage request of the required services. So that, the authentication process provides the identification of the user, and the authentication of its identity before getting into the system. It can be divided into two distinct phases:

- Identification: it answers the question of *who is the user?*.
- Data-origin authentication: it answers the question of *is the user really who he/she claims to be?*.

Identification scheme (IDS)

Identification is the concept whereby one entity ascertains through cryptographic methods the identity of another entity involved in a protocol which is known as identification protocol. Consider a party A Alice that needs to identify herself to a second party B Bob seeking for example Bob's help. If both parties meet face to face, Bob can recognize Alice by her face, voice, or other physical characteristics. People identify

one another by associating unique characteristics for each other. But, the situation is different when Alice and Bob are communicating over an insecure channel intercepted by an adversary C Charly.

During a digital conversation, and similarly to the physical world, each entity must be associated and represented with something unique that should be provided only by the entity itself. Most importantly, an identification protocol over an open line must be protected from impersonation attacks; so that an adversary C Charly fails to assume the identity of the legitimate party A Alice in the communication with the other party B Bob. Even Bob should be incapable to impersonate Alice.

Informally, an identification scheme consists of a challenge-response interaction between a Prover $\mathcal{P}$ and a Verifier $\mathcal{V}$:

- The Verifier $\mathcal{V}$ sends a challenge to the Prover $\mathcal{P}$,
- The Prover $\mathcal{P}$ responds with a valid answer that only $\mathcal{P}$ has knowledge of in order to be identified.

The Verifier $\mathcal{V}$ checks the response given by the Prover $\mathcal{P}$ and then verifies if he is whomever he claims to be.

4.2 Formal definitions and general structure

In an identification scheme (IDS), a Prover $\mathcal{P}$ tries to convince a Verifier $\mathcal{V}$ about its identity without letting a read adversary successfully assume the identity of the legitimate Prover $\mathcal{P}$. The goal is to have a secure identification scheme IDS that prevents an impersonation attack. More precisely:

Definition 17 (General structure) *. Let $\lambda \in \mathbb{N}$ be a security parameter.*
An identification scheme IDS consists of three probabilistic polynomial time (PPT) algorithms $IDS(1^\lambda) = (\textsf{KeyGen}, \mathcal{P}, \mathcal{V})$ such that:

- *the key generation* KeyGen *algorithm takes as input a security parameter λ and returns a public and secret key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \textsf{KeyGen}(1^\lambda)$,*
- *the Prover $\mathcal{P}$ algorithm takes as input the secret key* sk *and the ongoing conversation with the Verifier $\mathcal{V}$ and then sends the next message to $\mathcal{V}$,*
- *the Verifier $\mathcal{V}$ deterministic algorithm takes as input the public key* pk *and a full conversation transcript. $\mathcal{V}$ outputs a boolean value b with $b = 1$ signaling an identity validation, and $b = 0$ a rejection.*

An identification scheme must satisfy a completeness condition that every true statement could be justified by the Prover $\mathcal{P}$, with knowledge of the secret key sk, and the probability that the Verifier is convinced is almost 1.
More formally:

Definition 18 (Completeness) *An Identification scheme $(\textsf{KeyGen}, \mathcal{P}, \mathcal{V})$ is called complete if when both parties $\mathcal{P}$ and $\mathcal{V}$ follow the protocol correctly and honestly, the verifier $\mathcal{V}$ accepts the identification with probability 1. In other words, we have:*

$$\Pr\begin{bmatrix}(\mathsf{pk}, \mathsf{sk}) \leftarrow \textsf{KeyGen}(1^\lambda) \\ \langle \mathcal{P}(\mathsf{sk}), \mathcal{V}(\mathsf{pk}) \rangle = 1\end{bmatrix} = 1\,,$$

where $\langle \mathcal{P}(\mathsf{sk}), \mathcal{V}(\mathsf{pk}) \rangle$ *stands for the interaction between* $\mathcal{P}$ *with input* sk *and* $\mathcal{V}$ *with input* pk*.*

Roughly speaking, the structure of an identification scheme (IDS) uses the concept of a commitment scheme which covers generally two important phases. The first, namely $\mathrm{commit\ phase}$, the Prover $\mathcal{P}$ commits to a statement or a value Com that remains hidden to the Verifier $\mathcal{V}$ ($\mathrm{hiding\ property}$, see Section 4.5.1). The second, so called $\mathrm{reveal\ phase}$, during which the committed statement or value is revealed with some additional information that helps the Verifier to check opening correctness of the commitment. The committed value Com must be unchangeable after the $\mathrm{commit\ phase}$, so that the Prover cannot open its committed value in multiple ways ($\mathrm{binding\ property}$, see Section 4.5.1).

More precisely, an IDS that uses the idea of a commitment scheme can be defined as follows:

Definition 19 (Commitment scheme) *A commitment scheme consists of a triplet algorithms* (SETUP, COMMIT, VERIFY) *such that:*

- *the setup algorithm* SETUP(1^λ) *generates, for a given security parameter* λ*, the public parameters of the protocol,*
- *the commit algorithm takes as input the message to commit to* m *and a uniformly random bit string* r*, and outputs a commitment* com $\leftarrow$ COMMIT(r, m)*,*
- *the Verify algorithm* VERIFY *takes as input the commitment/random value pair* (com, r) *and the original message* m*, computes a commitment* $\widetilde{\mathsf{com}}$ *using the message/random pair communicated to the Verifier during the opening phase, and outputs a boolean value* b *with* $b = 1$ *whether* $\widetilde{\mathsf{com}} = \mathsf{com}$*, and* $b = 0$ *otherwise.*

The commitment scheme is $\mathrm{non\text{-}interactive}$ in terms of the VERIFY algorithm and can be considered as a derandomization of the COMMIT algorithm. The random string r used to commit to a statement or value is, at the same time, used to open and reveal the commitment to the Verifier.

By abusing the notation, we often omit mentioning the commitment randomness. We use com $\leftarrow$ COMMIT(m) which denotes implicitly the process of the commitment randomness using r (com $\leftarrow$ COMMIT(r, m)). Similarly, during the verification phase, com $=$ COMMIT(m) actually includes that the Prover communicates r to the Verifier, and that the Verifier checks COMMIT(r, m).

4.3 Canonical Identification Scheme (IDS)

A general case of an identification scheme is a three-move protocol between a Prover $\mathcal{P}$ and a Verifier $\mathcal{V}$. It is presented in Fig. 4.1 and defined as follows:

Definition 20 (Canonical three-move IDS) *A three-move canonical identification scheme* $IDS = (\text{COMMIT}, \text{CHALLENGE}, \text{PROVE}, \text{VERIFY})$ *is a protocol in which both of the prover* $\mathcal{P}$ *and the Verifier* $\mathcal{V}$ *algorithms are divided into two sub-algorithms* $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2)$ *respectively* $\mathcal{V} = (\mathcal{V}_1, \mathcal{V}f)$*:*

- $\mathcal{P}_1$ *is the* COMMIT *algorithm that takes as input the secret key* sk *and outputs the first move which is the initial commitment* com *and a state* St*,*

- $\mathcal{V}_1$ *is the* CHALLENGE *algorithm that samples the first challenge* ch *from the challenge set* ChSet*,*
- $\mathcal{P}_2$ *is the* PROVE *algorithm that takes as input* $(\mathsf{sk}, \mathsf{com}, \mathsf{ch}, \mathsf{St})$ *and outputs a response* rsp*,*
- $\mathcal{V}f$ *is the* VERIFY *algorithm that takes as input the interaction transcript* $(\mathsf{pk}, \mathsf{com}, \mathsf{St}, \mathsf{ch}, \mathsf{rsp})$ *and outputs a final decision* b*,* $b = 1$ *for acceptance or* $b = 0$ *for rejection.*

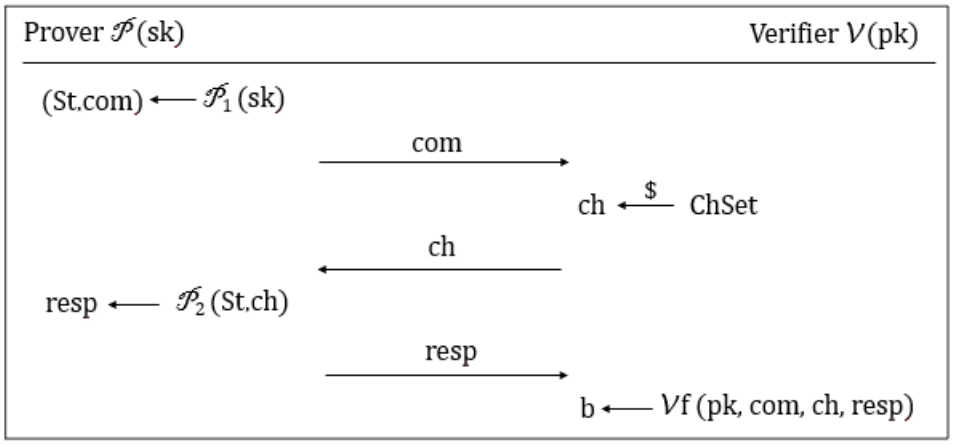

Figure 4.1: Canonical three-move identification scheme

It is possible to extend a canonical three-move identification scheme and perform $2n + 1$-move identification scheme. The conversation-transcript of this latter consists of the first move taken by $\mathcal{P}$ and multiple rounds of challenge and response messages: n challenges ch_i randomly sampled from ChSet_i and n responses rsp.

Similarly to a three-move IDS, the formal definition of a $2n + 1$-move IDS is:

Definition 21 ($2n + 1$**-move IDS)** *A* $2n + 1$*-move canonical identification scheme* $IDS = (\text{COMMIT}, \text{CHALLENGE}, \text{PROVE}, \text{VERIFY})$ *is a protocol in which both of the prover* $\mathcal{P}$ *and the Verifier* $\mathcal{V}$ *algorithms are divided into* $n + 1$ *sub-algorithms* $\mathcal{P} = (\mathcal{P}_1, \cdots, \mathcal{P}_{n+1})$ *respectively* $\mathcal{V} = (\mathcal{V}_1, \cdots, \mathcal{V}_n, \mathcal{V}f)$*:*

- $\mathcal{P}_1$ *is the* COMMIT *algorithm that takes as input the secret key* sk *and outputs the first move which is the initial commitment* com *and a state* St*,*
- $\mathcal{V}_1$ *is the* CHALLENGE *algorithm that samples the first challenge* ch_1 *from the challenge set* ChSet_1*,*
- $\mathcal{P}_2$ *is the* PROVE *algorithm that takes as input* $(\mathsf{sk}, \mathsf{com}, \mathsf{ch}, \mathsf{St})$ *and outputs the first response* rsp_1*,*
- *for* $i \in \{3, \cdots, n + 1\}$*:*

1. $\mathcal{V}_i$ *is the* CHALLENGE *algorithm that samples the i-th challenge* ch_i *from the challenge set* ChSet_i*,*
2. $\mathcal{P}_i$ *is the* PROVE *algorithm that takes as input* $(\mathsf{sk}, \mathsf{com}, \mathsf{ch}_i, \mathsf{St})$ *and outputs the i-th response* rsp_i*,*

- $\mathcal{V}f$ *is the* VERIFY *algorithm that takes as input the interaction transcript* $(\mathsf{pk}, \mathsf{com}, \mathsf{St}, \mathsf{ch}_1, \mathsf{rsp}_1, \cdots, \mathsf{ch}_{n+1}, \mathsf{rsp}_{n+1})$ *and outputs a final decision* b*,* $b = 1$ *for acceptance or* $b = 0$ *for rejection.*

A classic example of $2n + 1$-move identification schemes IDS is a five-move IDS for $n = 2$. This variant involves two rounds of both challenges and responses as presented below in Fig. 4.2:

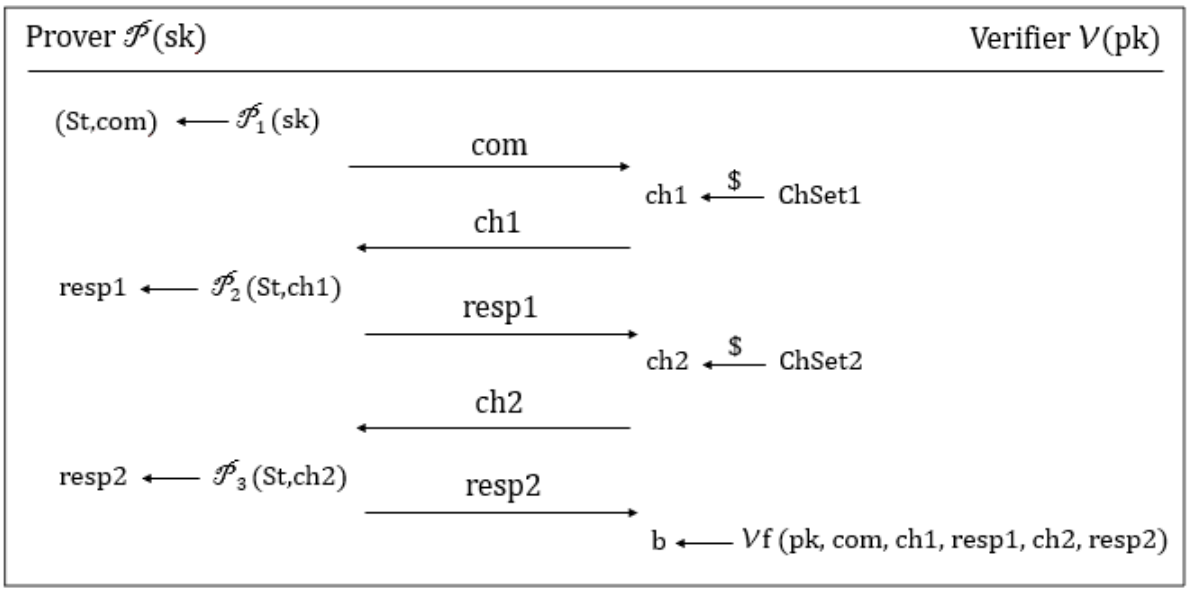

Figure 4.2: Five-move identification scheme

We also provide a useful definition of a special case of a five-move identification scheme where the challenge sets ChSet_1 and ChSet_2 verify respectively $|\mathsf{ChSet}_1| = q$ and $|\mathsf{ChSet}_2| = 2$.

Definition 22 (Five-move q2-identification scheme) *A five-move identification scheme is said to be a q2-identification scheme if the sizes of its two challenge sets* ChSet_1 *and* ChSet_2 *are respectively* q *and* 2*.*

4.4 Zero-Knowledge protocols

The most known class of identification schemes are the so-called zero-knowledge identification proofs or zero-knowledge identification protocols, commonly referred to as ZKPs, which allow one entity (the Prover $\mathcal{P}$) to prove to an another entity (the Verifier $\mathcal{V}$) the knowledge of a certain secret without sharing or revealing underlying information apart the knowledge of the claimed secret.
One can trivially demonstrate the possession of some secret data by simply exposing the data itself, the essential potential of using zero-knowledge proof is that such protocols eliminate the transfer of any sensitive data when communicating with one another.

Zero-knowledge protocols ZKPs are probabilistic proofs and not proofs in the mathematical sense of the term. In other terms, zero-knowledge protocols ZKPs are not deterministic which implies ZKPs don't demonstrate the possession of a confidential information with the same surety and certainty as revealing the entire information. Instead, ZKPs provide the accumulation of small unrelated information which shows that the possession of the claimed secret is strongly probable.

There are two variants of zero-knowledge proofs:

- **Interactive $\mathbf{ZKP}$** whereby two entities a Prover $\mathcal{P}$ and a Verifier $\mathcal{V}$ interact by exchanging a series of messages involving commitments (generated by $\mathcal{P}$), challenges (generated by $\mathcal{V}$) and responses (given by $\mathcal{P}$) over multiple rounds.
- **Non-interactive $\mathbf{ZKP}$** is an almost one round protocol whereby no interaction is needed between a Prover $\mathcal{P}$ and a Verifier $\mathcal{V}$.
  $\mathcal{P}$ sends a proof transcript in a single message to $\mathcal{V}$. Then, the Verifier $\mathcal{V}$ uses the proof transcript handed by the Prover $\mathcal{P}$, and verify its (in)validity by himself without needing any further assistance from the Prover $\mathcal{P}$.
  Unlike an interactive ZKP, it is not necessary to repeat interaction between $\mathcal{P}$ and $\mathcal{V}$.

Note that, it is possible to convert an interactive protocol into a non-interactive protocol by replacing the Verifier $\mathcal{V}$ by a hash function (or something similar) that computes the challenges over the set of commitments without the intervention of $\mathcal{V}$.

4.5 Security properties of an IDS

The classical form of identification schemes are the commitment schemes based on zero-knowledge concepts. And when it comes to such schemes, several security properties must be considered.

4.5.1 Security of a commitment identification scheme

As stated informally in Section 4.2, a commitment scheme must verify two essential security properties: the hiding property guarantees that there is no leak information about the message to commit to by sending the commitment to the Verifier, and the binding property guarantees the inability of the Prover to change the commitment and open it in two distinct ways.

Definition 23 (Computationally hiding) *Let* $\textsc{Commit}(1^\lambda)$ *be a commitment scheme with security parameter* $\lambda \in \mathbb{N}$*.* COMMIT *is computationally hiding if and only if, for any probabilistic polynomial time* PPT *algorithm* $\mathcal{A}$*, it is computationally hard to distinguish between* $\mathsf{com} = \textsc{Commit}(r, m)$ *and* $\mathsf{com}' = \textsc{Commit}(r, m')$ *for any two messages* m, m' *and a uniformly random string* r*. More formally:*

$$\Big| Pr\big[\mathcal{A}\big(\textsc{Commit}(r, m)\big) = 1\big] - Pr\big[\mathcal{A}\big(\textsc{Commit}(r, m')\big) = 1\big] \Big| = negl(\lambda).$$

Definition 24 (Computationally binding) *Let* $\textsc{Commit}(1^\lambda)$ *be a commitment scheme with security parameter* $\lambda \in \mathbb{N}$*.* COMMIT *is computationally hiding if and only if, for any* PPT *algorithm* $\mathcal{A}$*, it is computationally hard to come up with two valid commitments* $\mathsf{com} = \textsc{Commit}(r, m)$*,* $\mathsf{com}' = \textsc{Commit}(r', m')$ *for two different messages* $m \neq m'$ *such that* $\mathsf{com} = \mathsf{com}'$*. More formally:*

$$Pr\Big[(r, m, r', m') \leftarrow \mathcal{A}(1^\lambda) \ \wedge \ m \neq m' \quad \Big| \quad \textsc{Commit}(r, m) = \textsc{Commit}(r', m')\Big] = negl(\lambda).$$

4.5.2 Security of a zero-knowledge identification scheme $\mathrm{ZK\text{-}IDS}$

This section formally states the essential properties of a secure zero-knowledge identification scheme $\mathrm{ZK\text{-}IDS}$ against passive impersonation attack without knowledge of the secret key.

The notion of passive attacks implies that an attacker (called an impersonator), besides the public key, has an access to the view of some number of valid interaction transcripts between a legitimate prover and a verifier. On the other hand, the notion of active attacks is much stronger where an attacker is able to actively cheat and modify a transcript during an interactive protocol. Therefore, the security against active attacks is obviously harder to be achieved than against passive attacks.

However, an identification scheme IDS must satisfy the properties given below that ensure the security against passive attacks.

- **Completeness**
  The completeness property of zero-knowledge proofs guarantees that the Prover $\mathcal{P}$, with knowledge of secret with an overwhelming probability, will be able to convince the Verifier $\mathcal{V}$ to accept the proof.

  A formal definition of completeness is given in Definition 18, Section 4.2.

- **Soundness**
  The Prover $\mathcal{P}$ can always anticipate the Verifier challenge, and generate a response to send. Thus, a non-legitimate Prover, with no possession of the claimed secret, is able to convince the Verifier $\mathcal{V}$ to accept a false statement with some small probability called soundness error.
  More formally:

  Definition 25 (Soundness with soundness error κ) *An identification scheme* $(\textsc{KeyGen}, \mathcal{P}, \mathcal{V})$ *is called Sound with soundness error* κ*, if for any probabilistic polynomial time adversary* $\mathcal{A}$*, we have*

$$\Pr\left[\begin{matrix}(\mathsf{pk}, \mathsf{sk}) \leftarrow \textsc{KeyGen}(1^\lambda) \\ \langle \mathcal{A}(1^\lambda, \mathsf{pk}), \mathcal{V}(\mathsf{pk}) \rangle = 1\end{matrix}\right] \leq \kappa + \varepsilon(\lambda)\,,$$

for a given security parameter λ, and some negligible function $\varepsilon(\lambda)$.

- **Zero-knowledge**
  The zero knowledge property guarantees that the Verifier $\mathcal{V}$ extracts no information beyond the fact that the statement claimend by the Prover $\mathcal{P}$ is true.

  The idea of zero-knowledge property is that whatever the Verifier $\mathcal{V}$ learns from the execution of the protocol, he could have learned by himself without any intervention from the Prover. In other words, any Verifier $\mathcal{V}$ can use a simulator (a non-interactive algorithm) of the scheme that outputs, in polynomial time, a transcript indistinguishable from a real interaction between the Prover and the honest Verifier (who is following the proof properly (see Definition 26)) in question.

  This property is what makes the protocol zero-knowledge.

Definition 26 ((computational) Honest-Verifier Zero-Knowledge HVZK) *An identification scheme $(\textsc{KeyGen}, \mathcal{P}, \mathcal{V})$ is called HVZK if there exists a Simulator $\mathcal{S}$ (a probabilistic polynomial time algorithm PPT) that outputs transcripts that are computationally indistinguishable from transcripts of honest executions of the protocol. This implies that, for any PPT adversary $\mathcal{A}$, we have:*

$$Pr\big[1 \leftarrow \mathcal{A}\big(\mathsf{sk}, \mathsf{pk}, \langle \mathcal{P}(\mathsf{sk}), \mathcal{V}(\mathsf{pk})\rangle\big)\big] \approx Pr\big[1 \leftarrow \mathcal{A}\big(\mathsf{sk}, \mathsf{pk}, \mathcal{S}(\mathsf{pk})\big)\big]$$

where $\langle \mathcal{P}(\mathsf{sk}), \mathcal{V}(\mathsf{pk})\rangle$ stands for a valid transcript of $\mathcal{V}$ interacting with an honest Prover $\mathcal{P}$.

CHAPTER 5

Digital Signature Schemes (DSS)

Contents

5.1 Introduction

This chapter focuses on Digital Signature Schemes (DSS), which are public-key cryptographic primitives generally, that provide the authenticity of electronic messages and documents.In the physical world, traditional handwritten signatures or stamped seals on typed documents associate a specific sender to the content of the document.

Similarly, in a digital medium, digital signature schemes guarantee validation to the receiver that the document was sent by the claimed signer. When properly implemented, digitally signed messages may be more effective and resistant to forgery than the handwritten signatures.

Digital Signatures based on symmetric/asymmetric cryptography

A digital signature scheme (DSS) facilitate validation and authentication of electronic documents, and allows a digital identification for users.

The digitally signing technique is employed in two ways, namely DSS based on symmetric (private-key) cryptography and DSS based on asymmetric (public-key) cryptography. Symmetric based signatures involve only a single private key shared between the signer and the receiver. The private key is used at the same time to sign a document (by the signer) and also to verify the signature (by the receiver). While asymmetric based signatures use a pair of different keys: a private key used to produce the signature is kept as secret and never shared, a public key used to verify the message authenticity is freely shared with

any user.

An asymmetric based signature scheme is secure against message forgery by any user who does not have the private key used to sign the document. But, when using symmetric based signature scheme, both of the signer and the verifier share the same private key thus, the verifier can forge any message since he holds the private key that produces the signature.

Therefore, public-key based signature schemes are more commonly used and have various advantages over the private-key based signature schemes.However, this work is limited to public-key based signature schemes only.

Digital Signature Algorithm

As mentioned previously, digital signature schemes (
DSS) are typically based on public-key schemes and use a pair of public key and a private key.

A digital signature scheme performs essentially two separate algorithms:

- **A private signing algorithm** authorizing an entity, namely a signer, to securely sign an electronic document using a private key which is referred to as the signing key. The signing algorithm produces a digital signature.
- **A public verification algorithm** allowing an entity, namely a receiver of the signature, to verify the authenticity of the message using the corresponding public key which is referred to as the verification key. The verification algorithm outputs a boolean value to express validity or invalidity of the signature.

In order to obtain an efficient scheme, both of signing and verification algorithms must be relatively fast and not too complex in terms of computation.

The properties of a DSS

When properly implemented, a digital signature schemeDSSensures several properties (detailed in Section 5.2) that make the scheme effective.

Firstly, a DSS bind the signer to the signed document or message, so the receiver is assured that the document is indeed signed by the claimed signer (authenticity property). Moreover, every document has its unique signature which cannot be moved and used to sign other documents (uniqueness property).

Secondly, once a document has been digitally signed, it is not possible to modify the content of the document and the signature remains valid (integrity protection).

Finally, a digital signature scheme prevents an entity (generally the signer) from denying consent and the signature of the content of the signed document (non-repudiation property).

The relation between an IDS and a DSS

As stated previously, an Identification Scheme IDS is an interactive protocol between a Prover $\mathcal{P}$ and a Verifier $\mathcal{V}$ allowing $\mathcal{P}$ to prove its identity to $\mathcal{V}$. $\mathcal{P}$ is able to prove to $\mathcal{V}$ its identity, but $\mathcal{V}$ is not capable of proving to an another entity.

A Digital Signature Scheme (DSS) is a non-interactive scheme nearly equivalent to an IDS whereby $\mathcal{P}$ (or the signer) is able to $\mathcal{V}$ (or the receiver) its identity, but $\mathcal{P}$ is not capable even to prove it to himself.

The difference between an identification and a signature scheme can be shown when the Verifier $\mathcal{V}$ tries to prove the Prover's identity to a third party. In a identification scheme, $\mathcal{V}$ is able to establish a reasonable transcript of a fictitious conversation by properly choosing the challenges and the answers of the transcript, although such scenario is not possible in a digital signature scheme. $\mathcal{V}$ cannot create a credible transcript.

However, signature schemes provide signatures (and thus, identifications) that can be verified anytime, while identification schemes provide identifications that should be verified immediately during the execution of the protocol.
In some cases, it is possible to transform an identification scheme into a signature scheme by applying a standard technique detailed later.

5.2 Formal definitions and general structure

The most-known digital signature schemes are the public-key based schemes. This Section describes how a digital signature schemes (DSS) based on asymmetric cryptography work, and provides formal definitions and properties of such schemes.

General structure of Digital Signature Scheme DSS

Technically speaking, a public-key based signature scheme consists basically of three components: a key and parameters generation algorithm, a signing algorithm, and a verification algorithm.

Protocol steps of a DSS

A basic digital signature of a message m can be summarized by the following steps:

- The two participants in the digital signature scheme (DSS), the signer (or the Prover $\mathcal{P}$) and the receiver (or the Verifier $\mathcal{V}$), generate each its own public/secret key pair denoted by $(pk/\mathsf{sk})_{\mathcal{P}}$, respectively $(\mathsf{pk}/\mathsf{sk})_{\mathcal{V}}$.
  The secret (private) key sk is used for signing the message m, while the public key pk is used to verify the signature.
- The signer (Prover $\mathcal{P}$) encrypts the message m, or more generally a hashed value of m, using his secret key $\mathsf{sk}_{\mathcal{P}}$ and a signing algorithm. This latter outputs a digital signature of m that is sent to the receiver (Verifier $\mathcal{V}$) along with the original message m.

- The receiver feeds the original message m, its corresponding digital signature, and the public-key of the signer $\mathsf{pk}_{\mathcal{P}}$ to a verification algorithm. This latter outputs a boleean value which indicates the acceptance or rejection of the digital signature.
  When using a hashed value of the original message m instead of m, the verifier must compute first a hashed value of m using the same public hash function utilized by the signer.

Formal definition of a DSS

The signing/verification process for a digital signature scheme described above can be formulated as follows:

Definition 27 (General structure) *Let $\lambda \in \mathbb{N}$ be a security parameter.*
A digital signature scheme DSS consists of three probabilistic polynomial time (PPT) algorithms $DSS(1^\lambda) = (\textsc{KeyGen}, \textsc{sign}, \textsc{Verify})$ such that:

- *the key generation* KeyGen *algorithm takes as input a security parameter λ and returns a public and secret key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \textsc{KeyGen}(1^\lambda)$,*
- *the signing* sign *algorithm takes as input the secret key* sk *and the message m, then produces a digital signature σ of the message m,*
- *the* Verify *deterministic algorithm takes as input the public key* pk*, the message m, and the signature σ.* Verify *outputs a boolean value b with $b = 1$ signaling the signature validation, and $b = 0$ the rejection.*

Importance of a Digital Signature Scheme DSS

Among various cryptographic schemes, a public-key based digital signature scheme yields information security and added guarantees to the origin, identity, and a signer's approval of an electronic message or document.
The advantages of signing a message digitally is presented below:

- **Message authentication**
  When the receiving party verifies and accepts the digital signature using the public-key of the sender, he is then convinced that signature has been signed by sender who is the only owner of the corresponding secret-key sk. This authenticates the sender as the legitimate signer.
- **Integrity**
  When an adversary tries to modify a digitally signed message, the receiver is able to detect the changes and then revokes the signature. Note that, in most cryptographic signing algorithms, it is computationally impossible to change a message and its digital signature to obtain a new message with valid signature.
- **Non-repudiation**
  When signing a message digitally, sufficient evidence exists to prevent the signer from denying later time signing the message. Non-repudiation property is assured by a DSS since that each signature is binded to a unique signer who possess the signature key (the secret-key).

5.3 Attacks against Digital Signature Schemes DSSs

Digital signature schemes are vulnerable to different attacks where an outside adversary attempts to intervene with the transmission of an electronic document or message.

5.3.1 Basic attacks scenarios

An adversary $\mathcal{A}$ is capable of attacking a legitimate signer's digital signature if $\mathcal{A}$ succeeds to launch, with non-negligible probability, any of the following attacks models (listed in ascending order of hardness) :

- **A key-only attack** where an adversary $\mathcal{A}$ is given only the public-key pk of the signer. As mentioned earlier, the public-key serves as the verification key in a digital signature protocol.
- **Message attacks** in which an adversary has access to examine in advance signatures that correspond to known messages or furthermore to chosen messages. Message attacks are typically subdivided into the following classes:
  - **Known-message attack,** the adversary $\mathcal{A}$ is able to obtain valid signatures for a set of messages which are known only and not chosen by adversary.
  - **Generic chosen-message attack,** the adversary $\mathcal{A}$ chooses a variety of messages before trying to attack the signer's scheme. Then $\mathcal{A}$ proceeds by asking the signer for a valid signature of each chosen message. Such attack is generic because the messages may be 'randomly' chosen and independently of the signer's public-key pk. Therefore, a generic chosen-message attack can be used against any signer.
  - **Directed chosen-message attack,** such attack is nearly the same as the generic chosen-message attack (listed above), with the only distinction that the adversary $\mathcal{A}$ chooses the set of messages for which $\mathcal{A}$ wants to obtain signatures from the signer after knowing the signer's public-key but before obtaining any valid signature.
    Note that, both of the generic and directed chosen-message attacks are non-adaptive attacks.
  - **Adaptive chosen-message attack,** the adversary $\mathcal{A}$ is able to demand signatures of chosen messages that rely on the signer's public key, and most importantly $\mathcal{A}$ can adeptly request signatures of messages of his choice that depend on the signatures of previously chosen messages

  In principle, the most difficult attack to achieve is the adaptive chosen-message attack. In such type of attack, an adversary given access to several message/signature pairs can possibly derive a pattern and then forge a signature. Thus, a digital signature scheme must be designed in a way to be secure against the adaptive chosen-message attacks.

5.3.2 Goal of an adversary

An adversary $\mathcal{A}$ performs a successful digital signature forgery when $\mathcal{A}$ is able to generate a valid signature of a message m and deceive the receiver as if m is signed by a legitimate signer.
The different goals of an adversary to probabilistically forge a signature can be classified as follows ((listed in descending order of hardness)):

- **A total break,** the adversary $\mathcal{A}$ is able to either determine the secret-key that is the signing key of the scheme.
  Thus, the scheme is totally breakable.

- **A universal forgery attack,** the adversary $\mathcal{A}$ efficiently constructs a signing algorithm identical to the valid signing algorithm of the signer. $\mathcal{A}$ is then able to output a valid signature for any given message.
  Thus, the scheme is universally forgeable.

- **A selective forgery attack,** the adversary $\mathcal{A}$ is capable of producing a valid signature for a given target message chosen by $\mathcal{A}$ prior to the attack.
  Thus, the scheme is selectively forgeable.

- **An existential forgery** the adversary $\mathcal{A}$ is capable of producing some valid signatures of messages. In such attacks, $\mathcal{A}$ needs to have no control over the messages, and the content of the messages may be irrelevant; as long as the messages/signatures pairs are valid, then $\mathcal{A}$ succeeds the forgery.
  Thus, the scheme is existentially forgeable.

Basically, a secure digital signature secure must not even be existentially forgeable.

5.3.3 Security notions for digital signature schemes

The security notions for digital signature schemes (DSSs) manifest as combinations of attack scenarios (5.3.1) and security goals (Section 5.3.2). A DSS is considered secure if an adversary is unable to achieve the weakest goal when performing the strongest attack scenario.
Therefore, the standard security notion for digital signature scheme is namely existential unforgeability against adaptive chosen message attacks.

EUF-CMA security

In this section, an informal definition of the EUF-CMA security is presented using an experiment denoted by $\mathrm{Exp}^{\text{EUF-CMA}}_{\mathrm{DSS}(1^\lambda)}(\mathcal{A})$ that the adversary $\mathcal{A}$ should follow to break a digitally signed message.
Let λ be the security parameter of the scheme $\mathrm{DSS}(1^\lambda) = (\textsc{KeyGen}, \textsc{sign}, \textsc{Verify})$.
An adversary $\mathcal{A}$ proceeds the EUF-CMA game $\mathrm{Exp}^{\text{EUF-CMA}}_{\mathrm{DSS}(1^\lambda)}(\mathcal{A})$ as follows:

- **Setup.** Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \textsc{KeyGen}(1^\lambda)$ be the key pair of the scheme, the public key pk is then handed to the adversary $\mathcal{A}$.

- **Queries.** The adversary $\mathcal{A}$ queries for signatures (at most k times) on a list Q of chosen messages $Q = (m_1, \cdots, m_k)$, for some $k \in \mathbb{N}^*$ (Adaptive chosen-messages).
  Given access to a signing oracle $\textsc{sign}$, $\mathcal{A}$ obtains corresponding signatures $(\sigma_1, \cdots, \sigma_k)$ to the messages $m_i s$. The adversary's query m_i may adeptly depend on signatures of previous queries $m_1, \ldots, m_{i-1}$ (Adaptive chosen-message attack).

- **(Existential) Forgery.** The adversary $\mathcal{A}$ must output a message/signature pair (m^*, σ^*) such that:

1. the message m^* is not one of the previously queried messages $m_1, \ldots, m_k$, it means that $m^* \notin Q$.
2. the message/signature pair (m^*, σ^*) constitute a valid forgery. The VERIFY algorithm $\mathcal{V}f$ of the signature scheme accepts σ^* as a valid signature of m^* which means that $1 \leftarrow \mathcal{V}f(\mathsf{pk}, m^*, \sigma^*)$.

The scheme is said to be EUF-CMA secure if the probability that $\mathcal{A}$ wins the game $\mathrm{Exp}^{\text{EUF-CMA}}_{\mathrm{DSS}(1^\lambda)}(\mathcal{A})$ described above is negligible. More formally:

Definition 28 (EUF-CMA security) *A digital signature scheme $DSS(1^\lambda) = \big(\text{KEYGEN}, \text{SIGN}, \text{VERIFY}\big)$ is existentially unforgeable under chosen-message attacks if the advantage of any probabilistic polynomial time PPT adversary $\mathcal{A}$ running at most k queries to sign in $Exp^{\text{EUF-CMA}}_{DSS(1^\lambda)}(\mathcal{A})$ is negligible in λ:*

$$Adv^{\text{EUF-CMA}}_{\mathcal{A}} \;=\; Pr\Big[\; Succ^{\text{EUF-CMA}}_{\mathcal{A}}\Big(\; Exp^{\text{EUF-CMA}}_{DSS(1^\lambda)}(\mathcal{A}) \;\Big)\Big] \;=\; negl(\lambda),$$

where λ is the security parameter of the scheme.

5.4 The Fiat-Shamir Transform

This Section describes a well-known technique due to A. Fiat and A. Shamir, namely the Fiat-Shamir FS paradigm [35], for transforming a secure Zero-Knowledge Identification Scheme $\mathrm{ZK\text{-}IDS}$ into a Digital Signature Scheme DSS.

The basic idea behind this technique is to replace the random pick of the Verifier $\mathcal{V}$ with the output of some deterministic hash function. Therefore, the interaction between a Prover $\mathcal{P}$ and a Verifier $\mathcal{V}$ is omitted and replaced by a non-interactive proof of knowledge.

The Fiat-Shamir paradigm

The FS transform was originally designed to convert a three-move identification scheme into an existentially secure digital signature scheme against adaptive chosen-message attacks.

Recall the definition of an identification scheme given in Section 4.3.
Let $\mathrm{IDS} = \big(\text{COMMIT}, \text{CHALLENGE}, \text{PROVE}, \text{VERIFY}\big)$ be a canonical three-move identification protocol that is summarized by the following steps:

- the COMMIT algorithm, executed by the Prover $\mathcal{P}$, on input a pair of secret and public key $(\mathsf{sk}, \mathsf{pk})$ outputs the initial commitment com and a state St,
- the CHALLENGE algorithm, executed by the Verifier $\mathcal{V}$, outputs the first challenge ch from the challenge set ChSet,
- the PROVE algorithm, executed by the Prover $\mathcal{P}$, on input $(\mathsf{sk}, \mathsf{com}, \mathsf{ch}, \mathsf{St})$ returns a response rsp,
- the VERIFY algorithm, executed by the Verifier $\mathcal{V}$, on input the interaction transcript $(\mathsf{pk}, \mathsf{com}, \mathsf{St}, \mathsf{ch}, \mathsf{rsp})$ returns a final decision b, $b = 1$ for acceptance or $b = 0$ for rejection.

The Fiat-Shamir FS transform [35] uses a cryptographic hash function $\mathcal{H}$ as a random oracle to replace the challenge of the Verifier $\mathcal{V}$. The hash function $\mathcal{H}$ must output a hashed value having the same length as the challenge of the identification scheme. Note that, the KEYGEN algorithm that generates the signing and verification key pair $(\mathsf{sk}, \mathsf{pk})$ is the same for both identification and signature schemes.
The FS paradigm can be defined as follows:

Definition 29 (The Fiat-Shamir transform) *Let $IDS = (\text{KEYGEN}, \text{COMMIT}, \text{CHALLENGE}, \text{PROVE}, \text{VERIFY})$ be a three-move canonical identification scheme, and let $\mathcal{H}$ be a cryptographic hash function. The Fiat-Shamir FS method converts IDS into a signature scheme $DSS = (\text{KEYGEN}, \text{SIGN}, \text{VERIFY})$, by proceeding as follows to digitally sign a message m :*

- *$\mathcal{P}$ the Prover (signer) computes, using the* COMMIT *algorithm that takes on input the secret key* sk*, a commitment value* com *and a state* St*,*
- *$\mathcal{P}$ computes,using a publicly known hash function $\mathcal{H}$, the challenge* $\mathsf{ch} \leftarrow \mathcal{H}(\mathsf{com}, m)$*,*
- *$\mathcal{P}$ computes, using the* PROVE *algorithm that takes on input* $(\mathsf{sk}, \mathsf{com}, \mathsf{ch}, \mathsf{St})$*, a response* $\mathsf{rsp} \leftarrow \text{PROVE}(\mathsf{sk}, \mathsf{com}, \mathsf{ch}, \mathsf{St})$*,*
- *$\mathcal{P}$ returns a signature $\sigma = (\mathsf{com}, \mathsf{rsp})$.*

In order to verify the signature, the Verifier $\mathcal{V}$ given the verification key pk*, the message m, and the signature σ proceeds as follows:*

- *$\mathcal{V}$ parses the signature σ as* com *and* rsp*,*
- *$\mathcal{V}$ computes, using the same hash function of the signing algorithm $\mathcal{H}$, the challenge* $\mathsf{ch} \leftarrow \mathcal{H}(\mathsf{com}, m)$*,*
- *$\mathcal{V}$ outputs, using the* VERIFY *algorithm that takes on input the interaction transcript* $(\mathsf{pk}, \mathsf{com}, \mathsf{St}, \mathsf{ch}, \mathsf{rsp})$*, a final decision b, $b = 1$ for acceptance or $b = 0$ for rejection.*

Security of the FS paradigm

The (in)security of the Fiat-Shamir FS transform has been studied by many researchers. Some of the studies have established sufficient and necessary assumptions on the identification schemes to provide secure digital signature schemes [1, 7, 38, 59].

