A Cryptanalysis of the 2R Cryptosystem and An improved Commitment Range Proof

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften

vorgelegt beim Fachbereich Mathematik
der Johann Wolfgang Goethe-Universität
in Frankfurt am Main

von
Antoine Scemama

vom Fachbereich Mathematik der Johann Wolfgang Goethe-Universität als Dissertation angenommen.

Gutachter : Prof. Dr. C.P. Schnorr
Prof. Dr. T. Theobald

Prüfungskommission : Prof. Dr. C.P. Schnorr
Prof. Dr. T. Theobald
Prof. Dr. G. Schnitger
Prof. Dr. A. Werner

Datum der Disputation : 6 Mai 2009

Preface

The present thesis put an end to my PhD which began in October 2004 at the Frankfurt University, under the direction of Pr. Dr. Claus Schnorr.

It is made of two parts which correspond to two different articles that I wrote, in two different fields.

The first part presents my cryptanalysis of a cryptosystem called "Double-Round Quadratic" (or simply "2R"). The related article [47] was published in the proceeding of the ICISC '07 conference (International Conference on Information Security and Cryptology).
In order to understand the cryptanalysis, we first give an introduction to the field of Multivariate Cryptography. It allows not only to understand how the "Double-Round Quadratic Cryptosystem" works but also the paradigms which led the field to be one of the most fruitful in cryptography, in the last fifteen years.
Our cryptanalysis is based on another attack (of another scheme) which we explain. We also present another cryptanalysis of the "2R" cipher which was published in '99, but contains mistakes and whose heuristic is unclear.
Finally we give our cryptanalysis which is very general and uses a well known heuristic.

The second part of this thesis is devoted to "zero-knowledge proof of knowledge". In such protocols a person called the prover, convinces someone that he knows a secret value, without giving any information about this value.
In many cryptographic protocols (such as electronic voting, publicly veryfiable secret sharing, etc..) the prover needs also to show that the secret value lies in a specific interval. Such proofs are called "Commitment Range Proof".
After the necessary introduction to the theory, we show how to substantially improve the most efficient "Commitment Range Proof" known so far. Our related article "Improved Commitment Range Proof" is not yet published.

To finish, I would like to thank Pr. Dr. Schnorr, for having given me the opportunity to do my PhD here in Frankfurt. At the beginning it was not easy, because I didn't speak German, but as I learned the language and the people here, I became very attached to this beautiful country.
I would like to thank Ali Akhavi and Brigitte Vallee, who helped me and trusted me at the beginning of my "crypto journey". Without them, I would not be here today.
I also thank the people who helped me reading and correcting the thesis : Claus Schnorr, Marc Stoppelbein and Fritz Förkel.

Finally, all the people who contributed directly or inderectly to my rese-

arch and my life during this period, among others : my family, Yi-jin, Jean-loup, Christine, Marc and Xiaokun, Uhlrich and Florina, Cordian, Fritz, all the students I worked with, the unknown referees, and all the people I forgot to mention !

Inhaltsverzeichnis

Teil I

Multivariate Cryptography

1

Introduction to Multivariate Cryptography

1.1 Multivariate Cryptography: Definition

In order to develop cryptographic primitives, hard mathematical problems are to be used. The most famous one is to factor integers, which leads to several cryptographic primitives such as RSA/Rabin (encryption and signature), Paillier scheme [43] etc..

The purpose of the first part of this thesis is to present my work on the feasability of designing cryptographic primitives with another specific (hard) problem, namely the **MQ** (**M**ultivariate **Q**uadratic) problem. This field of research is known in the community as **Multivariate Cryptography**.

Multivariate Cryptography refers to all the cryptographic primitives which are based on the **MQ** problem, defined as follow :

- *Given a finite field* $\mathbb{F}_q$, $n, m \in \mathbb{Z}$ *and polynomials* $P_i \in \mathbb{F}_q[X_1, \cdots, X_n]$ *for* $1 \leq i \leq m$, *of total degree* 2.