The authors of [59] are the first to prove the security against adaptive chosen- message attacks for signature schemes built from zero-knowledge IDS using the FS transform under some conditions.

The Fiat-Shamir FS transform yields efficient digital signature schemes unforgeable under chosen-message attack in the random oracle model when the underlying identification scheme is secure against impersonation under passive attack [1].

CHAPTER 6

Multivariate cryptography

Contents

6.1 Introduction

Post-quantum cryptography involves public-key cryptosystems that are secure against both classical and quantum attacks. The most promising classes for public key post-quantum cryptographic systems [9] involve code-based, hash-based, lattice-based, multivariate based, and supersingular elliptic curve isogeny.

At present, several international competitions have been launched to standardize quantum-resistant public-key cryptographic schemes.

Multivariate cryptography is a branch of post-quantum cryptography whose security relies on the difficulty of solving systems of multivariate equations over a finite field which is NP-hard [41]. This problem is known as MQ (Multivariate Quadratic equations) when the polynomials are quadratic, otherwise it is known as the PoSSo (Polynomial System Solving) problem [33].

Various multivariate-based scheme have failed. However, some multivariate schemes as UOV [45], LUOV [15] , variants of HFE [25, 55], Rainbow [28], and Gemss [19] could make the basis for quantum-safe schemes.

The HFE family cryptosystems, proposed by J. PATARIN [55], is one of the best known and most studied schemes among all multivariate public-key cryptosystems MPKCs.

It can be used for public-key encryption and also for signatures, but it is generally more efficient for signatures schemes.

The HFE family consists of many variants due to the fact that it is always possible to apply different modifications (namely perturbations), and even to use different finite field ($\mathbb{F}_2$ is mostly used).

Moreover, as we will see next, one can use another degree than 2 for the public multivariate equations.

6.2 Multivariate public-key cryptography

Let $\mathbb{F}_q$ be a finite fiel with q elements, where q is a prime power.
Multivariate cryptography is the analysis of public-key cryptographic schemes based on multivariate polynomials in several variables over a finite field $\mathbb{F}_q$.

The security of multivariate public-key cryprosystems, known as MPKCs, relies on the existence of a specific family of functions that is the "trapdoor one-way functions" (See Section 3.3.3 for basic definitions). The employed trapdoor one-way function is a nonlinear (usually quadratic) polynomial map over a finite field $\mathbb{F}_q$.

Multivariate asymmetric cryptosystems are considered as potential candidates for post-quantum cryptography due to the NP-hardness of the problem of solving systems of multivariate equations [36].

There exists a polynomial-time quantum algorithm, namely Shor's algorithm [**Shor**], that weakens the computational difficulty for solving the mathematical problems behind today's most utilized public-key schemes (RSA, $\mathrm{ElGamal}$, ...).

Then, the advent of quantum computer makes the security of existing public-key schemes potentially vulnerable to efficient quantum attacks.
Therefore, it is important to build new quantum resistant cryptosystems that rely on a class of problems known to be unbreakable by a quantum computer such as NP-hard problems.

Recently, there has been a massive progress in multivariate public key cryptography. Some of the multivariate cryprosystems are not as safe as originally claimed, however, stronger schemes have withstood the cryptanalysis and are still viable.
Multivariate cryptography can be used for public-key encryption and also for signatures, but it is generally more efficient for signatures schemes.

6.2.1 General construction

A multivariate public-key cryptosystem MPKC involves a set $(\mathcal{P})$ of m multivariate polynomials, generally of degree two, in n variables over a finite field $\mathbb{F}_q$ as follows:

$$(\mathcal{P}) := \begin{cases} p_1(x_1,\ldots,x_n) & = \sum\limits_{1\le i\le j\le n} \alpha_{ij}^{(1)} x_i x_j + \sum\limits_{i=1}^{n} \beta_i^{(1)} x_i + \gamma^{(1)} \\ p_2(x_1,\ldots,x_n) & = \sum\limits_{1\le i\le j\le n} \alpha_{ij}^{(2)} x_i x_j + \sum\limits_{i=1}^{n} \beta_i^{(2)} x_i + \gamma^{(2)} \\ & \vdots \\ p_m(x_1,\ldots,x_n) & = \sum\limits_{1\le i\le j\le n} \alpha_{ij}^{(m)} x_i x_j + \sum\limits_{i=1}^{n} \beta_i^{(m)} x_i + \gamma^{(m)} \end{cases}$$

where the coefficients α_{ij}, β_i and γ are in $\mathbb{F}_q$, the field with q elements. Each multivariate polynomial p_ks, for $k \in \{1,\ldots,m\}$, is of a small degree d (in this case d is equal to two).
Note that, the set of polynomials $(\mathcal{P})$ (given above) presents quadratic polynomials $p_i s$.
Next Section provides a more general definition where the polynomials p_is are of any small degree d.

PoSSo problem:

Let $\mathbb{F}_q$ be a finite field of order q, and $\{p_1, p_2, \ldots, , p_m\} \subset \mathbb{F}_q[x_1, x_2, \ldots, x_n]$ be a set of multivariate polynomials.
The Polynomial System Solving (PoSSo) problem is defined as follows:
Input: Given a set of polynomials $\mathcal{P} = \{p_1, p_2, \ldots, p_m\} \subset \mathbb{F}_q[x_1, x_2, \ \ldots\ , x_n]$ of small degree d.
Question: Find - if there exists - a vector $\tilde{x} = (\tilde{x}_1, ..., \tilde{x}_n) \in {\mathbb{F}_q}^n$, such that:

$$\mathcal{P}(\tilde{x}) := \begin{cases} p_1(\tilde{x}_1,\ldots,\tilde{x}_n) & = 0 \\ p_2(\tilde{x}_1,\ldots,\tilde{x}_n) & = 0 \\ \vdots & \\ p_m(\tilde{x}_1,\ldots,\tilde{x}_n) & = 0 \end{cases}$$

The PoSSo problem is proven to be *NP*-Hard [36]. When $d = 2$, the problem is known as the MQ pproblem, where MQ stands for Multivariate Quadratic.

In all multivariate cryptosystems, the public key consists of a set of multivariate polynomials. In practice, a PoSSo-based multivariate scheme is built using an easily invertible map $\mathcal{P}' : {\mathbb{F}_q}^n \mapsto {\mathbb{F}_q}^m$, known as the central map. The structure of $\mathcal{P}'$ is then hidden via two invertible affine maps $\mathcal{S} : {\mathbb{F}_q}^m \mapsto {\mathbb{F}_q}^m$ and $\mathcal{T} : {\mathbb{F}_q}^n \mapsto {\mathbb{F}_q}^n$.

Therefore, the public key $\mathcal{P} = (p_1, \ldots, p_m)$ is the composition, on the left and on the right, $\mathcal{P} = \mathcal{S} \circ \mathcal{P}' \circ \mathcal{T} : {\mathbb{F}_q}^n \mapsto {\mathbb{F}_q}^m$ that is by assumption hard to invert without any information on the trapdoor.
The private key is the triplet of the maps $\mathcal{P}', \mathcal{S},$ and $\mathcal{T}$.

6.2.2 The standard processes

This section summarizes the major types of processes in multivariate cryptography.

Encryption process

For most PoSSo-based multivariate schemes, the encryption step is the same.
The messages belong to the vector sace ${\mathbb{F}_q}^n$, while the encrypted messages belong to the vector space ${\mathbb{F}_q}^m$.
More precisely, in order to encrypt a message $a = (a_1, \ldots, a_n) \in {\mathbb{F}_q}^n$, it suffices to evaluate the public key $\mathcal{P}(a) = \mathcal{S} \circ \mathcal{P}' \circ \mathcal{T}(a) = (p_1, \ldots, p_m)$.
Then, the encrypted message (ciphertext) $c = (c_1, \ldots, c_m) \in {\mathbb{F}_q}^m$ is given by:

$$c = (c_1, \ldots, c_m) = (p_1(a_1, \ldots, a_n), \ldots, p_m(a_1, \ldots, a_m)).$$

The computational complexity of the encryption process depends on the degree d of the public key polynomials.

Assuming that the polynomials p_is are quadratic, then the number of operations required to proceed the encryption thus to evaluate the public key is in:

$$\mathcal{O}(mn^2) \text{ operations.}$$

In fact, the number of monomials of degree $d = 2$ in n variables is given by $\binom{n+d}{d}$. An evaluation of a quadratic polynomial of the public key requires $\binom{n+2}{2}$ multiplications and $\binom{n+2}{2} - 1$ additions at most. Thus, each evaluation involves $\left(2\binom{n+2}{2} - 1\right) \approx n^2$ arithmetic operations.
Therefore, the overall complexity of an evaluation of the public key is given by $\mathcal{O}(mn^2)$ operations.

Decryption process

In order to decrypt a given encrypted message $c \in {\mathbb{F}_q}^m$, one must invert the computation of the ciphertext $c = \mathcal{P}(a)$ corresponding to the plaintext $a \in {\mathbb{F}_q}^n$.

Since that $\mathcal{P}$ is the composition of maps $\mathcal{P} = \mathcal{S} \circ \mathcal{P}' \circ \mathcal{T}$. Therefore, only the entity who possesses the trapdoor information $(\mathcal{P}', \mathcal{S}, \mathcal{T})$ is able to achieve the decryption process:

$$a = \mathcal{P}^{-1}(c) = \mathcal{T}^{-1} \circ \mathcal{P}'^{-1} \circ \mathcal{S}^{-1}(c).$$

One must compute respectively $x = \mathcal{S}^{-1}(c) \in {\mathbb{F}_q}^m$, $y = \mathcal{P}'^{-1}(x) \in {\mathbb{F}_q}^n$, and $a = \mathcal{T}^{-1}(y)$. Note that, the number of public polynomials m must be greater than or equal to the number of variables n in order to have a unique solution.

Signature process

On the contrary of the encryption process, a signature scheme does not require the uniqueness property. In fact, the following condition holds $m \leq n$ for signature schemes in order to guarantee that one can sign any message.

A signature of a given message a is generated using a public hash function $H : \{0,1\}^* \mapsto {\mathbb{F}_q}^m$, and the private key (the trapdoor information).

Similarly to the decryption process, the generation of a signature s employs the inversion of P on the hashed value $h = H(a)$ of the message a:

$$s = \mathcal{P}^{-1}(h) = \mathcal{T}^{-1} \circ \mathcal{P}'^{-1} \circ \mathcal{S}^{-1}(h).$$

One must compute respectively $x = \mathcal{S}^{-1}(h) \in {\mathbb{F}_q}^m$, $y = \mathcal{P}'^{-1}(x) \in {\mathbb{F}_q}^n$, and $s = \mathcal{T}^{-1}(y)$.
The computation of $\mathcal{P}'^{-1}(x)$ yields a possible (among several) pre-image of x under the special map $\mathcal{P}'$.
Since that $m \leq n$, then every message has a signature

Verification process

In order to verify a signature $s \in {\mathbb{F}_q}^n$ of a message a, one must first compute the hashed value $h = H(a) \in {\mathbb{F}_q}^m$, then evaluate the public map $\mathcal{P}$ at the value s to get $h' = \mathcal{P}(s)$.
The signature is valid if and only if $h' = h$, otherwise the signature is rejected.

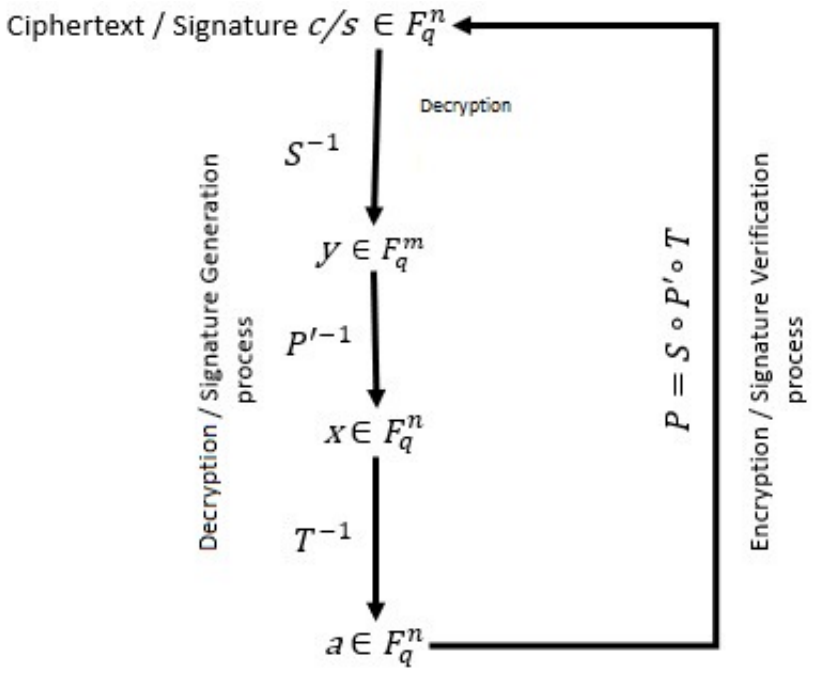

Figure 6.1: The classical processes of Multivariate Cryptography

6.2.3 Major attacks on MPKCs

The standard techniques generally utilized to attack multivariate cryptosystems can be classified into three main classes.
Most of the methods intend to analyze the difficulty of the PoSSo problem or to recover the secret keys.
Next Section provides briefly the major attacks on a multivariate public key crypstosystem MPKC;

Direct attacks

The direct attack, also known as the Gröbner basis attack, aims to solve the public system $y = \mathcal{P}(x)$ directly by finding a solution $x \in {\mathbb{F}_q}^n$ for the PoSSo problem.
The most common technique for solving a set of multivariate polynomials is to use Gröbner basis computations (see Section 7.2) that involve generally the following algorithms: $\mathrm{F4}$ [32], $\mathrm{F5}$ [31], and/or the hybrid approach [11].
A detailed description of such method will be given in next Sections 7.2.

Rank attacks

The main goal of using rank attack is to benefit from the special structure of the central map $\mathcal{P}'$ utilized by a multivariate cryptosystem in order to find (a part of) the private key.

The rank attack is related to the $\mathrm{MinRank}$ problem [23]. The security of various multivariate public-key scheme relies on the hardness of solving $\mathrm{MinRank}$. For instance, this problem appears in the cryptanalysis of several MPKC (such as HFE [46, 55], TTM [39], etc, ...).
By associating to the public (quadratic) polynomials their quadratic forms, the $\mathrm{MinRank}$ problem consist of finding a linear combination of these forms of low rank.
Therefore, it is possible to extract partial information or even extract the private key of the scheme.

Differential attacks

The differential attack consists of studying the effect of inserting differential information input to the public key $\Delta(x, x') = \mathcal{P}(x + x') - \mathcal{P}(x) - \mathcal{P}(x')$ for $x, x' \in {\mathbb{F}_q}^n$.
The differential attack helps exploiting the structure and the behavior of the scheme in order to recover the central map.

6.2.4 Classical trapdoor functions

This Section describes the basic types of central maps $\mathcal{P}'$ usually used in multivariate public-key cryptographic schemes MPKCs.

Triangular systems

An efficient method to obtain an easily invertible central map $\mathcal{P}' = (p'_1, \ldots, p'_m)$ is to employ the triangular structure over a finite field $\mathbb{F}_q$.
Given a system $\mathcal{P}'(x) = y \in {\mathbb{F}_q}^n$ of m equations and n variables. The special structure of $\mathcal{P}'$ is the following:

$$\mathcal{P}'(x) := y \begin{cases} p'_1(x_1, \ldots, x_n) & = y_1 \\ p'_2(x_2, \ldots, x_n) & = y_2 \\ \vdots & \\ p'_{m-1}(x_{m-1}, x_n) & = y_{m-1} \\ p'_m(x_n) & = y_m \end{cases}$$

By the construction given above, the system can be inverted by finding first a solution x_n for the last equation $p'_m(x_n) = y_m$, and next, substitute the value found in the previous equation $p'_{m-1}(x_{m-1}, x_n) = y_{m-1}$ in order to find this time x_{n-1} and so on until the first equation $p'_1(x) = p'_1(x_1 \ldots, x - n) = y_1$.
The main idea of this method can be generalized in order to solve at each step a small number of equations in less number of variables. Proceeding this way enables the inversion of the central map.
This type of construction was used in several number of multivariate schemes such as TTM [52], TTS [70], etc... . However, various attacks were proposed on such schemes to recover the key [39, 69].

Big field type

Unlike the previous approach, the construction method described in this Section involves a bigger Field $\mathbb{L} = \mathbb{F}_{q^n}$ that is an extension of the ground field $\mathbb{F}_q$ of degree n.
More precisely, the Big Field approach uses a morphism $\phi : \mathbb{L} \mapsto {\mathbb{F}_q}^n$ between the extension $\mathbb{F}_{q^n}$ and the corresponding vector space ${\mathbb{F}_q}^n$. Given an easily invertible map $\tilde{\mathcal{P}'} : \mathbb{L} \mapsto \mathbb{L}$, the multivariate polynomial map is a map over ${\mathbb{F}_q}^n$:

$$\mathcal{P}' = \phi \circ \tilde{\mathcal{P}'} \circ \phi^{-1},$$

then, the structure of $\mathcal{P}'$ is then masked by two invertible affine linear maps $\mathcal{S}$ and $\mathcal{T}$ in ${\mathbb{F}_q}^n$.
The Big Field type was used in the C^* of Matsumoto and Imai [51], and then it was attacked later by J. Patarin in [54] and generalized in the HFE [55] cryptosystem.

Unbalanced Oil and Vinegar variants

The Oil and Vinegar method used in the construction of the Unbalanced Oil and Vinegar scheme UOV [45]. Given two integers o and v such that $n = o + v$, $V = \{1, \ldots, v\}$, and $O = \{v+1, \ldots, n\}$. The variables $x_1, \ldots, x_v$ are the vinegar variables and the $x_{v+1}, \ldots, x_n$ are the oil variables. The quadratic polynomials of the central map have the following UOV-form:

$$p'_k = \sum_{(x_i, x_j) \in O \times V} \alpha_{ij}^{(k)} x_i x_j + \sum_{(x_i, x_j) \in V \times V} \beta_{ij}^{(k)} x_i x_j + \sum_{x_i \in O \cup V} \gamma_i^{(k)} x_i + \delta^{(k)},$$

where $1 \leq k \leq m$, and the coefficients α_{ij}, β_{ij}, γ_i and δ are in $\mathbb{F}_q$.
The main goal is to specialize certain variables, in order to obtain linear polynomials. Since that, the vinegar variables are combined quadratically while the oil variables are only combined with vinegar variables in a quadratic way. Therefore, by assigning random values to the vinegar variables, the system of equations becomes linear in the oil variables that can be efficiently solved.

The robustness of this approach is based on the fact that is difficult to distinguish the oil variables from the vinegar variables. Even though, some attacks were introduced when using weak parameters, this type has a good practical resistance to attacks with reasonable parameters.

6.3 HFE

One of the most known multivariate public-key cryptosystem is the Hidden Field Equations HFE, that uses an HFE trapdoor function. Hidden field Equations (HFE) scheme was proposed by J. PATARIN at Eurocrypt [55] to repair the algorithm C* of Matsumoto and Imai [51]. The basic idea of HFE is to hide the special structure of a univariate polynomial F over some finite field (usually $\mathbb{F}_{2^n}$) which allows F to have a quadratic polynomials representation in the small field.

6.3.1 HFE(q,n,D) shape

Let $\mathbb{F} = \mathbb{F}_q$ be a finite field of $q = p^m$ elements for some prime number p, $\mathbb{E} = \mathbb{F}_{q^n}$ its n-th degree extension, and $\phi : \mathbb{E} \to \mathbb{F}^n$ the canonical isomorphism between $\mathbb{E}$ and the corresponding vector space $\mathbb{F}^n$. Given $(\theta_1, \ldots, \theta_n)$ a basis of $\mathbb{E}$ as an $\mathbb{F}$-vector space, we have:

$$\begin{array}{rccc} \phi: & \mathbb{E} = \mathbb{F}_{q^n} & \longrightarrow & \mathbb{F}^n \\ & V = \sum_{i=1}^{n} v_i \theta_i & \longmapsto & (v_1, \ldots, v_n). \end{array}$$

Let $\mathcal{F}^*$ be the following map:

$$\begin{array}{rccc} \mathcal{F}^*: & \mathbb{F}_{q^n} & \longrightarrow & \mathbb{F}_{q^n} \\ & V & \longmapsto & F(V), \end{array}$$

with $F \in \mathbb{E}[X]$ is a univariate polynomial of the special form:

$$F = \sum_{0 \le i \le j \le n}^{q^i + q^j \le D} \alpha_{i,j} X^{q^i + q^j} + \sum_{0 \le i \le n}^{q^i \le D} \beta_i X^{q^i} + \gamma, \tag{6.1}$$

where $\alpha_{i,j},\ \beta_i, \gamma\ \in \mathbb{F}_{q^n}$, and F is of degree at most $D \in \mathbb{N}$. Then, F has the HFE(D) shape that allows to have multivariate quadratic polynomials representation over $\mathbb{F}$ using the map $\mathcal{F} = \phi \circ \mathcal{F}^* \circ \phi^{-1}$:

$$\begin{array}{rccc} \mathcal{F}: & \mathbb{F}_q^n & \longrightarrow & \mathbb{F}_q^n \\ & (v_1, \ldots, v_n) & \longmapsto & (f_1(v_1, \ldots, v_n), \ldots, f_n(v_1, \ldots, v_n)), \end{array}$$

with the quadratic polynomials $(f_1, \ldots, f_n) \in (\mathbb{F}_q[x_1, \ldots, x_n])^n$ such that:

$$F\big(\phi^{-1}(x_1, \ldots, x_n)\big) = \phi^{-1}(f_1, \ldots, f_n)$$

$$F\Big(\sum_{i=1}^{n} \theta_i x_i\Big) = \sum_{i=1}^{n} \theta_i f_i$$

.

6.3.2 HFE problem

Basically, $\mathcal{F}^*$ is chosen to be an easily invertible and evaluated map. Using the canonical isomorphism ϕ, the map $\mathcal{F}^*$ can be transformed into a quadratic map $\mathcal{F} = \phi \circ \mathcal{F}^* \circ \phi^{-1}$. Thus, F can be written as a set of n quadratic polynomials $(f_1, \ldots, f_n)$ in n variables $(x_1, \ldots, x_n)$ over $\mathbb{F}$.
In order to build a PoSSo-based cryptosystem, the original structure of $\mathcal{F}$ must be hidden since it is possible to find solutions of $F(x) = a,\ a \in \mathbb{F}_{q^n}$ in polynomial time. To do so, one uses two invertible affine maps

$$\mathcal{S}, \mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n.$$

Therefore, the public key consists of

$$\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} = \mathcal{S} \circ \phi \circ \mathcal{F}^* \circ \phi^{-1} \circ \mathcal{T}, \tag{6.2}$$

and the secret key that yields the inversion of the public key is given by $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{F}^*$.
Thus, it is difficult to compute the inverse of $\mathcal{P}$ when its decomposition remains secret.
HFE is one of the most studied algorithm in cryptography. It can be used for authentication, encryption and also signature purposes.

6.3.3 HFE perturbations

The previous sections present the basic HFE. This latter, namely "Nude HFE", is a basic version of degree 2.
This section introduces several variations that can increase the security of HFE or its efficiency [55].

6.3.3.1 The minus modifier HFE-

This variant consists of omitting some multivariate polynomials from the public key of an HFE-based scheme $\mathcal{P} = (p_1, \ldots, p_n)$. It is named HFE- due to the act of hiding public equations which leads to an under determined system. Hence, this modifications is more often used for signatures schemes.
More precisely, HFE- involves using the following projection:

$$\pi : \qquad \mathbb{F}^n \longrightarrow \qquad \mathbb{F}^{n-s},$$

where $s \in \mathbb{N}^*$ denotes the number of removed polynomials. Thus, the construction of the public key comes as $\mathcal{P} = \pi \circ \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ for the original HFE-based scheme. Hence, the public key is seen as $n - s$ quadratic multivariate polynomials in n variables.

6.3.3.2 The vinegar modifiers HFEv and HFEw

While the first modification discussed earlier affects the public key, this section presents a variation which affects the structure of the private HFE shaped polynomial.
It consists of adding some extra variables, namely vinegar variables $(z_1, \ldots, z_v) \in \mathbb{F}^v$, where $v \in \mathbb{N}$ denotes

the number of vinegar variables. The secret HFE shaped polynomial is now a function of these vinegar variables. Its multivariate representation over the ground field should remain quadratic in the original HFE case.
Note that, there are two classifications of vinegar variants:

- External vinegar modifier v, where the extra variables $(z_1, \ldots, z_v) \in \mathbb{F}^v$ are randomly chosen.
- Internal vinegar modifier w, where each extra variable $z_i \in \mathbb{F},$ for $i \in \{1, \ldots, v\}$ is a linear combination of the original variables $(x_1, \ldots, x_n) \in \mathbb{F}^n$.

Recall the special form of the the HFE shaped polynomial

$$F(X) = \sum_{0 \leq i \leq j \leq n}^{q^i + q^j \leq D} \alpha_{i,j} X^{q^i + q^j} + \sum_{0 \leq i \leq n}^{q^i \leq D} \beta_i X^{q^i} + \gamma,$$

where $\alpha_{i,j},\ \beta_i, \gamma\ \in \mathbb{F}_{q^n}$.
HFEv (respectively HFEw) consists of replacing F with a complicated map from $\mathbb{F}_{q^n} \times \mathbb{F}^v$ to $\mathbb{F}_{q^n}$:

$$F(X, z_1, \ldots, z_v) = \sum_{0 \leq i \leq j \leq n}^{q^i + q^j \leq D} \alpha_{i,j} X^{q^i + q^j} + \sum_{0 \leq i \leq n}^{q^i \leq D} \beta_i(z_1, \ldots, z_v) X^{q^i} + \gamma(z_1, \ldots, z_v),$$

where $\alpha_{i,j} \in \mathbb{F}_{q^n}$, and $\beta_i, \gamma : \mathbb{F}^v \longrightarrow \mathbb{F}_{q^n}$ are respectively a linear and a quadratic random maps.

CHAPTER 7

Polynomial systems solving

Contents

7.1 Introduction

This Section provides an introduction to the Gröbner basis technique that is a classical method for solving systems of polynomial equations in several variables. The polynomial systems solving problem, known as PoSSo (see Section 6.2.1), arises widely in many scientific areas. Therefore, it is important to determine the complexity of this problem.
We review known algorithms for computing Gröbner bases as well as for solving algebraic systems of equations. In order to define their complexity bounds, essential properties of such systems like the **degree of regularity** (d_{reg}) will be detailed.

Recall first the PoSSo problem: Let $\mathbb{K} = \mathbb{F}_q$ be a finite field of order q, and $\{f_1, f_2, \ldots, , f_m\} \subset \mathbb{K}[x_1, x_2, \ldots, x_n]$ be a set of multivariate polynomials.
The Polynomial System Solving (PoSSo) problem is defined as follows:
Input: Given a set of polynomials $\{f_1, f_2, \ldots, f_m\}$ in $\mathbb{K}[x_1, x_2, \ ... \ , x_n]$ of small degree d.
Question: Find - if there exists - a vector $\tilde{x} = (\tilde{x}_1, ..., \tilde{x}_n) \in {\mathbb{F}_q}^n$, such that:

$$\begin{cases} f_1(\tilde{x}_1, \ldots, \tilde{x}_n) &= 0 \\ f_2(\tilde{x}_1, \ldots, \tilde{x}_n) &= 0 \\ \quad \vdots & \\ f_m(\tilde{x}_1, \ldots, \tilde{x}_n) &= 0 \end{cases}$$

The PoSSo problem, proven to be *NP-Hard* [**michael1979computers**], is a fundamental problem which regularly occurs in the cryptanalysis of most cryptographic problems. Hence, several methodologies have been developed for its resolution.

7.2 Computing a Gröbner basis

The classical technique for solving the PoSSo problem and for studying the solutions of a system of multivariate polynomials involves Gröbner basis computations. Specific notations and definitions related to the Gröbner basis solving tool will detailed in next Sections. We refer the readers to [2, 11, 26] for more details.

7.2.1 Monomial orderings

Let $\mathbb{K} = \mathbb{F}_q$ denote a finite field of characteristic p, and I an ideal of the polynomial ring in n variables $\mathbb{K}[x_1, x_2, \ldots, x_n]$ over $\mathbb{K}$.
Basically, solving a system of m polynomials $\{f_1, f_2, \ldots, f_m\}$ in $\mathbb{K}[x_1, x_2, \ldots, x_n]$ consists of finding the *zero-locus* of the system that is the set of points $x = (x_1, x_2, \ldots, x_n) \in \mathbb{K}^n$ on which the functions f_is simultaneously vanish.
In algebraic geometry, the set of roots corresponding to a system of polynomials constitutes an algebraic variety.

Definition 30 *(Algebraic variety)*
Let $\mathbb{L} \supseteq \mathbb{K}$ be a field extension of $\mathbb{K}$.
Given an ideal $I = \langle f_1, \ldots, f_m \rangle$, the algebraic variety $V_{\mathbb{L}}(I)$ is the set of all common roots in $\mathbb{L}^n$ on which the elements of the ideal I vanish:

$$V_{\mathbb{L}}(I) \;=\; V_{\mathbb{L}}(f_1, \ldots, f_m) \;=\; \{(z_1, \ldots, z_n) \in \mathbb{L}^n \mid f_i(z_1, \ldots, z_n) = 0,\ \forall\, 1 \leq i \leq m\}\,.$$

In multivariate cryptography, the most interesting systems of polynomials are the ones that have a finite number of solutions.

Definition 31 *(zero-dimensional ideal)*
An ideal I is zero-dimensional if its corresponding variety $V(I)$ is finite.

The degree of an ideal I, denoted by *deg(I)*, is the cardinal of the associated variety V where the elements are counted with multiplicities.
Given a system of polynomial equations, the Gröbner basis of the ideal I generated by the polynomials allows the study of the algebraic variety $V(I)$. Such solving tool requires a specific monomial order.

Definition 32 *(Monomial order)*
A monomial ordering, also known as an admissible ordering, is a relation $\prec$ on the set of all monomials M_n, satisfying the following properties:

1. *Total order: $\forall m_1, m_2 \in M_n,\ if\ m_1 \neq m_2$ then $m_1 \prec m_2$ or $m_2 \prec m_1$.*
2. *Respecting-multiplication: $\forall m_1, m_2, m_3 \in M_n,\ if\ m_1 \prec m_2$ then $m_1 m_3 \prec m_2 m_3$.*
3. *Well-ordering: every non-empty subset of monomials has a smallest element with respect to the ordering $\prec$.*

Considering the order on the variables $x_1 \succ x_2 \succ \ldots \succ x_n$, the most used orders, among several monomial orderings, are the following:

- Lexicographic order lex: also known as dictionary order due to its similarity with the alphabetical order used in lexicography for dictionaries.

$$x_1^{\alpha_1}x_2^{\alpha_2}...x_n^{\alpha_n} \prec_{lex} x_1^{\beta_1}x_2^{\beta_2}...x_n^{\beta_n} \quad \text{if there exists} \quad 1 \leq i \leq n \quad \text{such that}$$
$$\alpha_j = \beta_j \quad for \quad 1 \leq j < i \quad \text{and} \quad \alpha_i < \beta_i.$$

- Graded reverse lexicographic order $\mathrm{grevlex}$: it compares the total degree first (sum of all exponents), and when a tie appears, it applies a reverse lexicographic order

$$x_1^{\alpha_1}x_2^{\alpha_2}...x_n^{\alpha_n} \prec_{grevlex} x_1^{\beta_1}x_2^{\beta_2}...x_n^{\beta_n} \quad \text{if} \quad \sum_{i=1}^{n}\alpha_i < \sum_{i=1}^{n}\beta_i$$

$$\text{or} \sum_{i=1}^{n}\alpha_i = \sum_{i=1}^{n}\beta_i \quad \text{and there exists } i, \quad 1 \leq i \leq n \quad \text{such that} \quad \forall j,\ i < j \leq n,$$
$$\alpha_j = \beta_j \quad and \quad \alpha_i > \beta_i.$$

When a specific monomial order is chosen, it is then possible to sort the terms of a given polynomial.
The leading monomial of a polynomial f, denoted by $LM_{\prec}(f)$, is the largest (monic) monomial in the set of all monomials occurring in f. The leading term, denoted by $LT_{\prec}(f)$, is the product of $LM_{\prec}(f)$ and its corresponding coefficient c that is $c \times LM_{\prec}(f)$.
Moreover, $LT_{\prec}(I)$ is the set of all leading terms of the elements of an ideal I.

7.2.2 Gröbner basis

Let I be an ideal $I \subseteq \mathbb{K}[x_1, \ldots, x_n]$, and $\prec$ a specific monomial order.

Definition 33 *(Gröbner basis [17])*

A finite set of polynomials $G = \{g_1, ..., g_s\} \subset I$ is a Gröbner basis of I, with respect to a monomial order $\prec$, if $LT_{\prec}(I)$ the set of all leading terms of I is generated by the leading terms of the g_is:

$$\langle LT_{\prec}(g_1), ..., LT_{\prec}(g_s)\rangle \ = \ \langle LT_{\prec}(I)\rangle$$

In other words, $\forall f \in I,$ there exists $g \in G$ such that $LT_{\prec}(f)$ is divisible by $LT_{\prec}(g)$.

Given an ideal I and with respect to a monomial order, a Gröbner basis G of I always exists, but it is not unique. For a given monomial order, the uniqueness can be guaranteed by defining the so-called $\mathrm{reduced}$ Gröbner basis.

Definition 34 *(Reduced Gröbner basis)*
Let $I \subseteq \mathbb{K}[x_1, \ldots, x_n]$. A finite subset $G = \{g_1, ..., g_s\} \subset I$ is a reduced Gröbner basis of I if

1. *G forms a Gröbner basis of I*
2. *The corresponding coefficient of $LT(g)$ is equal to 1, for every $g \in G$*
3. *No monomial of g lies in $\langle LT(G-g)\rangle$.*

Buchberger's algorithm is the oldest method of transforming a set generators G of an ideal I into a Gröbner basis [17]. This latter generalizes the *Euclidean* algorithm for GCD computation and the *Gaussian* elimination for linear systems. Numerous methods were developed to improve this algorithm like the **Faugère**'s $F4$ and $F5$ algorithms [31, 32].
Below, an alternative algorithm to compute a Gröbner basis that applies the *Gaussian elimination* on the so-called matrices of *Macaulay* will be introduced.

Let $M(d)$ be the set of all monomials of degree less than or equal to d and $\mu_i^{\leq d}$ the i-th element of $M(d)$ with a descending order according to $\prec$. Let $\ell_d = \binom{n+d}{d}$ be the cardinal of $M(d)$. Finally, for $f \in \mathbb{K}[x_1, \ldots, x_n]$, let $Coeff(f, \mu_i^{\leq d})$ be the coefficient of f associated to $\mu_i^{\leq d}$.

Definition 35 *(Macaulay's matrix)*
Given a monomial order $\prec$, an integer $D \succ 0$, and a set of polynomials $\{f_1, ..., f_m\} \subset \mathbb{K}[x_1, \ldots, x_n]$, each of degree d_i, such that $d_1 \leq d_2 \leq ... \leq d_m$ respectively, the Macaulay's matrix of degree D, denoted by $Mac_D(\{f_1, \ldots, f_m\})$, is the matrix containing the coefficients of the polynomials $\mu_k^{\leq (D-d_i)} f_i$ of degree less than or equal to D for every i, $1 \leq i \leq m$ and for every k, $1 \leq k \leq \ell_{(D-d_i)}$, where $\ell_{(D-d_i)}$ denoted the cardinal of $M(D-d_i)$.

$$
\begin{array}{c}
 \\
\mu_1^{\leq (D-d_1)} f_1 \\
\mu_2^{\leq (D-d_1)} f_1 \\
\vdots \\
\mu_{\ell_D - d_1}^{\leq (D-d_1)} f_1 \\
\vdots \\
\vdots \\
\mu_1^{\leq (D-d_m)} f_m \\
\mu_2^{\leq (D-d_m)} f_m \\
\vdots \\
\mu_{\ell_D - d_m}^{\leq (D-d_m)} f_m
\end{array}
\begin{array}{c}
\begin{array}{cccc}
\mu_1^{\leq D} \succ & \mu_2^{\leq D} \succ & ldots \succ & \mu_{\ell_D}^{\leq D}
\end{array} \\
\left(\begin{array}{cccc}
\cdots & \cdots & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots \\
\cdots & Coeff(\, \mu_k^{\leq (D-d_j)} f_j,\ \mu_i^{\leq D}) & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots
\end{array}\right)
\end{array}
$$

It was shown in D. Lazard [49, 50], that it is possible to deduce a Gröbner basis of $I = \langle f_1, \ldots, f_m \rangle$ by considering the $\mathrm{Echelon\ form}$ (where the only authorized elementary operations is the combination rows) of the $\mathrm{Macaulay}$'s matrix.
The system of polynomials can be solved using the Gröbner basis technique by computing, successively,

the Echelon forms of the Macaulay's matrix. The $F4$ [32] utilizes the same principles of Buchberger's algorithm, and when the degree D up to which one should go, the algorithm $F4$ provides a clear picture on the computational complexity of the Gröbner basis technique.
Therefore,in order to determine the complexity of such algorithms, it is important to introduce the notion of degree of regularity.
Intuitively, d_{reg} is the minimal degree for which a set of polynomials of degree $d = d_{reg}$ can form a Gröbner basis. Usually, it is defined for homogeneous polynomials, but it can be extended for the other case.

Definition 36 *(degree of regularity)*
Given a zero-dimensional homogeneous ideal $I \subseteq \mathbb{K}[x_1, x_2, \ldots, x_n]$, the degree of regularity of I is the smallest integer $d > 0$ such that the polynomials of degree d in I constitutes a generating set of all monomials of degree d in n variables, i.e.

$$d_{reg}(I) \;=\; \min\left\{ d \in \mathbb{N} \mid dim_{\mathbb{K}}(\{f \in I \;:\; deg(f) = d\}) \;=\; \binom{n+d-1}{d} \right\},$$

where, $dim_{\mathbb{K}}$ is seen as the dimension of a $\mathbb{K}$-vector space.

As stated before, the degree of regularity involves the dimension of the vector space $I_d = \{f \in I \;:\; deg(f) = d\}$. This notion of dimension is strongly related to the corresponding *Hilbert series* of the ideal I. The most known method for computing the dimension of an ideal and the degree of its algebraic variety is to define an explicit form of the Hilbert series of the ideal. Denote by $S_n = \mathbb{K}[x_1, ..., x_n]$.
For a given $d \in \mathbb{N}$, let $(\mathbb{S}_n)_d = \{f \in S_n \;:\; deg(f) = d\}$ be a $\mathbb{K}$-vector space.

Definition 37 *(Hilbert function)*
The Hilbert function associated to a homogeneous ideal $I = \langle f_1, \ldots, f_m \rangle$ is defined as follows:

$$HF_{d,m,d_m}(n) \;=\; dim(S_n/I)_d \;=\; dim\big((S_n)_d/I_d\big) \;=\; dim(S_n)_d - dim(I_d),$$

where $\underline{d_m} = (d_1, \cdots, d_m)$ and each d_i corresponds to the degree of the homogeneous polynomial f_i.

After a certain degree (the degree of regularity), the Hilbert function that depends on d will be equal to a certain polynomial, the so-called Hilbert polynomial. The degree of this latteris exactly the dimension of the ideal I.
The Hilbert series is given by : $HS_{d,m,\underline{d_m}}(z) = \sum_{d \geq 0} HF_{d,m,\underline{d_m}}(n) z^d$.
Now, it is possible to give a concrete bound for the complexity of a Gröbner basis computation.

7.2.3 Complexity analysis and solving strategy

In order to obtain an effective solving strategy in terms of speed and low memory usage, it is important to adapt a very precise monomial order.
In practice, it is not always simple and easy to compute a Gröbner basis directly. Actually, the choice of a monomial order affects directly the computation's complexity.

When dealing with zero-dimensional ideals, the computation of a Gröbner basis with respect to the lex order is not complicated numerically. In fact, the structure of such basis can be given as follows:

Theorem 2 *(Elimination)[***P.113***, 26]*
Given an ideal $I \in \mathbb{K}[x_1, \ldots, x_n]$*, let* G *be a Gröbner basis of* I *with respect to the* lex *order with* $x_1 \succ \ldots \succ x_n$*. For every* $\ell,\ 0 \leq \ell \leq n-1\ G_\ell = G \cap \mathbb{K}[x_{\ell+1}, ..., x_n]$ *constitutes a Gröbner basis of the ideal* $I_\ell = I \cap \mathbb{K}[x_{\ell+1}, ..., x_n]$.