- *find* $(x_1, \cdots, x_n)$ *in* $(\mathbb{F}_q)^n$, *such that :*

$$\begin{cases} P_1(x_1, \cdots, x_n) = 0 \\ \vdots \\ P_m(x_1, \cdots, x_n) = 0 \end{cases}$$

Remark. We say that a cryptosystem is "based on" a problem, if a cryptanalyst can break this cryptosystem by solving (in polynomial time) a corresponding instance of the problem.

1.2 Multivariate Cryptography: An Overview

1.2.1 Some Historical Elements

The first attempt to create cryptosystems based on the **MQ** problem was made by Imai and Matsumoto in 1986 [28]. Surprisingly, at that time, the three new schemes that they presented didn't seem to have caught much attention from the community, despite their elegance and simplicity.

One reason for this may be that in the '80's RSA and the cryptosystems based on the discrete logarithm problem [1] were still very recent, so the community devoted most of its attention on studying the security of these schemes, because it was a more urgent matter.

To the best of our knowledge, no articles related to the one of Imai and Matsumoto was published until Jacques Patarin et al. in 1995 gave a new life to the field in a series of articles [45, 46, 44, 26, 32].

From Patarin's work it was realised that, along the line of Imai and Matsumoto's cryptosystem, a lot of new schemes can be built.
From then on, multivariate cryptography became a hot topic, with contribution to the field in most of the major conferences.

Between 1996 and 2003 many cryptosystems and signatures schemes were designed. Most of them were also cryptanalyzed, which eventually lowered the confidence in cryptographic schemes based on **MQ** problem.
Despite this fact, some constructions generated much enthusiasm, particularly **SFLASH**, a signature scheme designed by Patarin which was selected by the *NESSIE* consortium in 2003.
Founded in 2000, this consortium had to select a portfolio of strong cyptographic primitives of various types to be used as a standard within the European Union.
After a 3 year process, **SFLASH** was considered to be among the 3 most robust and efficient signature schemes, and was selected for standardization.

Unfortunately, in 2007 Dubois et al. [18] broke **SFLASH**. This convinced many people that building new schemes in multivariate cryptography was no longer worth trying.
On the other hand, some systems are still unbroken, in particular, a signature scheme named **QUARTZ** still seems very attractive.
Whether or not the field will die may be linked with the future of **QUARTZ**.

[1] Given N, g and y. Find x such that $g^x = y \mod N$

1.2.2 Motivations

For some time, multivariate cryptography has been considered one of the best alternative to RSA and discrete logarithm schemes.
The reasons are the following :

- The schemes are very fast for many cryptographic operations. They outperform RSA in some cases, and are very well suited to be implemented on low weight devices such as smartcards.

- It has been known since 1994 that factorization and the discrete logarithm problem can be solved in polynomial time with the help of a quantum computers [49]. At present, quantum computer already exist, a private company even sell some (see D-wave, http://www.dwavesys.com). But they are capable of working only with a few (qu)bits. No one can predict if quantum computers will break practical instances of RSA in the near future. The optimists think it will become a reality within the decade, others think it will never happen. Nevertheless, it is important to develop cryptographic schemes which will resist quantum computers. At this early stage, no scheme is provably secure against quantum computers, but some are not (yet) broken, in particular multivariate cryptographic schemes are in this category.

1.3 Thesis Part I: Organisation

In this part of the thesis, we will present the work done in the area of Multivariate Cryptography through a cryptanalysis of a scheme named **2R**, which was published at the ICISC '07 conference [47].
First we will introduce the necessary mathematical background and the general "design principles" to create multivariate schemes. As an example we will explain how the original scheme of Imai and Matsumoto works, and how it was cryptanalyzed. In a second step, we will introduce all the materials required to understand the **2R** scheme and its cryptanalysis.

The chapters are organised as follow:

- The present chapter consists of an introduction to the **MQ** problem with emphasis on its complexity from both the theoretical and the practical perspective.
  Then we describe the main ideas to build cryptographic schemes based on the **MQ** problem.

- In chapter 2, we will present one of the first three schemes of Imai and Matsumoto [28] along with Patarin's cryptanalysis [45]. The signature version of this scheme is **SFLASH**.

- Chapter 3 studies a generalisation of the original Imai-Matsumoto scheme called **HFE** [46]. A cryptanalysis of **HFE** by Kipnis and Shamir [3] will also be explained.

- The **2R** scheme will be introduced in Chapter 4, along with a first cryptanalysis by Ding and Feng. Then, based on the attack against **HFE**, we will explain our cryptanalysis and its advantages over the one by Ding and Feng.

1.4 Multivariate Cryptography: Mathematical Background and General Ideas

1.4.1 A Few Mathematical Notations

- $\mathbb{K} = \mathbb{F}_{q^n}$ denotes the field of q^n elements.

- Let $\beta_1, \cdots, \beta_n$ be a basis of the $\mathbb{F}_q$-vector space $\mathbb{K}$.

- We will use both vector and field representations to refer to $x \in \mathbb{K}$. We denote $\mathbf{x} = x_1\beta_1 + \cdots + x_n\beta_n \in \mathbb{K}$ to refer to the field element, whereas $\bar{x} = (x_1, \cdots, x_n)$ corresponds to the vector notation.

1.4.2 The (Hard) Problem

In this section we study the hard problem underlying multivariate cryptography.

- *Given a finite field* $\mathbb{F}_q$, $n, m \in \mathbb{Z}$ *and polynomials* $P_i \in \mathbb{F}_q[X_1, \cdots, X_n]$ *for* $1 \leq i \leq m$, *of total degree* 2.

- *Find* $(x_1, \cdots, x_n)$ *in* $\mathbb{F}_q^n$, *such that:*

$$\begin{cases} P_1(x_1, \cdots, x_n) = 0 \\ \vdots \\ P_m(x_1, \cdots, x_n) = 0 \end{cases}$$

In the definition of the problem, instead of having quadratic polynomials P_i, we could have required the total degree to be more than (or equal to) 2. However, a n-ary polynomial of degree d has $O(n^d)$ terms. To keep the instance's length as small as possible, d will be 2 most of the time.

As it is usually the case in cryptography, when working with a new problem, we would like to find some arguments that this problem is "difficult" to solve.

Theoretical Hardness of the MQ Problem

A first argument that the **MQ** Problem is difficult to solve comes from complexity theory :

Theorem 1.4.1 *The* **MQ** *Problem is NP-hard.*

A proof of this theorem can be found in [26].

On the (un)importance of NP-hardness in Cryptography.

NP-hard problem are difficult in the worst case. If $P \neq NP$, there are no polynomial algorithms which solve all the instances of any NP-hard problem.

However, it may be possible that a polynomial time algorithm solves many of the instances. If a cryptosystem uses easy instances, the scheme can be broken.
Thus, to ensure security of a cryptosystem, one has to prove that there is no polynomial time algorithm which solves the problem on a non-negligible fraction of the instances which appear with the cryptosystem.
Such a proof does not exist for schemes in multivariate cryptography.
So, the hardness result from theorem 1.4.1 does not guarantee security of the schemes.

Solving the MQ Problem in Practice

What matters for the cryptanalyst is that the best algorithm can solve (in a reasonable amount of time) a non-negligible fraction of the instances.
For this reason, it is important to know the best techniques at our disposal to solve the **MQ** problem.

Gröbner Basis Algorithms. Nowadays, the best way to solve "random" instances of the **MQ** problem is to use Gröbner basis algorithms (see [39] for a general introduction on Gröbner bases).
The theory of Gröbner basis was invented in 1976 by Buchberger and is dedicated to the study of polynomial systems over a general ring.