Therefore, when a lexicographic order lex is used, the reduced Gröbner basis $G = (g_1, \ldots, g_s)$ of I has the following triangular shape:

$$
\begin{array}{r}
g_1(x_1, \quad x_2, \quad \ldots, \quad x_n) \\
\vdots \\
g_i(x_1, \quad x_2, \quad \ldots, \quad x_n) \\
g_{i+1}(x_2, \quad \ldots, \quad x_n) \\
\vdots \\
g_j(x_2, \quad \ldots, \quad x_n) \\
g_{j+1}(x_3, \ \ldots, \ x_n) \\
\vdots \\
\vdots \\
\vdots \\
g_s(x_n)
\end{array}
$$

In practice, the lex order tends to be the most costly while the $\mathrm{grevlex}$ order seems to be easier: it produces, almost all the time, the Gröbner basis that are the easiest to compute.
Thus, the maximal degree of polynomials intervening during the computation of the basis is smaller for the *grevlex* order than other monomial orders [49].
In terms of complexity, d_{reg} forms an upper bound to the maximal degree reached during Gröbner basis computation of a zero dimensional ideal.

Proposition 1 *(Complexity bound [31]) Given a zero-dimensional ideal* I*, the computational complexity of the corresponding Gröbner basis is given by:*

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

where $2 \leq \omega \leq 3$ *is the linear algebra constant.*

The general strategy for solving a system of polynomials using the Gröbner basis technique can be summarized by:

Solving strategy.

- First, compute a Gröbner basis of the system of polynomials with respect to the *GrevLex* order;
- Then, apply a specific algorithm to convert the used monomial order, such as the FGLM algorithm [34].
- Finally, a Gröbner basis for the lex order is carried out. This monomial order is very useful to express the solutions of the corresponding system of polynomials.

The FGLM algorithm [34], named after its designers, Faugère, Gianni, Lazard and Mora, takes as input a Gröbner basis with respect to a monomial order and gives a second Gröbner basis with respect to an another order.
Generally, The main interest is to solve systems of polynomials over a finite field. For this reason, a suitable algorithm, for the resolution over a field, is designed that is the *Hybrid approach* [11].

7.3 Resolution with the $\mathrm{Hybrid\ Approach}$

The basic idea of this approach is to combine two known techniques for solving systems of polynomials: the exhaustive search and the Gröbner basis.
By fixing certain variables, not only one but several, some subsystems are expected to be easier to solve. Therefore, it is important to carefully choose the number of variables to be fixed between exhaustive search and the computation using the Gröbner basis.
Technically, the $\mathrm{Hybrid\ approach}$ brings a gain in the number of operations needed for solving a system of equations.

$\mathrm{Hybrid\ approach}$'s complexity

[[11]] The complexity of the $\mathrm{Hybrid\ approach}$ can be easily deduced from the Gröbner basis method. As mentioned above, solving systems of polynomials requires two steps: the computation of a first Gröbner basis and then the change of the monomial order.
Also, as stated before, this algorithm involves an exhaustive search, so its complexity depends on the number of variables $k \geq 0$ to fix, and it can be determined as follows.

Proposition 2 *Let $\{f_1, \ldots, f_m\} \in \mathbb{K}[x_1, \ldots, x_n]$ be a zero-dimensional system, where $\mathbb{K}$ is of order equal to q.*
Let $D^{max}(k)$ be the maximum number of solution in $\bar{\mathbb{K}}$ counted with multiplicities of the following system:

$$\{f_1(x_1, \ldots, x_{n-k}, v_1, \ldots, v_k), \ldots, f_m(x_1, \ldots, x_{n-k}, v_1, \ldots, v_k)\}$$

for any fixed values $(v_1, \ldots, v_k) \in \mathbb{K}^k$.
The complexity of the $\mathrm{Hybrid\ approach}$ is given by:

$$\min_{0 \leq k \leq n} \left(q^k \left(Complexity_{GB}(n-k, d_{reg}^{\max}(k)) + \mathcal{O}\left((n-k) D^{\max}(k)^{\omega} \right) \right) \right).$$

Proof: This bound is the sum of the complexity of both the Gröbner basis technique and FGLM algorithm for solving a system of $n-k$ variables. It is multiplied by the cost of an exhaustive search on the k fixed variables. ■

PART II:

CONTRIBUTIONS ON PKP

CHAPTER 8

The Permuted Kernel Problem (PKP)

Contents

8.1 Introduction to PKP

The permuted kernel problem, also known as PKP, was first introduced by A. Shamir in 1989 [63]. The author of [63] proposed a new public-key scheme, a Zero-Knowledge (ZK) Identification scheme, based on the intractability of PKP that possesses a very efficient implementation on low-cost smart cards. The permuted kernel problem is an algebraic problem where the goal is to find a permutation of a given vector so that the reordered vector is in the kernel of a publicly known matrix.

Informally, the permuted kernel problem is defined as follows:

- Given a matrix A and a vector V,
- Find a permutation π such that $AV_\pi = 0$.

Even though PKP requires only simple mathematical operations that involve basic linear algebra computations, it has been proven to be an NP-Hard problem [36].

The simplicity of PKP has led to a larger theoretical analysis, and to a significant attention for using this latter in cryptographic applications. The so-called PKP problem has been widely studied in several papers [6, 37, 40, 57, 60]. Despite the research efforts and for appropriate parameters, the problem PKP is still exponential.

This chapter investigates the theoretical analysis behind the permuted kernel problem PKP and its complexity over a finite field. It also provides a summary of previously known algorithms used to solve this problem. The main contribution of this chapter is to provide an updated complexity analysis of the most efficient algorithm for solving instances of the permuted kernel problem.
The permuted kernel problem [63] is defined next over a finite field $\mathbb{F}_p$.

8.1.1 Description of the permuted kernel problem

The permuted kernel problem (PKP) is a combinatorial problem on which the security of an efficient ZK Identification scheme is based [63]. A reduction of the $3\text{-}\mathrm{Partition}$ problem proves that PKP over a finite field $\mathbb{F}_p$ is an NP-Hard problem [36, pg.224] in the good reasoning (i.e the hardness of PKP grows exponentially with p).
Given a prime number p and two random numbers $m, n \in \mathbb{N}^*$, PKP is defined as follows:

Input. A finite field $\mathbb{F}_p$, a matrix $A \in \mathcal{M}_{m\times n}(\mathbb{F}_p)$ and an n-vector $V \in {\mathbb{F}_p}^n$.
Question. Find a permutation π over $(1, \ldots, n)$ such that $A \times V_\pi = 0$ (mod p), where $V_\pi = (V_{\pi(1)}, \ldots, V_{\pi(n)})$.

Note that, all the arithmetic operations carried out are modulo p.

8.1.2 Practical complexity considerations

It is more suitable to assume that the matrix $A \in \mathbb{M}_{m\times n}(\mathbb{F}_p)$ is of rank exactly m. Otherwise an equivalent matrix of A could be expressed with fewer significant lines. In order to have a hard instance of PKP, $n-m$ (which is the dimension of the kernel of A of rank m) must be large enough so that the kernel of the matrix A has sufficiently enough elements.

By denoting $A_\sigma = (a_{i\sigma(j)})$, the effect of a permutation σ over the columns of A, it is easy to see that $A_\sigma V_\sigma = AV$. Also, up to a certain reordering of the columns of A, one may assume that A is given in its systematic form:

$$A = (a_{ij})_{1\leq i\leq m, 1\leq j\leq n} \;=\; \big[A'|I\big],$$

where $A' = (a'_{ij})_{1\leq i\leq m, 1\leq j\leq n-m} \in \mathcal{M}_{m\times(n-m)}(\mathbb{F}_p)$ and I is the identity matrix of size m.
It is more preferable that the n-vector V be with no double (distinct coordinates) in $\mathbb{F}_p$. So that, the number of possible solutions decreases, and at the same time the search space expands.

The probability of an arbitrary vector to be in the kernel of the matrix $A \in \mathbb{M}_{m\times n}$ with coefficients in $\mathbb{F}_p$ and whose rank is equal to m, is equal to p^{-m}. Since that the n-vector V has distinct entries, then its orbit under the possible permutations consists of $n!$ vectors. Therefore, in order to obtain in average one

solution for a PKP instance, then $n!$ must be approximately equal to p^m ($n! \approx p^m$).

The robustness of PKP comes from, on one hand, the big number of permutations, on the other hand, from the small number of possible permutations which may suit the kernel equations. More precisely, PKP is hard because it obligates the choice of a vector, with already fixed set of entries, from the kernel of the matrix A.

8.2 Solving PKP: best known algorithms

Due to the efficient Identification scheme IDS based on PKP and introduced in the rump session of Crypto 89 [63], various solving tools and attacks were proposed that are all exponential.

8.2.1 Exhaustive search

The exhaustive search, also known as brute-force attack, consists of examining all possible candidates (permutations of a set on n elements) for the solution in order to determine whether a candidate satisfies the problem's conditions. Despite the fact that this search technique is very general and naive, mainly when the search space is large, it is important to consider its complexity which is in $n!$.

8.2.2 J. GEORGIADES method

J. GEORGIADES [37] was the first to improve the exhaustive search for solving an instance of the permuted kernel problem. The main idea of the algorithm is to decrease the set of suitable permutations by finding some new equations.

Assuming that the rank of the matrix A is equal to m, the rank-nullity theorem states that the dimension of the kernel (null space) of A is equal to $dim\big(ker(A)\big) = n-m$. Hence, there are $n-m$ linearly independent vectors that span the kernel of A. It is possible to fix the last m components of each vector while first $n-m$ components are constants depending on A. Consequently, the kernel of the matrix A can be stated as follows:

$$Ker(A) = \lambda_1 \begin{pmatrix} u_{1,1} \\ u_{1,2} \\ \vdots \\ u_{1,m} \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \lambda_2 \begin{pmatrix} u_{2,1} \\ u_{2,2} \\ \vdots \\ u_{2,m} \\ 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} + \cdots + \lambda_{n-m} \begin{pmatrix} u_{n-m,1} \\ u_{n-m,2} \\ \vdots \\ u_{n-m,m} \\ 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}, \tag{8.1}$$

where $u_{1,1}, \ldots, u_{n-m,m}$ belong to $\mathbb{F}_p$, and so does the $\lambda_i s$.
Thus, the kernel of A is the set of vectors:

$$\Big(f_1, f_2, \ldots, f_m, \lambda_1, \lambda_2, \ldots, \lambda_{n-m}\Big), \tag{8.2}$$

where $f_j = \sum_{i=1}^{n-m} u_{i,j}\lambda_i, \quad j \in \{1, \ldots, m\}$.
So that, to find the secret permutation π, it suffices to find the right $n-m$ components (the $\lambda_i s$) of the corresponding vector V_π, and the other m components can be easily deduced from the kernel equations 8.1 and 8.2.
In other words, it suffices to correctly pick and place $(n-m)$ values out of n. Therefore, the number of permutations to be considered will be reduced from $n!$ to:

$$\frac{n!}{\big(n-(n-m)\big)!} = \frac{n!}{m!}.$$

Moreover, since that V is publicly given, then the components of V_π are also known, as well as all their symmetrical expressions. For example the sum of the vector components, the sum of their squares, etc... Hence, one may successfully establish relations between the $\lambda_i s$ to decrease the cost of the attack. As a matter of fact, $V = (v_1, \ldots, v_n)$ is known, and its permutation $V_\pi = (x_1, \ldots, x_n) \in Ker(A)$ is of the form given in 8.2. Thus, the following relations in $\mathbb{F}_p$ hold:

$$\sum_{i=1}^{n} v_i^r \mod p \quad = \quad \sum_{i=1}^{n} x_i^r \mod p \quad = \quad \sum_{i=1}^{m} f_i^r + \sum_{i=1}^{n-m} \lambda_i^r \mod p, \qquad (G_r)$$

where r is a positive integer.

When giving small values for r (for example $r = 1, 2$), the equations G_r stated above might be very useful and easy to compute.
For $r = 1$ ($resp.$ $r = 2$), it is easy to represent some of the λ_is ($resp.$ $\lambda_{j\neq i}$) as a linear combination ($resp.$ quadratic equation) of the other $n-m-1$ ($resp.$ $n-m-2$) values. This will reduce, by taking $r = 2$, the possible permutations to:

$$\frac{n!}{\big(n-(n-m-2)\big)!} = \frac{n!}{(m+2)!}.$$

Note that, the use of symmetric equations of higher degree ($r > 2$) might not be very beneficial due to its computational complexity.

8.2.3 A time-memory trade-off

The authors of [6] introduce a new type of attack that is a time-memory trade-off leading to a faster method for solving PKP instances. The proposed approach reduces both of time and space complexities required to solve the PKP problem.

Recall that, solving a random PKP instance means to find a permutation π of a vector V such that $A \times V_\pi = 0$. Hence, using the reduced systematic form of the matrix A, PKP can be represented as:

$$\begin{pmatrix} a'_{1,1} & \ldots & a'_{1,n-m} & 1 & & \\ \vdots & & \vdots & & \ddots & \\ a'_{m,1} & \ldots & a'_{m,n-m} & & & 1 \end{pmatrix} \begin{pmatrix} V_{\pi(1)} \\ \vdots \\ V_{\pi(n)} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

Consequently, in order to find the permutation π one must solve a system $\mathcal{S}$ of m equations in n variables carried out from the matrix-vector product described above.
The algorithm consists of choosing of the system $\mathcal{S}$ only $0 \leq k \leq m$ equations to solve out of m. Due to the block form of A, a sub-system of $\mathcal{S}$ formed by k equations involves only $n-m+k$ variables namely $V_{\pi(1)}, \ldots, V_{\pi(n-m+k)}$. The algorithm employs an another parameter $0 \leq k' \leq n-m+k$ that determine the storage amount needed in the first step of the algorithm.
The approach is essentially divided into two steps:
Step1: precomputation. Recall that the aim is to solve k relations in $n-m+k$ variables. The algorithm computes, for each $k'-uple$ $(V_{\pi(1)}, \ldots, V_{\pi(k')})$, the following values:

$$b_1 = \sum_{j=1}^{k'} a'_{1,j} V_{\pi(j)}$$

$$\vdots$$

$$b_k = \sum_{j=1}^{k'} a'_{k,j} V_{\pi(j)}$$

Then, the $\frac{n!}{(n-k')!}$ possible $k'-uples$ and their corresponding values $(b_1, \cdots, b_k)$ are stored. Note that, for each of the p^k possible value of the vector $(b_1, \cdots, b_k)$ the $k'-uple$ $(V_{\pi(1)}, \ldots, V_{\pi(k')})$ can be easily accessed.
The first step of the algorithm costs $\frac{n!}{(n-k')!}$ matrix-vector product. The memory required is nearly equal to $\frac{n!}{(n-k')!}$ $k'-uples$. Also, for each $(b_1, \cdots, b_k)$ value corresponds *approx.* $p^{-k}\frac{n!}{(n-k')!}$ $k'-uples$.
Step2: exhaustive trial. The exhaustive search is performed over the remaining components $(V_{\pi(k'+1)}, \ldots, V_{\pi(n-m+k)})$. There are $\frac{n!}{(m+k'-k)!}$ possible values of such vector.
For each tested vector, the corresponding values are computed from the k equations:

$$c_1 = \sum_{j=k'+1}^{n-m+k} a'_{1,j} V_{\pi(j)}$$

$$\vdots$$

$$c_k = \sum_{j=k+1}^{n-m+k} a'_{k,j} V_{\pi(j)}$$

Now, using the precomputation step, a list of probable $(V_{\pi(1)}, \ldots, V_{\pi(k')})$ is obtained. Obviously, the k relations can be represented as:

$$b_1 + c_1 = 0$$

$$\vdots$$

$$b_k + c_k = 0.$$

Moreover, the $k'-uple$ $(V_{\pi(1)}, \ldots, V_{\pi(k')})$ is certainly one of the possible $k'-uples$ for the $(-c_1, \ldots, -c_k)$ value of $(b_1, \ldots, b_k)$.

For every vector $(V_{\pi(k'+1)}, \ldots, V_{\pi(n-m+k)})$ generated, there are in average $p^{-k}\frac{n!}{(n-k')!}$ $(V_{\pi(1)}, \ldots, V_{\pi(k')})$ values. For each probable solution $(V_{\pi(1)}, \ldots, V_{\pi(n-m+k)})$, the remaining unsolved equations from $(k'+1)$ to (m) give successively only one possible value for the last components $V_{\pi(n-m+k+1)}, \ldots, V_{\pi(n)}$.
The required storage space of this step is negligible. In contrast, the required time is approximately equal to $\sup(\frac{n!}{(m+k'-k)!}\frac{n!}{(n-k')!}p^{-k}, \frac{n!}{(m+k'-k)})$ matrix vector product.
Thus, for every pair (k, k'), the total time complexity for solving PKP, using this time-memory attack is about:

$$\frac{n!}{(n-k')!} + \sup(\frac{n!}{(m+k'-k)!}\frac{n!}{(n-k')!}p^{-k}, \frac{n!}{(m+k'-k)}),$$

and the total space required is about:

$$\frac{n!}{(n-k')!} \text{ k'-vectors.}$$

8.2.4 Patarin-Chauvaud attack

J. Patarin and P. Chauvaud combine in [57] the two ideas presented in the previous attacks (see Sections 8.2.2, 8.2.3). The resulting algorithm decreases the time required to attack PKP. It also presents some new ideas in order to reduce this time the memory storage.
Thus, this leads to a new algorithm which is quicker and more efficient than the attacks given above [6, 37].

Due to the different variants and the numerous technical considerations proposed by the authors, we refer the readers to the main article [57] for a more detailed description.

8.2.5 Joux-Jaulmes algorithm

In [40], A. Joux and E. Jaulmes use an algorithm introduced in [42] to present a new time-memory trade-off algorithm for solving the permuted kernel problem PKP. This latter involves the so-called $4\mathrm{SET}$ problem (see [40, 42] for more details) that can be stated as follows:

Input. An n-vector P of prime numbers p_is $P = (p_1, \ldots, p_n)$, four sets S_i of $n-$vectors such that $|S_i| = N_i$ for $i = 1 \ldots 4$, and n sets $D_1, \ldots, D_n$.
Question. Find $v^{(1)} \in S_1, \ldots, v^{(4)} \in S_4$, $d_1 \in D_1, \ldots, d_n \in D_n$ such that:

$$\forall i \in [1, \ldots, n], \quad v_i^{(1)} + v_i^{(2)} + v_i^{(3)} + v_i^{(4)} \equiv d_i \pmod{p_i}$$

The solving tool of the $4\mathrm{SET}$ problem is divided into two phases: a precomputation phase called the A-Phase, and a main loop called the B-Phase that consists essentially of two enumeration steps (detailed in [40]). For reasonable choice of parameters, the authors of [40] define a time complexity bound of the algorithm that is given by:

$$\mathcal{O}\Big((n-k)\psi N_1 N_2 N_3 N_4\Big),$$

Where $\psi = \prod_{i=1}^{k} \frac{|D_i|}{p_i}$ for suitable choices of $1 \leq k \leq n$. As stated in [40], it is feasible to reduce an instance of PKP to the $4\mathrm{SET}$ problem. The algorithm builds upon the useful G_r equations [37] described in Section

8.2.2. The linear equation G_r (for $r = 1$) describes the fact that the sum σ of the components of the vector V (that is the same for V_π) is independent of the secret permutation π. By taking into consideration the linear equation G_r, the kernel vector V_π verifies:

$$(A'_0|I_{m+1})V_\pi = \begin{pmatrix} \sigma \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

when considering the systematic form of $A = \big[A'|I\big]$. Thus, $A'_0 \in \mathcal{M}_{(m+1)\times(n-m-1)}(\mathbb{F}_p)$, and I_{m+1} is the identity matrix of order $(m+1)$.
In order to view a PKP instance as $4\mathrm{SET}$ instance, A'_0 can be seen as a composition of four roughly equal parts, so:

$$(A'_1A'_2A'_3A'_4|I_{m+1})V_\pi = \begin{pmatrix} \sigma \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{8.3}$$

where A'_i is a $(m+1)\times(n_i)$ matrix and $n_1+n_2+n_3+n_4 = n-m-1$.
Recall that, V_π is the reordered vector under the permutation π of the vector V. The sets S_i need to be determined in order to apply the $4\mathrm{SET}$ problem. Since A'_i is an $(m+1)\times(n_i)$ matrix, S_i is the set of $m+1$-vectors: the product of A'_i by all the possible n_i combinations of the components of V. So, the size of S_i is equal to $N_i = \frac{n!}{(n-n_i)!}$.

The Joux-Jaulmes algorithm considers all the prime numbers p_is to be equal to the prime number p given in the PKP instance. To determine the $m+1$ sets D_i, the vector V_π must be decomposed, similarly to the matrix A'_0, $V_\pi = (V_1^{(\pi)}V_2^{(\pi)}V_3^{(\pi)}V_4^{(\pi)}V_5^{(\pi)})$ such that for $i \in [1,\ldots,4]$, $V_i^{(\pi)}$ is an n_i-vector where, as we quoted before, $n_1+n_2+n_3+n_4 = n-m-1$. So, $V_5^{(\pi)}$ is the vector formed by the last $(m+1)$ components of V_π. Hence, the matrix-vector product (8.3) can be reformulated as follow:

$$A'_1V_1^{(\pi)} + A'_2V_2^{(\pi)} + A'_3V_3^{(\pi)} + A'_4V_4^{(\pi)} = \begin{pmatrix} \sigma \\ 0 \\ \vdots \\ 0 \end{pmatrix} - V_5^{(\pi)} = \begin{pmatrix} \sigma - v_{n-m}^{(\pi)} \\ -v_{n-m+1}^{(\pi)} \\ \vdots \\ -v_n^{(\pi)} \end{pmatrix} = \mathcal{D}$$

The value of $V_5^{(\pi)}$ depends on the V_i^{π}s , for $i \in [1,\ldots,4]$, so does the $m+1$-vector $\mathcal{D}$. The first component of $\mathcal{D}$ depends on $v_{n-m}^{(\pi)}$ that has n possible values. Thus, D_1 is the set of these possible n values. Since V has no double entries, the set D_2 is composed by $n-1$ elements, and so on. In this way, the sets $D_1,\ldots,D_{m+1}$ are built such that each one has in average:

$$\frac{n+(n-1)+,\ \ldots\ ,+(n-m)}{m+1} \quad elements.$$

In order to define the solving time complexity, when dealing with bit operations, it is indispensable to pack 32 or even $64 = 2^6$ bit operations in one word operation. It is equivalent to divide the complexity by 2^6. In summary, the authors of [40] gives the following time complexity for their algorithm (see [40] for more details):

$$\mathcal{O}\Big((m+1-k) \times \psi \times \frac{n!^2}{(n-n_1-n_2)!(n-n_3-n_4)!} \times 2^{-6}\Big),$$

Where $k = \log_{\frac{|D_i|}{p}}(\psi)$.
It appears in [40] that this new approach is the most efficient to solve PKP.

The following sections show that the performance of Joux-Jaulmes attack on PKP had been misjudged. Joux-Jaulmes attack is much more complicated than expected.
Thus, the next section details the choice of the most efficient solving tool for PKP.

8.3 Concrete security of PKP

The goals of this Section are twofold: to fix the complexity bound of the algorithm introduced by Joux-Jaulmes, and to provide the best solving tool for PKP.

8.3.1 Complexity Analysis of Joux-Jaulmes algorithm

Note that, this section the notations of Section 8.2.5.
As stated before, any PKP instance can be reduced and seen as an instance of the $4\mathrm{SET}$ problem. Thus, it is possible to apply on PKP the same solving strategy of the $4\mathrm{SET}$ problem. In [40], the Joux-Jaulmes algorithm was introduced to solve PKP and it consists of a main loop involving two enumerations phases: A-Phase and B-Phase. The authors assume that the second enumerating phase (B-Phase) dominates the time complexity of their technique. Without going too far into the complexity analysis of Joux-Jaulmes approach, we found that the overall complexity is wrongly estimated. By taking into consideration a reasonable choices of parameters, it appears that the time complexity of Joux-Jaulmes algorithm is controlled most of the time by the A-Phase.
Recall that the analysis of Joux-Jaulmes [40] leads to the following:

The number of operations required to proceed the A-Phase is in:

$$\mathcal{C}_{A-Phase} \;=\; \mathcal{O}\Big(\; \max\big\{N_1 \log(N_2)\psi\phi,\; (m+1-k)\psi\frac{n!}{(n-n_1-n_2)!}\big\}\;\Big).$$

While the B-phase needs approximately:

$$\mathcal{C}_{B-Phase} \;=\; \mathcal{O}\Big(\; \max\big\{N_3 \log(N_4), (m+1-k)\psi\frac{n!^2}{(n-n_1-n_2)!(n-n_3-n_4)!}\phi^{-1}\big\}\;\Big),$$

Where, $N_i \;=\; \frac{n!}{(n-n_i)!}$.
Therefore, the total time complexity which is determined by the effect of the main loop involving the two Phases, can be expressed as follows:

$$2^{-6} \times \phi \times \big(\mathcal{C}_{A-Phase} + \mathcal{C}_{B-Phase}\big),$$

Where $\phi = p^k$ is the cost of the main loop containing the two phases A and B, and 2^{-6} is for packing bit operations.
The following table approves what it is claimed previously: the algorithm's overall complexity is dominated by the A-phase.
Note that, Table 8.1 uses the same parameters given in [40], and of the form $\big(\mathrm{PKP}_p(m,n)\big)$, .

Parameters Sets	A-phase Complexity	B-phase Complexity	Overall Complexity
$\big(\mathrm{PKP}_{251}(16,32)\big)$	$2^{45.72}$	$2^{18.02}$	$2^{93.55}$
$\big(\mathrm{PKP}_{251}(15,32)\big)$	$2^{46.13}$	$2^{18.02}$	$2^{93.96}$
$\big(\mathrm{PKP}_{251}(24,48)\big)$	$2^{94.45}$	$2^{32.09}$	$2^{190.1}$
$\big(\mathrm{PKP}_{251}(34,64)\big)$	$2^{135.67}$	$2^{40.67}$	$2^{270.19}$

Table 8.1: The A/B-Phase complexity of JOUX-JAULMES algorithm

Experimental results confirm that, for suitable sets of parameters minimizing the total time complexity, the A-phase dominates, and the total complexity is way more greater than announced. Therefore, the JOUX-JAULMES algorithm is not very efficient for solving PKP.

8.3.2 Simplest and most efficient algorithm

This Section presents complexity bounds estimations for some techniques that helps solving the permuted kernel problem (PKP). It is convenient to assume that an elementary operation is defined by the computation of a tuple, such as a vector-matrix product in $\mathbb{F}_p$. Therefore, a memory unit is defined by the space needed to store a tuple.

8.3.2.1 Improvement and Generalization of existing attacks.

The robustness of the permuted kernel problem (PKP) was widely investigated. Some of the most common algorithms were introduced in previous Sections.
This Section is a build-up on existing solving techniques for PKP in order to upgrade the solving complexity bound of the permuted kernel problem.

As summarized in Section 8.2.3, the time-memory trade-off technique reduces the complexity time for solving instances of PKP at a significant cost of increasing the storage (memory). The algorithm is designed to speed up the solving time by performing a pre-computation step on a subset of the search space.
Moreover, J. Patarin and P. Chauvaud bring together the solving tool proposed by J. Georgiades (see Section 8.2.2) along with the time-memory trade-off algorithm (see Section 8.2.3) in a way to decrease the time needed to solve a PKP-based instance [57]. Furthermore, the authors of [57] propose some

interesting ideas that decreases the memory usage.

Since that several variants were presented in [57], only few variants will be used and cited in this work (See [57] for further details):

- "Set integration" technique: performs an exhaustive search (also known as brute-force attack) on only a subset of values, instead of all the ordered tuples of values. Such method yields a considerable decrease of the size of the initial pre-computation step.
- "Middle values" technique: performs an exhaustive search on a number of "middle values" in order to find solutions with respect to some fixed middle values.
- "Pre-computation on the matrix A" method: investigates the possibility of dealing with a sub-system that can be expressed with less variables. This technique leads to probabilistic algorithms.

Likewise, G. Poupard in [60] provides a generalization of the "Middle values" technique, in addition to its corresponding complexity analysis. But, it seems to be imprecise since that no clear details were given.
Thus, next section considers all the existing algorithms for solving the permuted kernel problem, and provides a fresher look on the best solving techniques.

8.3.2.2 Our method : Extension of the most efficient attacks.

This Section combines most of the previously described solving methods for the permuted kernel problem. It provides a new software that yields an efficient complexity analysis. Moreover, an approximate time (and space) complexity bound is established next.
Let kV denotes a tuple of k values, where k denotes conveniently at the same time a subset of indexes and the number of elements of this subset. Similarly, $k.iA$ is a sub-matrix of A with some given subsets of indexes.

The algorithm provided in this Section is mainly based on the techniques proposed by Patarin-Chauvaud in [57] and G. Poupard [60]. The extension of these two techniques pushes further the implementation in order to provide higher security levels.
More specifically, the algorithm employs four essential variants:

1. The use of symmetric polynomial equations of small degrees G_r.
2. A time complexity reduction: by performing a pre-computation step and an exhaustive search on a sub-set of variables, instead of the whole ordered tuples of variables [6, 57].
3. A memory complexity reduction: by introducing some middle values that helps solving a simple system of equations [57].
4. Carry out a pre-computation on the matrix A which leads to probabilistic algorithms (See Section 8.3.2.3).

The starting point is to bring together, in Alg. 1, the first two ideas given above (Section 8.3.2.2) for solving PKP. This method exploits usefully the special form of the matrix $A = (A' \| I)$.

Algorithm 1 A1 : Solve PKP (n, m, p)

Require: $0 \leq k \leq m$ and $l + r + m - k = n$

select k equations, and split their variables into two sets l and r
for all l-tuple lV **do**
  compute $C \leftarrow k.lA \times lV$
  store (C, lV) in a file F_0 so that given a value C, one can efficiently access all the corresponding l-tuples
end for
for all r-tuple rV **do**
  compute $C \leftarrow -k.rA \times rV$
  pick from F_0 a list L of l-tuples associated with C
  filter the list L by keeping only the l-tuples compatible with rV
  for all l-tuple lV in L **do**
    compute the last variables $sV \leftarrow (m-k).(l+r)A \times (l+r)V$
    if the values s are compatible with $(l + r)$ **then**
      $(l + r + s)$ is a solution
    end if
  end for
end for

For this algorithm, the time and space complexities are given by the following formulas:

$$\text{space} = \frac{n!}{(n-l)!}, \tag{8.4}$$

$$\text{time} = \frac{n!}{(n-l)!} + \frac{n!}{(n-r)!} + \frac{n!}{p^k(n-(l+r))!}. \tag{8.5}$$

In order to provide a first clear image on the parameter sets, several tests are carried out, using Alg. 1, on prime fields for different values of n. Thus, Fig. 8.1 provides the time complexity for solving PKP by Alg. 1.
Note that, the number m of equations is implicitly estimated by the closest integer for $\log(n!)/\log(p)$.

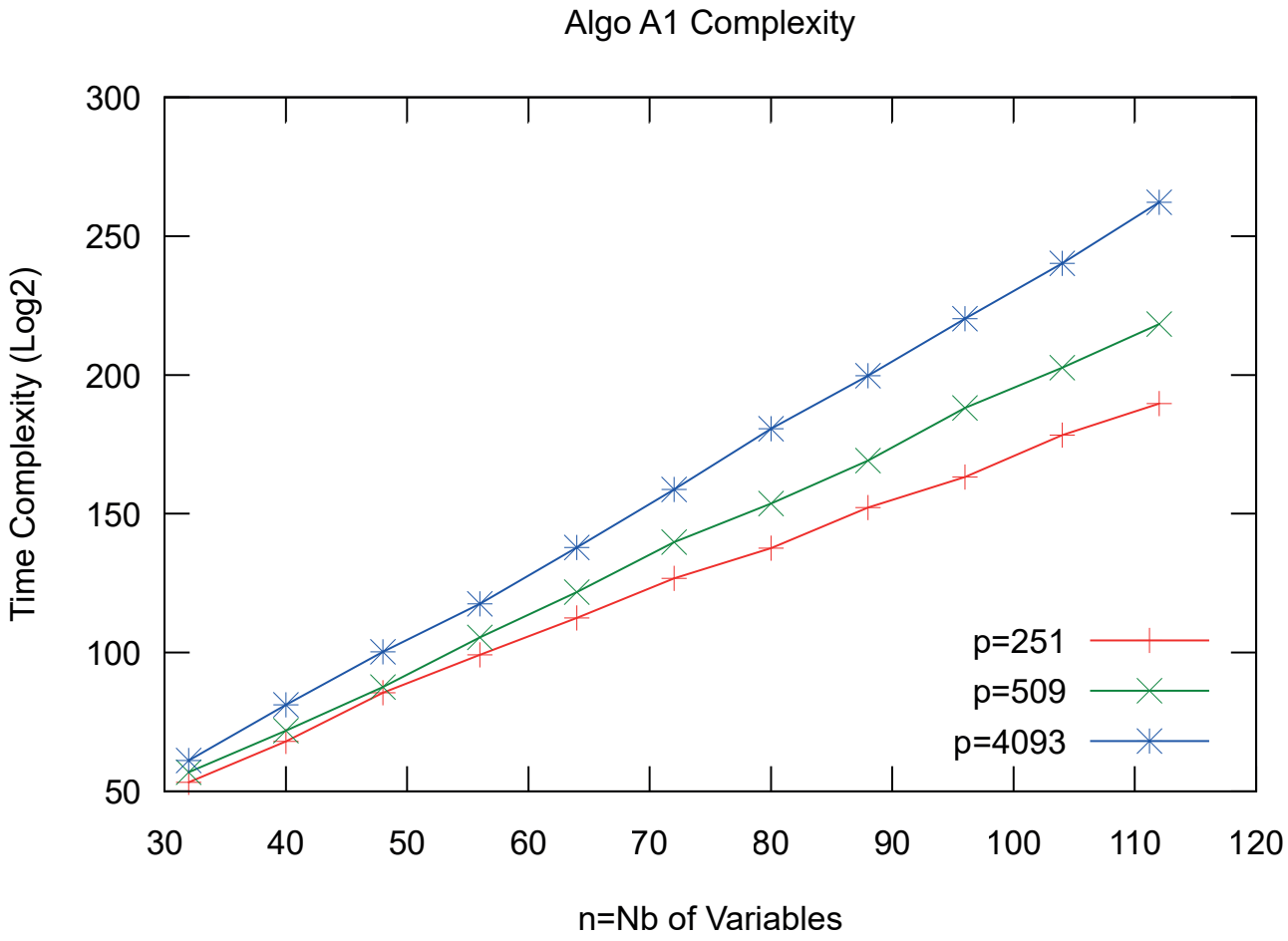

Figure 8.1: Time complexity of Alg. 1 for various values of p

The second step is to combine almost all of the variations cited in Section 8.3.2.2. Therefore, Alg. 2 leads to a more synthetic description of Poupard's algorithm mixed with the techniques of pre-computed files and "Middle values". Moreover, this Section provides a detailed complexity of the extended method which has not been estimated before (cf. equations 8.6 and 8.7 below).
Note that, we keep the notations used in [60, Fig. 3].

Algorithm 2 A2 : Solve PKP (n, m, p)

Require: $i + j + r + m = n$ and $c + c' \leq m$ and $d \leq c$ and $l + k = r + d$

for all j-tuple jV **do**
  compute $C_0 \leftarrow c.jA \times jV$
  store (C_0, jV) in a file F_0 so that, given a value C_0, one can efficiently access all the corresponding j-tuples
end for
for all l-tuple lV **do**
  compute $C_2 \leftarrow d.lA \times lV$
  store (C_2, lV) in a file F_2 so that given a value C_2, one can efficiently access all the corresponding l-tuples
end for
for all c-tuple of "Middle values" cC **do**
  for all i-tuple iV **do**
    compute $C_0 \leftarrow cC - c.iA \times iV$
    pick from F_0 a list L of j-tuples associated with C_0
    filter the list L_0 by keeping only the j-tuples compatible with iV and get $(j + i)$-tuples
    for all $(j + i)$-tuple $j + iV$ of L_0 **do**
      compute $C_1 \leftarrow c'.(j + i)A \times (j + i)V$
      store $(C_1, (j + i)V)$ in a file F_1 so that given a value C_1, one can efficiently access all the associated $(j + i)$-tuples
    end for
    for all k-tuple kV **do**
      compute $C_2 \leftarrow dC - d.kA \times kV$
      pick from F_2 a list L_2 of l-tuples associated with C_2
      filter the list L_2 by keeping only the l-tuples compatible with $kV$and get $(l + k)$-tuples
      for all $(l + k)$-tuple $(l + k)V$ of L_2 **do**
        compute $(c - d)V \leftarrow (c - d)C - (c - d).(l + k)A \times (l + k)V$ and keep only the compatible values with $(l + k)V$ and get $(r + c)$-tuples
        for all c'-tuple $c'V$ compatible with $(r + c)V$ **do**
          compute $C1 \leftarrow -(c').(r + c)A \times (r + c)V$
          pick from F_1 a list L_1 of $j + i$-tuples associated with C_1
          filter the list L_1 by keeping only the $(j + i)$-tuples compatible with $r + c + c'V$ and get $(j + i + r + c + c')$-tuples
          for all $(j + i + r + c + c')$-tuple $(j + i + r + c + c')V$ in L_1 **do**
            compute the last variables $sV \leftarrow (m - c - c').(j + i + r + c + c')A \times (j + i + r + c + c')V$
            if the values s are compatible with $(j + i + r + c + c')$ **then**
              $(j + i + r + c + c' + s)$ is a solution
            end if
          end for
        end for
      end for
    end for
  end for
end for

For this algorithm, the time and space complexities are estimated as follows:

$$\text{space} = \frac{n!}{(n-j)!} + \frac{n!}{(n-l)!} + \frac{n!}{p^c(n-(i+j))!} \tag{8.6}$$

$$\begin{aligned}\text{time} = &\frac{n!}{(n-j)!} + \frac{n!}{(n-l)!} + \frac{n!}{(n-(i+j))!} + \\ &\frac{p^c n!}{(n-k)!} + \frac{p^{c-d} n!}{(n-(k+l))!} + \frac{p^{c-d}(n-(c-d))!}{(n-(r+c+c'))!} \\ &+ \frac{(n-(c-d))!}{p^d(n-(i+j+r+c+c'))!}\end{aligned} \tag{8.7}$$

As stated in [60], the importance of this algorithm is for realistic attacks, where memory storage is limited. However, from a theoretical point of view, the previous algorithm is the most efficient method for solving the permuted kernel problem.

8.3.2.3 Probabilistic method

This Section describes the method used for the step of "Pre-computation on the A matrix" introduced in [57]. Similarly to other techniques cited in 8.3.2.1, the aim of the pre-computation on A is to reduce the complexity of solving the permuted kernel problem. The particularity of such method is that it limits the search space to a subsystem with less variables than expected.

First, when dealing with a set of m equations in n variables over $\mathbb{F}_p$, an accurate estimation of the probability to find a subset of k equations in only r variables is explicitly detailed below.
The following results hold:

Claim: The probability that a random linear equation in $\mathbb{F}_p[x_1, \ldots, x_n]$ is expressed in only r variables $x_1, \ldots, x_r$ is give by:

$$Pr = \binom{n}{r}\left(\frac{1}{p}\right)^{n-r}\left(1-\frac{1}{p}\right)^{r}.$$

Proof: Consider a random linear equation in n variables over $\mathbb{F}_p$. The coefficients of the equation are randomly and uniformly chosen in $\mathbb{F}_p$. The probability that a given variable x_i does not occur in the equation is the probability that its corresponding coefficient is equal to 0, which is in $\frac{1}{p}$. There are $n-r$ variables with coefficients equal to zero.
The coefficients of the r variables are assumed to be independently chosen at random. Therefore, the number of variables that appears in the equation follows the binomial distribution with parameters n and $\frac{1}{p}$; hence the result. ■

Claim: The probability that a set of k random linear equations in n variables over $\mathbb{F}_p$ is expressed in only r variables is given by:

$$\binom{n}{r}\left(\frac{1}{p^k}\right)^{n-r}\left(1-\frac{1}{p^k}\right)^{r}.$$

Proof: The probability that a given variable does not occur in the k equations $\frac{1}{p^k}$. Therefore, the number of variables that appears in k equations follows the binomial distribution with parameters n and $\frac{1}{p^k}$; hence the result. ■

Claim: Given a set of m linear equations over $\mathbb{F}_p$, the number of distinct linear sub-spaces formed by k equations is defined by :

$$\prod_{i=0}^{k-1}\frac{p^m-p^i}{p^k-p^i}.$$

Proof: The number of k-tuples of linearly independent equations is given by :

$$\prod_{i=0}^{k-1}p^m-p^i.$$

All the tuples that have equivalent bases of the same sub-space are related by a given isomorphism over $\mathbb{F}_p^k$. These isomorphisms amount to $\prod_{i=0}^{k-1}p^k-p^i$. Hence the number of distinct linear sub-spaces formed by k equations is equal to:

$$\prod_{i=0}^{k-1}\frac{p^m-p^i}{p^k-p^i}.$$

■

Claim: Given a set of m linear equations over $\mathbb{F}_p$, the number of distinct linear sub-spaces of k equations that can be expressed in r variables is approximately given by:

$$\binom{n}{r}\left(\frac{1}{p^k}\right)^{n-r}\left(1-\frac{1}{p^k}\right)^{r}\times\prod_{i=0}^{k-1}\frac{p^m-p^i}{p^k-p^i}.$$

Proof: Even though, the sub-spaces of dimension k in the linear space are not uniformly distributed over all the possibilities, but it is very close to the product of the number of sub-spaces by the probability that one subspace has the required property. This result has been experimentally verified. ■

The method of using sub-spaces of equations with less variables speeds up the search algorithm. It appears that the probability of finding such sub-sets is massively small.