In the framework of his theory, Buchberger designed an algorithm which, from a set of polynomials, computes a so-called "Gröbner basis". This basis directly leads to the common zero of the polynomial system.
Although it made it possible to solve problems in many fields of mathematics

and engineering, Gröbner basis computation was still highly time consuming in practice.
A breakthrough occured in 1999 with the algorithm $F4$ of J-C Faugere [21] and later with the $F5$ algorithm [20]. Both $F4$ and $F5$ are very efficient refinement of Buchberger original algorithm. They enable us to solve a much wider range of problems in practice.
In 2003, Faugere et al. used the $F4$ algorithm to successfully cryptanalyzed a challenge of the **HFE** cryptosystem by solving a system of 80 equations with 80 variables, see [29].

Other Methods. There are several other methods to find the common roots of systems of polynomials. They may not be as effective as the Gröbner algorithm in general, but may be advantageous in some special cases. The following list is not exhaustive:

- **X(S)L** Algorithms
  The algorithms XL [12] and XSL [41] were invented in 2000 and 2002. They are by nature very similar to the $F4/F5$ algorithms of Faugere, but seem to be somewhat slower (see [38, 42]).

- **Zhuang-Zi** Algorithms
  In [31], the author used results from Kipnis and Shamir (that are presented in chapter 3) to transfom a system of multivariate polynomials over a small finite field into a sparse univariate polynomial over a big extension field. In some special cases, the polynomial has a special shape, which makes it easy to solve.

- **Resultant** Algorithms
  In [52], the authors propose to use resultant computations to solve system of multivariates polynomial equations. They also show some experiments (with less than a dozen variables) where their method is faster than $F4$. Nevertheless, it is unclear if their method can be of any help in cryptography.

- **SAT Solver** Algorithms
  The authors of [24], inspired by the reduction between the **MQ** and the **3-SAT**, transform the polynomial system in a **SAT** instance on which they apply known **SAT**-solver algorithms.
  This method seems somewhat "exotic", but the authors claim that it is efficient (especially for sparse systems).

1.4.3 General Construction

The (hard) problem has been presented. Now, the question is: how to make a cryptosystem out of it ?

How to build a cryptosystem

A hard problem is not enough to build cryptographic primitives.
In addition, one must embed a trapdoor that allows to solve the hard problem using secret information.
Anyone possessing the trapdoor can solve the instance of the problem. On the other hand, without it, the problem must remain hard to solve.

Note that by using instances on which a trapdoor is embedded, we may exclude other instances of the problem. The problem might not remains NP-hard on the trapdoor embedded instances. Thus, the difficulty of solving in practice, the problem on these instances, without the trapdoor, can only be assumed.

In our case the hard problem is **MQ**. Let us write the set of polynomials P_i as a function **S** such that :

$$\begin{array}{rccl} \mathbf{S}: & \mathbb{F}_{q^n} & \longrightarrow & \mathbb{F}_{q^n} \\ & (x_{01}, \cdots, x_{0n}) & \longrightarrow & (P_1(x_{01}, \cdots, x_{0n}), \cdots, (P_n(x_{01}, \cdots, x_{0n})) \end{array}$$

The most obvious trapdoor would be a (secret) way to invert the function **S**.

First, suppose that one knows how to create instances in which this trapdoor is embedded.

A cryptosystem can be defined the following way:

Public	**Private**
$\mathbb{F}_q, n$ $\mathbf{S}\begin{cases} P_1(x_1, \cdots, x_n) \\ \vdots \\ P_n(x_1, \cdots, x_n) \end{cases}$	A way to invert the function **S**

Encryption
Let $\bar{m} = (m_1, \cdots, m_n) \in (\mathbb{F}_q)^n$ *be the plaintext.* The ciphertext is $\bar{c} = (c_1, \cdots, c_n) = (P_1(m_1, \cdots, m_n), \cdots, P_n(m_1, \cdots, m_n)) = \mathbf{S}(\bar{m})$.

Decryption
$\bar{m} = (m_1, \cdots, m_n) = \mathbf{S}^{-1}(\bar{c})$.

Remarks.

- If the system $\mathbf{S}$ is one to one, the encryption and decryption work. Moreover, an attacker who wants to decrypt a ciphertext without the secret key has to solve an instance of the **MQ** problem.

- The public key size is $O(n^2)$ because the P_i's are quadratic.

- Increasing the total degree d of the polynomials might make the **MQ** problem harder to solve in practice. However if $d = 2$ the problem is already intractable for small values of n. Thus one takes $d = 2$ to minimize the size of the public key.

The Trapdoor

It is easy to design invertible quadratic functions $\mathbf{S}'$ which do not have a trapdoor. We will see several examples in this thesis.
The problem is : how to embed the trapdoor ?

In multivariate cryptography the method is always the same :

1. Choose an easily invertible quadratic system of equations. This system will be public, anyone is able to invert it.

2. Apply a linear transformation to the variables of this quadratic system, and keep this transformation secret.
   Publish the resulting quadratic system of equations as the public key.

Knowing the public key does not enable us to recover the original easily invertible system of polynomials, unless the secret linear transformation is also known.

Designers of the scheme hope that the only way to invert the public system of polynomials is by finding the linear transformation, because without knowing it, the quadratic system seems "random" and solving the **MQ** problem on "random instance" is a very hard task in practice.

Generic construction of the key pair

$$\bar{x} = (x_1, \cdots, x_n)$$

$$\boxed{\bar{x}' = A\bar{x}}$$

$$\bar{x}' = (\sum_{j=1}^{n} a_{1j}x_j, \cdots, \sum_{j=1}^{n} a_{nj}x_j)$$

$$\Downarrow$$

linear with matrix A (Private)

$$\boxed{\bar{x}'' = \mathbf{S}'(\bar{x}')}$$

$$\bar{x}'' = (\sum_{i,j=1}^{n} r_{1ij}x_ix_j, \cdots, \sum_{j=1}^{n} r_{nj}x_ix_j)$$

$$\Downarrow$$

nonlinear invertible (Public)
e.g. quadratic

$$\boxed{\bar{y} = B\bar{x}''}$$

$$\bar{y} = (y_1 = \sum_{i,j=1}^{n} s_{1ij}x_ix_j, \cdots, y_n = \sum_{j=1}^{n} s_{nj}x_ix_j)$$

$$\Downarrow$$

linear with matrix B (Private)

$$S \begin{cases} y_1 = P_1(x_1, \cdots, x_n) \\ \vdots \\ y_n = P_n(x_1, \cdots, x_n) \end{cases} \quad \textbf{Public}$$

This means $S(\mathbf{x}) = \sum_{i=1}^{n} y_i\beta_i$

Abbildung 1.1: Generic Construction of a Multivariate Cryptosystems

In the following we explain why two matrices are to be used:

- Suppose that someone constructs the public key using only the linear transformation B but not A. In this case it is easy to recover the (private) matrix $B = (b_{i,j})_{1 \le i,j \le n}$.

  Starting from any plaintext we compute the corresponding ciphertext with the help of the public key. Because the function $\mathbf{S}'$ is public, each plaintext/ciphertext pair leads to linear equations in the $b_{i,j}$'s. From sufficiently many such pairs (and equations) the $b_{i,j}$'s can be recovered.

- If the public key was build without B, the same method as in the last paragraph enables us to recover the private key A.

Notice that an attacker who knows one of the two matrices, can compute the second with this technique.

General Construction of a Multivariate Cryptosystem

The next frames show the general construction of a multivariate cyptosystem. Note that S' is not specified because it is the component which will change from one particular cryptosystem to another.

Keys of the cryptosystem

Public	*Private*
$n, q \in \mathbb{N}$, $\mathbf{S}'$ quadratic (invertible), $\mathbf{S}$ $\begin{cases} P_1(x_1, \cdots, x_n) \\ \vdots \\ P_n(x_1, \cdots, x_n) \end{cases}$	Matrices $A, B \in M_n(\mathbb{F}_q)$ nonsingular

Encryption/Decryption:

Encryption
Let $\bar{m} = (m_1, \cdots, m_n) \in (\mathbb{F}_q)^n$ *be the plaintext.* The ciphertext is $\bar{c} = (c_1, \cdots, c_n) = (P_1(m_1, \cdots, m_n), \cdots, P_n(m_1, \cdots, m_n)) = \mathbf{S}(\bar{m})$.