8.4 Secure Parameters Sets and PKP application

As stated before, PKP was utilized in 1989 to construct a $\mathrm{Zero\text{-}Knowledge}$ Identification scheme (IDS) [63]. There has been renewed interest in the PKP-based IDS. In fact, it is possible to transform the PKP-based IDS into a post-quantum digital signature scheme. Thus, it is important to analyze the security of PKP, in order to upgrade the complexity bounds of each existing solving tool for the permuted kernel problem.

A realistic picture of the efficiency and the complexity bound of nearly all of the known methods for solving PKP is now given in Section 8.3.2.2, in addition to an upgraded complexity bound for the most efficient attack. Consequently, it is possible to define by now secure parameters sets for PKP instances.

The following table shows that the most efficient attack to solve PKP is our approach that is an extended version of Patarin-Chauvaud [57] and Poupard [60] techniques. The corresponding complexity bound for the extended approach which is established in Section 8.3.2.2, is computed using a Magma code given in Appendix A.
Note that, the same parameters sets, of the form $\big(\mathrm{PKP}_p(m,n)\big)$, used in [13] will be presented here.

Parameters Sets	$\big(\mathrm{PKP}_{251}(41,69)\big)$	$\big(\mathrm{PKP}_{509}(54,94)\big)$	$\big(\mathrm{PKP}_{4093}(47,106)\big)$
Security level	2^{128}	2^{192}	2^{256}
Brute force attack	2^{326}	2^{485}	2^{565}
J. Georgiades attack	2^{151}	2^{236}	2^{356}
Time-memory trade-off	2^{131}	2^{196}	2^{262}
Joux-Jaulmes attack	2^{286}	2^{413}	2^{432}
Extended approach	$\mathbf{2^{130}}$	$\mathbf{2^{193}}$	$\mathbf{2^{257}}$

Table 8.2: Complexity bounds for PKP's best known algorithms

8.5 Conclusion

This Chapter investigated the complexity of the so-called problem PKP. We presented a summary of the previously best known algorithms devoted to solve this problem.
Some of these published algorithms and results are not accurate or genuine. Therefore, this chapter reviewed and upgraded some of these algorithms.

Contrary to what is shown in [40], and after a thorough analysis of the State-of-the-art attacks of PKP, we claimed that the Joux-Jaulmes attack [40] is not the most efficient algorithm for solving PKP.
In fact, the complexity of the Joux-Jaulmes attack underestimate the amount of a certain important phase

of the algorithm.

Moreover, by combining methods, namely the approach of Patarin-Chauvaud and Poupard [60], we have been able to provide an explicit complexity bound (see equations 8.6 and 8.7 given above) of the best algorithm for solving hard instances of PKP.

Also, we have been able to develop a Magma code (see Appendix A) that gives a realistic picture on the security level of the permuted kernel problem. Such code is very useful to establish secure sets of parameters in order to arise hard instances of PKP.

CHAPTER 9

PKP-based Signature Scheme (PKP-DSS)

Contents

9.1 Introduction

The most widely used public-key schemes today rely on the discrete logarithm problem or the integer factorization problem. Since that quantum computers are believed to be able to solve these two computational problems and to outperform the classical computers, thus building a functional quantum computer would endanger a large number of common public-key cryptographic protocols. Even though, it might take a few years for quantum computations to be feasible, it is important to anticipate such breakthrough and build new public-key cryptosystems that are resistant to quantum attacks. Therefore, researchers are making great effort and progress in order to develop new quantum-based secure algorithms.

Moreover, the American National Institute of Standards and Technology (NIST) has launched a Post-Quantum Cryptography standardization process (`https://www.nist.gov/`) to standardize new quantum public-key algorithms. Due to this competition, there has been renewed interest in developing signature

schemes by applying the Fiat-Shamir transform (FS) [35] to zero-knowledge identification schemes ($\mathrm{ZK\text{-}IDS}$) whose security relies on the quantum hardness of some NP-Hard problem [8].

Quantum computers are expected to be incapable of solving NP-Hard problems in sub-exponential time (the worst case). Therefore, zero-knowledge identification schemes based on such type of problems constitute potential candidates for post-quantum cryptography. One of those problems is the permuted kernel problem (PKP, described in Section 8.1.1): the problem of finding a permutation of a given vector such that the reordered vector is in the kernel of a known matrix. This is a classical NP-Hard combinatorial problem that requires only simple operations such as basic linear algebra and permuting the entries of a vector. It has been a while since no new attacks have been reported on PKP. Hence, it is convenient to utilize the concrete hardness of this problem to build secure cryptographic schemes.

A five-move zero-knowledge identification scheme ($\mathrm{ZK\text{-}IDS}$) based on the permuted kernel problem was introduced by A. Shamir in [63]. A well known technique, namely the Fiat-Shamir transform [35], is commonly used to convert zero-knowledge identification schemes into signature schemes. In this work, the Fiat-Shamir paradigm [35] will be used to transform the PKP-based IDS into a new digital signature scheme DSS that is provably secure in the Random Oracle Model (ROM).

9.2 PKP-based Identification Scheme (PKP-IDS)

This section presents first the original 5-move zero-knowledge identification scheme ($\mathrm{ZK\text{-}IDS}$) based on the computational hardness of the permuted kernel problem [63, 48], noted here PKP-IDS. Then, our optimized version of PKP-IDS will be introduced in addition to the proof that the optimized identification scheme based on PKP is secure.

9.2.1 The original 5-move PKP-IDS

This Section provides the original PKP-IDS [48, 63], and also its slightly modified version. The original PKP-IDS consists of three probabilistic polynomial time algorithms $IDS = (\textsc{KeyGen}, \mathcal{P}, \mathcal{V})$ that will be described next.

Generation of the public key and secret key in PKP-IDS

The users first agree on a prime number p, and on n, m, the dimensions of the matrix $\mathbf{A}$. The public-key in PKP-IDS is an instance of the problem PKP, a solution to this instance is the secret-key.
Thus, the prover picks a (right) kernel-vector $\mathbf{w} \in \mathrm{Ker(A)}$, then randomly generates a secret permutation of n elements $\mathsf{sk} = \pi$ and finishes by computing $\mathbf{v} = \mathbf{w}_{\pi^{-1}}$.

The key generation algorithm is summarized in Alg. 3.

Algorithm 3 KEYGEN(n,m,p)

$\mathbf{A} \xleftarrow{\$} \mathbb{F}_p^{m \times n}$
$\mathbf{w} \xleftarrow{\$} \mathrm{Ker}(\mathbf{A})$
$\pi \xleftarrow{\$} S_n$
$\mathbf{v} \leftarrow \mathbf{w}_{\pi^{-1}}$
Return $(\mathsf{pk} = (\mathbf{A}, \mathbf{v}), \mathsf{sk} = \pi)$

9.2.1.1 5-move identification protocol: Prover $\mathcal{P}$ and Verifier $\mathcal{V}$.

The prover $\mathcal{P}$ and the verifier $\mathcal{V}$ are both interactive algorithms that accomplish the identification protocol in five moves.
The five moves are composed of one commitment, two challenges addressed from the verifier to the prover, and two responses sent from the prover to the verifier.

The identification protocol is summarized in Alg. 4.

Recall, the definitions of completeness, honest-verifier zero-knowledge (HVZK), and of soundness error given in Section 4.5.2.

It was shown in [63] that PKP-IDS is complete. If the commitment scheme utilized during the process is computationally hiding then PKP-IDS is computationally honest-verifier zero-knowledge, and if the commitment scheme is computationally binding, then PKP-IDS is sound with soundness error:

$$\kappa = \frac{p+1}{2p}.$$

It is usually possible to successfully cheat in such zero-knowledge identification schemes. A dishonest prover in the underlying identification scheme may be able to previously guess and then anticipate some of the challenges. Therefore, there is a nonzero probability, namely a soundness error κ, that a dishonest prover impersonate the legitimate prover.

In the case of PKP-IDS, the soundness error is equal to $\frac{p+1}{2p}$. Thus, it is necessary to repeat the protocol several times in order to reduce the probability of fraud. Sequentially repeating the zero-knowledge proof N times yields an identification scheme with knowledge error equal to

$$\kappa_{\mathsf{repeated}} = \kappa^N.$$

Hence, it suffices to repeat the protocol $\lceil \lambda / \log_2(\frac{2p}{p+1}) \rceil$ times to get a soundness error $\kappa \leq 2^{-\lambda}$, where λ is the security parameter.
The scheme is built in such a way that executing the protocol does not reveal any secrets (Zero-knowledge).

Algorithm 4 The original 5-move PKP-based identification protocol

$\mathcal{P}(\mathsf{sk}, \mathsf{pk})$		$\mathcal{V}(\mathsf{pk})$
$\sigma \xleftarrow{\$} S_n$		
$\mathbf{r} \xleftarrow{\$} \mathbb{F}_p^n$		
$\mathsf{C}_0 \leftarrow \textsc{Commit}(\sigma, \mathbf{A}\mathbf{r})$		
$\mathsf{C}_1 \leftarrow \textsc{Commit}(\pi\sigma, \mathbf{r}_\sigma)$		
	$\xrightarrow{\mathsf{C}_0, \mathsf{C}_1}$	
		$c \xleftarrow{\$} \mathbb{F}_p$
	$\xleftarrow{c}$	
$\mathbf{z} \leftarrow \mathbf{r}_\sigma + c\mathbf{v}_{\pi\sigma}$		
	$\xrightarrow{\mathbf{z}}$	
		$b \xleftarrow{\$} \{0, 1\}$
	$\xleftarrow{b}$	
if $b = 0$ **then**		
$\quad \mathsf{rsp} \leftarrow \sigma$		
else		
$\quad \mathsf{rsp} \leftarrow \pi\sigma$		
end if		
	$\xrightarrow{\mathsf{rsp}}$	
		if $b = 0$ **then**
		$\quad$ accept if $\mathsf{C}_0 = \textsc{Commit}(\sigma, A\mathbf{z}_{\sigma^{-1}})$
		else
		$\quad$ accept if $\mathsf{C}_1 = \textsc{Commit}(\pi\sigma, \mathbf{z} - c\mathbf{v}_{\pi\sigma})$
		end if

9.2.2 The modified version of PKP-IDS

Several optimizations will be provided in order to reduce the communication cost of the identification scheme, as well as the computational cost of the algorithm.
This Section presents first an explanation of few standard optimizations that are common for identification protocols based on zero-knowledge proofs.
Then, some novel optimizations will be given that are appropriate to the specific context of the PKP-based identification scheme PKP-IDS.

Hashing the commitments.

During the commitment phase of the protocol, instead of sending all the $2N$ commitments $\mathsf{C}_0^{(1)}, \mathsf{C}_1^{(1)}, \cdots, \mathsf{C}_0^{(N)}, \mathsf{C}_1^{(N)}$ the prover can simply hash all the commitments together using a collision resistant hash function $\mathcal{H}$ and sends only the hash $h = \mathcal{H}(\mathsf{C}_0^{(1)}, \cdots, \mathsf{C}_1^{(N)})$.
Then, the prover includes the N commitments $\mathsf{C}_{1-b_i}^{(i)}$ in the second response. Since that the verifier can recover the $\mathsf{C}_{b_i}^{i}$ by himself, he then has all the $2N$ commitments, so it is possible to hash them together and check if their hashed value matches h.
This optimization reduces the number of communicated commitments from $2N$ to N, at the cost of transmitting a single hashed value.

Use seeds and PRG.

Instead of directly choosing the permutation σ at random, it is always possible to choose instead a random seed of λ bits and use a pseudo random generator PRG to expand this seed into a permutation σ.
Thus, instead of transmitting σ, it is more practical to just transfer the λ-bit seed. This reduces the communication cost per permutation from $\log_2(n!)$ bits to just λ bits.
For example for 128-bits of security, n is equal to $n = 69$, so the communication cost per permutation drops from $\log_2(69!) \approx 327$ bits to just 128 bits.

Matrix $\mathbf{A}$ in systematic form.

This Section describes specific optimizations for PKP-IDS.
With high probability, elementary row operations can be performed on the matrix $\mathbf{A}$ in order to put $\mathbf{A}$ in its symmetric form $\left(I_m || \mathbf{A}'\right)$, for some $(n-m) \times m$ matrix $\mathbf{A}'$.
Since row operations do not affect the right kernel of A, one can just choose the matrix $\mathbf{A}$ of this form during key generation, without affecting the security of the scheme.
Such optimization makes the protocol more efficient because multiplying by a matrix of this form requires only $(n-m) \times m$ multiplications instead of $n \times m$ multiplications for a general matrix multiplication.

Key generation optimization.

Obviously, it is not very efficient to include in the public key the matrix $\mathbf{A} = \left[\mathbf{c_i^A},\ \mathbf{i} \in \{\mathbf{1}, \cdots, \mathbf{n}\}\right]$, where c_i^A is the i-th column of $\mathcal{A}$.
The main idea is to just pick a random seed, and use a pseudo random generator PRG to expand this seed in order to obtain the matrix $\mathbf{A}$.

The public key is then composed of a random seed, and the vector $\mathbf{v}$ of length n. However, it is possible to slightly improve the algorithm better than this. We can use a seed to generate **A*** which is formed by the first $n-1$ columns $c_1^A, \cdots, c_{n-1}^A$ of $\mathbf{A}$ and the vector v.
Then, a random permutation π is picked, and one must solve for the last column c_n^A of $\mathbf{A}$ such that $\mathbf{v}_\pi$ is in the right kernel of $\mathbf{A}$.
Now, the public key only consists of a seed and a vector of length m (instead of a vector of length n). Another important advantage of this approach is there is no need to perform a Gaussian elimination this way (and in fact this was the motivation behind this optimization).

The optimized key generation procedure is given by the following algorithm Alg. 5.

Algorithm 5 KEYGEN(n,m,p)

$sk.seed \leftarrow$ Randomly sample λ $bits$
$(seed_\pi, pk.seed) \leftarrow \mathsf{PRG}_0(sk.seed)$
$\pi \leftarrow \mathsf{PRG}_1(seed_\pi)$
$(\mathbf{A}^*, \mathbf{v}) \leftarrow \mathsf{PRG}_2(pk.seed)$
Compute $\mathbf{c_n^A}$ from A^* and $\mathbf{v}_\pi$
sk $\leftarrow \mathbf{sk.seed}$
pk $\leftarrow (\mathbf{pk.seed}, \mathbf{c_n^A})$
Return (pk, sk)

Sending seeds instead of permutations.

Due to the second optimization cited above, it is useful to transfer a λ-bit seed instead of σ, if the challenge bit $b = 0$. On the other hand, when the challenge $b = 1$, one still need to transmit the permutation $\pi\sigma$, because it is not possible to generate both of the permutations σ and $\pi\sigma$ using a pseudo random generator PRG.

However, such problem can be solved. It is possible to generate $\mathbf{r}_\sigma$ using a pseudo random generator PRG, and then send its seed instead of transmitting $\pi\sigma$. The $\mathbf{r}_\sigma$ seed can be utilized to compute $\pi\sigma$, because if the verifier possesses $\mathbf{z}$ and $\mathbf{r}_\sigma$, then he is able to compute $\mathbf{z} - \mathbf{r}_\sigma = c\mathbf{v}_{\pi\sigma}$.
And since $\mathbf{v}$ and c are known, it is easy to recover $\pi\sigma$ from $c\mathbf{v}_{\pi\sigma}$ (the parameters are chosen in a way that the entries of v are all distinct, so there is a unique permutation that maps $\mathbf{v}$ to $\mathbf{v}_{\pi\sigma}$).

Moreover, sending the seed for $\mathbf{r}_\sigma$ does not reveal additional information than sending $\pi\sigma$ itself, because given $\mathbf{z}$ and $\pi\sigma$ it is trivial to compute $\mathbf{r}_\sigma$, so this optimization does not affect the security of the scheme.
However, the problem is when $c = 0$, then the $c\mathbf{v}_{\pi\sigma} = 0$, and so the verifier cannot recover $\pi\sigma$. In order

to solve this problem, the challenge space is then restricted to be $\mathbb{F}_p \setminus \{0\}$. Such condition affects the soundness error by increasing it to $\frac{p}{2p-2}$ instead of $\frac{p+1}{2p}$. But this is not a big deal.

The main advantage of this optimization is that the signature size is now constant. Without using this optimization, a response to the challenge $b = 0$ would be smaller than a response to $b = 1$. By applying the optimization described in this Section, the response to the second challenge is always a random seed, regardless the value of b.

A one round of the modified version of the original PKP-based IDS is summarized in Algorithm 6.

Algorithm 6 The modified 5-move of PKP-IDS

$\mathcal{P}(\mathsf{sk}, \mathsf{pk})$		$\mathcal{V}(\mathsf{pk})$
$\mathsf{seed}_0, \mathsf{seed}_1 \xleftarrow{\$} \{0,1\}^\lambda$		
$\sigma \leftarrow \mathsf{PRG}_1(seed_\sigma)$		
$\mathbf{r}_\sigma \leftarrow \mathsf{PRG}_2(\mathbf{r}_\sigma.seed)$		
$\mathsf{C}_0 \leftarrow \textsc{Commit}(\sigma, \mathbf{Ar})$		
$\mathsf{C}_1 \leftarrow \textsc{Commit}(\pi\sigma, \mathbf{r}_\sigma)$		
	$\xrightarrow{\mathsf{C}_0,\mathsf{C}_1}$	
		$c \xleftarrow{\$} \mathbb{F}_p \setminus \{0\}$
	$\xleftarrow{c}$	
$\mathbf{z} \leftarrow \mathbf{r}_\sigma + c\mathbf{v}_{\pi\sigma}$		
	$\xrightarrow{\mathbf{z}}$	
		$b \xleftarrow{\$} \{0,1\}$
	$\xleftarrow{b}$	
$\mathsf{rsp} \leftarrow \mathsf{seed}_b$		
	$\xrightarrow{\mathsf{rsp}}$	
		if $b = 0$ **then**
		$\sigma \leftarrow \mathsf{PRG}_1(\mathsf{rsp})$
		accept if $\mathsf{C}_0 = \textsc{Commit}(\sigma, A\mathbf{z}_{\sigma^{-1}})$
		else
		$\mathbf{r}_\sigma \leftarrow \mathsf{PRG}_2(\mathsf{rsp})$
		if $\mathbf{z} - \mathbf{r}_\sigma$ is not a permutation of $c\mathbf{v}$ **then**
		Return reject
		else
		Let $\rho \in S_n$ such that $c\mathbf{v}_\rho = \mathbf{z} - \mathbf{r}_\sigma$.
		end if
		accept if $\mathsf{C}_1 = \textsc{Commit}(\rho, \mathbf{r}_\sigma)$
		end if

9.2.3 Security Proofs of the modified scheme

This Section presents some essential definitions to an identification scheme and details the three essential properties that a zero-knowledge protocol must satisfy. We show that the modified version of PKP-IDS still verify these properties.

$2q$-Identification schemes and $2q$-extractors.

A $2q$-Identification Scheme [21] is a 5-move identification scheme, where the first challenge is drawn uniformly at random from a challenge space of size q, and the second challenge is a random bit. Therefore, a transcript of an execution of a $2q$-protocol looks like $(\mathsf{com}, c, \mathsf{rsp}_1, b, \mathsf{rsp}_2)$. We now state the properties of a $2q$-protocol more formally:

Definition 38 ($2q$-Identification scheme,[21]) *A $2q$-Identification scheme is a canonical five-move identification scheme $(\mathsf{KeyGen}, \mathcal{P}, \mathcal{V})$ with challenge spaces $\mathcal{C}h_1$ and $\mathcal{C}h_2$ for which it holds that $|\mathcal{C}h_1| = q$ and $|\mathcal{C}h_2| = 2$. Moreover, we require that the probability of the commitment com to take a certain value is a negligible function of the security parameter.*

we define the notion of a $2q$-extractor, which is an algorithm that can extract the secret key from 4 transcripts that satisfy some properties. This is useful because when there exists a $2q$-extractor for a $2q$-Identification scheme, this implies that the identification scheme has soundness with knowledge error at most $\frac{q+1}{2q}$. Moreover, this implies that applying the Fiat-Shamir transform to the identification scheme results in a secure signature scheme.

Definition 39 ($2q$-extractability) *A $2q$-identification scheme $(\mathsf{KeyGen}, \mathcal{P}, \mathcal{V})$ has $2q$-extractability, if there exists a polynomial-time algorithm that given four transcripts $(\mathsf{com}, c^{(i)}, \mathsf{rsp}_1^{(i)}, b^{(i)}, \mathsf{rsp}_2^{(i)})$ for i from 1 to 4, such that*

$$c^{(1)} = c^{(2)} \neq c^{(3)} = c^{(4)}$$
$$\mathsf{rsp}_1^{(1)} = \mathsf{rsp}_1^{(2)} \quad \mathsf{rsp}_1^{(3)} = \mathsf{rsp}_1^{(4)}$$
$$b^{(1)} = b^{(3)} \neq b^{(2)} = b^{(4)}$$

can efficiently extract a secret key.

Theorem 3
- *The modified version of PKP-IDS is complete.*
- *If the commitment scheme is computationally binding, then the scheme is sound with soundness error $\kappa = \frac{p}{2p-2}$.*
- *If the commitment scheme used is computationally hiding and the outputs of PRG_1 and PRG_2 are indistinguishable from uniform randomness, then the scheme is computationally honest-verifier zero-knowledge.*

Proof:
Completeness.

When $b = 0$, if the prover follows the protocol honestly, then the verification of the commitment will succeed if

$$\mathbf{Ar} = \mathbf{Az}_\sigma^{-1} = \mathbf{A}(\mathbf{r} + \mathbf{v}_{\pi\sigma\sigma^{-1}}),$$

which holds if and only if $\mathbf{Av}_\pi = 0$.
Therefore, if π is a solution for the permuted kernel problem, then the verifier will accept the transcript.

During an honest execution when is equal to $b = 1$, the verifier will always accept the procedure, regardless of whether π was a solution for the PKP problem or it is not.

Soundness.

First, one must show that the underlying scheme possesses a $q2$-extractor.
Therefore, we prove that, given four accepted transcripts of the following form $(\mathsf{C}_0, \mathsf{C}_1, c^{(i)}, \mathbf{z}^{(i)}, b^{(i)}, \mathsf{rsp}^{(i)})$ for i from 1 to 4, such that

$$\begin{aligned} c^{(1)} = c^{(2)} \neq c^{(3)} = c^{(4)} \\ \mathbf{z}^{(1)} = \mathbf{z}^{(2)} \quad \mathbf{z}^{(3)} = \mathbf{z}^{(4)} \\ b^{(1)} = b^{(3)} \neq b^{(2)} = b^{(4)} \end{aligned}$$

It is possible to efficiently extract a solution for the permuted kernel problem PKP.
By relabeling the transcripts if necessary, it is convenient to assume that $b^{(1)} = b^{(3)} = 0$ and $b^{(2)} = b^{(4)} = 1$.
Consider first the transcripts 1 and 3.
Let $\sigma = \mathsf{PRG}_1(\mathsf{rsp}^{(1)})$ and $\sigma' = \mathsf{PRG}_1(\mathsf{rsp}^{(3)})$, and let $\mathbf{x} = \mathbf{A}\mathbf{z}^{(1)}_{\sigma^{-1}}$ and $\mathbf{x}' = \mathbf{A}\mathbf{z}^{(3)}_{\sigma'^{-1}}$.
Since that both of the transcripts are accepted, then

$$\mathsf{C}_0 = \textsc{Commit}(\sigma, \mathbf{x}) = \textsc{Commit}(\sigma', \mathbf{x}').$$

Therefore, the computationally binding property of Com implies that, with overwhelming probability, the following holds $\sigma = \sigma'$ and $\mathbf{x} = \mathbf{x}'$.
Now, consider the transcripts 2 and 4.
Let $\mathbf{y} = \mathsf{PRG}_2(\mathsf{rsp}^{(2)})$ and $\mathbf{y}' = \mathsf{PRG}_2(\mathsf{rsp}^{(4)})$.
Since that both of the transcripts are accepted, then $\mathbf{z}^{(2)} - \mathbf{y}$ and $\mathbf{z}^{(4)} - \mathbf{y}'$ are the corresponding permutations of $c^{(2)}\mathbf{v}$ and $c^{(4)}\mathbf{v}$ respectively.
Let ρ and ρ' be the permutations such that $c^{(2)}\mathbf{v}_\rho = \mathbf{z}^{(2)} - \mathbf{y}$ and $c^{(4)}\mathbf{v}'_\rho = \mathbf{z}^{(4)} - \mathbf{y}'$.
Since that both of the transcripts are accepted, then the following holds

$$\mathsf{C}_1 = \textsc{Commit}(\rho, y) = \textsc{Commit}(\rho', y')\,,$$

so the computationally binding property of Com implies, with overwhelming probability, the following $\rho = \rho'$ and $\mathbf{y} = \mathbf{y}'$.
Now, by putting everything together, the following holds

$$\begin{aligned} 0 &= \mathbf{A}(\mathbf{z}^{(1)}_{\sigma^{-1}} - \mathbf{z}^{(3)}_{\sigma^{-1}}) \\ &= \mathbf{A}(\mathbf{z}^{(2)}_{\sigma^{-1}} - \mathbf{z}^{(4)}_{\sigma^{-1}}) \\ &= \mathbf{A}(c^{(2)}\mathbf{v}_{\rho\sigma^{-1}} - \mathbf{y}_{\sigma^{-1}} - c^{(4)}\mathbf{v}_{\rho\sigma^{-1}} + \mathbf{y}_{\sigma^{-1}}) \\ &= (c^{(2)} - c^{(4)})\mathbf{A}\mathbf{v}_{\rho\sigma^{-1}}\,. \end{aligned}$$

Since $c^{(2)} - c^{(4)}$ is nonzero, this means that $\rho\sigma^{-1}$ is a solution for the permuted kernel problem PKP.
Moreover, the extractor can efficiently extract such solution, because he is able to extract ρ from either transcript 2 or 4, and he can also extract σ from either transcript 1 or 3.

It is known that $2q$-extractability implies soundness with error $\frac{q+1}{2q}$, where q is the size of the first challenge space [21, 62].
In our case, the first challenge space has $p-1$ elements, so the optimized IDS employs a soundness error of $\frac{p}{2p-2}$.

Honest-Verifier Zero-knowledge.

To prove the property of Honest-Verifier Zero-Knowledge, one must construct a simulator that outputs transcripts that are computationally indistinguishable from transcripts of honest executions of the identification scheme.
First, the simulator picks a uniformly random value $c \in \mathbb{F}_p \setminus \{0\}$ and a random bit b (also uniformly).

The cases of $b = 0$ and $b = 1$ will be separately discussed:

Case $\mathbf{b = 0}$: The simulator picks a random seed seed_0, and a uniformly random vector $\mathbf{z}$, computes $\sigma = \mathsf{PRG}_1(\mathsf{seed}_0)$ and $\mathsf{C}_0 = \textsc{Commit}(\sigma, \mathbf{Az})$.
The simulator also commits to a dummy value in order to get C_1.
Next, the simulator outputs $(\mathsf{C}_0, \mathsf{C}_1, c, \mathbf{z}, b, \mathsf{seed}_\sigma)$.

This distribution is indistinguishable from honestly generated transcripts with $b = 0$.
Indeed, the values $c, \mathbf{z}, \mathsf{seed}_0$ are indistinguishable from uniformly random values in both simulated and honest transcripts (with the assumption that the output of PRG_2 is indistinguishable from the uniform distribution).

The first commitment $\mathsf{C}_0 = \textsc{Commit}(\sigma, \mathbf{Az}_{\sigma^{-1}})$ is a function of seed_0 and $\mathbf{z}$, so it also has the same distribution in both simulated and honest transcripts.

Finally, the commitment C_1 is never opened, so the computationally hiding property of the $\textsc{Commit}$ function guarantees that C_1 in the simulated transcript is computationally indistinguishable from the C_1 in an honest transcript.
Case $\mathbf{b = 1}$: The simulator picks a random seed seed_1 and a random permutation ρ uniformly, and computes $\mathbf{r}_\sigma = \mathsf{PRG}_2(\mathsf{seed}_1)$, $z = c\mathbf{v}_\rho + \mathbf{r}_\sigma$ and C_1.
The simulator also commits to a dummy value in order to produce a commitment C_0, then the simulator outputs the transcript $(\mathsf{C}_0, \mathsf{C}_1, c, \mathbf{z}, b, \mathsf{seed}_1)$.
The simulated transcripts are indistinguishable from honestly generated transcripts when $b = 1$.
It is clear that c and seed_1 are uniformly random values in both simulated transcripts and honestly generated transcripts.
Moreover, in both simulated and real transcripts, $\mathbf{z}$ is equal to $\mathsf{PRG}_2(\mathsf{seed}_1)+c\mathbf{v}_\rho$, and $\mathsf{C}_1 = \mathsf{Com}(\rho, \mathsf{PRG}_2(\mathsf{seed}_1))$ where ρ is indistinguishable from a uniformly random permutation (with the assumption that the output of PRG_1 is indistinguishable from a uniformly random permutation).

Therefore, $\mathbf{z}$ and C_1 have the same distribution in the simulated and the honest transcripts.

Finally, the computationally hiding properties of the COMMIT function guarantees that the value of C_0 in the simulated transcripts is indistinguishable from that of C_0 in the honestly generated transcripts.

■

9.2.4 Communication cost

This Section provides the communication complexity of N rounds of the modified version of the pkp-based IDS, where the soundness error of N iterations is given by $\kappa = \left(\frac{p}{2p-2}\right)^N$.
The commitment consists of a single hashed value, which is only 2λ bits.
The first response consists of N vectors of length n over $\mathbb{F}_p$, so this costs $Nn\lceil \log_2 p \rceil$ bits of communication.

Lastly, the final responses composed of N random λ-bit seeds, N commitments (which consist of 2λ bits each) and N commitment random strings (which consist of λ bits each), so this costs $4N\lambda$ bits of communication.

In total, the communication cost (ignoring the challenges) is

$$2\lambda + N\left(n\lceil \log_2 p \rceil + 4\lambda\right) .$$

9.3 PKP-based Digital Signature Scheme (PKP-DSS)

This Section provides the main contribution of this thesis that is to build a digital signature scheme, based on the computational hardness of the permuted kernel PKP problem.

The digital signature scheme will be derived from the modified version of the PKP-based IDS described in Section 9.2.2. It is a simple and direct application of the well-known paradigm of Fiat-Shamir [35] detailed in Section 5.4.

Note that, the derived signature scheme is the Fiat-Shamir Transform of N parallel rounds of the 5-move identification protocol. All random generations are turned into deterministic generations using Pseudo-random generators and secret seeds.

The key generation algorithm is the exact same as the key generation algorithm employed in the modified identification scheme that is detailed in Alg. 5.

In order to digitally sign a message m, the signer performs the first phase of the commitment scheme to get a commitment com. Then, the first challenge $\mathbf{c} = (c_1, \cdots, c_N)$ is computed using the message m and the commitment com by evaluating a hash function $\mathcal{H}_1(m||\mathsf{com})$.

Next, the signer proceeds the next phase of the identification protocol to get the N response vectors $\mathsf{rsp}_1 = (\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(N)})$.

After that, the signer uses a second hash function in order to obtain the challenges $\mathbf{b} = (b_1, \ldots, b_N)$ from m, com and rsp_1 as $\mathcal{H}_2(m||\mathsf{com}, \mathsf{rsp}_1)$.

Then, the last step of the identification protocol is to derive a vector composed of the second responses $\mathsf{rsp}_2 = (\mathsf{rsp}^{(1)}, \cdots, \mathsf{rsp}^{(N)})$.
Therefore, the signature is simply given by $(\mathsf{com}, \mathsf{rsp}_1, \mathsf{rsp}_2)$.

In order to verify the signature $(\mathsf{com}, \mathsf{rsp}_1, \mathsf{rsp}_2)$ for a given message m, the verifier simply applies the hash functions $\mathcal{H}_1$ and $\mathcal{H}_2$ to obtain respectively $\mathbf{c}$ and $\mathbf{b}$.
Next, the verifier checks if $(\mathsf{com}, \mathbf{c}, \mathsf{rsp}_1, \mathbf{b}, \mathsf{rsp}_2)$ is a valid transcript for the PKP-based identification protocol.

The signing and the verification algorithms are detailed in Algorithm 7 and 8.

Algorithm 7 Sign(sk, m)

derive $\mathbf{A}, \mathbf{v}$ and π from sk.
for i from 1 to N **do**
  pick λ-bit seeds $\mathsf{seed}_0^{(i)}$ and $\mathsf{seed}_1^{(i)}$ uniformly at random
  $\sigma^{(i)} \leftarrow \mathsf{PRG}_1(\mathsf{seed}_0^{(i)})$
  $\mathbf{r}_\sigma^{(i)} \leftarrow \mathsf{PRG}_2(\mathsf{seed}_1^{(i)})$
  $\mathsf{C}_0^{(i)} = \textsc{Commit}\big(\sigma^{(i)}, \mathbf{A}\mathbf{r}^{(i)}\big)$,
  $\mathsf{C}_1^{(i)} = \textsc{Commit}\big(\pi\sigma^{(i)}, \mathbf{r}_\sigma^{(i)}\big)$.
end for
$\mathsf{com} := \mathcal{H}_{\mathsf{com}}\big(\mathsf{C}_0^{(1)}, \mathsf{C}_1^{(1)}, \cdots, \mathsf{C}_0^{(N)}, \mathsf{C}_1^{(N)}\big)$
$c^{(1)}, \cdots, c^{(N)} \leftarrow \mathcal{H}_1(m||\mathsf{com})$. $c^i \in \mathbb{F}_p \setminus \{0\}$
for i from 1 to N **do**
  $\mathbf{z}^{(i)} \leftarrow \mathbf{r}_\sigma^{(i)} + c^{(i)}\mathbf{v}_{\pi\sigma^{(i)}}$
end for
$\mathsf{rsp}_1 \leftarrow (\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(N)})$
$b^{(1)}, \cdots, b^{(N)} \leftarrow \mathcal{H}_2(m||\mathsf{com}||\mathsf{rsp}_1)$
for i from 1 to N **do**
  $\mathsf{rsp}_2^{(\mathrm{i})} \leftarrow (\mathsf{seed}_{b^{(i)}}^{(i)} || \mathsf{C}_{1-b^{(i)}}^{(i)})$
end for
$\mathsf{rsp}_2 \leftarrow (\mathsf{rsp}_2{}^{(1)}, \cdots, \mathsf{rsp}_2{}^{(N)})$
Return $(\mathsf{com}, \mathsf{rsp}_1, \mathsf{rsp}_2)$

A valid signature of a message m by PKP-DSS 8 is then a tuple $(\mathsf{com}, \mathsf{rsp}_1, \mathsf{rsp}_2)$, where com, rsp_1 and rsp_2 hold the (vector of parallel) commitments and responses of the non interactive prover.

The implicit values of $\mathcal{H}_1(m||\mathsf{com})$ and $\mathcal{H}_2(m||\mathsf{com}||\mathsf{rsp}_1)$ represent the (vector of parallel) challenges of the non interactive verifier.

Algorithm 8 Verify($m, \mathsf{pk}, \sigma = (\mathsf{com}, \mathsf{rsp}_1, \mathsf{rsp}_2)$)

$c^{(1)}, \cdots, c^{(N)} \leftarrow \mathcal{H}_1\big(m||\mathsf{com}\big)$.
$b^{(1)}, \cdots, b^{(N)} \leftarrow \mathcal{H}_2(m||\mathsf{com}||\mathsf{rsp}_1)$
Parse rsp_1 as $\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(N)}$
Parse rsp_2 as $\mathsf{seed}^{(1)}, \cdots, \mathsf{seed}^{(N)}, \mathsf{C}^{(1)}_{1-b^{(1)}}, \cdots, \mathsf{C}^{(N)}_{1-b^{(N)}}$
for i from 1 to N **do**
  if $b^{(i)} = 0$ **then**
    $\sigma^{(i)} \leftarrow \mathsf{PRG}_1(\mathsf{seed}^{(i)})$
    $\mathsf{C}^{(i)}_0 \leftarrow \textsc{Commit}(\sigma^{(i)}, \mathbf{A}\mathsf{z}_{\sigma^{(i)-1}})$
  else
    $\mathbf{r}^{(i)}_\sigma \leftarrow \mathsf{PRG}_2(\mathsf{seed}^{(i)})$
    if $\mathbf{z}^{(i)} - \mathbf{r}_\sigma$ is not a permutation of $c\mathbf{v}$ **then**
      Return reject
    else
      $\pi\sigma^{(i)} \leftarrow$ the permutation that maps $c\mathbf{v}$ to $\mathbf{z}^{(i)} - \mathbf{r}_\sigma$.
    end if
    $\mathsf{C}^{(i)}_1 \leftarrow \textsc{Commit}(\pi\sigma^{(i)}, \mathbf{r}^{(i)}_\sigma)$
  end if
end for
$\mathsf{com}' := \mathcal{H}_{\mathsf{com}}\big(\mathsf{C}^{(1)}_0,\ \mathsf{C}^{(1)}_1, \cdots, \mathsf{C}^{(N)}_0,\ \mathsf{C}^{(N)}_1\big)$
Return accept if and only if $\mathsf{com} = \mathsf{com}'$

The signature protocol is essentially designed to achieve the standard security concept of a DSS: Existential-Unforgeability under Chosen Adaptive Message attacks EUF-CMA.

It is known that applying the Fiat-Shamir transform to a $2q$-extractable identification scheme results a EUF-CMA secure signature scheme in the Random Oracle Model [22, 35].
The EUF-CMA property is explicitly detailed in Section 5.3.3 in terms of an experiment. The classical method of Fiat-Shamir (FS) transforms an interactive proof of knowledge (identification scheme) into a non interactive one (signature scheme). This work is a direct application of this method to get PKP-DSS from PKP-IDS. Thus, the following security result holds:

Theorem 4 *PKP-DSS is Existential-Unforgeable under Chosen Adaptive Message Attacks (*EUF-CMA*) in the random oracle model, if*

- *the search version of the Permuted Kernel problem is intractable,*
- *the hash functions and pseudo-random generators are modeled as random oracles,*
- *the commitment functions are computationally binding, computationally hiding, and the probability that their output takes a given value is negligible in the security parameter.*

The proof is the same as in [22].

9.3.1 Generic attack

In order to achieve a security level of 2^λ bits, the cheating probability of the identification scheme based on the permuted kernel problem PKP is bounded by $2^{-\lambda}$.
Therefore, the scheme PKP-DSS must be iterated N times, where N is chosen such that

$$(\frac{p}{2p-2})^N \leq 2^{-\lambda}.$$

However, Kales and Zaverucha introduce a recent attack on MQDSS (similarly to PKP-DSS, MQDSS is a digital signature scheme built from an identification scheme using the Fiat shamir FS transform, the scheme is based on the Multivariate Quadratic MQ problem) showing that by considering the condition, given above on the number of iterations N, does not guarantee that the Fiat-Shamir signature scheme provide a λ bits level of security [43].
The authors of [43] present a generic attack that can be also applied to PKP-DSS.
The attack exploits the fact that if it is possible for an attacker to guess the first challenge **or** the second challenge, he can deliver responses that will be accepted by the verifier.

The idea is to split up the attack in two phases.
During the first phase, the attacker tries to guess the values of the N first challenges, and then utilizes these values in order to produce commitments.
Then, the attacker derives the challenges from the commitment by assuming that at least k of his N guesses are correct.

This phase requires on average

$$\mathsf{Cost}_1(N,k) = \sum_{i=k}^{N} \left(\frac{1}{p-1}\right)^k \left(\frac{p-2}{p-1}\right)^{N-k} \binom{N}{k}$$

trials.
In the second phase, the attacker tried to guess the values of the second challenges, and uses these values to generate a response.
Then, the attacker derives the second challenges with a hash function and by assuming that his guess was correct for the $N-k$ rounds of the identification protocol, where he did not guess the first challenge correctly. This phase requires on average 2^{N-k} tries.

Therefore, the total cost of the attack is

$$\min_{0\leq k\leq N} \mathsf{Cost}_1(N,k) + 2^{N-k}\,.$$

9.4 Parameter choice and Implementation

9.4.1 Parameter choice

The main parameters that affects mostly the signature scheme PKP-DSS are (p,n,m).

This Section explicitly details the choice of each set of parameters (p, n, m) following the security level required.
Note that the first identification scheme based on the permuted kernel problem PKP-IDS [63] was designed with the intention to fit small devices. Therefore, A. Shamir suggested the size of p to be equal to $p = 251$.
In order to keep the efficiency of the implementation, one must choose p to be a prime number close to a power of 2, such as 251, 509 and 4093.

A random instance of the permuted kernel problem PKP is solved by finding a kernel n-vector $(\mathbf{v}_\pi)$ with distinct coordinates in $\mathbb{F}_p$. Hence, the probability to find such a vector shouldn't be too small. The probability of an arbitrary vector to be in the kernel of the matrix $A \in \mathcal{M}_{m\times n}$ whose rank is equal to m, is p^{-m}.
Moreover, if the n-vector v has no repeated entries, its orbit under the possible permutations π contains $n!$ vectors.
Thus, to get on average one solution, the following constraint must be imposed: $n! \approx p^m$.

On the other hand, by using the complexity of our extended approach of Poupard's algorithm [60] combined with Patarin-Chauvaud's method (See Section 8.3.2.2), triplets of (p, n, m) were selected matching the security requirements and the optimizations of the signatures size.

The parameter choices presented in Table 9.1 provide a secure scheme against all the attacks described in [47].
Note that, the value of N is chosen to be just large enough such that

$$\min_{0\leq k\leq N} \mathsf{Cost}_1(N, k) + 2^{N-k} \geq 2^\lambda \,,$$

such that the scheme is secure against the generic attack of Kales and Zaverucha [43] detailed in Section 9.3.1.

The following parameter sets given in Table 9.1 are chosen to provide three different security levels.

Parameter Set	Security level	p	n	m	Iterations N	Attack cost
PKP-DSS-128	128	251	69	41	157	2^{130}
PKP-DSS-192	192	509	94	54	229	2^{193}
PKP-DSS-256	256	4093	106	47	289	2^{257}

Table 9.1: PKP-DSS Parameters sets

9.4.2 Key and signature sizes

This Section provides the sizes of the public/secret-keys and the signature of PKP-DSS with respect to the security levels and the parameters given above.

Public key.

The public key consists of the last column $\mathbf{c}_{\mathbf{n}}^{\mathbf{A}}$ of $\mathbf{A}$ and a random seed **pk.seed**, which is used to generate **A*** that is formed by all but the last column of $\mathbf{A}$ and the vector $\mathbf{v}$.
Therefore, the public key consist of $\lambda + m\lfloor \log_2(p) \rfloor$ bits.

Secret key.

The secret key is just a random seed **pk.seed** that was used to seed the key generation algorithm.
Therefore, it consists of only λ bits.

Signature.

Finally, PKP-DSS signature consists of a transcript of the identification protocol (excluding the challenges, because they are computed with a hash function).
Sect 9.2.4 shows that a transcript can be represented with $2\lambda + N\left(n\lceil \log_2 p \rceil + 4\lambda\right)$ bits, which is also the signature size.

In Table 9.2 we summarize the key and signature sizes for the parameter sets proposed in the previous section.

Security level	Parameters (p, n, m, N)	|sk| Bytes	|pk| Bytes	|sig| KiloBytes
128	$(251, 69, 41, 157)$	16	57	20.4
192	$(509, 94, 54, 229)$	24	85	45.2
256	$(4093, 106, 47, 289)$	32	103	81.1

Table 9.2: Key and signature sizes for PKP-DSS with the three proposed parameter sets.

9.4.3 Implementation

In order to demonstrate the efficiency of PKP-DSS and to compare the performance of the scheme to the other existing Fiat-Shamir signatures, a proof-of-concept implementation in plain C has been made.
The code of our implementation is available on GitHub at [14].
The hash and the commitment functions employed are represented by SHA-3, and SHAKE128 is used as a function with possible extended output.

The running time of the signing and verification algorithms is dominated by expanding seeds into random vectors and random permutations. This can be sped up by using a vectorized implementation of SHAKE128, and by using vector instructions to convert the random bitstring into a vector over $\mathbb{F}_p$ or a permutation in S_n. We leave this task for future work.

Making the implementation constant time.

Most of the key generation and signing algorithms are inherently constant time (signing branches on the value of the challenge bits b, but this does not leak information because b is public). The only problem was that applying a secret permutation to the entries of a vector, when implemented naively, involves accessing data at secret indices.

To prevent this potential timing leak, we use the "djbsort" constant time sorting code [65]. More specifically, (see Alg. 9) we combine the permutation and the vector into a single list of n integers, where the permutation is stored in the most significant bits, and the entries of the vector are stored in the least significant bits.

Then, the list of integers is sorted in constant time and the permuted vector is extracted from the low order bits. Comparing to the naive implementation, this method slows down the signing algorithm by only 11%, while, there is no significant slowdown for the key generation algorithm.

Algorithm 9 Constant time computation of $v' = v_\sigma$

1: Initialize a list of integers $L \leftarrow \emptyset$
2: $L := \big[\sigma[1] * B + v[1], \cdots, \sigma[n] * B + v[n]\big]$, where $B > n$ is a constant
3: sort L in constant time
4: $v' := \big[L[1] \ mod \ B, \cdots, L[n] \ mod \ B\big]$
5: Return v'

9.4.4 Performance results

In order to measure the performance of our implementation, experiments were carried out on a device with a i5-8250U CPU running at 1.8 GHz.

The C code was compiled with gcc version 7.4.0 with the compile option `-O3`.

Table 9.3 presents the average number of cycles counted over nearly 10000 key generations, signings, and verifications.

Security level	Parameters (p, n, m, N)	KeyGen 10^3 cycles	Sign 10^3 cycles	Verify 10^3 cycles
128	$(251, 69, 41, 157)$	72	2518	896
192	$(509, 94, 54, 229)$	121	5486	2088
256	$(4093, 106, 47, 289)$	151	7411	3491

Table 9.3: The average number of cycles for PKP-DSS with the proposed parameter sets.

9.4.5 Comparison with existing FS signatures

Table 9.4 provides a comparison of PKP-DSS to MQDSS [22], Picnic, and Picnic2 [20].

One can notice that, for all schemes, the public and secret keys are very small. The main differences between those schemes are in the signature's size and speed.

comparing to MQDSS [22], the signature sizes of PKP-DSS are roughly 30% smaller, while our scheme is faster than the other schemes by a factor of 14 (resp. 30) for the signing (resp. verification) algorithm.

Compared to Picnic [20], PKP-DSS signatures are roughly 40% smaller, and the signing and verification algorithms are 4 and 9 times faster respectively.

Compared to $\mathrm{Picinc2}$ [20] our scheme is 153 and 170 times faster for signing and verification, but this comes at the cost of 50% larger signatures.

Finally, compared to $\mathrm{SUSHSYFISH}$ [12], a different scheme based on the Permuted Kernel Problem, our scheme is 3.4 and 6.6 times faster, but at the cost of 45% larger signatures.

Security level	Scheme	Secret key (Bytes)	Public key (Bytes)	Signature (KBytes)	Sign 10^6 cycles	Verify 10^6 cycles
128	PKP-DSS-128	16	57	20.4	2.5	0.9
	MQDSS-31-48	16	46	28.0	36	27
	Picnic-L1-FS	16	32	33.2	10	8.4
	Picnic2-L1-FS	16	32	13.5	384	153
	SUSHSYFISH-1	16	72	14.0	8.6	6
192	PKP-DSS-192	24	85	45.2	5.5	2.1
	MQDSS-31-64	24	64	58.6	116	85
	Picnic-L3-FS	24	48	74.9	24	20
	Picnic2-L3-FS	24	48	29.1	1183	357
	SUSHSYFISH-3	24	108	30.8	22.7	16.5
256	PKP-DSS-256	32	103	81.1	7.4	3.5
	Picnic-L5-FS	32	64	129.7	44	38
	Picnic2-L5-FS	32	64	53.5	2551	643
	SUSHSYFISH-5	32	142	54.9	25.7	18

Table 9.4: Comparison of different post-quantum Fiat-Shamir schemes

9.4.6 Quantum analysis of PKP-DSS

The security of all known digital signature schemes depends on the intractability of certain computational problems in mathematics. Today's security proofs are reductions. The goal of such a reduction is to show that the ability of an attacker to mount a successful attack on a signature scheme implies his ability of solving a basic computational problem in mathematics.

Till now there are no quantum versions of the known attacks on PKP cited in above, as opposed to MQDSS where Grover's algorithm can be used for solving the MQ problem.
Grover's search algorithm is a quantum algorithm that can speedup the exhaustive search quadratically, for an unstructured large list of size $N = 2^k$ which is encoded in terms of a quantum black-box function $f : \{0,1\}^k \rightarrow \{0,1\}$, where $f(x) = 1$ if and only if the search condition is satisfied for a winner $x = w$, otherwise $f(x) = 0$. Grover's algorithm finds with high probability the input $x = w$ using just $\mathcal{O}(\sqrt{N})$ evaluations of f.
The following Table shows that the parameters sets proposed for our scheme are still efficient against the quantum exhaustive search.

Parameters Sets	$N = n!$	Grover's algorithm
PKP-DSS-128	$\approx 2^{326}$	2^{163}
PKP-DSS-192	$\approx 2^{485}$	2^{243}
PKP-DSS-256	$\approx 2^{565}$	2^{282}

Table 9.5: The complexity of the Quantum exhaustive search

However, since the goal is to develop a post-quantum scheme, we should also consider attackers in the Quantum Random Oracle Model (QROM). The security of the Fiat-Shamir transform in the QROM has been studied in [44, 66, 67], where the authors of [44, 66] explain that the Fiat-Shamir transform might not be secure against quantum computers. Thus, new techniques with extra properties (such as "lossy IDS") were developed to obtain a quantum-secure transform.
However, more recently, a number of works have proven the Fiat-Shamir construction secure in the QROM [30, 67] under very mild conditions. So far, none of these works apply to five-move protocols (which is the kind of protocol we are considering in this work), but it is very likely that the results can be generalized to five-move protocols, including ours. We consider this an important open problem in post-quantum cryptography. Meanwhile, we can keep the initial FIAT-SHAMIR as long as there is neither perfect proof nor quantum attack.

9.5 Conclusion

In this Chapter, we have introduced a new post-quantum secure signature scheme PKP-DSS: a Digital Signature Scheme based on the Permuted Kernel Problem (PKP) [63].

PKP is a simple NP-Hard [36] combinatorial problem that consists of finding a kernel for a publicly known matrix, such that the kernel vector is a permutation of a publicly known vector.
This problem was used to develop an Identification Scheme (IDS) which has a very efficient implementation on low-cost smart cards.

We optimized this zero-knowledge identification scheme, and then derived PKP-DSS using the traditional Fiat-Shamir transform [35] to make it non-interactive. Thus, PKP-DSS has a security that can be provably reduced, in the *(classical) random oracle model*, to the hardness of random instances of PKP (or, if wanted, to any specific family of PKP instances).

We proposed parameter sets following the thorough analysis of the State-of-the-art attacks on PKP presented in [47].
We developed a constant-time implementation of PKP-DSS [14]. We showed that PKP-DSS is competitive with other signatures derived from Zero-Knowledge identification schemes such as MQDSS, Picnic/Picnic2, and SUSHSYFISH.

The main advantages of our scheme are that signing and verification are much faster than existing Fiat-Shamir signatures and that the scheme is very simple to implement. Our implementation takes only 440 lines of C code.
Since the PKP is NP-Hard and since there are no known quantum attacks for solving PKP significantly better than classical attacks, we believe that our scheme is post-quantum secure.

PART III:

CONTRIBUTIONS ON MULTIVARIATE SIGNATURES

CHAPTER 10

Ultra-Short Multivariate Public Key Signatures

Contents

10.1 Introduction

This Chapter introduces multivariate public key signature schemes that come with “ultra”-short signature sizes. To do so, both of the signing and the verification algorithms might take up to one minute of computation using a modern personal computer. The proposed schemes are easily adaptable in order to require

only one second of computation if needed, at the cost of increasing the signatures of few additional bits; and, more generally, a trade-off has to be found between computation time and signature size, it usually depends on the final purpose of the scheme.

The ultra-short length of signatures leaves the schemes with a minor drawback in terms of the speed complexity when comparing with other multivariate-based signature schemes (such as GeMMS [19] or Quartz [25]).

10.1.1 General context.

The RSA cryptosystem [61] is one of the first public-key cryptosystems and is the most-widely used signature algorithm. The security of the RSA scheme relies on the difficulty of factoring large numbers. Following the best successful factorization technique, the so-called General Number Field Sieve GNFS [18] whose complexity is sub-exponential, RSA provides a public key of a size at least equal to 1024 bits in order to achieve a security of 80 bits (in other words, an attacker would need at least 2^{80} operations to recover the secret key), and in this case the length of the signature is at least 1024 bits. Similarly, in order to achieve a security of 128 bits, the RSA cryptosystem provides signature of length greater than 3000 bits.

Lately, the interest is rather directed towards post-quantum branch of cryptography. Post-quantum cryptography refers to cryptographic schemes that could be secure against attacks performed by quantum computers. Currently, there are mainly six different classes of post-quantum cryptography : Multivariate cryptography, Symmetric key quantum resistance, Supersingular elliptic curve isogeny cryptography, Lattice, code, and hash-based cryptography.

Multivariate-based cryptography started in 1988 with the C* algorithm of T. Matsumoto and H. Imai [51]. It was later broken by J. Patarin in [54]. Then, J. Patarin suggested a way to repair the scheme with what is called the Hidden Field Equations (HFE) in 1999 [56].

Later-on , several variations of the scheme were proposed in order to strengthen the security and overcome the weak instances of HFE, such as GeMSS [19], Quartz [25], and $\mathrm{HFE}v-$ [56].

It turns out that multivariate cryptography is the branch that successfully provides the shortest signatures among all post-quantum signature schemes. For example, GeMSS produces signatures of size 256 bits for an expected security level of 128 bits.

10.1.2 Multivariate Signatures

Multivariate cryptography is well-known nowadays due to the robustness of its signature schemes that constitute potential candidates for post-quantum cryptography. There exists a large number of promising and practical multivariate signature schemes over a finite field. In general, the existing signature schemes are classified by the size of the corresponding field into two main categories: single field schemes (such as UOV [45] and $\mathrm{Rainbow}$ [28]), and big field schemes (such as HFEv- [56]).

Signature generation and verification

Recall that the classical signature and verification process of a multivariate-based scheme over a finite field $\mathbb{F}$ requires:

- A specially chosen invertible map $\mathcal{F}^*$
- A canonical isomorphism ϕ to transform the map $\mathcal{F}^*$ into a quadratic map $\mathcal{F} = \phi \circ \mathcal{F}^* \circ \phi^{-1}$
- Two invertible affine maps $\mathcal{S}, \mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$, to hide the the original structure of $\mathcal{F}$

The public key of a multivariate signature scheme is given by a set of m polynomials in n variables: $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} = \mathcal{S} \circ \phi \circ \mathcal{F}^* \circ \phi^{-1} \circ \mathcal{T} : \mathbb{F}^n \mapsto \mathbb{F}^n$, and the secret key is given by $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{F}^*$.

In order to generate a multivariate signature on a message m, one must proceed as follows:

- Compute a hash value of the original message $h = H(m)$ using a publicly known hash function $H : \{0,1\}^* \mapsto \mathbb{F}^n$.
- Using the decomposition of $\mathcal{P}$, compute $\mathcal{S}^{-1}(h) = x$ and $X = \phi^{-1}(x)$.
- Solve $\mathcal{F}(Y) = X$ using Berlekamp's algorithm
- Compute $\phi(Y) = y \in \mathbb{F}^n$ and $z = \mathcal{T}^{-1}(y)$.

Therefore, the signature of the message m is $z \in \mathbb{F}^n$.
In order to check the veracity of a signature z, one must evaluate $\mathcal{P}(z) = h' \in \mathbb{F}^n$. If $h' = h$ holds, then the signature is accepted, otherwise it is rejected.
The main goal of this Chapter is to provide parameters in order to have ultra-short signatures. Short signatures can be useful especially for application in watermarking, QR codes, etc.

In order to design secure multivariate public key signature schemes, one must consider four essential types of attacks:

- **Type 1** refers to attacks that are valid for any public key signature of length L bits. In what follows, the required security level is of λ bits, which means that attacking the signature would require at least 2^λ operations.
- **Type 2**. The most common way to compute a signature S of a message m using a public function $\mathcal{P} : \mathbb{F}_q^n \mapsto {\mathbb{F}_q}^m$ is to first compute a hash value $h = H(m)$ using a public hash function H, then the signature is given by:
$$S = \mathcal{P}^{-1}(h).$$
To verify the signature, one must check if $\mathcal{P}(S) = h = H(m)$, where $\mathcal{P}$ is usually a function that can be easily computed based on a set of quadratic equations.
Type 2 refers to attacks that target this verification process $\mathcal{P}(S) = H(m)$.
Note that this verification process is different from the more classical one where the verification process takes as input the couple (S, m) and returns 1 if S is a valid signature of the message m and 0 otherwise.

- **Type 3** refers to attacks that target directly the underlying trapdoor $\mathcal{P}$ that is a set of m multivariate (usually quadratic) equations over a finite field $\mathbb{F}_q$.
- **Type 4.** When using an HFE based trapdoor $\mathcal{P}$ with some perturbations, Type 4 refers to attacks that target the HFE polynomial itself.

These different type of attacks, namely the generic attacks, will be detailed in next sections. Moreover, different tools will be also given in order to avoid such attacks.

10.2 Generic attacks against multivariate signatures

This Section details each of the attacks cited above. In what follows, let λ be the security level required, and L be the length of a signature S of a given message m.

10.2.1 Type 1 attacks

Let VERIFY be the verification algorithm, that takes as input the signature S and the original message m, and returns VERIFY$(S, m) = 1$ if S is a valid signature of the message m, and 0 otherwise.

In order to find a valid signature for a given message m, an attacker could check if VERIFY$(S, m) = 1$ by enumerating all the possible signatures S of length L; this brute force attack requires to run at most 2^L times the verification algorithm VERIFY. By assuming that a single execution of the algorithm VERIFY requires 2^t operations, it is possible to reach a security of λ bits (against this generic attack) as long as $L \geq \lambda - t$.

For example, in order to have a security of 80 bits with signatures of less than 80 bits, the evaluation of the VERIFY algorithm needs to be "slow". To do so, we introduce a new mode of operations called the "slow" mode of operations (see Section 10.3) that will be used to define parameters for short signatures.

However, by considering that the maximum time required to check an ultra-short signature will not exceed a minute on a current personal computer (that is to say a computer performing around 2^{31} operations per second), the computation power needed to verify a signature has to be around $2^{37} = 2^{31} \times 64$ operations per minute. Therefore, to achieve a security level of 80 bits, L must be greater or equal to $L \geq 80 - 37 = 43$ bits.

The aim of this work is to find values for L (larger than this bound, but as small as possible) that are possible to obtain for multivariate signatures.

Remark 1 *If the signature has only 60 bits of length for example, then by taking into consideration the birthday paradox, with high probability when signing 2^{30} messages, two messages might have the same signature.*
However, this may not be an issue, because these two messages are signed by a legitimate user, and therefore there is no dangerous possible attack based on the fact that the two messages possesses the

same signature.
Moreover, it is also possible to force the legitimate user to not sign more than one billion messages with the same public key in order to avoid this issue, but it seems to be unnecessary.

10.2.2 Type 2 attacks

For most of the studied signature schemes, the goal is to check if S (of the form $S = \mathcal{P}^{-1}(H(m))$) is a valid signature of a given message m by the following equality :

$$\mathcal{P}(S) = H(m),$$

where $\mathcal{P}$ is publicly given, and H is a public Hash function. Such form of equation open the way to more possible generic attacks.

Let z be the output size of the hash function H.

Hash collision attack

A collision attack finds two different inputs m_1 and m_2 of a hash function H that produce the same hash value $H(m_1) = H(m_2)$.
Since that a signature is given by $S = \mathcal{P}^{-1}(H(m))$, a hash collision of the form $H(m_1) = H(m_2)$ yields the fact that a signature of m_1 is also a valid signature of m_2.

Assume that the hash function H requires 2^t operations to compute a Hash value h.
Digital signature schemes are vulnerable to a birthday-paradox attack in order to create hash collisions. Assuming that H is a z-bit hash function that provide z bits of output, an attacker who computes only $2^{z/2}$ (or $\sqrt{2^z}$) hash operations is likely to find two matching outputs.
Therefore, a collision attack occurs when $z/2 < \lambda - t$.
In order to secure a signature scheme against the ability of brute forcing hash collisions on H, the following must hold:

$$z \geq 2\lambda - 2t.$$

Consider that the maximum time required to compute a hash value using H (as stated before for signatures verification) will not exceed a minute on a current personal computer performing around 2^{31} operations per second, or around 2^{37} operations per minute. In order to achieve a security for example of $\lambda = 80$ bits with $t \leq 37$, then z must verify :

$$z \geq 2 \times 80 - 2 \times 37 = 86 \text{ bits.}$$

Generally, there is no problem if z is greater than 86. In case of problem, it is possible to use a verification algorithm that requires to check if $\textsc{Verify}(S, m) = 1$ instead of checking if $\mathcal{P}(S) = H(m)$ in order to verify a signature. The main drawback of using $\textsc{Verify}(S, m)$ is that it limits the number of possible designs.

Collision of type $\mathcal{P}(S) = H(m)$

As cited above, let H be a hash function that requires 2^t in order to output a hash value of size z, and let $\mathcal{P}$ be a function that takes 2^w operations in order to be evaluated.
It is possible to carry out an another variation of the birthday attack as follows:

- Generate $2^{\lambda-t}$ values $H(m)$ and store the hash-values (grouped with corresponding message) such that they can be subsequently searched on hash-value.
- Generate $2^{\lambda-w}$ values $\mathcal{P}(S)$ and check for matches with any m above; continue until a match is found *i.e.* a collision of the form $\mathcal{P}(S) = H(m)$.

Due to the birthday paradox, a match can be expected when computing $z/2$ random evaluations of $\mathcal{P}$; recall that λ is the security parameter. To ensure that the parameters provide a secure scheme where no attack coming with a complexity smaller than 2^λ operations could be performed, therefore z, t, and w must fulfill the following condition:

$$2^{t+\frac{z}{2}} + 2^{w+\frac{z}{2}} \geq 2^\lambda.$$

Hence, in order to avoid this attack, the following must hold:

$$z \geq 2\lambda - t - w. \tag{10.1}$$

Let Δ denotes $\Delta = z - 2\lambda + t + w$. Hence, the condition to have a secure signature scheme given in (10.1) can be written as $\Delta \geq 0$.
In Section 10.3, Different mode of operations against the attack cited above will be presented following the sign of Δ. Note that, when $\Delta \geq 0$ more modes tend to work. By considering that $t = 37$ for example, to reach a security level of:

- 2^{80}, Δ is equal to $\Delta = z - 123 + w$.
- 2^{90}, Δ is equal to $\Delta = z - 143 + w$.
- 2^{100}, Δ is equal to $\Delta = z - 163 + w$.
- 2^{128}, Δ is equal to $\Delta = z - 219 + w$, etc.

10.2.3 Type 3 attacks

As defined above, Type 3 attacks refers to attacks that concern the underlying trapdoor $\mathcal{P} : \mathbb{F}_q^n \mapsto \mathbb{F}_q^m$ that is a set of m multivariate (usually quadratic) equations in n variables over a finite field $\mathbb{F}_q$.

In order to forge a signature $S = \mathcal{P}^{-1}(H(m))$ for a given message m, an attacker must solve a system of multivariate equations in n variables.
The Gröbner basis technique is a fundamental tool for solving systems of multivariate polynomial equations. Gröbner basis method is a non-linear generalization of Euclid's algorithm for the GCD, as well as a generalization of Gaussian elimination for linear systems. Roughly speaking, a Gröbner basis is a set of multivariate polynomials having special properties that allow easy solutions derivation for complex polynomial systems.

As a matter of fact, it is possible to transform any multivariate polynomial system, even complicated ones, into Gröbner basis form using specific algorithms (like $\mathrm{F4}$ [32] and $\mathrm{F5}$ [31] algorithms).

Recall from Section 7.2.3 that the computational complexity of such method relies strongly on an important notion, namely the degree of regularity d_{reg}, that is the minimal degree for which a set of polynomials of degree d can form a Gröbner basis, and thus can be solved (see [3, 31] for more details).
The complexity of a Gröbner basis computation detailed in Section 7.2.3 is in:

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

where $2 \leq \omega \leq 3$ is the linear algebra constant.

Note also that for random systems, the degree of regularity can be evaluated by the computation of the first non negative coefficient of a Hilbert Series, see [3].

Moreover, as detailed in Section 7.3, the Gröbner basis basis technique can be improved by combining the two general techniques for solving polynomial systems over finite fields [11] :

- **Exhaustive search technique** (brute-force search) is a very general solving technique that consists of enumerating explicitly all the possible solutions in order to find the right one. The computational cost of such method is quite large.
- **Gröbner basis computation**, defined previously.

The complexity of the Hybrid approach is given in [11]:

$$\min_{0\leq k\leq n}\left(q^k\left(\mathcal{C}_{F5}(n-k, d_{reg}^{\max}(k)) + \mathcal{O}\left((n-k)D^{\max}(k)^{\omega}\right)\right)\right),$$

where k is the number of fixed variables, $2 \leq w \leq 3$ is the linear algebra constant, $D^{max}(k)$ is the maximum number of solutions of the system in $\bar{\mathbb{F}}_q$ counted with multiplicity, and $\mathcal{C}_{F5}$ is the Gröbner basis computation complexity using the $\mathrm{F5}$ algorithm [31].
When the trade-off k is well chosen, then the hybrid approach constitutes one of the most efficient algorithms for solving polynomial systems.

Choice of the constant of linear algebra ω

For practical reasons, the linear algebra constant should be $\omega = 2.81$ due to Strassen algorithm.

However, by taking into consideration the fact that the systems are usually not dense, it could be useful, as a rough evaluation to choose $\omega = 2.37$ or $\omega = 2$.

When $q = 2$, the best-known asymptotic complexity for the direct attack is in $2^{0.792m}$ [5]. A more realistic estimation is in $2^{0.88m}$.

When $q \geq 5$, the complexity results of [11] will be used. The best-known algorithm is the Hybrid approach that performs an exhaustive search on some variables, and then performs a Gröbner base algorithm (like $\mathrm{F4}$ or $\mathrm{F5}$).

10.2.4 Type 4 attacks

The principle of Type 4 attacks is to find weakness in the trapdoor function $\mathcal{P}$ itself used to digitally sign a message m.
This Section presents the possible attacks when the employed trapdoor is an HFE based function.

Description of $\mathbf{HFE}$

Hidden field Equations (HFE [55]) cryptosystem is one the most promising asymmetric multivariate schemes. HFE provides essentially very short signature schemes, fast asymmetric encryption schemes, etc. Since that the goal is to introduce efficient ultra-short signatures, therefore the trapdoor function used to digitally sign messages will mainly be an HFE based function.

The main idea of HFE is to represent a univariate polynomial function as a set of multivariate equations (usually quadratic) in order to hide the special structure of the polynomial.

Given a finite field $\mathbb{F}_q$, the special shape of an HFE based univariate polynomial is given by (see Section 6.3):

$$\sum_{0\leq i\leq j\leq n}^{q^i+q^j\leq D} \alpha_{i,j} X^{q^i+q^j} + \sum_{0\leq i\leq n}^{q^i\leq D} \beta_i X^{q^i} + \gamma,$$

where $\alpha_{i,j},\ \beta_i, \gamma\ \in \mathbb{F}_{q^n}$, and F is of degree at most $D \in \mathbb{N}$.
The special form of F allows a multivariate representation $\mathcal{P}$ (see Eq. 6.3) over $\mathbb{F}_q^n$.

Various attacks are known on HFE. Even though all of those attacks share nearly the same sub-exponential asymptotic complexity, several variations were proposed to increase the security of HFE or its efficiency [55] such as (see Section 6.3.3 for details): the $-$ (minus), v (the external vinegar), w (the internal vinegar) modifiers. Basically, we will consider the HFEv- variant due to its robustness despite all the known attacks, moreover HFEv- is a potential variant to digital signature schemes.

Probability of having zero solutions

Given a random function f from $\mathbb{F}$ to $\mathbb{F}$, where $\mathbb{F}$ is a finite set, the probability of $f(x) - y = 0$ having at least one solution x for a given value $y \in \mathbb{F}$ is about 1-1/e (i.e. 63.2%).

Given a random homogeneous/non-homogeneous polynomial of degree 2 in n variables in $\mathbb{F}_{2^n}$, the probability of having at least one solution is also about 63%. Since the signature generation process involves the inversion of a univariate polynomial, therefore, we will need to test in average about 1.5 values in order

to find a signature.

However, given an homogeneous random polynomial of degree 2 over $\mathbb{F}_{2^n}$ when q is odd, the probability is only 31.6% (2 times less). In this case, one must test in average about 3 values in order to find a signature. The main reason behind this probability is that for every X, we have $f(X) = f(-X)$ and $X \neq -X$ (except if $X = 0$).

Therefore, we will generally choose a non-homogeneous polynomial when q is odd.
Then, the secret linear transformations S and T defined in HFE will be chosen to be affine maps (linear maps with constants).

We expect that the non-homogeneous choice (when an odd q) won't affect the security of the scheme: since S and T are linear bijective maps, there is no expected attack that is able to exploits only the degree 1 part of the public equations. However, if it appears that non-homogeneous equations are not a good choice, then one must use homogeneous solutions, with a probability divided by 2 to be invertible .

Best known attacks

HFEv- was first introduced in [56] due to J. Patarin. It is the basis of the so-called Quartz signature scheme presented in [25] (see also Appendix B for more details on Quartz). The security of the HFEv- family has been extensively studied. Hence, there are various type of attacks applicable on multivariate schemes, in particular on HFEv- based schemes:

- Key recovery attacks
- Direct attacks
- Differential attacks

A brief description of these attacks will be given next (see also Section 6.2.3 for more details).

10.2.5 Key recovery attacks

Let m be the number of public equations, a be the number of equations removed, v be the number of external vinegar variables, and let $r = \lceil \log_q(D) \rceil + (a + v)$ the value that determines the maximum degree D of HFE in order to have computations in less than a minute, and let $n = m + a$.

It is possible to attack an instance of HFEv-, that involves the parameters (m, D, r, a, v), by solving a $\mathrm{MinRank}$ instance that has the following form: the goal is to find a linear combination with coefficients in $\mathbb{F}_{q^n}$ of $K := m$ square matrices of size n (with entries in $\mathbb{F}_q$) with a small rank r or less. This reduction corresponds to the key recovery attack described in [68].

Recent improvements have been presented in [4] for solving the $\mathrm{MinRank}$ problem, and the key recovery attack is currently the most threatening attack against HFEv-; previously it was the attack described in [27]. The authors of [4] use an algebraic modeling of the $\mathrm{MinRank}$ problem in order to solve it by direct

linearization instead of using generic Gröbner basis algorithms or the hybrid approach.

The complexity of the most efficient attack is given by

$$\mathcal{O}\left(K(r+1)\left(\binom{n}{r}\binom{K+b-1}{b}\right)^2\right) \tag{10.2}$$

operations in $\mathbb{F}_{q^n}$ as long as there exists an integer $b < q$ in $\{1, \ldots, r+2\}$ that fulfills the condition given in Eq. (10.3):

$$\binom{n}{r}\binom{K+b-1}{b} - 1 \leq \sum_{i=1}^{b}(-1)^{i+1}\binom{n}{r+i}\binom{l+i-1}{i}\binom{K+b-i-1}{b-i}, \tag{10.3}$$

where $l := n$ is the number of rows of the matrices; in order to define the optimal complexity, it is possible to omit few columns to get a new instance with n' columns as mentioned in [4]. Thus, n will be replaced n' in (10.2) and (10.3), while $l := n$ must remain the same.
Note that this optimization works if and only if the new $\mathrm{MinRank}$ instance still has a single solution. Hence, the complexity of the attack is the minimum value obtained from (10.2) for valid choices of b and n'.

10.2.6 Direct algebraic attacks

Direct algebraic attacks aim to solve directly the public equations. More precisely, given a message m, the goal of an attacker is to find a valid signature $S = \mathcal{P}^{-1}(m)$ by using the public equations given by $\mathcal{P}$). A direct attack can be performed using the Gröbner basis techniques as $\mathrm{F4}$ or $\mathrm{F5}$ algorithms.
Let D denotes the degree of the HFE polynomial, a be the number of equations removed, and by v the number of (external) vinegar variables.
Let $r = \lfloor \log_q(D-1) \rfloor + 1$ be defined as the rank of the quadratic forms associated to the HFE polynomial.

As stated in Section 1, the complexity of the direct attack that involves the Gröbner basis technique is given by

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

Where d_{reg} is the degree of regularity of the system, and $2 \leq \omega \leq 3$ is the linear algebra constant. Usually, ω takes the values $\omega = 2$ or $\omega = 2.37$.
In order to determine the complexity of the Gröbner basis methodology, one must first evaluate the value d_{reg}.
An upper bound on the degree of regularity d_{reg} is defined in [29]:

$$d_{reg} \leq 2 + (q-1).(r+a+v)/2. \tag{10.4}$$

The complexity bound given in Eq. 10.4 shows that an HFEv- based scheme presents smaller degrees of regularity d_{reg} than random systems. Moreover, generally d_{reg} is significantly smaller (cf below).

Remark 2 *The Hybrid approach, that is a combination of the exhaustive search and the Gröbner basis technique (see Section 7.3), is known to be the best algorithms for solving random systems. By analyzing*

the security of the HFEv- schemes, it appears that the Hybrid algorithms does not bring remarkable improvements. Therefore, the direct algebraic attack that will be considered is the pure Gröbner basis algorithm.

As long as the degree of regularity d_{reg} of an HFEv- scheme is not larger than the d_{reg} of a random system, the experiments show that, when $q = 2$, d_{reg} can be estimated to be

$$d_{reg} = \lfloor \frac{a + v + r + 7}{3} \rfloor.$$

However, there are few results known when q is different from 2. Hence, our own experiments are made with Magma (cf Appendix E.1).
We remark that, the degree of regularity d_{reg} starts with the value q (when $D = 2$), and when $q = 5$, it increases of almost one whenever a, v or r increases by one. The situation is different when $q = 2$, where a, v, and r generally have to be increased by three.in order to increase the degree of regularity.

Therefore, we will assume that when $q \geq 5$, the degree of regularity verifies:

$$d_{reg} \geq q - 2 + r + a + v,$$

Remark 3 *When $q \geq n$, we have noticed that $d_{reg} = n + 1$. However, our cases of study concerns only the instances when $q < n$.*

10.2.7 Differential attacks

Differential attacks (defined in Section 6.2.3) are very efficient when only one monomial is used, even if some perturbations are used.
In order to avoid a differential attacks, the secret HFE polynomial will involve at least two monomials of weight two. In other words, the secret HFE polynomial will contain at least the monomials X^2 and X^{q+1} if q is not a power of two, and at least the monomials X^{q+1} and X^{2q+1} otherwise.
Note that, this type of attacks are also efficient when using the w perturbation.

10.3 Mode of operations against the birthday-paradox attack

The mode of operations define the way to deal with plain-texts and cipher-texts during the process of a cryptographic algorithm. Several mode of operations have been introduced to avoid the birthday-paradox attack.
Given a multivariate based trapdoor that can be represented as a set of m equations in n variables, if m is small, therefore the scheme is vulnerable to an efficient birthday-paradox attack against the hash function that might be the most efficient among all the possible attacks.

This Section presents various modes of operation, including our new modes, namely " Slow mode of operation", and "Multiple independent public keys mode of operation".

10.3.1 Most known mode of operations

To avoid the birthday-paradox attack against the ultra-short signatures, several ways have been designed, usually known as “modes of operation”.

Note that classical signature schemes such as RSA or ECDSA are not threatened by this kind of attack since their signature lengths are way bigger than the multivariate-based signatures, which makes the complexity of the birthday paradox attack larger than any other attacks.

Mainly, there exist the Feistel-Patarin, the Gui, the UOV, and the Dragon mode of operations:

Feistel-Patarin mode

Feistel-Patarin mode of operation was first introduced and studied by J. Patarin in [56], and then analyzed in [24] by N. Courtois. This mode of operation is used in Quartz [25] and in GeMMS [19].
The basic idea of the Feistel-Patarin mode of operation is to roughly iterate the signature process several times. More precisely, a legitimate signer uses the trapdoor $\mathcal{P}^{-1}$ in order to sign a message m, and obtain as signature $S = \mathcal{P}^{-1}(m)$, Feistel-Patarin impose the use of the computation $\mathcal{P}^{-1}$ more than once (typically 4 times).
We refer the readers to [24, 25, 56] for more details.

Gui mode

Gui mode of operation [58] generate k (typically k is equal to two) HFEv- signatures for different hash values of the same message and by the same public key.
This mode prevents the collision attack, while the length of the public key stays relatively small.
However, Gui mode increases the length of the signature by a factor k. Since that the goal is to obtain signatures as small as possible, hence, such mode of operation will not be used.

UOV mode

In a UOV scheme, the computation of the secret key remains exactly the same if some specific extra equations were added to the public key.

Therefore, there is no collision limitations with UOV: the signature can be checked as if $\textsc{Verify}(S, m) = 1$ instead of $\mathcal{P}(S) = H(m)$.
However, the length of UOV signatures is bigger than the length of HFEv- signatures. So, UOV mode of operation will not be used to develop ultra-short signatures.

Dragon mode

The public key of a multivariate cryptographic scheme is given usually by a set of m equations in n variables of the form:

$$\begin{aligned} y_1 &= p_1(x_1, \ldots, x_n) \\ y_2 &= p_2(x_1, \ldots, x_n) \\ &\vdots \\ y_m &= p_m(x_1, \ldots, x_n) \end{aligned}$$

where $p_1, p_2, \ldots, p_m$ are multivariate polynomials of small degree (typically quadratic polynomials).

Sometimes, it is possible to have a public key of the following form:

$$\begin{aligned} p_1(x_1, \ldots, x_n, y_1, \ldots, y_m) &= 0 \\ p_2(x_1, \ldots, x_n, y_1, \ldots, y_m) &= 0 \\ &\vdots \\ p_m(x_1, \ldots, x_n, y_1, \ldots, y_m) &= 0 \end{aligned}$$

where $p_1, p_2, \ldots, p_m$ are polynomials of total small degree (typically quadratic or cubic polynomials). Such systems of polynomials are called "Dragons schemes" in [53]. Dragon systems might provide shorter signatures since that they avoid Type 2 attacks (see Section 6.2.3: a signature can be verified by checking if VERIFY$(S, m) = 1$ and not if $\mathcal{P}(S) = H(m)$.

However, it is extremely risky to design a "Dragon scheme" based on an HFE trapdoor function because some attacks (such as the generalization of the Kipnis-Shamir attack) are much more efficient on Dragon schemes.
Moreover, when other trapdoor functions (other than HFE) are used, the length of the signature is much larger. For example, Dragon mode works very well with UOV signatures, but the signature length is larger than the one given by an HFE trapdoor. Therefore, in this thesis the Dragon mode also will not be used.

10.3.2 The mode of operations developed in this thesis

As stated above, we will not use the aforementioned modes of operation since they usually increase the size of the signatures, which is not suitable with our main goal that is to provide ultra-short signature schemes.

Therefore, new modes of operations, namely the "Multiple independent public keys", the "Slow" mode of operation, and both of them mixed.

The Slow mode of operation relies on the use of a slow hash function, that is to say a hash function which requires around one minute to be computed.
Indeed, this new mode of operation perfectly fulfills the requirement of ultra-short signature schemes

since it does not raise the length of the signature and it is compatible with the one minute of computation requirement described above.