Decryption
$\bar{m} = (m_1, \cdots, m_n) = A^{-1}\bar{S'}^{-1}B^{-1}\bar{c}$.

These frames show the general way to create multivariate cryptosystems. The cryptosystem of Imai and Matsumoto [28] in the next chapter follows this general approach.
On the other hand, some systems which will later be presented in this thesis, such as **2R** or the **SFLASH** signature scheme, are tiny variations of this general scheme.

General Remarks on Multivariate Scheme

- Depending on the specific system the field $\mathbb{F}_q$ will typically have 2 to 100 elements, whereas the value of n is between a few dozens and up to 200.

- Encryption consists only of computing the image of a vector of $(\mathbb{F}_q)^n$ via the system of quadratic polynomials over $\mathbb{F}_q$. Since only small numbers are used, and the computation is very fast. These computations are also well suited for implementation on (very) low-weight devices.

- Decryption speed depends on the time required to invert the public quadratic invertible system. For some systems it is done very fast, but for others it can become quite slow.

- The public key is the system of quadratic polynomials and has size $O(n^3)$, whereas the private key (A and B) is in $O(n^2)$. Theoretically it is worse than RSA for which the keylength is $O(log(N))$. Keep in mind that in practice n is a few bits long whereas in RSA the public modulus N is thousand bits long.

- Signature schemes can be produced in the same way. Let the signature $\bar{s}$ of a message $\bar{m}$ correspond to the decryption of $\bar{m}$. To verify the validity of $\bar{s}$ one tests if $\mathbf{S}(\bar{s}) = \bar{m}$.

- Any attack succeeds in a given amount of time which is a function of the length of the public key parameters.
  In general the length of one of those parameters plays a dominent role in the running time of the attack, increasing it slightly substantially increases the running time of the attacks.
  This parameter is called a security parameter, it is the one which is enlarge in order to increase the workload of the attacks.

Minimal size of n when no specific attacks are known

Two attacks can break the cryptosystem presented above : either via solving the **MQ** problem or by doing an exhaustive search. The security parameter n has to be large enough to prevent these attacks.

Attack via MQ solver.

This attack tries to solve the quadratic system given by the public key. ie given a ciphertext, to recover the plaintext as a solution of a set of quadratic equations over $\mathbb{F}_q$.
In 1.4.2, different methods to solve systems of quadratic polynomials (over a finite field) were listed. Amongs the most efficient seems to be the Gröbner basis method. But until the breakthrough of Faugere in 2002 [20], Gröbner

basis algorithms were only capable of solving systems with very low number of equations and unknowns. Hence this attack was not a threat, and n was rather chosen to resist brute force attacks.

Brute Force Attack/Exhaustive Search

Given a ciphertext, brute force attacks consist in exhausting all plaintexts, encrypt them, and check wether it leads to the given ciphertext. Obviously, there are q^n possible plaintexts, hence n has to be chosen so that q^n trials are out of reach by computers.[2]

[2]Brute force attack can be applied on the key as well, but here the number of trials with regards to the key would be $2q^{n^2}$ so it is less efficient than an exhaustive search of the plaintext.

2

The Imai-Matsumoto Cryptosystem

We present the Imai-Matsumoto scheme, the first multivariate cryptosystem, which was published in 1986 in [28] (along with 2 other schemes that we won't discuss in this thesis).
Apart from being of historical interest, the Imai-Matsumoto cryptosystem is the basis upon which were constructed the **2R** and **HFE** cryptosystems. Understanding this crypotsystem is a first step toward the understanding of the other schemes, which are the topic the following chapters.

First a simplified version of the scheme will be presented (as in [36]), then its cryptanalysis due to Patarin [45]. Afterwards, we will describe the general scheme and how Patarin's cryptanalysis naturally adapts to it.