10.3.2.1 Slow mode of operation

The main idea behind the slow mode of operation is the use of a slow hash function H. It can be chosen to be the iterations of a standard hash function such as SHA-3 or SHA-256.
Usually, a standard hash function requires 13 cycles per byte, which means that for a data of 72 bytes (36 words of 16 bits), it requires about 210 cycles. For instance, if one wants to build a hash function for which the value of t is 37 (*i.e.* around one minute to compute a hash value), one needs to consider using (SHA-3) , that is to say the hash function consisting in 2^{27} iterations of SHA-3.

The core of the Slow mode of operation is the slow hash function, nevertheless it is possible to parallelize the procedure. The security of our ultra-short signature scheme relies on the fact that the user or the attacker needs to perform 2^{37} operations to sign a message or to verify a signature, the time (that is one minute) is given as an indication, it is not essential to follow it. In order to have a parallelization version of our scheme, one could look for parallelizable hash functions that require 2^{37} operations.

For a given security level, this mode of operation is efficient as long as the time required by the slow hash function is acceptable. If the time required is too large, then it is possible to combine the Slow mode with other mode of operations (see Section 10.3.2.2), or even to choose an another mode.

10.3.2.2 Multiple independent public keys mode

The main idea of this mode of operation is to use a set of k independent public keys. The new public key is a set of k previous public keys. Therefore, the length of the new public key is k times the length of the previous public key, while the security of the scheme remains the same. Such mode permits to avoid attacks based on the birthday paradox, but this mode is only realistic when k is not too large.

In this work, a Slow Hash mode (see Section 10.3.2.1) will be used and sometimes a combination of the Slow Hash mode with the idea of Multiple independent public keys.

Signature generation

In order to sign a message m using the Multiple independent public key mode, one must proceed as follows:

- First compute $H(m)$ using a slow hash of m. This step requires 2^t computations (typically $t = 37$).
- Generate a random value $1 \leq R(H(m)) \leq k$, where R is a random generator of numbers between 1 and k. This value gives the number of the public keys that will be used (i.e. the m public quadratic equations that must be satisfied to sign m).
- Using the corresponding secret key, the signature is then computed.

Possible attack

It is possible to attack this mode of operation with a complexity in 2^{λ} bits using the birthday-paradox attack as follows:

- The attacker computes $2^{\lambda-t}$ values $R(H(m))$.
- The attacker selects the public key that was obtained the most. In general, it is possible to obtain about $2^{\lambda-t}/k$ values for the selected public key (see Remark below).
- The attacker then computes $2^{\lambda-w}$ values $\mathcal{P}(S)$ and looks for a collision with a value obtained in the previous step. (2^{w} denotes the time required to compute a value $\mathcal{P}(S)$).

This attack succeeds with a good probability when $2^{2\lambda-t-w} \geq k \times 2^{z}$ where z denotes the output size of the Hash function.

Since by definition Δ is equal to $\Delta = z - 2\lambda + t + w$, in order to avoid this attack, the following condition must be satisfied: $k \geq 2^{-\Delta}$. This yields an acceptable public key length only if $-\Delta$ is not too large.

Remark 4 *When $2^{\lambda-t}$ is way larger than k, the number of values obtained in step 2, for a given public key, is a variable of mean value $2^{\lambda-t}/k$, and with a standard deviation about the square root of this latter. Therefore, for the public key with the most solutions, there will be about $2^{\lambda-t}/k$ solutions as claimed.*
For example, a simple simulation that generates about 10 millions random values between 1 and 100 shows that the most obtained number appeared nearly 100 732 times in our simulation. This number is very close to 100 000 as expected.

10.4 Parameters choice

In this Section, we first present a large number of experiments and also we define different set of parameters for ultra-short signatures.

10.4.1 Gröbner basis experiments on random systems

Tables 10.1 and 10.2 show the results of our experiments with Magma limited to one minute of computation over a finite field $\mathbb{F}_q$.
These experiments evaluate, for different levels of security $\lambda = 80, 90,$ etc., the minimum values of m needed (when solving a random system of m equations in m variables) to reach a security level of 2^λ bits. Moreover, the tables provide the minimal m required to solve a system of degree two (resp. of degree three) when $m - s$ variables are known in nearly one minute of computation.

Signature size with a perfect trapdoor

Similarly to the previous Sections of the Type 1 and 2 attacks, it is possible to provide minimal parameters for ultra-short multivariate signature schemes according to Type 3 attacks.

As the verifier has a computational power up to 2^{37} operations (i.e. around one minute of compytation), the signature can be a part S' of the signature S, containing only ms elements over the considered finite field.
The value s (30 in our example cell) is chosen in a way that recovering the signature S from S' takes up to one minute by the verifier according to our MAGMA implementation.
Once m and s are known, then the length of the signature can be computed as follows:

$$L := (m - s) \times \log_2(q).$$

As stated above, Δ must not be too small in order to avoid large public keys for the multivariate signature schemes.
Recall that the goal is to have ultra-short signatures with acceptable public key such that the verification and signing processes require a reasonable time.

For example, in Table 10.3 the value of $\Delta = 10.4$ is too large. So, the purpose of the arrows and the new values on the right (m', s', L') is to set new parameters with the same level of security, this time with Δ as close to zero as possible.
Note that when Δ is positive, there is no need to raise the parameters, so there is no right part in the cells of the Tables.

Moreover, the "+" symbol next to the signature length L means that this length is a lower bound which will naturally go up while taking into account all the other possible attacks.

Remark 5 *Due to the memory restrictions with Magma (a limitation of 366 M Bytes of RAM), in some cases, there are memory limitations before even reaching the one minute of computations. Therefore, all*

the values in Tables 10.1 and 10.2 can be explicitly computed in less than two minutes and with less than 350 M bytes of RAM.

Surprisingly, Tables 10.1 and 10.2 shows that $q = 2$ does not appear so far to be the best choice for a finite field in order to produce very short signatures. Moreover, these tables shows that the length L are very similar for different values of q ($q = 5, 7, 11, \ldots$). However, the experiments given here are not sufficient: it is necessary to consider many more attacks since that the best choice for q is not at all clear yet.

Degree 2 $\omega = \mathbf{2.37}$	**80 bits**	**90 bits**	**100 bits**	**128 bits**	**192 bits**	**256 bits**
$\mathbf{q = 2}$	$m = 86$ $s = 29$ $L = 57+$	$m = 100$ $s = 30$ $L = 70+$	$m = 112$ $s = 31$ $L = 81+$	$m = 145$ $s = 33$ $L = 112+$	$m = 218$ $s = 37$ $L = 181+$	$m = 290$ $s = 42$ $L = 248+$
$\mathbf{q = 4}$	$m = 43$ $s = 21$ $L = 44+$	$m = 50$ $s = 22$ $L = 56+$	$m = 56$ $s = 23$ $L = 66+$	$m = 73$ $s = 26$ $L = 94+$	$m = 113$ $s = 30$ $L = 166+$	$m = 154$ $s = 32$ $L = 244+$
$\mathbf{q = 5}$	$m = 40$ $s = 20$ $L = 47+$	$m = 45$ $s = 20$ $L = 59+$	$m = 50$ $s = 21$ $L = 68+$	$m = 66$ $s = 24$ $L = 98+$	$m = 102$ $s = 27$ $L = 175+$	$m = 139$ $s = 30$ $L = 254+$
$\mathbf{q = 7}$	$m = 35$ $s = 19$ $L = 45+$	$m = 40$ $s = 20$ $L = 57+$	$m = 45$ $s = 20$ $L = 71+$	$m = 59$ $s = 22$ $L = 104+$	$m = 91$ $s = 26$ $L = 183+$	$m = 124$ $s = 29$ $L = 267+$
$\mathbf{q = 8}$	$m = 34$ $s = 19$ $L = 45+$	$m = 39$ $s = 19$ $L = 60+$	$m = 43$ $s = 20$ $L = 69+$	$m = 57$ $s = 23$ $L = 102+$	$m = 88$ $s = 27$ $L = 183+$	$m = 119$ $s = 29$ $L = 270+$
$\mathbf{q = 11}$	$m = 32$ $s = 18$ $L = 49+$	$m = 36$ $s = 19$ $L = 59+$	$m = 40$ $s = 20$ $L = 70$	$m = 52$ $s = 21$ $L = 108+$	$m = 81$ $s = 26$ $L = 191+$	$m = 111$ $s = 28$ $L = 288+$
$\mathbf{q = 13}$	$m = 31$ $s = 18$ $L = 49+$	$m = 35$ $s = 19$ $L = 60+$	$m = 39$ $s = 20$ $L = 71+$	$m = 51$ $s = 21$ $L = 112+$	$m = 79$ $s = 25$ $L = 200+$	$m = 107$ $s = 28$ $L = 293+$
$\mathbf{q = 16}$	$m = 30$ $s = 18$ $L = 48+$	$m = 34$ $s = 18$ $L = 64+$	$m = 38$ $s = 19$ $L = 76+$	$m = 50$ $s = 21$ $L = 116+$	$m = 77$ $s = 26$ $L = 204+$	$m = 104$ $s = 28$ $L = 304+$
$\mathbf{q = 17}$	$m = 29$ $s = 17$ $L = 50+$	$m = 33$ $s = 18$ $L = 62+$	$m = 37$ $s = 19$ $L = 74+$	$m = 49$ $s = 21$ $L = 115+$	$m = 76$ $s = 25$ $L = 209+$	$m = 103$ $s = 28$ $L = 307+$

Table 10.1: Experiments with Magma in one minute of computation on random systems of degree two ($\omega = 2.37$).

Degree 3 $\omega = 2.37$	**80 bits**	**90 bits**	**100 bits**	**128 bits**	**192 bits**	**256 bits**
$\mathbf{q = 2}$	$m = 82$ $s = 22$ $L = 60+$	$m = 92$ $s = 22$ $L = 70+$	$m = 103$ $s = 23$ $L = 80+$	$m = 131$ $s = 23$ $L = 108+$	$m = 196$ $s = 23$ $L = 173+$	$m = 262$ $s = 23$ $L = 239+$
$\mathbf{q = 4}$	$m = 40$ $s = 13$ $L = 54+$	$m = 45$ $s = 14$ $L = 62+$	$m = 50$ $s = 14$ $L = 72+$	$m = 64$ $s = 14$ $L = 100+$	$m = 96$ $s = 16$ $L = 160+$	$m = 128$ $s = 17$ $L = 222+$
$\mathbf{q = 5}$	$m = 35$ $s = 12$ $L = 54+$	$m = 39$ $s = 12$ $L = 63+$	$m = 44$ $s = 12$ $L = 75+$	$m = 56$ $s = 13$ $L = 100+$	$m = 83$ $s = 14$ $L = 161+$	$m = 112$ $s = 15$ $L = 226+$
$\mathbf{q = 7}$	$m = 29$ $s = 11$ $L = 51+$	$m = 33$ $s = 12$ $L = 59+$	$m = 36$ $s = 12$ $L = 68+$	$m = 47$ $s = 13$ $L = 96+$	$m = 71$ $s = 14$ $L = 161+$	$m = 96$ $s = 15$ $L = 228+$
$\mathbf{q = 8}$	$m = 27$ $s = 11$ $L = 48+$	$m = 31$ $s = 12$ $L = 57+$	$m = 34$ $s = 12$ $L = 66+$	$m = 45$ $s = 13$ $L = 96+$	$m = 68$ $s = 14$ $L = 162+$	$m = 92$ $s = 15$ $L = 231+$
$\mathbf{q = 11}$	$m = 25$ $s = 11$ $L = 49+$	$m = 28$ $s = 11$ $L = 63+$	$m = 31$ $s = 11$ $L = 70$	$m = 40$ $s = 12$ $L = 104+$	$m = 62$ $s = 13$ $L = 170+$	$m = 83$ $s = 14$ $L = 239+$
$\mathbf{q = 13}$	$m = 24$ $s = 11$ $L = 49+$	$m = 27$ $s = 11$ $L = 60+$	$m = 30$ $s = 11$ $L = 71+$	$m = 39$ $s = 12$ $L = 100+$	$m = 59$ $s = 13$ $L = 171+$	$m = 80$ $s = 14$ $L = 245+$
$\mathbf{q = 16}$	$m = 23$ $s = 11$ $L = 48+$	$m = 26$ $s = 11$ $L = 60+$	$m = 28$ $s = 11$ $L = 68+$	$m = 37$ $s = 12$ $L = 100+$	$m = 56$ $s = 13$ $L = 172+$	$m = 76$ $s = 14$ $L = 248+$
$\mathbf{q = 17}$	$m = 22$ $s = 10$ $L = 50+$	$m = 25$ $s = 11$ $L = 58+$	$m = 28$ $s = 11$ $L = 70+$	$m = 37$ $s = 12$ $L = 103+$	$m = 56$ $s = 13$ $L = 176+$	$m = 75$ $s = 13$ $L = 254+$

Table 10.2: Experiments with Magma in one minute of computation on random systems of degree three ($\omega = 2.37$).

10.4.2 Minimal parameters in the general case

Recall from Section 10.3.2.2 the value $\Delta = z - 2\lambda + t + w$, where z denotes the output of the hash function, 2^t the complexity of computing hash values, and 2^w the time required to compute $\mathcal{P}(S)$.

The results shown in Table 10.3 are given in the following form $\begin{bmatrix} \Delta \\ m \mapsto m' \\ s \mapsto s' \\ L \mapsto L' \end{bmatrix}$.

Degree 2 $\omega = 2$ $\Delta \to \Delta \approx 0$	**80 bits**	**90 bits**	**100 bits**	**128 bits**	**192 bits**	**256 bits**
$\mathbf{q = 2}$	$\Delta = -10.4$ $94 \to 104$ $30 \to 31$ $64 \to 73+$	$\Delta = -15.8$ $108 \to 124$ $31 \to 32$ $77 \to 92+$	$\Delta = -23.3$ $120 \to 144$ $32 \to 33$ $88 \to 111+$	$\Delta = -40.1$ $158 \to 198$ $34 \to 37$ $124 \to 162+$	$\Delta = -82.2$ $242 \to 325$ $39 \to 44$ $203 \to 281+$	$\Delta = -128.0$ $323 \to 451$ $44 \to 51$ $279 \to 400+$
$\mathbf{q = 4}$	$\Delta = -13.4$ $47 \to 54$ $21 \to 23$ $52 \to 62+$	$\Delta = -18.8$ $54 \to 64$ $23 \to 24$ $62 \to 80+$	$\Delta = -26.3$ $60 \to 73$ $24 \to 26$ $72 \to 94+$	$\Delta = -43.1$ $79 \to 100$ $27 \to 29$ $104 \to 142+$	$\Delta = -83.2$ $122 \to 164$ $31 \to 32$ $182 \to 264+$	$\Delta = -121.8$ $166 \to 227$ $32 \to 37$ $268 \to 380+$
$\mathbf{q = 5}$	$\Delta = -8$ $43 \to 47$ $20 \to 21$ $54 \to 61+$	$\Delta = -13.6$ $49 \to 55$ $21 \to 22$ $65 \to 77+$	$\Delta = -19.1$ $55 \to 63$ $22 \to 23$ $77 \to 93+$	$\Delta = -36.8$ $71 \to 87$ $25 \to 26$ $107 \to 142+$	$\Delta = -72.4$ $110 \to 142$ $28 \to 30$ $191 \to 260+$	$\Delta = -106.3$ $150 \to 196$ $31 \to 34$ $276 \to 376+$
$\mathbf{q = 7}$	$\Delta = +1$ $m = 39$ $s = 19$ $L = 56+$	$\Delta = -4.4$ $44 \to 46$ $20 \to 21$ $68 \to 70+$	$\Delta = -7.1$ $50 \to 53$ $21 \to 22$ $82 \to 87+$	$\Delta = -20.1$ $65 \to 72$ $23 \to 25$ $118 \to 132+$	$\Delta = -48.1$ $100 \to 117$ $27 \to 29$ $206 \to 247+$	$\Delta = -76.7$ $135 \to 163$ $30 \to 32$ $295 \to 367+$
$\mathbf{q = 8}$	$\Delta = +5.7$ $m = 38$ $s = 19$ $L = 57+$	$\Delta = +1.3$ $m = 43$ $s = 20$ $L = 69+$	$\Delta = -3.2$ $48 \to 50$ $21 \to 21$ $81 \to 87+$	$\Delta = -13.1$ $63 \to 68$ $23 \to 24$ $120 \to 132+$	$\Delta = -40.2$ $96 \to 1104$ $27 \to 29$ $207 \to 243+$	$\Delta = -61.9$ $131 \to 152$ $30 \to 30$ $303 \to 366+$
$\mathbf{q = 11}$	$\Delta = +12.1$ $m = 35$ $s = 19$ $L = 56+$	$\Delta = +13.5$ $m = 41$ $s = 20$ $L = 73+$	$\Delta = +7.7$ $m = 45$ $s = 21$ $L = 84+$	$\Delta = +1.1$ $m = 59$ $s = 23$ $L = 125+$	$\Delta = -14.5$ $91 \to 96$ $27 \to 27$ $222 \to 239+$	$\Delta = -30.8$ $123 \to 132$ $30 \to 30$ $322 \to 353+$
$\mathbf{q = 13}$	$\Delta = +20.9$ $m = 35$ $s = 19$ $L = 60+$	$\Delta = +16.1$ $m = 39$ $s = 20$ $L = 71+$	$\Delta = +15.2$ $m = 44$ $s = 20$ $L = 89+$	$\Delta = +8.4$ $m = 57$ $s = 22$ $L = 130+$	$\Delta = +0.7$ $m = 89$ $s = 26$ $L = 233+$	$\Delta = -11.2$ $120 \to 123$ 29 $337 \to 348+$
$\mathbf{q = 16}$	$\Delta = +23.1$ $m = 33$ $s = 18$ $L = 60+$	$\Delta = +23.7$ $m = 38$ $s = 19$ $L = 76+$	$\Delta = +24.3$ $m = 43$ $s = 20$ $L = 92+$	$\Delta = +21.4$ $m = 56$ $s = 21$ $L = 140+$	$\Delta = +15.3$ $m = 86$ $s = 26$ $L = 240+$	$\Delta = +8.5$ $m = 116$ $s = 28$ $L = 352+$
$\mathbf{q = 17}$	$\Delta = +25.7$ $m = 33$ $s = 18$ $L = 62+$	$\Delta = +26.7$ $m = 38$ $s = 19$ $L = 78+$	$\Delta = +27.7$ $m = 43$ $s = 20$ $L = 94+$	$\Delta = +21.7$ $m = 55$ $s = 21$ $L = 139+$	$\Delta = +18$ $m = 85$ $s = 26$ $L = 241+$	$\Delta = +13.7$ $m = 115$ $s = 28$ $L = 355+$

Table 10.3: Experiments with Magma in one minute of computation to avoid generic attacks ($\omega = 2$).

Table 10.3 provides potential parameters of multivariate signatures in degree 2 when using a perfect trap-

door, while Δ keeps a value close to 0, and $\omega = 2$, and when the verification process of a signature requires at most one minute with MAGMA.

Note that, the value of ω is generally not so important here: when m is increased in order to have $\Delta \approx 0$ then very often the same value m will be obtained whatever was the value of ω.

10.4.3 Our proposed ultra-short signature and its sets of parameters

This Section presents our ultra-short multivariate signature scheme based on HFEv in degree 2 and on the slow mode of operation described in Section 10.3.2.1. In order to do so, we will explain the choice of its parameters step by step. Recall that we want our signatures to be as short as possible, with reasonable public key size, and verifiable in at most about 2 minutes on a modern personal computer (1 minutes for the slow hash and 1 minute to recover the whole signature).

10.4.3.1 The choice of the degree D of the HFE polynomial

As stated previously in Section 10.2.7, the trapdoor function must contains at least two monomials in order to avoid the powerful cryptanalysis on employing only one monomial (even with perturbations [16]). So, the HFE polynomial will always have at least the two monomials: X^2 and X^{1+q} .
Given $r = \lceil \log_q(D) \rceil$.
From our Magma simulations given in Appendix D, we derive the following maximum values for r according to q (in order to spend less than one minute of computation for the verification process):

- if $q = 2$ then $r = 16$,
- if $q = 4$ then $r = 9$,
- if $q = 5$ then $r =$7,
- if $q = 7$ then $r = 6$,
- if $q = 8$ then $r = 5$,
- if $q = 11$ then $r = 5$,
- if $q = 13$ then $r = 4$,
- if $q = 16$ then $r = 4$,
- if $q = 17$ then $r = 4$.

An HFEv- scheme combines the two perturbations $-$ (minus) and v (vinegar) (detailed in Section 6.3.3) which cost almost nothing in signature s whereas they increase the security of the scheme.
Therefore, by considering an HFEv- based trapdoor, it is then possible to choose suitable (maximum) values for D compatible with one minute of computation. For other costly perturbations (such as the w perturbation), one must choose smaller values for D.

10.4.3.2 The choice of the v and $-$ parameters

When the value of D is chosen, then the required (and minimum) number of vinegar variables v and the minus $-$ parameters can be chosen in order to avoid the direct attacks and the key recovery attacks from [4].

10.4.3.3 Our candidate solution

This Section presents parameters of our ultra-short multivariate public key signature based on HFEv scheme with our new slow mode of operation.

Our parameters are given for the level of security 80 bits to have the shortest possible signatures with a "reasonable" security, then additional parameters are given for the more classical security levels: 128, 192 and 256 bits, and also for other interesting levels of security. The complexities are evaluated according to the best known attack, that is usually from the recent key recovery attack mentioned in Section 6.2.3.

Security of 80 bits								
q	r	m	a	v	Complexity	s	Signature Size (bits)	Public Key Size (KBytes)
2	16	104	0	0	84.4	31	73	69.3
4	9	54	3	4	81.6	23	76	24.9
5	7	47	5	4	81.0	21	82	21.2
7	6	39	5	5	80.2	19	85	16.3
8	5	38	6	5	80.1	19	90	17.0
11	5	35	6	6	83.7	19	97	16.6
13	4	35	7	6	83.7	19	108	18.6
16	4	33	7	6	83.5	18	112	17.4
17	4	33	7	6	83.5	18	115	17.8

Security of 90 bits								
q	r	m	a	v	Complexity	f	Signature Size (bits)	Public Key Size (kBits)
2	16	124	1	1	93.2	32	94	121.1
4	9	64	5	4	90.3	24	98	42.2
5	7	55	6	6	93.7	22	105	35.5
7	6	46	7	6	92.9	21	107	27.9
8	5	43	7	7	92.6	20	111	26.0
11	5	41	7	7	92.4	20	122	26.6
13	4	39	8	7	92.2	20	126	26.1
16	4	38	8	7	92.1	19	136	26.5
17	4	38	8	7	92.1	19	139	27.1

Security of 100 bits								
q	r	m	a	v	Complexity	s	Signature Size (bits)	Public Key Size (KBytes)
2	16	144	2	2	101.8	33	115	193.8
4	9	73	6	6	102.9	26	118	65.2
5	7	63	7	7	102.3	23	126	53.6
7	6	53	8	7	101.5	22	130	42.6
8	5	50	8	8	101.3	21	135	40.4
11	5	45	8	8	100.8	21	139	35.9
13	4	44	9	8	100.7	20	152	37.5
16	4	43	9	8	100.6	20	160	38.4
17	4	43	9	8	100.6	20	164	39.2

Security of 128 bits								
q	r	m	a	v	Complexity	s	Signature Size (bits)	Public Key Size (KBytes)
2	16	198	6	5	131.2	37	173	530
4	9	100	9	9	128.3	29	178	171.4
5	7	87	11	10	131.7	26	191	145.1
7	6	72	11	11	130.8	25	194	110.1
8	5	68	12	11	130.6	24	201	104.2
11	5	59	12	11	130.0	23	205	84.7
13	4	57	12	12	129.8	22	219	85.5
16	4	56	12	12	129.8	21	236	88.6
17	4	55	12	12	129.7	21	238	86.7

Security of 192 bits								
q	r	m	a	v	Complexity	s	Signature Size (bits)	Public Key Size (KBytes)
2	16	325	13	13	193.4	44	307	2450
4	9	164	17	17	194.4	32	332	788.8
5	7	142	18	18	193.6	30	344	641.2
7	6	117	19	18	192.4	29	351	478.5
8	5	110	19	19	192.0	29	357	444.1
11	5	96	19	19	192.1	27	371	366.6
13	4	89	20	20	195.8	26	382	337.1
16	4	86	20	20	195.6	26	400	335.9
17	4	85	20	20	195.6	26	405	334.0

Security of 256 bits								
q	r	m	a	v	Complexity	s	Signature Size (bits)	Public Key Size (kBytes)
2	16	451	21	21	257.9	51	444	6704
4	9	227	25	25	257.0	37	488	2134
5	7	196	26	26	256.8	34	509	1715
7	6	163	27	26	256.7	32	517	1309
8	5	152	27	27	256.2	30	528	1186
11	5	132	28	27	259.2	30	544	980
13	4	123	28	27	256.7	29	552	885
16	4	116	28	27	256.1	28	572	833
17	4	115	28	27	256.0	28	581	834

Variants

Many variants of the proposed ultra-short multivariate signature schemes are possible.
For instance, instead of using the $-$ and v perturbations to reach the required level of security, one could use w (the internal vinegar modifier). We expect that the result security-wise would be very similar.

Another idea is to set $r = 2$ (since there exists an attack when $r = 1$ that is not extendable to $r = 2$) and to control the complexity of the best known attack with ω, with making sure that the computation will never exceed 1 minute.
Then, the final adjustments can be done by increasing a or v as before. We also expect that the signatures lengths and the security will be similar.

So far, for our schemes, we have decided to eliminate the use of the “multiple independent public keys” mode of operation in order to avoid having large public keys. Nevertheless, a trade-off can be found between large public keys and larger signatures. For some applications, one might prefer to lose few bytes of memory to store a public key instead of having larger signatures.

Table 10.4 provides parameters when using this “multiple independent public keys” mode of operation. This shows that when the public key is larger, the signature is only a few bits smaller.

Recall that 2^w denotes the time required to compute a value $\mathcal{P}(S)$, $C1$ refers to the complexity of the aforementioned key recovery attack whereas $C2$ refers to the direct attack described in [10]. The parameters given below present the case of a “Nude" HFE trapdoor with $q = 2$, $D = 32769$ (which means $r = 16$), and with k independent public keys. The time considered to sign and verify a signature is about one minute (2^{37} operations).

m	r	w	Δ	k	s	$C1$	$C2$	Signature Size (bits)	Public Key Size
86	16	18.2	-18.8	456000	29	83.6	32.32ω	57	17 GBytes
88	16	18.3	-16.7	106000	29	83.7	32.56ω	59	4.2 GBytes
92	16	18.5	-12.5	5800	30	83.9	33.02ω	62	271 MBytes
94	16	18.6	-10.4	1352	30	84.0	33.27ω	64	67.6 MBytes
96	16	18.7	-8.3	316	30	84.1	33.47ω	66	16.8 MBytes
100	16	18.9	-4.1	18	31	84.3	33.89ω	69	1.08 MBytes
104	16	18.2	0.1	1	31	84.4	34.30ω	73	69.3 kBytes

Table 10.4: Ultra-short signatures for a security level of 2^{80} bits using the Nude HFE and k independent public keys

Remark 6 *In Table 10.4, the original (nude) version of HFE with no perturbations is used ($a = v = 0$). This may look surprising since several super-polynomial attacks are known on nude HFE and generally, it is not recommended to use nude HFE.*

However, here the goal is to have a scheme that produces very short signatures with 2^{37} operations for legitimate users and 2^{80} operations for the non-legitimate users.
Therefore, for this application, a super-polynomial attack (or even a polynomial attack) might not be a problem.

Remark 7 *The complexity of C2 is larger than 2^{80} if $32.32\omega > 80$, it means if $\omega > 2.47$. Such assumption is generally considered very reasonable.*

However, if we want to assume a smaller value of ω, then it is convenient to choose $a = 1$ and $v = 0$ (or $a = 0$ and $v = 1$) instead of $a = v = 0$. Such variation will add only one bit to the signature's length, and the degree of regularity of C2 should became equal to 8, instead of 7 (because it is expected that this degree is the integer part of $((r + a + v + 7)/3)$).
Then, the complexity of C2 becomes at least 35.6ω, that is larger than 80 if $\omega > 2.24$.

Fast Verification.

It is possible to have a slow hash computation to sign a message, and a fast verification to check the signature. For example, by adding 10 bits to the signature, the verification can be 1024 times faster. However, the aim is to have the shortest signatures, so this is not a main concern.

10.5 Discussion about our security model

10.5.1 Examples of implementation of our security model

Recall that, in this paper, all of our schemes rely on a security model where the oracles for the verification and for the computation of hash values have limited resources; that is to say that each request to one of

those oracles has a non-negligible cost for an attacker.

As our signatures are “ultra-short”, one of their most valuable interests is for instance that they could be easily spelled by a human to another one over the phone. For example in a standard PC one can manage signatures of about 64 bits that can be represented as 16 hexadecimal values such as: `45A5F352CDE20240`. This is less than the size of a strong Wifi-Box password to a smartphone for example. Moreover, if we use Alphanumeric values (Numbers $0 \dots 10$ + Letters $a \dots z$ and $A \dots Z$, i.e. 62 characters) then 64 bits represent only 11 Alphanumeric characters such as: `4fDjK457GfD`.

A concrete example involving the non-negligible cost of verification and hash computation could be the following: in order to activate a software, a user needs to enter an ultra-short signature, this signature will be given to the user over the phone; in this case, the cost of the verification would not be negligible as it would be done by the user’s personal computer. In such examples, if the user manages to get an illegal copy of the software and he is looking for a valid signature to activate it, he would have to “pay” the cost of every verification on his computer.

Another example of our security model involves QR codes; as the ultra-short signatures can fit in small QR codes, a hand device such a phone could be used to verify them. In such a case, the verification cost would not be free as well.

Last but not least, if the verification process is done online by a server that receives requests from a user (a client), it is really easy to create artificially a verification cost. Indeed, the server just has to ask the client to solve a puzzle before answering. The cost of solving the puzzle, to match our security parameters, would never be more than 1 minute or, similarly, 2^{37} bit operations.

These real-life applications of our security model seem legitimate since for all of them an attacker would require billions of cell phones or personal computers to be able to verify a signature or to compute a hash value instantly.

10.5.2 Security of our ultra-short signature scheme in a classical security model

Since we deal with multivariate-based cryptography, our scheme might resist quantum attacks. However, this work focuses only on attacks with non-quantum computers.

In the classical security model used to prove the existential unforgeability under chosen message attack (EUF-CMA), an attacker has access to an oracle that can determine if a given string of bits is a valid signature for a message. As our signature scheme involves the use of hash functions, the attacker also needs an oracle that computes hash values. If, like in the classical security model, the attacker has access to those oracle “for free” (that is to say at no extra cost than generating the request), our scheme does not hold since it relies on slow verification and hash computations. Nevertheless, this section shows that as long as the access to the hash-oracle has a non-negligible cost, our scheme remains secure only by changing a few of its parameters. Indeed, if the access to the hash-oracle is free, thus the birthday paradox would be significantly improved (such as the one given in Section 10.2).

If the access to the verification oracle is free, for a given signature of length L bits and for a security of 2^λ operations, the attacker can perform a brute force attack on the 2^L possible signatures. Usually, when $L \geq \lambda$, this is not a problem, but for the few parameters for which the length of the signature is smaller than the security λ, the parameters need to be slightly adjusted. For example, for a security level of 2^{80}, with $q = 2$ (see Section 10.4.3.3), we should set $a + v = 7$ (instead of 0), and $r = 8$ (instead of 16), then the signature length would be 80 bits instead of 73 bits. Similarly, with $q = 4$, we should set $a + v = 9$ (instead of 7), and $r = 8$ (instead of 9), then the signature length would be 80 bits as well. With these parameters, the other attacks are less efficient, so it enables us to reduce the value of D (from 65536 to 256 for the former and from 262144 to 65536 for the latter) and thus obtain significantly faster signatures.

Note that for 90 bits of security (or more), we do not have to change the parameters given in Section 10.4.3.3 since the lengths of the signatures are already larger than the security parameter λ.

Remark 8 *To ensure that our proposed signatures are secure against brute-force attacks (i.e. finding at least one valid signature among the 2^L possible bit strings), there is no need for the verification cost to be as big as 2^{37} operations. Indeed, for the two sets of parameters for which the signatures are shorter than the security parameter λ, if the verification oracle asks the user/attacker to solve a* simple *puzzle before answering his request, it is usually enough. More precisely, with $q = 2$ and 73 bit-long signatures (c.f. Section 10.4.3.3), a puzzle requiring 2^7 operations to be solved would usually be enough to reach a security of 80 bits, and with $q = 2$ and 64 bit-long signatures (Table 10.4 page 124), a puzzle in 2^{16} operations would usually be enough.*

10.6 Conclusion

At present, the shortest public key signatures are obtained with multivariate signature schemes such as Quartz, or GeMMS, with signature size typically between 128 and 256 bits, and time to sign and verify the signature in milliseconds.

In this Chapter, we have studied how to design even shorter signatures when there is no problem for the signing and verification algorithms to require nearly one minute to be performed (i.e. about 2^{37} operations).

Considering an expected security of 2^{80} bits, we have designed a signature schemes with about 64 bits. Interestingly, there are many other designs, variants, and parameters that achieve also such results.

In order to avoid the birthday-paradox attack, we have also designed some specific new modes of operations such as the “Slow Hash” and the “Multiple independent public keys” mode of operations.

The proposed signature schemes provide signatures of length much shorter than other known public key signatures schemes.
The main idea to have ultra-short signatures is to consider the following:

- The use of very slow hash functions.
- The use of many independent public keys.
- Some missing bits of the signature can be found much faster by using Gröbner basis algorithms than by using the exhaustive search.

With future progress in cryptanalysis, it is possible to have more efficient attacks against signature schemes, and therefore there will be a need to add some extra bits to the signature in order to keep the same expected security level.
As long as the best known attacks against the variants of HFE (for example the $-$ and v perturbations) are still exponential, then only a few more bits should be enough to keep the expected security level.

CHAPTER 11

Conclusions

Chapter 8 focuses on the analysis of the Permuted Kernel Problem (PKP). PKP is the problem of finding a permutation of a known vector such that the resulting vector is in the kernel of a given matrix.

It was proved to be an NP-Hard combinatorial problem [63].
PKP requires simple operations which involve basic linear algebra computations. Due to its simplicity, the problem has received significant attentions from theory and application in cryptography.

Chapter 8 provides the theoretical analysis behind the PKP problem over a finite field. We were essentially concerned about this problem because it can be used to build a post-quantum signature scheme based on the hardness of solving random instances of PKP.
Since that quantum computers are expected to be incapable to solve *NP*-Hard problems [8], therefore,algebraic problems such as PKP, are very interesting nowadays.

We presented an updated complexity analysis for the most efficient algorithm for solving instances of the Permuted Kernel Problem.
Many researchers investigated the security of PKP, and proposed several solving algorithms.
All the suggested attacks combine exhaustive search with some form of time-memory trade-off.

In Chapter 8, we performed a complexity analysis of all the existing attacks for solving the Permuted Kernel Problem. Interestingly, it appears that the complexity bound given in Joux-Jaulmes paper [40] is not quite precise. Therefore, we showed that Joux-Jaulmes attack is not the best algorithm for solving PKP.
In order to estimate a concrete security of PKP, we reviewed and compared the best known attack's efficiency, in terms of the number of operations performed, for different finite fields.

After all, we have been able to bring together Patarin-Chauvaud attack [57] and Poupard algorithm [60] to provide an accurate program for solving PKP. This latter yields better security estimates that contribute for secure parameters sets of the Permuted Kernel Problem.
After updating the complexity bounds of PKP's best algorithms, we performed a Magma implementation for our program to identify hard instances of PKP, and therefore, to define secure sets of parameters of this problem for different security levels and with respect to the best attack currently known.

After reviewing in Chapter 8 the complexity analysis of the Permuted Kernel Problem, we were particularly interested in the design of a post-quantum digital signature scheme based on PKP.
In 1989, A. Shamir [63] introduced a five-move Zero-Knowledge Identification scheme (ZK-IDS), based

on the complexity of PKP.
Chapter 9 uses the well-known Fiat-Shamir (FS) transform [35] on this identification scheme to develop a digital signature scheme that is provably secure in the Random Oracle Model (ROM).
The FS method is important since it yields efficient signature schemes in terms of minimal and sufficient security assumptions.
The main contribution of Chapter 9 is to present PKP-DSS: a Digital Signature Scheme based on PKP that is post-quantum secure. We used the Fiat-Shamir transform to build PKP-DSS from the 5-move PKP identification scheme introduced by Shamir [63].
Following the complexity analysis of PKP [47] given in Chapter 8, we have choosen secure parameter sets of the signature scheme for $128/192/256$ bits of classical security level.

Till now, there are no known quantum algorithms for solving PKP (other than combining Grover search with the classical algorithms), so we claimed that our signatures achieve the NIST security levels I/III and V respectively.
However, we recognize that the (quantum) hardness of PKP deserves more research, and we hope that this work will inspire researchers to investigate this topic further.

In Chapter 9, we have developed a constant-time C implementation of the new signature scheme. By constant-time we mean that the running time and the memory access pattern of the implementation are independent of secret material, therefore blocking attacks from timing side channels.
The resulting signature scheme compares well with other schemes of the same nature as MQDSS and Picnic/Picnic2. Our scheme is much faster than MQDSS and Picnic/Picnic2 in terms of signing and verification, we have small public and private keys, and the signature sizes of our scheme are comparable to those of MQDSS and Picnic2. This makes our signature scheme PKP-DSS competitive with state of the art post-quantum signature schemes.

PKP-DSS took part of the Chinese competition for the standardization of new post-quantum cryptographic schemes organized by the Chinese association CACR. PKP-DSS won the third prize of the competition.

Chapter 10 studies another type of signature schemes: multivariate public key signature schemes with "ultra"-short signatures over finite fields.
In order to build signature schemes with ultra-short signatures, we investigated an efficient idea that is the time constraint: signing and verifying a signature could require about one minute of computation on a modern personal computer with a 3GHz frequency processor, a computation power around $3.10^9 \times 60 \approx 2^{37}$ word operations.
Chapter 10 describes generic attacks against multivariate signature schemes. We used these attacks to determine inferences for optimal parameters that can provide secure ultra-short multivariate signature schemes.
Chapter 10 provided explicit examples of ultra-short multivariate signature schemes based on the Hidden Field Equations (HFE) variants.
Additionally, we provided various parameters for such schemes with respect to different security levels (from 80 to 256 bits) and over different finite fields (from $\mathbb{F}_2$ to $\mathbb{F}_{17}$).

CHAPTER 12

Conclusions

Le chapitre 8 se concentre sur l'analyse du problème du noyau permuté (PKP). PKP est le problème de trouver une permutation d'un vecteur connu tel que le vecteur résultant soit dans le noyau d'une matrice donnée.

Il s'est avéré être un problème combinatoire NP-dur [63].
PKP nécessite des opérations simples qui impliquent des calculs d'algèbre linéaire de base. En raison de sa simplicité, le problème a reçu une attention particulière de la part de la théorie et de l'application en cryptographie.

Le chapitre 8 fournit l'analyse théorique du problème PKP sur un corps fini. Nous étions essentiellement intéressés par ce problème car il peut être utilisé pour construire un schéma de signature post-quantique basé sur la dureté de la résolution d'instances aléatoires de PKP.
Comme on s'attend à ce que les ordinateurs quantiques soient incapables de résoudre les problèmes *NP* -Hard s [8], par conséquent, les problèmes algébriques tels que PKP sont très intéressants de nos jours.

Nous avons présenté une analyse de complexité mise à jour pour l'algorithme le plus efficace pour résoudre les instances du problème du noyau permuté.
De nombreux chercheurs ont étudié la sécurité de PKP et ils ont proposé plusieurs algorithmes de résolution.
Toutes les attaques suggérées combinent une recherche exhaustive avec une forme de compromis temps-mémoire.

Dans le chapitre 8, nous avons effectué une analyse de complexité de toutes les attaques existantes pour résoudre le problème du noyau permuté. Il semble que la borne de complexité donnée dans l'article de Joux-Jaulmes [40] n'est pas tout à fait précise. Par conséquent, nous avons montré que l'attaque Joux-Jaulmes n'est pas le meilleur algorithme pour résoudre PKP.
Afin d'estimer une sécurité concrète de PKP, nous avons examiné et comparé l'efficacité de l'attaque la plus connue, en termes de nombre d'opérations effectuées, pour différents corps finis.

Nous avons pu rassembler l'attaque de Patarin-Chauvaud [57] et l'algorithme de Poupard [60] pour fournir un programme précis pour résoudre PKP. Ce dernier produit de meilleures estimations de sécurité qui contribuent aux ensembles de paramètres sécurisés du problème du noyau permuté.
Après avoir mis à jour les bornes de complexité des meilleurs algorithmes de PKP, nous avons effectué une implémentation Magma pour notre programme afin d'identifier les instances efficaces de PKP, et par

conséquent, de définir des ensembles sécurisés de paramètres de ce problème pour différents niveaux de sécurité et par rapport à la meilleure attaque connue actuellement.

Après avoir examiné dans le chapitre 8 l'analyse de complexité du problème du noyau permuté, nous nous sommes particulièrement intéressés par la conception d'un schéma de signature digitale post-quantique basé sur PKP.
En 1989, A. Shamir [63] a introduit un schéma d'identification à connaissance zéro en cinq passes ($\mathrm{ZK\text{-}IDS}$), basé sur la complexité de PKP.
Le chapitre 9 utilise la célèbre transformation Fiat-Shamir (FS) [35] sur ce schéma d'identification pour développer un schéma de signature digitale qui est prouvé sécurisé dans le Random Oracle Model (ROM). La méthode FS est importante car elle produit des schémas de signature efficaces en termes d'hypothèses de sécurité minimales et suffisantes.
La principale contribution du chapitre 9 est de présenter PKP-DSS: un schéma de signature digitale basé sur PKP qui est post-quantique sécurisé. Nous avons utilisé la transformation Fiat-Shamir pour construire PKP-DSS à partir du schéma d'identification à 5 passes PKP introduit par Shamir [63].
Suite à l'analyse de complexité de PKP [47] donnée dans le chapitre 8, nous avons choisi des ensembles de paramètres sécurisés du schéma de signature pour les niveaux de sécurité classiques $128/192/256$ bits.

Jusqu'à présent, il n'y a pas d'algorithme quantique connu pour résoudre PKP (autre que de combiner la recherche de Grover avec les algorithmes classiques), nous avons donc affirmé que nos signatures atteignent respectivement les niveaux de sécurité induiqués par le NIST I/III et V.
Cependant, nous reconnaissons que la dureté (quantique) de PKP mérite plus de recherche, et nous espérons que ce travail inspirera les chercheurs à approfondir ce sujet.

Dans le chapitre 9, nous avons développé une implémentation C en temps constant du nouveau schéma de signature. Par temps constant, nous entendons que le temps d'exécution et le modèle d'accès à la mémoire de l'implémentation sont indépendants du matériel secret, bloquant ainsi les attaques temporelles. Le schéma de signature qui en résulte se compare bien avec d'autres schémas de même nature que MQDSS et Picnic/Picnic2. Notre schéma est beaucoup plus rapide que MQDSS et Picnic/Picnic2 en termes de signature et de vérification, nous avons de petites clés publiques et privées, et les tailles de signature de notre schéma sont comparables à celles de MQDSS et Picnic2. Cela rend notre schéma de signature PKP-DSS compétitif par rapport aux schémas de signature post-quantique de même nature.

On a participé avec PKP-DSS à une compétition chinoise pour la standardisation de nouveaux schémas cryptographiques post-quantiques organisé par l'association chinoise CACR. PKP-DSS a gagné le troisième prix de la compétition.

Le chapitre 10 étudie un autre type de schémas de signature: les schémas de signature à clé publique multivariée avec "ultra"-courtes sur des corps finis.
Afin de construire des schémas de signature avec des signatures ultra-courtes, nous avons étudié une idée efficace qui est la contrainte de temps: la signature et la vérification d'une signature peuvent nécessiter environ une minute de calcul sur un ordinateur personnel moderne avec un processeur de fréquence

3 GHz, une puissance de calcul d'environ 3,10 $3.10^9 \times 60 \approx 2^{37}$ opérations de mots.
Le chapitre 10 décrit les attaques génériques contre les schémas de signatures multivariées. Nous avons utilisé ces attaques pour déterminer des inférences pour des paramètres optimaux qui peuvent fournir des schémas de signatures multivariées ultra-courtes sécurisées.
Le chapitre 10 fourni des exemples explicites de schémas de signatures multivariées ultra-courtes basées sur des variantes de (HFE).
De plus, nous avons fourni divers paramètres pour de tels schémas en ce qui concerne différents niveaux de sécurité (de 80 à 256 bits) et sur différents corps finis (de $\mathbb{F}_2$ à $\mathbb{F}_{17}$).

Bibliography

[1] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. “From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security”. In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2002, pp. 418–433.

[2] William Wells Adams, William W Adams, WILLIAM HOWARD ADAMS, Philippe Loustaunau, and Wm W Adams. *An introduction to Grobner bases*. 3. American Mathematical Soc., 1994.

[3] Magali Bardet. “Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie”. PhD thesis. 2004.

[4] Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier Verbel. “Improvements of Algebraic Attacks for solving the Rank Decoding and MinRank problems”. In: (2020).

[5] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Pierre-Jean Spaenlehauer. “On the complexity of solving quadratic boolean systems”. In: *Journal of Complexity* 29.1 (2013), pp. 53–75.

[6] Thierry Baritaud, Mireille Campana, Pascal Chauvaud, and Henri Gilbert. “On the security of the permuted kernel identification scheme”. In: *Annual International Cryptology Conference*. Springer. 1992, pp. 305–311.

[7] Mihir Bellare and Phillip Rogaway. “Random oracles are practical: A paradigm for designing efficient protocols”. In: *Proceedings of the 1st ACM conference on Computer and communications security*. 1993, pp. 62–73.

[8] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. “Strengths and weaknesses of quantum computing”. In: *SIAM journal on Computing* 26.5 (1997), pp. 1510–1523.

[9] Daniel J Bernstein. “Introduction to post-quantum cryptography”. In: *Post-quantum cryptography*. Springer, 2009, pp. 1–14.

[10] Luk Bettale, Jean-Charles Faugere, and Ludovic Perret. “Cryptanalysis of HFE, multi-HFE and variants for odd and even characteristic”. In: *Designs, Codes and Cryptography* 69.1 (2013), pp. 1–52.

[11] Luk Bettale, Jean-Charles Faugere, and Ludovic Perret. “Hybrid approach for solving multivariate systems over finite fields”. In: *Journal of Mathematical Cryptology* 3.3 (2009), pp. 177–197.

[12] Ward Beullens. “On sigma protocols with helper for MQ and PKP, fishy signature schemes and more.” In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 490.

[13] Ward Beullens, Jean-Charles Faugère, Eliane Koussa, Gilles Macario-Rat, Jacques Patarin, and Ludovic Perret. “PKP-based signature scheme”. In: *International Conference on Cryptology in India*. Springer. 2019, pp. 3–22.

[14] Ward Beullens, Eliane Koussa, Gilles Macario-Rat, and Jacques Patarin. “PKPDSS (2019) Public GitHub repository”. In: URL: `https://github.com/WardBeullens/PKPDSS`.

[15] Ward Beullens, Bart Preneel, Alan Szepieniec, and Frederik Vercauteren. “LUOV: Lifted Unbalanced Oil and Vinegar, NIST Submission, 2017”. In: *URL: https://csrc. nist. gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions* ().

[16] Charles Bouillaguet, Pierre-Alain Fouque, and Gilles Macario-Rat. “Practical key-recovery for all possible parameters of SFLASH”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2011, pp. 667–685.

[17] Bruno Buchberger. “Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal”. In: *PhD thesis, Universitat Innsbruck* (1965).

[18] Joe P Buhler, Hendrik W Lenstra, and Carl Pomerance. “Factoring integers with the number field sieve”. In: *The development of the number field sieve*. Springer, 1993, pp. 50–94.

[19] A Casanova, JC Faugère, G Macario-Rat, J Patarin, L Perret, and J Ryckeghem. “GeMSS: A Great Multivariate Short Signature, NIST Submission, 2017.[On-line]”. In: *Internet: https://csrc. nist. gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions* ().

[20] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. “Post-quantum zero-knowledge and signatures from symmetric-key primitives”. In: *Proceedings of the 2017 acm sigsac conference on computer and communications security*. 2017, pp. 1825–1842.

[21] Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe. “From 5-Pass MQ-Based Identification to MQ-Based Signatures”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2016, pp. 135–165.

[22] Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe. “MQDSS specifications”. In: *NIST PQC Round* 2 (2018), p. 13.

[23] Nicolas T Courtois. “Efficient zero-knowledge authentication based on a linear algebra problem MinRank”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2001, pp. 402–421.

[24] Nicolas T Courtois. “Short Signatures, Provable Security, Generic Attacks and Computational Security of Multivariate Polynomial Schemes such as HFE, Quartz and Sflash.” In: *IACR Cryptol. ePrint Arch.* 2004 (2004), p. 143.

[25] Nicolas Courtois, Louis Goubin, and Jacques Patarin. *Quartz, 128-bit long digital signatures: in cryptographers’ Track Rsa Confrence 2001, LNCS 2020*.

[26] David Cox, John Little, and Donal OShea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer Science & Business Media, 2013.

[27] Jintai Ding, Ray Perlner, Albrecht Petzoldt, and Daniel Smith-Tone. “Improved cryptanalysis of hfev-via projection”. In: *International Conference on Post-Quantum Cryptography*. Springer. 2018, pp. 375–395.

[28] Jintai Ding and Dieter Schmidt. “Rainbow, a new multivariable polynomial signature scheme”. In: *International Conference on Applied Cryptography and Network Security*. Springer. 2005, pp. 164–175.

[29] Jintai Ding and Bo-Yin Yang. “Degree of regularity for HFEv and HFEv”. In: *International Workshop on Post-Quantum Cryptography*. Springer. 2013, pp. 52–66.

[30] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. “Security of the Fiat-Shamir transformation in the quantum random-oracle model”. In: *Annual International Cryptology Conference*. Springer. 2019, pp. 356–383.

[31] Jean Charles Faugère. “A new efficient algorithm for computing Gröbner bases without reduction to zero (F 5)”. In: *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*. 2002, pp. 75–83.

[32] Jean-Charles Faugere. “A new efficient algorithm for computing Gröbner bases (F4)”. In: *Journal of pure and applied algebra* 139.1-3 (1999), pp. 61–88.

[33] Jean-Charles Faugère, Pierrick Gaudry, Louise Huot, and Guénaël Renault. “Polynomial systems solving by fast linear algebra”. In: *arXiv preprint arXiv:1304.6039* (2013).

[34] Jean-Charles Faugere, Patrizia Gianni, Daniel Lazard, and Teo Mora. “Efficient computation of zero-dimensional Gröbner bases by change of ordering”. In: *Journal of Symbolic Computation* 16.4 (1993), pp. 329–344.

[35] Amos Fiat and Adi Shamir. “How to prove yourself: Practical solutions to identification and signature problems”. In: *Conference on the theory and application of cryptographic techniques*. Springer. 1986, pp. 186–194.

[36] MR Garey and DS Johnson. *Computers and Intractability–A guide to NP-completeness.(1979)*.

[37] Jean Georgiades. “Some remarks on the security of the identification scheme based on permuted kernels”. In: *Journal of Cryptology* 5.2 (1992), pp. 133–137.

[38] Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. “A digital signature scheme secure against adaptive chosen-message attacks”. In: *SIAM Journal on Computing* 17.2 (1988), pp. 281–308.

[39] Louis Goubin and Nicolas T Courtois. “Cryptanalysis of the TTM cryptosystem”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2000, pp. 44–57.

[40] Éliane Jaulmes and Antoine Joux. “Cryptanalysis of pkp: a new approach”. In: *International Workshop on Public Key Cryptography*. Springer. 2001, pp. 165–172.

[41] David S Johnson. “The NP-completeness column: an ongoing guide”. In: *Journal of Algorithms* 6.3 (1985), pp. 434–451.

[42] A Joux and R Lercier. “Chinese and Match, an alternative to Atkin’s Match and Sort method used in the SEA algorithm.(1999)”. In: *Preprint* ().

[43] D Kales and G Zaverucha. *Forgery attacks against MQDSSv2. 0 Note postes on the NIST PQC forum*.

[44] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. “A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2018, pp. 552–586.

[45] Aviad Kipnis, Jacques Patarin, and Louis Goubin. “Unbalanced oil and vinegar signature schemes”. In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 1999, pp. 206–222.

[46] Aviad Kipnis and Adi Shamir. “Cryptanalysis of the HFE public key cryptosystem by relinearization”. In: *Annual International Cryptology Conference*. Springer. 1999, pp. 19–30.

[47] Eliane Koussa, Gilles Macario-Rat, and Jacques Patarin. “On the complexity of the Permuted Kernel Problem.” In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 412.

[48] Rodolphe Lampe and Jacques Patarin. “Analysis of Some Natural Variants of the PKP Algorithm.” In: *IACR Cryptol. ePrint Arch.* 2011 (2011), p. 686.

[49] Daniel Lazard. “Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations”. In: *European Conference on Computer Algebra*. Springer. 1983, pp. 146–156.

[50] Daniel Lazard. “Solving systems of algebraic equations”. In: *ACM SIGSAM Bulletin* 35.3 (2001), pp. 11–37.

[51] Tsutomu Matsumoto and Hideki Imai. “Public quadratic polynomial-tuples for efficient signature-verification and message-encryption”. In: *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer. 1988, pp. 419–453.

[52] T Moh. “A public key system with signature and master key functions”. In: *Communications in Algebra* 27.5 (1999), pp. 2207–2222.

[53] Jacques Patarin. “Asymmetric cryptography with a hidden monomial”. In: *Annual International Cryptology Conference*. Springer. 1996, pp. 45–60.

[54] Jacques Patarin. “Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt’88”. In: *Annual International Cryptology Conference*. Springer. 1995, pp. 248–261.

[55] Jacques Patarin. “Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms”. In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 1996, pp. 33–48.

[56] Jacques Patarin. “La Cryptographie Multivariable”. In: *Mémoire d’habilitationa diriger des recherches de l’Université Paris* 7 (1999), p. 354.

[57] Jaques Patarin and Pascal Chauvaud. “Improved algorithms for the permuted kernel problem”. In: *Annual International Cryptology Conference*. Springer. 1993, pp. 391–402.

[58] Albrecht Petzoldt, Ming-Shing Chen, Bo-Yin Yang, Chengdong Tao, and Jintai Ding. “Design principles for HFEv-based multivariate signature schemes”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2015, pp. 311–334.

[59] David Pointcheval and Jacques Stern. “Security proofs for signature schemes”. In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 1996, pp. 387–398.

[60] Guillaume Poupard. “A realistic security analysis of identification schemes based on combinatorial problems”. In: *European transactions on telecommunications* 8.5 (1997), pp. 471–480.

[61] Ronald L Rivest, Adi Shamir, and Leonard Adleman. “A method for obtaining digital signatures and public-key cryptosystems”. In: *Communications of the ACM* 21.2 (1978), pp. 120–126.

[62] Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari. “Public-key identification schemes based on multivariate quadratic polynomials”. In: *Annual Cryptology Conference*. Springer. 2011, pp. 706–723.

[63] Adi Shamir. “An efficient identification scheme based on permuted kernels”. In: *Conference on the Theory and Application of Cryptology*. Springer. 1989, pp. 606–609.

[64] Peter W Shor. “Algorithms for quantum computation: discrete logarithms and factoring”. In: *Proceedings 35th annual symposium on foundations of computer science*. Ieee. 1994, pp. 124–134.

[65] “The djbsort software library for sorting arrays of integers or floating-point numbers in constant time”. In: URL: https://sorting.cr.yp.to/.

[66] Dominique Unruh. “Non-interactive zero-knowledge proofs in the quantum random oracle model”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2015, pp. 755–784.

[67] Dominique Unruh. “Post-quantum security of Fiat-Shamir”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2017, pp. 65–95.

[68] Jeremy Vates and Daniel Smith-Tone. “Key recovery attack for all parameters of HFE”. In: *International Workshop on Post-Quantum Cryptography*. Springer. 2017, pp. 272–288.

[69] Christopher Wolf, An Braeken, and Bart Preneel. “Efficient cryptanalysis of rse (2) pkc and rsse (2) pkc”. In: *International Conference on Security in Communication Networks*. Springer. 2004, pp. 294–309.

[70] Bo-Yin Yang and Jiun-Ming Chen. “Building secure tame-like multivariate public-key cryptosystems: The new TTS”. In: *Australasian Conference on Information Security and Privacy*. Springer. 2005, pp. 518–531.

APPENDICES

APPENDIX A

Magma code for the complexity of PKP

```
//The first algorithm A1
m:=Round(Log(p,Factorial(n)));
m+:=1; //georgA1:=function(n,p);iades
SpaceMax:=8*10^15;
TimeMin:=2^1000;

for k in [1..m] do;
 l:=(n-m+k) div 2;
 r:=n-m+k-r;
 F0:=1.0*Factorial(n) / Factorial(n-l);
 size0:=F0;
  if size0 gt SpaceMax then continue; end if;
 nb1:=1.0*Factorial(n)/Factorial(n-r);
 nb2:=1.0*Factorial(n)/Factorial(n-(l+r))/p^k;
 tps0:= F0;//*(k*l);
 tps1:= nb1;//*(r-k)*k;
 tps2:= nb2;//*(l+r-k)*(m-k);
 tps:=tps0+tps1+tps2;
  if tps lt TimeMin then
    TimeMin:=tps;
    U:=<Log(2,tps),Log(10,size0), Log(2,tps0), Log(2,tps1),
    Log(2,tps2), Log(2,nb1),Log(2,nb2),l, r, k>;
  end if;
end for;

return U;
end function;

// The second algorithm : A2
//the use of sets instead of tuples
for given n,p;
m:=Round(Log(p,Factorial(n)));
m+:=1; //georgiades
SpaceMax:=8*10^15;
TimeMin:=2^1000;
```

```
 for k in [1..m], l in [1..n-m] do;
  r:=n-m-l+k;
  nb0:=Binomial(n,r);
  F0:=1.0*Factorial(n-r) / Factorial(n-l-r);
  size0:=F0*l;
   if size0 gt SpaceMax then continue; end if;
  nb1:=1.0*Factorial(r);
  nb2:=1.0*nb1*F0/p^k;
  tps0:= F0;//*(k*l);
  tps1:= nb1;//*(r-k)*k;
  tps2:= nb2;//*(l+r-k)*(m-k);
  tps:=nb0*(tps0+tps1+tps2);
   if tps lt TimeMin then
     TimeMin:=tps;
     U:=<Log(2,tps),Log(10,size0), Log(2,tps0), Log(2,tps1),
     Log(2,tps2), Log(2,nb1),Log(2,nb2),l, r, k>;
   end if;
  end for;
 U; exit;

poupard:= function(n,p);    //poupard
m:=Round(Log(p,Factorial(n)));
m+:=1; //georgiades
SpaceMax:=8*10^150;
TimeMin:=2^1000;
 for j in [0..n-m], i in [0..n-m-j],
 l in [0..n-m-j-i], c in [0..m], d in [0..c], cc in [0..m-c] do;
  r:=n-m-i-j;
  k:=r+d-l;
  mm:=c-d;
  nf0:=Factorial(n)/Factorial(n-j);
  size_f0:=nf0*j;
  time_f0:=nf0*(j*c);
  nf2:=Factorial(n)/Factorial(n-l);
  size_f2:=nf2*l;
  time_f2:=nf2*(l*d);
  nf1:=Ceiling(Factorial(n)/Factorial(n-(i+j))/p^c);
  size_f1:=nf1*(i+j);
  time_f1:=Factorial(n)/Factorial(n-i)*(c*i) // -> compute candidate F0
  +nf1*((i+j)*cc);
  size0:=size_f0+size_f1+size_f2;
   if size0 gt SpaceMax then continue; end if;
  //choice of k-tuples
```

```
    nb1:=Factorial(n)/Factorial(n-k); // k-tuples_V
    time_1:=nb1*((r-l)*d); //calcul entr e F2
   //solution F2 in (l+k)
    nb2:=1.0*Factorial(n)/Factorial(n-(k+l))/p^d;
    time_2:=1+nb2*(r*mm); //-> donne candidat M
   //M (compatible KL)
    nb3_:=1.0*nb2*Factorial(n-mm)*Factorial(n-(k+l))
     /Factorial(n)/Factorial(n-(k+l+mm));
   nb3:=1.0* Factorial(n-mm)/Factorial(n-(k+l+mm))/p^d;
  //We complete with N
   nb4:=nb3*Factorial(n-(k+l+mm))/Factorial(n-(k+l+mm+cc));
   time_4:=nb4*(cc); //-> candidat F1
  //We read F1
   nb5:=Ceiling(nb4*nf1/(p^cc)*Factorial(n-(i+j))*Factorial(n-(l+k+mm+cc))
   /Factorial(n)/Factorial(n-(i+j+l+k+mm+cc)));
  time_5:=nb5*((m+1)-(c+cc))*(i+j+r);
  tps:=time_f0+time_f2+ p^c *
  ( time_f1 + time_1 + time_2
   + time_4 + time_5);
  if tps lt TimeMin then
    TimeMin:=tps;
    U:=<Log(2,tps),Log(10,size0),
    j,i,l, r, k>;
   end if;
end for;
return U;
end function;
U;
exit;
```

APPENDIX B

"Slow-Quartz" signatures (77 bits signatures)

Quartz belongs to the HFE family, it is a variant that involves both of the vinegar and the minus perturbations. Quartz is an HFEv- based signature scheme introduced in [25] that generates practical signatures of length 128 bits.

The ground field is $\mathbb{F}_2$, with $n = 103$, $D = 129$, $a = 3$ equations removed, and $v = 4$ external vinegar variables.

Quartz was first designed to achieve a security level of 2^{80} bits. At present, the complexity of the best known attack on Quartz is in 2^{80} (see [27]).

More precisely, an evaluation of the direct Gröbner basis attack gives the following $r = 8$ (since $D = 129$) and $d_{reg} = 7$.
Therefore, the complexity of a Gröbner basis computation is in $\binom{110}{7}^{\omega}$. This leads to a complexity of $2^{82.7}$ when $\omega = 2.37$ (and $2^{69.8}$ when $\omega = 2$, however $\omega = 2.37$ is probably more realistic).
The complexity of finding the secret key is much higher due to the difficulty of solving a MinRank instance that can be evaluated to be about $2^{63\omega}$ in this case.

At present (2020), and by using Magma, the time required to compute the roots of a polynomial of degree 129 over $\mathbb{F}_{2^{103}}$is nearly 0.095 s.
With the improvement given in GeMMS [19], the time complexity is 23 times smaller: 4.1 ms (see Appendix D).

Quartz for encryption

The trapdoor used in Quartz is designed for signatures purposes. However, since that the time complexity is of 4.1 ms which is relatively small, then it is possible to consider using Quartz for an encryption scheme. Then, the time to decrypt will be $2^7 \times 0,095$ s = 12.1 s with Magma, and about 0.5 s with the improved algorithm.

Nevertheless, we are mainly interested in short signature, then only the signature version of Quartz will be considered.

Quartz for signatures

Quartz employs the so-called "Feistel-Patarin" (see [24]) mode of operation in order to be able to sign messages of any length.

Remark 9 *In some cases, the value $Y = SlowHash(M)$ to sign has no solution, and one must try again. This case occurs with probability of nearly $1/e$, i.e. about $36,79\%$. Then, it is convenient to compute $Y = SlowHash(U||M)$, where U takes the values 0, then 1, 2 etc. until a solution is found.*
The value U can be added to the signature in order to accelerate the signature verification process (this method was employed in Quartz and GeMMS). Since the goal is to have ultra-short signatures, we will prefer to keep U, so the various values of U will be tested while checking the signature .

In this section, we will modify the scheme by using the "Slow hash" mode instead of the Feistel-Patarin mode.
All the parameters of Quartz will remain the same, the only difference is that another mode of operation called "Slow mode" will be used.

The time required to sign a message will then be much bigger, but the signature will be shorter.
As seen in Section 10.3.2.1, the "Slow mode" operation involves the following two ideas:

1. The use of a slow Hash function in order to avoid the birthday paradox that finds collisions on the hash values.
2. Not all of the bits that constitute the HFEv- signature will be revealed. The missing bits can be recovered by using the public key and a Gröbner basis computation.

Several tests were performed by using the Magma calculator (Magma V2.25-4). This version of Magma is limited to two minutes of computation and 300 M Bytes of RAM.

The time required on Magma to solve a system of $n = 100$ quadratic equations over $\mathbb{F}_2$ with s variables is in:
$s = 15$ variables: 8,13 ms
$s = 16$ variables: 10,6 ms
$s = 17$ variables: 16,5 ms
$s = 18$ variables: 26,8 ms
$s = 19$ variables: 45 ms
$s = 20$ variables: 75,0 ms
$s = 21$ variables: 0,125 s
$s = 22$ variables: 0,198 s
$s = 23$ variables: 0,32 s
$s = 24$ variables: 0,89 s (or 0,64 s from $s = 23$)
$s = 25$ variables: 1,7 s (or 1,28 s from $s = 23$)
$s = 26$ variables: 3,36 s (or 2,56 s from $s = 23$)
$s = 27$ variables: 6,53 s (or 5,12 s from $s = 23$)

$s = 28$ variables: 12,6 s (or 10,2 s from $s = 23$)
$s = 29$ variables: 23,9 s (or 20,4 s from $s = 23$)
$s = 30$ variables: 40,9 s

When $s = 30$, the best way is maybe to perform an exhaustive search on 7 variables and to use the fact that only $0, 32$ s is needed to find the remaining 23 variables.
Therefore, less memory will be used while keeping nearly the same required time.

We present parameters for a "Slow-Quartz" solution based on a previous Quartz function mixed with our new slow hash mode, and with 16 independent public keys.
The parameters are given below in Table F.6:

Slow-Quartz 80 bit Security								
q	r	m	a	v	Complexity	s	Signature Size (bits)	Public Key Size (KBytes)
2	8	100	3	4	80.3	30	77	1128

Table B.1: "Slow-Quartz" parameters

If having 41s as the required time to verify a signature is not an issue, then the signature's length is given by 103 (value of n) + 4 (external vinegar) – 30 (fixed variables) = 77 bits (Instead of 128 bits for the initial Quartz signature).

However, in order to be able to sign messages of any length, one must check if the results of the "Slow Hash mode" are acceptable. The following holds:
$w = \log_2(100.100.101/2) = 18.94$, $\Delta = m \log_2(q) + w - 123 = -4.06$
Then, by considering our "Slow mode", a hash of $2^{4.06}$ minutes (16 minutes) is required. Or, a hash of one minute and 16 independent public keys. Also, a hash of 4 minutes and 4 independent public keys can be used.
Next, our proposition "Slow-Quartz" will be compared to the original Quartz (we present here the solution with 16 independent public keys):

Quartz ($s = 0$, Feistel-Patarin mode)
Length of the public key: 70.5 kBytes
Time to sign: 6×4.1 ms = 24.6 ms in average.
Time to check the signature: less than 1 ms
Signature length: 128 bits,
Expected security: 2^{80} (with $\omega = 2.37$).

Slow-Quartz ($s = 30$, Slow Hash and Memory trade-off mode)
Length of the public key: 16×70.5 kBytes = 1.1 MBytes
Time to sign: 1.5×4.1 ms + 1 minute of slow hash.
Time to check the signature: 41 s + 1 minute of slow hash.

Signature length: 77 bits,
Expected security: 2^{80} (with $\omega = 2.37$).

Note that the results given in Table F.6 are slightly better than what is obtained with the original Quartz. However, the Quartz parameters have been extensively studied, so it may be efficient to use these values. Moreover, these parameters can be used for encryption schemes, not only to signatures (see Appendix B).
Signatures of 77 bits is not too far from being the best signatures schemes. However, in this work we design schemes with even smaller signatures.
Moreover, we propose smaller signature with expected security of 2^{80} bits, even by assuming $\omega = 2$, and sometimes with smaller public keys.

APPENDIX C

"Slow- GeMMS-128" signatures

GeMMS-128 [19] is an HFEv- signature scheme with an expected security of 2^{128} bits on classical (non-quantum) computers. It is a potential candidate to the Post-Quantum NIST competition.

GeMMS parameters are the following: $q = 2$, $n = 174$, $m = 162$, $a = 12$, $v = 12$, $D = 513$ (so $r = 10$). It employs four rounds of the Feistel-Patarin mode of operation. The length of the signature is in 258 bits. The length of the public key is equal to 352 KBytes.

Next, we consider our proposed mode of operation, the so-called "Slow Hash and Multiple public keys", instead of the Feistel-Patarin mode, with exactly the same HFEv- parameters.

Here $\Delta = m - 219 + \log_2(m^3/2) = 162 - 219 + 21 = -36$. This value of Δ is so negative that it gives unrealistic size for the public key (nearly 22528 Terabyte), with computations of about one minute, and a signature size of 152 bits (with $s = 34$ variables to be found using the 162 quadratic equations).

Some trade-offs exist in order to reduce the huge size of the public key. For example by increasing the time required to sign, or by using more than one computer in a given time (parallel computing), but here it seems that it's more reasonable to increase the number m of equations in order to have a smaller Δ: the parameters of GeMMS-128 are well suited for the Feistel-Patarin mode, but not well suited for our new mode of operation.

For example, as cited before, with $q = 2$, $m = 198$, $a + v = 11$, the signature is of 173 bits with an expected security of 2^{131} bits.
Here the length of the signature, 173 bits, is larger than what might be obtained by using with Slow-GeMMS (152 bits with a huge public key), but significantly smaller than the classical GeMMS (258 bits).

APPENDIX D

Berlekamp algorithm, and roots finding with Magma

Berlekamp algorithm.

The Berlekamp algorithm is generally used to find the roots of a polynomial of degree D over $\mathbb{F}_{q^n}$ for typical cryptographic values.

The algorithm is divided into two main steps: the computation of the Frobenius application, and the computation of a GCD.
The complexity of the first step that demands the computation of the Frobenius is in $O(nD\log^2(D))$ (when D is larger than n). The complexity of the second step is in $O(nD^2)$. (Asymptotically the complexity is in $O(nD)$, but from a practical point of view the asymptotic algorithms are not expected to be useful for our parameters).

The tables given below provide the time required by Magma to find the roots of a polynomial of degree D over $\mathbb{F}_{q^n}$.

q=2, n=103								
D	5	9	17	33	65	129	257	513
Time (ms)	1.13	2.06	5.05	13.3	40.8	95	260	430
D	1025	2049	4097	8193	16385	32769		
Time (s)	1.1	4.4	9.8	21.0	45.5	98.3		

Table D.1: Time to compute roots of a polynomial of degree D, $q = 2$, $n = 103$.

Remark 10 *In Table D.1, D = 129 (parameter of* Quartz*) gives 95 ms. If the improved software of* GeMMS *is used, then the time required becomes 4.1 ms (i.e. 23 times faster than Magma) or 11,2 MegaCycles.*

q=4, n=47				
D	17	65	257	1025
Time (ms)	4.29	31.2	140	650
D	4097	16385	65537	
Time (s)	2.17	13.4	80.6	

Table D.2: Time to compute roots of a polynomial of degree D, $q = 4$, $n = 47$.

q=5, n=43			
D	6	26	126
Time (ms)	20.2	76	840
D	626	3126	15626
Time (s)	4.5	17.4	117

Table D.3: Time to compute roots of a polynomial of degree D, $q = 5$, $n = 43$.

q=7, n=40			
D	8	50	344
Time (ms)	9.35	105	1630
D	2402	16808	
Time (s)	10.2	110	

Table D.4: Time to compute roots of a polynomial of degree D, $q = 7$, $n = 40$.

q=11, n=35				
D	12	122	1332	14642
Time	15 ms	360 ms	11.7 s	86.9 s

Table D.5: Time to compute roots of a polynomial of degree D, $q = 11$, $n = 35$.

q=13, n=35			
D	14	170	2198
Time	23 ms	540 ms	11.4 s

Table D.6: Time to compute roots of a polynomial of degree D, $q = 13$, $n = 35$.

q=17, n=33			
D	18	290	4914
Time	55 ms	1680 ms	29.2 s

Table D.7: Time to compute roots of a polynomial of degree D, $q = 17$, $n = 33$.

APPENDIX E

Our Magma simulation on the direct attack against HFEv-

E.1 Experimental results for public equations of degree 2

Note that the computations are considered over a finite field $\mathbb{F}_q$, where m is the number of quadratic equations, and v is the number of vinegar variables (so, the total number of variables is $m + a + v$ variables). The number of removed equations is denoted by a (initially, there was $m+a$ equations and now m).

Considering a random system of quadratic equations, the degree of regularity denoted by d_{reg} verifies the following:

- If $n < q$ then d_{reg} = n + 1
- If $n \geq q$ then, we estimate that $d_{reg} = q + 1 + (n - q - 1).\alpha$ with $\alpha \approx 0.19$ for $q = 3$, $\alpha \approx 0.2$ for $q = 5$, $\alpha \approx 0.33$ for $q = 7$, $\alpha \approx 0.40$ for $q = 13$, $\alpha \approx 0.41$ for $q = 17$.

For HFEv- systems, we have got the following results using Magma:

$q = 3$, $m = 17$, $D = 4$ (so here $r = 2$, and there are two monomials X^2 and X^4). Here, the degree of regularity is $d_{regmax} = 7$.
$v = 0$ and $a = 0$: $d_{reg} = 4$
$v = 0$ and $a = 1$: $d_{reg} = 4$
$v = 0$ and $a = 2$: $d_{reg} = 5$
$v = 0$ and $a = 3$: $d_{reg} = 6$
$v = 1$ and $a = 0$: $d_{reg} = 4$
$v = 2$ and $a = 0$: $d_{reg} = 5$
$v = 3$ and $a = 0$: $d_{reg} = 6$

$q = 3$, $m = 16$, $a = v = 0$ (nude HFE)
$r = 3$: $d_{reg} = 4$
$r = 4$: $d_{reg} = 5$
$r = 5$: $d_{reg} = 6$ (Max)

$q = 4$, $m = 14$, $a = v = 0$ (nude HFE)
$r = 3$: $d_{reg} = 6$
$r = 4$: $d_{reg} = 6$

$r = 5 : d_{reg} = 7$ (Max)

$q = 5$, $m = 13$, $D = 6$ (there are here two monomials X^2 and X^6). Here, the degree of regularity is $d_{regmax} = 8$.
$v = 0$ and $a = 0 : d_{reg} = 6$
$v = 0$ and $a = 1 : d_{reg} = 7$
$v = 0$ and $a = 2 : d_{reg} = 8$ (Max)
$v = 1$ and $a = 0 : d_{reg} = 7$
$v = 2$ and $a = 0 : d_{reg} = 8$ (Max)
$v = 1$ and $a = 1 : d_{reg} = 8$ (Max)

E.2 Experimental results for public equations of degree 3

$q = 3$, $m = 11$, $a = v = 0$ (nude HFE with public equations of degree 3)
$r = 2 : d_{reg} = 4$
$r = 3 : d_{reg} = 5$
$r = 4 : d_{reg} = 6$
$r = 5 : d_{reg} = 7$ (Max)

$q = 4$, $m = 10$, $a = v = 0$ (nude HFE with public equations of degree 3)
$r = 2 : d_{reg} = 5$
$r = 3 : d_{reg} = 6$
$r = 4 : d_{reg} = 8$
$r = 5 : d_{reg} = 8$
$r = 6 : d_{reg} = 9$ (Max)

APPENDIX F

Résumé en Français: Analyse et conception d'algorithmes de cryptographie post-quantique

---

La cryptographie est une méthode de protection des informations et des communications par le biais de codes. C'est la science mathématique permettant d'assurer la confidentialité, l'intégrité et l'authenticité des données en présence de tiers malveillants appelés adversaires.
Les algorithmes cryptographiques sont en outre classés en trois catégories distinctes en fonction du nombre de clés utilisées pour le chiffrement et le déchiffrement:

- Cryptographie symétrique ou cryptographie à clé privée / secrète (SKC, Secret Key Cryptography): La cryptographie à clé symétrique permet l'utilisation d'une seule clé commune pour les processus de chiffrement et de déchiffrement à la fois. L'expéditeur et le destinataire doivent partager la même clé (qui reste secrète pour les autres parties) ou, rarement, des clés différentes mais liées d'une manière simple et calculable.
  L'inconvénient majeur de la cryptographie symétrique est l'échange sécurisé de la clé secrète utilisée par les deux parties pour le chiffrement et le déchiffrement également. Le problème de la distribution des clés donne naissance à un nouveau type de cryptographie: la cryptographie asymétrique.

- Cryptographie asymétrique ou cryptographie à clé publique (PKC, Public Key Cryptography): la cryptographie à clé publique permet l'utilisation de deux clés différentes pour les processus de chiffrement et de déchiffrement.
  Chaque partie possède une paire de clés (une clé publique et une clé privée). La clé privée doit rester secrète à tout le temps, alors que la clé publique correspondante peut être librement partagée sur le réseau.
  La clé publique est utilisée pour le chiffrement, par contre la clé secrète est utilisée pour le déchiffrement. En signature, la clé secrète est utilisée pour signer les messages.

- Fonctions de hachage: une fonction de hachage est une fonction conçue pour prendre comme entrée une donnée de taille arbitraire, et lui associer une image (une valeur de hachage) de taille fixe. Le concept des fonctions de hachage dépend d'une propriété essentielle: il est pratiquement impossible d'inverser la fonction, par contre le calcul direct d'une image par cette fonction s'effectue facilement.

La cryptographie à clef publique est un pilier de la cybersécurité que nous utilisons quotidiennement pour garantir la confidentialité des communications électroniques (via https, IPSEC, VPN, SSH. . .).

La sécurité des protocoles que nous employons en pratique repose essentiellement sur la difficulté mathématique de deux problèmes : le logarithme discret (DLOG) et la factorisation des entiers (FACT).

Il est bien connu qu'une percée technologique comme la construction d'un ordinateur quantique remettrait en cause la difficulté de ces deux problèmes [64] ; rendant ainsi complètement obsolète des primitives comme le chiffrement à clef publique RSA ou l'échange de clef Diffie-Hellman. Il est difficile d'imaginer la situation de panique que pourrait engendrer l'annonce de la conception d'ordinateurs quantiques puissants.

En effet, il est très probable que l'ensemble des canaux de communications électroniques soient paralysés ainsi que les transactions bancaires. Nous ne sommes heureusement pas dans cette situation, mais il semble maintenant indispensable d'anticiper dès aujourd'hui un tel basculement technologique.

L'objet de la cryptographique post-quantique est de concevoir des cryptosytèmes à clefs publiques qui sont résistants aux attaques classiques comme quantiques. C'est un sujet d'étude classique qui a débuté rapidement après l'algorithme de Shor [64]. Cette branche de la cryptographie repose, en particulier, sur l'hypothèse fondamentale qu'il n'existe pas d'algorithme polynomial quantique pour résoudre les problèmes NP-durs [8].

Les cryptosystèmes post-quantiques les plus prometteurs [36] incluent : les cryptosystèmes multivariés, les cryptosystèmes à base de codes, les cryptosystèmes à base de réseaux qui sont basés sur la difficulté de trouver un vecteur court dans un réseau Euclidien, les cryptosystème à base des isogénies sur les courbes elliptiques supersingulières, et les cryptosystèmes à base de hachage qui utilisent la difficulté de trouver une collision, ou une pré-image, d'une fonction de hachage.

De nombreux indicateurs démontrent que l'effort pour développer des cryptosystèmes post-quantiques s'intensifie actuellement. L'indicateur le plus significatif est certainement la nouvelle position de l'organisation de standardisation américain https://csrc.nist.gov/. Il est donc probable que la cryptographie post-quantique envahisse notre quotidien ces prochaines années mais le type de cryptographie post-quantique qui sera utilisé reste encore lui à définir. On peut donc anticiper une très forte activité, académique comme industrielle, dans le domaine ces prochaines années. Nous assistons en effet au renouvellement des piliers mathématiques fondamentaux de la cybersécurité.

Le but de cette thèse est d'une part d'analyser la sécurité des cryptosystèmes post-quantiques. D'autre part, nous allons proposer de nouveaux schémas de signature post-quantiques. Ceux-ci seront de type multivariés ou plus généralement basés sur des problèmes NP-dur.

La théorie de la complexité cherche à classer les problèmes de calcul selon leur difficulté intrinsèque. Plus précisément, la classification s'effectue au niveau de la quantité de ressources (espace mémoire, temps, etc.) requise par un algorithme pour résoudre un problème algorithmique.
La classe NP-dur est une classe de complexité comprenant des problèmes mathématiques dont la difficulté n'est a priori pas remise en cause par l'émergence d'une ordinateur quantique [41].

En 1989, A. Shamir a introduit un schéma à clé publique d'une nouvelle nature [63], un schéma d'identification à divulgation nulle de connaissance (Zero-Knowledge ZK), basé sur le problème PKP: le problème du noyau permuté PKP (Permuted Kernel Problem).
PKP est un problème algébrique NP-dur [36] qui a été largement étudié [6, 37, 40, 57, 60]. Malgré l'effort de recherche, le problème PKP reste toujours exponentiel. Ce problème a été utilisé pour développer un schéma d'identification (IDS) qui a une implémentation très efficace sur les cartes à puce à faible coût [63].

Comme les ordinateurs quantiques sont censés être incapables de résoudre des problèmes NP-dur, les problèmes algébriques tels PKP sont très intéressants de nos jours.

F.1 L'étude du problème PKP

F.1.1 Introduction du problème

Le problème PKP (Permuted Kernel Problem) est un problème algébrique prouvé NP-dur [36]. Il consiste à trouver un vecteur noyau, sous une contrainte de coordonnées, d'une matrice donnée. C'est un problème simple, et nécessite seulement des notions et des opérations basiques de l'algèbre linéaire.
PKP est une généralisation du "Partition Problem" [36, pg.224]. Plus précisément, il est défini ainsi:

Donnée. Un corps fini $\mathbb{F}_p$, une matrice $A \in \mathcal{M}_{m\times n}(\mathbb{F}_p)$ et un n-vecteur $V \in {\mathbb{F}_p}^n$.
Question. Trouver une permutation π de n éléments $sachant\ que\ A \times V_\pi = 0$, où
$V_\pi = (V_{\pi(j)}), j = 1, \ldots, n.$

Une réduction du "3-Partition Problem" prouve que PKP est un problème NP-dur [36].

En fait, la solidité de PKP provient, d'une part, du grand nombre de permutations possibles, d'autre part, du petit nombre de permutations pouvant convenir aux équations du noyau. Plus précisément, PKP est difficile car on oblige le choix d'un vecteur-noyau de la matrice A, sous la condition que les coordonnées du vecteur sont déjà fixés.

F.1.2 L'analyse des attaques

L'efficacité de l'implémentation du premier IDS, proposé par A. Shamir [63], basé sur le problème PKP a conduit à plusieurs outils de résolution. En fait, il existe diverses attaques contre PKP [6, 37, 40, 57, 60], qui sont tous exponentiels.
A titre de référence, nous mentionnons la recherche exhaustive consistant à examiner toutes les permutations possibles. Sa complexité est évidemment en n!.

J. Georgiades a été le premier à améliorer la recherche exhaustive dans [37]. L'idée de base est de trouver de nouvelles équations afin de réduire l'ensemble des permutations possibles. Ces équations seront utilisées par toutes les autres attaques.

Les auteurs de [6] étudient également la sécurité de PKP, où un compromis temps-mémoire a été introduit. Le schéma proposé dans [6] réduit le temps et l'espace nécessaires pour résoudre le problème PKP.

De plus, J. Patarin et P. Chauvaud améliorent les algorithmes du problème du noyau permuté [57].
Les auteurs de [6] réduisent le temps nécessaire pour résoudre le problème PKP au prix de l'utilisation d'une mémoire considérablement volumineuse. Par conséquent, J. Patarin et P. Chauvaud combinent la méthode présente dans [6] avec l'idée de J. Georgiades. Ainsi, l'ajout des équations de J. Geogiades conduit à une réduction du temps nécessaire pour attaquer PKP. Ils présentent également de nouvelles idées afin de réduire l'utilisation de la mémoire.

Dans [40], A. Joux et E. Jaulmes introduisent un nouvel algorithme de compromis temps-mémoire qui est une application au problème du noyau permuté de l'algorithme décrit dans [42] pour la résolution d'un problème nommé 4SET [42, 40].
La stratégie de résolution de 4SET est composée de deux phases: la phase A qui est une étape de précalcul, et la phase B qui est une boucle principale composée de deux étapes d'énumération. Les auteurs de [42] spécifient un choix raisonnable de paramètres pour la technique de résolution de 4SET appliquée à PKP.
Ils supposent avoir la plus meilleure attaque contre PKP. En fait, les auteurs de [40] supposent que la phase-B contrôle la complexité temporelle de l'algorithme de Joux-Jaulmes qui consiste en deux phases d'énumérations: Phase-A et Phase-B.

F.1.3 Sécurité concrète du problème du noyau permuté PKP

Dans cette section, nous présentons une des principales contributions de cette thèse. L'objectif ici est double: corriger la borne de complexité de l'algorithme de Joux-Jaulmes, et fournir la meilleure méthode connue pour résoudre PKP.

F.1.3.1 Analyse de complexité de l'algorithme de Joux-Jaulmes

Comme indiqué précédemment, il est possible de réduire une instance de PKP au problème 4SET. A. Joux et E. Jaulmes ont prouvé qu'il est possible d'appliquer la stratégie de résolution de 4SET à PKP. L'algorithme de Joux-Jaulmes consiste en deux phases, et ils supposent que la phase B contrôle la complexité temporelle de leur approche.

La première chose que nous examinons essentiellement est l'analyse de la complexité du problème PKP. Nous analysons les méthodes les plus connues pour le résoudre.
En utilisant un code MAGMA (logiciel de calcul formel) pour comparer les différentes attaques, nous constatons que la complexité globale de l'algorithme Joux-Jaulmes est mal estimée.
En considérant un choix raisonnable de paramètres, il s'avère que la complexité temporelle de l'algorithme est dominée dans la plupart des cas par la phase A au contraire à ce qui a été indiqué dans [40].

Le tableau suivant confirme que la complexité globale de l'algorithme est dominée par la phase A.
Notez que, nous utilisons ici les mêmes ensembles de paramètres, de la forme $\big(PKP_p(m,n)\big)$, donnés dans [40].

Paramètres	Phase A Complexité	Phase B Complexité	Complexité globale
$(PKP_{251}(16,32))$	$2^{45.72}$	$2^{18.02}$	$2^{93.55}$
$(PKP_{251}(15,32))$	$2^{46.13}$	$2^{18.02}$	$2^{93.96}$
$(PKP_{251}(24,48))$	$2^{94.45}$	$2^{32.09}$	$2^{190.1}$
$(PKP_{251}(34,64))$	$2^{135.67}$	$2^{40.67}$	$2^{270.19}$

Table F.1: La complexité des Phase A/B de l'algorithme Joux-Jaulmes

Les résultats expérimentaux montrent que pour un choix raisonnable des paramètres, la phase A domine, et la complexité globale est beaucoup plus élevé. Par conséquent, l'algorithme Joux-Jaulmes n'est pas un outil efficace pour résoudre PKP.

F.1.3.2 l'algorithme le plus simple et le plus efficace

Nous essayons d'estimer une borne de complexité explicite des algorithmes qui servent à résoudre PKP.

Amélioration et généralisation d'attaques déjà existantes.

Comme déjà mentionné, la méthode de compromis temps-mémoire présentée dans [6] réduit le temps nécessaire pour résoudre le problème PKP au prix de l'utilisation d'une mémoire très volumineuse.
Il est possible de réduire le temps de calcul en effectuant un pré-calcul sur un ensemble de recherche plus petit impliquant une sous-matrice A et le sous-système correspondant.

J. Patarin et P. Chauvaud combinent cette méthode avec les équations de J. Georgiades dans [57]. Par conséquent, l'ajout de ces équations conduit à une réduction du temps nécessaire pour attaquer PKP. Ils présentent également de nouvelles idées afin de réduire l'utilisation de la mémoire.
G. Poupard, dans [60] donne une belle généralisation de la méthode utilisée par J. Patarin et P. Chauvaud, et une analyse de complexité correspondante, mais elle semble imparfaite car les détails de la complexité ne sont pas clairement donnés.

Ainsi, nous considérons toutes les attaques existantes et leurs améliorations afin d'obtenir une évaluation plus précise et récente sur l'algorithme le plus efficace pour résoudre PKP.

Notre méthode: Extension des attaques les plus efficaces.

Nous combinons la plupart des méthodes précédemment connues pour résoudre PKP. Nous avons fourni un nouveau logiciel (voir A) qui produit une mesure de complexité efficace, et nous établissons des approximations pour la complexité en temps et en espace mémoire (voir 8.6 et 8.7).
Notre méthode s'appuie principalement sur les travaux de Patarin-Chauvaud [57] et Poupard [60], combinant leurs techniques et poussant plus loin l'implémentation pour maintenir des niveaux de sécurité plus élevés.

Après avoir ainsi obtenu une image réaliste de l'efficacité et de la complexité de presque toutes les méthodes connues pour résoudre PKP, nous comparons les performances de chaque technique.

Par conséquent, il est possible de définir désormais des ensembles de paramètres sécurisés pour les instances PKP.
Le tableau suivant montre que l'attaque la plus efficace pour résoudre PKP est notre version étendue de Patarin-Chauvaud [57] et Poupard [60].

Paramètres	$(PKP_{251}(41,69))$	$(PKP_{509}(54,94))$	$(PKP_{4093}(47,106))$
Niveau de securité	2^{128}	2^{192}	2^{256}
Recherche exhaustive	2^{326}	2^{485}	2^{565}
attaque de J. Georgiades	2^{151}	2^{236}	2^{356}
Temps-mémoire compromis	2^{131}	2^{196}	2^{262}
attaque de Joux-Jaulmes	2^{286}	2^{413}	2^{432}
Notre version étendue	$\mathbf{2^{130}}$	$\mathbf{2^{193}}$	$\mathbf{2^{257}}$

Table F.2: Borne de complexité pour les meilleurs attaques contre PKP

F.1.4 Conclusion

Nous avons examiné les méthodes les plus connues pour résoudre PKP. Nous avons brièvement présenté chacun d'eux et mis à jour certains résultats qui n'étaient pas exacts.

Plus précisément, nous avons trouvé que l'algorithme de Joux-Jaulmes ne représente pas la meilleure technique pour résoudre le problème du noyau permuté.

En combinant des méthodes, à savoir l'approche de Patarin-Chauvaud et Poupard, nous avons pu fournir une formule de complexité explicite du meilleur algorithme pour résoudre des instances dures de PKP. Aussi, nous avons construit un programme donnant une image réaliste du niveau de sécurité des instances PKP.
Ce programme est très utile pour établir des ensembles sécurisés de paramètres afin de créer des instances de PKP difficile à résoudre.

F.2 PKP-DSS: Schéma de signature basé sur PKP

Nous sommes intéressés par la conception de signatures post-quantiques construites à partir de schémas d'identifications à divulgation nulle de connaissance basés sur des problèmes NP-dur, comme par exemple le problème de noyau permuté PKP.

Pourquoi PKP

Ce problème a été utilisé pour développer le premier schéma d'identification (IDS) dont l'implémentation est efficace sur les cartes à puce à faible coût.
La principale raison d'étudier PKP est de construire un schéma de signature basé sur un simple problème NP-dur. Par conséquent, nous étudions l'application en cryptographie du problème PKP sur un corps fini.

Nous sommes essentiellement concernés par ce problème car depuis peu de temps, aucune attaque sur PKP n'a été signalée, ce qui rend la construction de systèmes basés sur des instances dures de ce problème plus applicable, et il est possible d'obtenir à partir de PKP des performances très intéressantes.

Un schéma à divulgation nulle de connaissance

Un protocole à divulgation nulle de connaissance (Zero-Knowledge ZK) implique une interaction entre deux entités: un prouveur P et un vérifieur V, où P, utilisant une clé secrète, prouve la possesion d'un secret à V qui ne peut pas, à son tour, le prouver à une autre personne, puisque le vérifieur ne possède pas la clé secrète.

Suite à l'appel lancé par le NIST pour la standardisation de nouvelles normes post-quantiques, il y a eu un regain d'intérêt pour les schémas d'identification ZK transformés en schémas de signatures digitales (DSS) via le paradigme de Fiat-Shamir [35]. Cette méthode de transformation est importante car elle donne lieu à des schémas de signature efficaces en termes d'hypothèses de sécurité minimales et suffisantes.

Plus précisément, La transformation dite "Fiat-Shamir" est une technique permettant de convertir un schéma d'identification à divulgation nulle de connaissance (schéma ZK) en un schéma de signature. Le principe de la transformation de Fiat-Shamir est d'utiliser comme signature les éléments échangés lors du déroulement d'un schéma ZK.
Cela permet d'obtenir un schéma dont la sécurité repose directement un problème algorithmique pour des instances aléatoires.

F.2.1 PKP 5-passes IDS

Dans cette section, nous présentons le schéma original d'identification PKP-IDS basé sur PKP [63]. Il se compose de trois algorithmes probabilistes en temps polynomial $IDS = (\text{KeyGen}, \mathcal{P}, \mathcal{V})$.

KEYGEN: Génération de la clé publique et de la clé secrète dans PKP-IDS.

Le prouveur $\mathcal{P}$ et le vérifieur $\mathcal{V}$ s'accordent d'abord sur un nombre premier p, et sur n, m, les dimensions de la matrice $\mathbf{A}$.
La clé publique dans PKP-IDS est une instance de PKP. Une solution à cette instance est la clé secrète.

Ainsi, le prouveur choisit un vecteur-noyau $\mathbf{w} \in \mathrm{Ker(A)}$, puis génère aléatoirement une permutation secrète de n éléments $\mathsf{sk} = \pi$ et termine en calculant $\mathbf{v} = \mathbf{w}_{\pi^{-1}}$.
Nous résumons l'algorithme de génération de clé dans Alg. 3.

Protocole d'identification à 5-passes: Prouveur $\mathcal{P}$ et Vérifieur $\mathcal{V}$.

Le prouveur et le vérifieur sont des algorithmes interactifs qui réalisent le protocole d'identification en cinq passes.

Les cinq passes se composent d'un engagement et de deux réponses transmises du prouveur au vérifieur et de deux défis transmis du vérifieur au prouveur.
Le protocole d'identification est résumé dans Alg. 4
Le schéma d'identification basé sur PKP vérifie les trois propriétés essentielles pour garantir la sécurité contre les attaques passives d'un schéma d'identification à divulgation nulle de connaissance ZK [63]:

- **dureness:** cette propriété garantit que le Prouveur $\mathcal{P}$, qui possède la connaissance du secret avec une probabilité écrasante, sera en mesure de convaincre le vérifieur $\mathcal{V}$ d'accepter la preuve de connaissance.

- **Soundness:** Le Prouveur $\mathcal{P}$ peut toujours anticiper le défi du vérifieur $\mathcal{V}$ et générer une réponse à envoyer. Ainsi, un prouveur non légitime, sans possession du secret revendiqué, est capable de convaincre le vérifieur d'accepter une fausse déclaration avec une petite probabilité appelée "soundness error κ".
  Pour PKP-IDS, on a la valeur suivante $\kappa = \frac{p+1}{2p}$.

- **Divulgation nulle de connaissance ZK:** La propriété de connaissance "zero-knowledge ZK" garantit que le vérifieur $\mathcal{V}$ n'extrait aucune information au-delà du fait que la déclaration revendiquée par le prouveur $\mathcal{P}$ est vraie.

F.2.2 La version optimisée de PKP-IDS

Nous décrivons brièvement un certain nombre d'optimisations pour réduire le coût de communication du schéma d'identification, ainsi que le coût de calcul des algorithmes.

Nous commencerons par expliquer quelques optimisations standard qui sont courantes pour les protocoles d'identification basés sur des preuves à divulgations nulle de connaissance zéro. Ensuite, nous expliquerons quelques nouvelles optimisations applicables à PKP-IDS:

- **Hachage des engagements.** Dans la phase d'engagement du protocole, au lieu de transmettre tous les engagements, le prouveur peut simplement hacher tous ces engagements avec une fonction de hachage résistante aux collisions et ne transmettre que le hash. Avec cette optimisation, nous réduisons de moitié le nombre d'engagements communiqués.

- **Utiliser des graines et Pseudo Random Generators PRG.** Au lieu de choisir directement la permutation au hasard, nous pouvons choisir une graine aléatoire de λ bits (où λ est le niveau de sécurité requis) et utiliser un PRG pour étendre cette graine en une permutation. De cette façon, au lieu de transmettre la permutation elle-même, nous pouvons simplement transmettre la graine de λ -bit qui est normalement de taille plus petite que la permutation.

- **Matrice $\mathbf{A}$ sous forme systématique.** C'est une optimisation spécifique à PKP-IDS. Avec une forte probabilité, nous pouvons effectuer des opérations élémentaires sur la matrice $\mathbf{A}$ et la mettre sous une forme spéciale, forme systématique, sans affecter la sécurité du schéma. Cette optimisation rend le protocole plus efficace et rend aussi les opérations mathématiques comme la multiplication plus rapide.

- **Optimisation de la génération de clé.** Il n'est bien sûr pas très efficace d'inclure dans la clé publique la matrice $\mathbf{A}$. En effet, c'est une grande matrice. La première idée est de choisir simplement une graine aléatoire et d'utiliser un PRG pour développer cette graine afin d'obtenir la matrice $\mathbf{A}$ de n colonnes et m lignes. Cependant, nous pouvons faire un peu mieux que cela. Nous pouvons utiliser une graine pour générer **A *** qui est formée par les premières $n-1$ colonnes de $\mathbf{A}$ et le vecteur v. Ensuite, nous choisissons une permutation aléatoire π, et nous résolvons pour la dernière colonne c_n^A de $\mathbf{A}$ telle que $\mathbf{v}_\pi$ soit dans le noyau de $\mathbf{A}$. Désormais, la clé publique se compose uniquement d'une graine et d'un vecteur de longueur m.
  Un autre avantage important de cette approche est que nous n'avons pas besoin de faire une élimination gaussienne de cette façon (c'était la motivation derrière cette optimisation). Ainsi, la procédure de génération de clé de la version modifiée est donnée dans Alg. 5.

Une autre modification essentielle est présentée dans Section 9.2.2.
La version modifiée de PKP-IDS vérifie également les propriétés d'un schéma d'identification à divulgation nulle de connaissance, mais cette fois-ci avec un "soundness error" $\kappa = \frac{p}{2p-2}$.
Il est possible de répéter l'exécution du schéma d'identification de N tours afin de minimiser la valeur de κ.

Coût de communication de N tours de l'IDS optimisé.

Sachant que le niveau de sécurité requis est de 2^λ bits, la phase d'engagement consiste en une seule valeur hachée de 2λ bits. La première réponse est constituée de N vecteurs de longueur n sur $\mathbb{F}_p$, donc cela coûte $Nn\lceil \log_2 p \rceil$ bits.
Enfin, les réponses finales étant composées de N graines aléatoires de taille λ, de N engagements (qui se composent de 2λ bits chacun) et de N chaînes aléatoires d'engagement (qui se composent de λ bits chacun), cela coûte $4N\lambda$ bits de communication.
Au total, le coût de communication est de $2\lambda + N\left(n\lceil \log_2 p \rceil + 4\lambda\right)$ bits.

F.2.3 PKP-DSS: un schéma de signatures basé sur PKP

Nous présentons dans cette Section une contribution principale de ce travail qui est de construire un schéma de signature digitale (Digital Signature Scheme DSS), basé sur le problème PKP, à partir de l'IDS optimisé défini dans la Section précédente.

Il s'agit simplement d'une application directe de la célèbre transformation Fiat-Shamir [35].

L'algorithme de génération de clé est identique à l'algorithme de génération de clé pour le schéma d'identification.

Pour signer un message m, le signataire exécute la première phase du schéma d'engagement pour obtenir un engagement com. Puis, il dérive le premier défi $\mathbf{c} = (c_1, \cdots, c_N)$ de m et com en évaluant une fonction de hachage $\mathcal{H}_1(m||\mathsf{com})$. Ensuite, il effectue la phase suivante du protocole d'identification pour obtenir les vecteurs de réponse $\mathsf{rsp}_1 = (\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(N)})$. Ensuite, il utilise une deuxième fonction de hachage $\mathcal{H}_2$ pour dériver le deuxième défi $\mathbf{b} = (b_1, \ldots, b_N)$ de m, com et rsp_1 comme $\mathcal{H}_2(m||\mathsf{com}, \mathsf{rsp}_1)$. Puis, il termine le protocole d'identification pour obtenir le vecteur des secondes réponses
$\mathsf{rsp}_2 = (\mathsf{rsp}^{(1)}, \cdots, \mathsf{rsp}^{(N)})$.
Ensuite, la signature est simplement composée de $(\mathsf{com}, \mathsf{rsp}_1, \mathsf{rsp}_2)$.

Pour vérifier une signature $(\mathsf{com}, \mathsf{rsp}_1, \mathsf{rsp}_2)$ pour un message m, le vérifieur utilise simplement les fonctions de hachage $\mathcal{H}_1$ et $\mathcal{H}_2$ pour obtenir respectivement $\mathbf{c}$ et $\mathbf{b}$.
Ensuite, il vérifie que $(\mathsf{com}, \mathbf{c}, \mathsf{rsp}_1, \mathbf{b}, \mathsf{rsp}_2)$ est une transcription valide du protocole d'identification.

Les algorithmes de signature et de vérification sont affichés plus en détail dans les algorithmes 3 et 4.

PKP-DSS vérifie la propriété essentielle d'un schéma de signature: *Existential-Unforgeable under Chosen Adaptive Message Attacks* (EU-CMA) dans le modèle de l'oracle aléatoire.

En prenant en considération les meilleurs algorithmes pour résoudre PKP, on présente dans le tableau suivant plusieurs ensembles de paramètres choisis pour trois niveaux de sécurité différents.

Ensemble de paramètre	niveau de Securité	p	n	m	Itérations N	Complexité d'attaque
PKP-DSS-128	128	251	69	41	157	2^{130}
PKP-DSS-192	192	509	94	54	229	2^{193}
PKP-DSS-256	256	4093	106	47	289	2^{257}

Table F.3: Ensemble de paramètres pour PKP-DSS

Tailles des clés et des signatures

Clé publique pk**.** Une clé publique se compose de la dernière colonne $\mathbf{c_n^A}$ de $\mathbf{A}$ et d'une graine aléatoire, qui est utilisée pour générer **A*** formée par tout sauf la dernière colonne de $\mathbf{A}$ et le vecteur $\mathbf{v}$.
Par conséquent, la clé publique se compose de $\lambda + m\lfloor \log_2(p) \rfloor$ bits.

Clef secrète sk**.** Une clé secrète est juste une graine aléatoire qui a été utilisée pour l'algorithme de génération de clé, elle se compose donc uniquement de λ bits.

Signature sig**.** Enfin, une signature consiste en N exécutions du protocole d'identification. Une transcription du protocole peut être représentée avec $2\lambda + N\left(n\lceil \log_2 p \rceil + 4\lambda\right)$, c'est donc aussi la taille de la signature.

Dans le tableau suivant, nous résumons les tailles de clé et de signature pour les ensembles de paramètres proposé dans la section précédente.

Niveau de sécurité	Paramètres (p, n, m, N)	|sk| Octets	|pk| Octets	|sig| KilOctets
128	$(251, 69, 41, 157)$	16	57	20,4
192	$(509, 94, 54, 229)$	24	85	45,2
256	$(4093, 106, 47, 289)$	32	103	81,1

Table F.4: Tailles des clés et des signatures pour PKP-DSS

F.2.4 Comparaison avec d'autres signatures existantes FS

Afin de démontrer l'efficacité de PKP-DSS et de comparer les performances du schéma aux autres signatures Fiat-Shamir existantes, une implémentation en temps constant en utilisant le langage C a été réalisée.
Le code de notre implémentation est disponible sur GitHub à [14].

Pour mesurer les performances de notre implémentation, nous avons mené des expériences sur un ordinateur portable possédant d'un processeur i5-8250U fonctionnant à 1,8 GHz.
Le code C a été compilé avec gcc version 7.4.0 avec l'option de compilation -O3.
Les nombres de cycles dans le tableau suivant sont des moyennes de 10 000 générations de clés, signatures et vérifications.

Dans Table F.5, nous comparons PKP-DSS à MQ-DSS, Picnic et Picnic2. Nous pouvons voir que pour tous les schémas, les clés publiques et secrètes sont toutes très petites.
Les principales différences sont la taille et la vitesse de la signature. Par rapport à MQDSS, les tailles de signature de PKP-DSS sont environ 30 $\%$ plus petites, tout en étant respectivement d'un facteur 14 (resp. 30) plus rapides pour la signature (resp. la vérification).
Par rapport à Picnic, les signatures PKP-DSS sont environ 40 $\%$ plus petites, et la signature et la vérifica-

tion sont respectivement 4 et 9 fois plus rapides.
Par rapport à Picinc2, notre système est 153 et 170 fois respectivement plus rapide pour la signature et la vérification respectivement, mais cela s'effectue au coût de signatures qui sont 50 $\%$ plus grandes.
Enfin, comparé à SUSHSYFISH, un autre schéma basé sur le problème du noyau permuté, notre schéma est 3,4 et 6,6 fois plus rapide, mais le coût de signatures est 45 $\%$ plus grand.

Niveau de sécurité	Schéma	Clé secrète (Octets)	Clé publique (Octets)	Signature (KOctets)	Signer 10^6 cycles	Verification 10^6 cycles
128	PKP-DSS-128	16	57	20.4	2.5	0.9
	MQDSS-31-48	16	46	28.0	36	27
	Picnic-L1-FS	16	32	33.2	10	8.4
	Picnic2-L1-FS	16	32	13.5	384	153
	SUSHSYFISH-1	16	72	14.0	8.6	6
192	PKP-DSS-192	24	85	45.2	5.5	2.1
	MQDSS-31-64	24	64	58.6	116	85
	Picnic-L3-FS	24	48	74.9	24	20
	Picnic2-L3-FS	24	48	29.1	1183	357
	SUSHSYFISH-3	24	108	30.8	22.7	16.5
256	PKP-DSS-256	32	103	81.1	7.4	3.5
	Picnic-L5-FS	32	64	129.7	44	38
	Picnic2-L5-FS	32	64	53.5	2551	643
	SUSHSYFISH-5	32	142	54.9	25.7	18

Table F.5: Comparaison de différents schémas Fiat-Shamir post-quantique

Jusqu'à présent, il n'y a pas de versions quantiques des attaques connues sur PKP, et puisque ce problème est connu d'être NP-dur, alors que nous pensons que notre schéma est post-quantique.

F.2.5 Conclusion

Nous avons introduit un nouveau schéma de signature sécurisée post-quantique PKP-DSS, qui est basé sur un schéma d'identification PKP Zero-knowledge.
Nous avons optimisé ce schéma d'identification, et pour le rendre non interactif, nous avons utilisé la transformation de Fiat-Shamir.

Nous avons développé une implémentation à temps constant de PKP-DSS et nous concluons que notre schéma est compétitif avec d'autres schémas de signature post-quantique Fiat-Shamir tels que MQDSS, Picnic / Picnic2 et SUSHSYFISH.

Les principaux avantages de notre schéma sont que la signature et la vérification sont beaucoup plus rapides que les signatures FS existantes et que le schéma est très simple à mettre en œuvre.
Notre implémentation ne prend que 600 lignes de code C, y compris les commentaires et les lignes vides.

F.3 Signature ultra-courtes de type multivarié

La cryptographie multivariée fournit des schémas cryptographiques à clé publique dont la sécurité repose sur la difficulté de résoudre un système de polynômes multivariés de petits degrés sur un corps fini.
Divers schémas multivariés ont été cassés, alors qu'il existe encore des schémas intéressants et résistants aux attaques: typiquement des variantes de HFE (Hidden Field Equations) ou de UOV (Unbalanced Oil and Vinegar) [25, 55, 45].

Le principal avantage de la cryptographie multivariée est qu'elle est historiquement connue pour son efficacité à construire des schémas de signature courts. De plus, en plus d'être flexible en termes de conception de schémas et de variantes distinctes, la cryptographie multivariée permet le développement de schémas cryptographiques sur des petits corps comme par exemple $\mathbb{F}_2$.

La famille des schémas de cryptographie basés sur HFE, proposé par J. Patarin [55], est l'une des famille les plus connues et les plus étudiées parmi tous les cryptosystèmes à clé publique multivariés. HFE peut être utilisé pour le chiffrement à clé publique et également pour les signatures, mais il est généralement plus efficace pour les schémas de signatures.

La famille HFE implique de nombreuses variantes en raison du fait qu'il est toujours possible d'appliquer différentes modifications (à savoir les perturbations), et même d'utiliser un corps fini différent ($\mathbb{F}_2$ est principalement utilisé).
Compte tenu de l'existence de quelques algorithmes de temps quasi-polynomiaux pour attaquer "HFE nu" (c'est-à-dire sans perturbations), il est fortement recommandé de toujours utiliser des perturbations adaptées.
Plusieurs variantes du schéma HFE ont été proposées afin de renforcer la sécurité et de surmonter les instances faibles de HFE nu, telles que HFEv- [56], Quartz [25] et GeMSS [19].

F.3.1 Signatures de type multivarié

Le processus classique de signature et de vérification d'un schéma multivarié sur un corps fini $\mathbb{F}_q$ nécessite:

- Une fonction inversible $\mathcal{F}^*$ ayant une structure spéciale.
- Un isomorphisme canonique ϕ pour transformer la fonction $\mathcal{F}^*$ en une application quadratique $\mathcal{F} = \phi \circ \mathcal{F}^* \circ \phi^{-1}$.
- Deux fonctions affines inversibles $\mathcal{S}, \mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$,; pour masquer la structure originale de $\mathcal{F}$.

La clé publique d'un schéma de signature multivariée est donnée par un ensemble de m polynômes en n variables: $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} = \mathcal{S} \circ \phi \circ \mathcal{F}^* \circ \phi^{-1} \circ \mathcal{T} : \mathbb{F}^n \mapsto \mathbb{F}^n$. Et la clé secrète est donnée par $\mathcal{S}$, $\mathcal{T}$ et $\mathcal{F}^*$.
Afin de générer une signature multivariée d'un message m, il faut procéder comme suit:

- Calculer une valeur de hachage du message d'origine $h = H(m)$ à l'aide d'une fonction de hachage publiquement connue $H : \{0,1\}^* \mapsto \mathbb{F}^n$.

- En utilisant la décomposition de $\mathcal{P}$, calculer $\mathcal{S}^{-1}(h) = x$ et $X = \phi^{-1}(x)$.
- Résoudre $\mathcal{F}(Y) = X$ en utilisant l'algorithme de Berlekamp (ou un analogue).
- Calculer $\phi(Y) = y \in \mathbb{F}^n$ et $z = \mathcal{T}^{-1}(y)$.

Par conséquent, la signature du message m est $z \in \mathbb{F}^n$.

Afin de vérifier la véracité d'une signature z, il faut évaluer $\mathcal{P}(z) = h' \in \mathbb{F}^n$. Si $h' = h$ tient, alors la signature est acceptée, sinon elle est rejetée.
L'objectif principal ici est de fournir des paramètres afin d'avoir des signatures ultra-courtes. Nous nous autorisons à ce que les algorithmes de signature et de vérification puissent nécessiter près d'une minute de calcul sur un ordinateur moderne.

Afin de concevoir des schémas de signature sécurisés, plusieurs types d'attaques génériques doivent être pris en compte, essentiellement les attaques suivantes:

- l'attaque du paradoxe des anniversaires qui crée des collisions sur la sortie de la fonction de hachage,
- l'attaque par les bases de Gröbner sur la trappe utilisée qui est un ensemble de m équations multivariées (généralement quadratiques) en n variables sur un corps fini $\mathbb{F}_q$.
- l'attaque qui consiste à trouver une faiblesse dans la fonction de trappe elle-même. Dans notre travail, la fonction employée est généralement basée sur HFE. Alors qu'on considère les attaques possibles sur HFE.

Un mode d'opération définit la manière de traiter les textes en clair et les textes chiffrés au cours du processus d'un algorithme cryptographique.
Plusieurs modes d'opérations ont été introduits pour éviter essentiellement l'attaque du paradoxe des anniversaires. On cite le mode Feistel-Patarin [56], Gui mode [58], UOV mode, Dragon mode [56], etc.

Dans cette thèse, on développe de nouveaux modes d'opérations:

- Mode "Plusieurs clés publiques indépendantes": L'idée principale de ce mode d'opération est d'utiliser un ensemble de k clés publiques indépendantes. La nouvelle clé publique est l'ensemble de k clés publiques précédemment calculées.
- Mode "Hachage lent" (également mode de compromis lent et plusieurs clés publiques): L'idée principale est d'utiliser une fonction de hachage lente. Pour un niveau de sécurité donné, ce mode d'opération est efficace tant que le temps requis par la fonction de hachage lente est acceptable.

F.3.2 Choix de paramètres pour des signatures ultra-courtes

Nous effectuons un grand nombre d'expériences en utilisant Magma avec une limitation à une minute de calcul sur un corps fini $\mathbb{F}_q$, et nous définissons également différents ensembles de paramètres pour les

signatures ultra-courtes en prenant en compte tous les types d'attaques possibles contre les signatures multivariées.

Ces expériences évaluent, pour différents niveaux de sécurité = 80; 90; etc., les valeurs minimales de m nécessaires (lors de la résolution d'un système aléatoire de m équations en m variables) pour atteindre un niveau de sécurité de 2^λ bits. De plus, les tableaux fournissent le minimum m requis pour résoudre un système de degré deux (resp. de degré trois) lorsque $m - s$ variables sont trouvées en effectuant d'une minute de calcul.
La taille de la signature est donnée par:

$$L := \text{ le premier entier supérieur ou égal à} (m - s) \log_2(q),$$

le symbole «+» qui apparaît dans les valeurs de L signifie que dans certains cas particuliers, comme lors de l'utilisation d'une fonction de trappe spécifique ou même lors de la prise en compte des attaques génériques, la longueur de la signature L peut être augmenté.

Notre solution candidate

Cette section fournit des ensembles de paramètres efficaces pour les signatures ultra-courtes sur un corps $\mathbb{F}_q$ lorsque la trappe utilisée est une fonction basée sur HFEv-.
Un schéma HFEv- est une combinaison de deux perturbations - (moins) et v (vinaigre) qui ne coûtent presque rien en signature.
Par conséquent, en considérant une trappe basée sur HFEv-, il faut d'abord fixer le degré D du polynômes HFE, ensuite le nombre requis (et minimum) de variables de vinaigre v et les paramètres moins - peuvent être choisis après afin d'éviter les attaques directes.
Soit $r = \lceil \log_q(D) \rceil$, et a le nombre d'équations omis avec la perturbation -, les complexités sont évaluées en fonction de la meilleure attaque contre un schéma de signature multivarié basé sur HFEv-.

Sécurité de 80 bit								
q	r	m	a	v	Complexité	s	Taille de signature (bits)	Taille clé publique (KOctets)
2	16	104	0	0	84.4	31	73	69.3
4	9	54	3	4	81.6	23	76	24.9
5	7	47	5	4	81.0	21	82	21.2
7	6	39	5	5	80.2	19	85	16.3
8	5	38	6	5	80.1	19	90	17.0
11	5	35	6	6	83.7	19	97	16.6
13	4	35	7	6	83.7	19	108	18.6
16	4	33	7	6	83.5	18	112	17.4
17	4	33	7	6	83.5	18	115	17.8

Securité de 90 bit								
q	r	m	a	v	Complexité	f	Taille de signature (bits)	Taille clé publique (kBits)
2	16	124	1	1	93.2	32	94	121.1
4	9	64	5	4	90.3	24	98	42.2
5	7	55	6	6	93.7	22	105	35.5
7	6	46	7	6	92.9	21	107	27.9
8	5	43	7	7	92.6	20	111	26.0
11	5	41	7	7	92.4	20	122	26.6
13	4	39	8	7	92.2	20	126	26.1
16	4	38	8	7	92.1	19	136	26.5
17	4	38	8	7	92.1	19	139	27.1

Sécurité de 100 bit								
q	r	m	a	v	Complexité	s	Signature Taille (bits)	clé publiqueTaille (KOctets)
2	16	144	2	2	101.8	33	115	193.8
4	9	73	6	6	102.9	26	118	65.2
5	7	63	7	7	102.3	23	126	53.6
7	6	53	8	7	101.5	22	130	42.6
8	5	50	8	8	101.3	21	135	40.4
11	5	45	8	8	100.8	21	139	35.9
13	4	44	9	8	100.7	20	152	37.5
16	4	43	9	8	100.6	20	160	38.4
17	4	43	9	8	100.6	20	164	39.2

Sécurité de 128 bit								
q	r	m	a	v	Complexité	s	Taille de signature (bits)	Taille clé publique (KOctets)
2	16	198	6	5	131.2	37	173	530
4	9	100	9	9	128.3	29	178	171.4
5	7	87	11	10	131.7	26	191	145.1
7	6	72	11	11	130.8	25	194	110.1
8	5	68	12	11	130.6	24	201	104.2
11	5	59	12	11	130.0	23	205	84.7
13	4	57	12	12	129.8	22	219	85.5
16	4	56	12	12	129.8	21	236	88.6
17	4	55	12	12	129.7	21	238	86.7

Sécurité de 192 bit								
q	r	m	a	v	Complexité	s	Taille de signature (bits)	Taille clé publique (KOctets)
2	16	325	13	13	193.4	44	307	2450
4	9	164	17	17	194.4	32	332	788.8
5	7	142	18	18	193.6	30	344	641.2
7	6	117	19	18	192.4	29	351	478.5
8	5	110	19	19	192.0	29	357	444.1
11	5	96	19	19	192.1	27	371	366.6
13	4	89	20	20	195.8	26	382	337.1
16	4	86	20	20	195.6	26	400	335.9
17	4	85	20	20	195.6	26	405	334.0

Sécurité de 256 bit								
q	r	m	a	v	Complexité	s	Taille de signature (bits)	Taille clé publique (KOctets)
2	16	451	21	21	257.9	51	444	6704
4	9	227	25	25	257.0	37	488	2134
5	7	196	26	26	256.8	34	509	1715
7	6	163	27	26	256.7	32	517	1309
8	5	152	27	27	256.2	30	528	1186
11	5	132	28	27	259.2	30	544	980
13	4	123	28	27	256.7	29	552	885
16	4	116	28	27	256.1	28	572	833
17	4	115	28	27	256.0	28	581	834

Dans l'annexe B, une solution "Slow-Quartz" est présentée basée sur une fonction "Quartz" combinée avec notre nouveau mode d'opération: hachage lent avec des clés publiques indépendantes.
Les paramètres sont donnés ci-dessous dans le tableau.

Slow-Quartz 80 bits de sécurité								
q	r	m	a	v	Complexité	s	Taille de signature (bits)	Taille clé publique (KOctets)
2	8	100	3	4	80.3	30	77	1128

Table F.6: Paramètres "Slow-Quartz"

F.3.3 Conclusion

À l'heure actuelle, les signatures à clé publique les plus courtes sont obtenues avec des schémas de signature multivariés avec une taille de signature généralement comprise entre 128 et 256 bits, et le temps de signer et de vérifier le signature en millisecondes.

Dans cette thèse, nous avons étudié comment concevoir des signatures encore plus courtes lorsqu'il n'y a pas de problème pour que les algorithmes de signature et de vérification nécessitent près d'une minute pour être exécutés.

L'idée principale pour avoir des signatures ultra-courtes est de prendre en compte les éléments suivants: l'utilisation de fonctions de hachage très lentes et aussi l'utilisation de nombreuses clés publiques indépendantes.

Du coup et afin d'éviter l'attaque du paradoxe de l'anniversaire, nous avons également proposé de nouveaux modes d'opérations spécifiques tels que le mode d'opérations «Hashage lent» et « Plusieurs clés publiques indépendantes».

Nous avons proposé plusieurs ensembles de paramètres pour des schémas de signature dont la taille de signature est beaucoup plus courte que les autres schémas de signatures de type multivarié.

List of Publications

REFEREED CONFERENCE PAPERS

- Eliane Koussa, Jean-Charles Faugère, Gilles Macario-Rat, Jacques Patarin, and Ludovic Perret. "Combinatorial Digital Signature Scheme". In: Proceedings of the 1st International Conference on Big Data and Cyber-Security Intelligence. BDCSIntell 2018. Hadath, Lebanon. December 13-15, pp. 48–54. URL: http://ceur-ws.org/Vol-2343/paper11.pdf

- Eliane Koussa, Gilles Macario-Rat, and Jacques Patarin. "On the complexity of the Permuted Kernel Problem". In: Mathematical Cryptology and Cybersecurity (MC&C 2020). Warsaw, Poland.
URL: https://imk.wat.edu.pl/index.php/ramowy-plan-konferencji,
https://pdfs.semanticscholar.org/0c91/d87cf288f589f54eabd7a4a70f8c5923c4c7.pdf

- Eliane Koussa, Ward Beullens, Jean-Charles Faugère, Gilles Macario-Rat, Jacques Patarin, and Ludovic Perret. "PKP-Based Signature Scheme". In: Progress in Cryptology, INDOCRYPT 2019, 20th International Conference on Cryptology in India. Hyderabad, India. December 15-18, pp. 3–22.
URL: https://link.springer.com/chapter/10.1007%2F978-3-030-35423-7_1

CRYPTOLOGY ePrint ARCHIVE

Jacques Patarin, Gilles Macario-Rat, Maxime Bros, and Eliane Koussa. "Ultra-Short Multivariate Public Key Signatures".
URL: https://eprint.iacr.org/2020/914.pdf

This article has been submitted to Eurocrypt2021.