The cryptanalysis consists in finding hidden linear relations between the plaintext and the ciphertext. Transforming the non linear problem posed by the public key to a linear one is "the way" to attack multivariate schemes. In this thesis, all the attacks are different techniques to achieve this goal.

The Imai-Matsumoto cryptosystem strictly follows the general construction presented in the last chapter. Thus only the (invertible quadratic) function $\mathbf{S}'$ has to be defined.

In the following, $\mathbf{S}'$ will be given as a function over the field $\mathbb{F}_{q^n}$ (instead of the vector space $(\mathbb{F}_q)^n$). Thus, before discussing the Imai-Matsumoto scheme, we shall mention some notations and basic facts from finite field theory.

2.0.4 Notations and Mathematical Background

- $\mathbb{K} = \mathbb{F}_{q^n}$ denotes the field with q^n elements, where q is a prime power.

- $(\beta_1, \cdots, \beta_n)$ denotes a basis of the $\mathbb{F}_q$-vector space $\mathbb{K} = \mathbb{F}_{q^n}$.

We use for the elements of $\mathbb{K} = \mathbb{F}_{q^n}$ the notations for vector spaces as well as those for finite fields.

- We denote $\mathbf{x} = x_1\beta_1 + \cdots + x_n\beta_n \in \mathbb{K}$ and $\bar{x} = (x_1, \cdots, x_n) \in (\mathbb{F}_q)^n$.

We will often move from field notations to vector space notations. It is convenient to define a function for this purpose.
Given a basis $(\beta_1, \cdots, \beta_n)$ of $\mathbb{F}_{q^n}$ (as $\mathbb{F}_q$-vector space), let π be the following group isomorphism :

$$\begin{array}{rccl} \pi: & (\mathbb{F}_q)^n & \longrightarrow & \mathbb{F}_{q^n} \\ & \bar{x} = (x_1, x_2, \cdots, x_n) & \longrightarrow & \mathbf{x} = x_1\beta_1 + \cdots + x_n\beta_n \end{array}$$

The following lemmas are used in the construction of $\mathbf{S}'$.

Lemma 1 *For all a, b in $\mathbb{F}_{q^n}$, and $i \in [0, n-1]$: $(a+b)^{q^i} = a^{q^i} + b^{q^i}$.*

Lemma 2 *For all x in $\mathbb{F}_q$, $x^q = x$ holds.*

2.0.5 The Quadratic Bijection $\mathbf{S}'$

Imai and Matsumoto chose the following quadratic function:

$$\begin{array}{rccll} \mathbf{S}': & \mathbb{F}_{q^n} & \longrightarrow & \mathbb{F}_{q^n} & \\ & \mathbf{x} & \longrightarrow & \mathbf{x}^{q^\theta+1} & , gcd(q^\theta+1, q^n-1) = 1 \end{array}$$

Notice that the condition $gcd(q^\theta+1, q^n-1) = 1$ requires that q is a power of 2.

- First let us explain why this function leads to a system of n quadratic polynomial in n variables.

  Let $\mathbf{x} = x_1\beta_1 + \cdots + x_n\beta_n$, $x_i \in \mathbb{F}_q$. We write $\mathbf{S}'(\mathbf{x})$ as a linear combination of the basis $\beta_1, \cdots, \beta_n$ whose coefficients are quadratic polynomials in $x_1, \cdots, x_n$. Thus $\pi^{-1}(\mathbf{x})$ is a vector whose n coordinates are quadratic polynomials.

  - From Lemma 1 and 2: $\mathbf{x}^{q^\theta} = \sum_{i=1}^n x_i\beta_i^{q^\theta}$
  - Each $\beta_i^{q^\theta}$ is an element of $\mathbb{F}_{q^n}$,
    it exist $l_{1,i}, \cdots, l_{n,i} \in \mathbb{F}_q$ such that $\beta_i^{q^\theta} = \sum_{t=0}^n l_{t,i}\beta_t$.
    So $\mathbf{x}^{q^\theta} = (\sum_{k=1}^n l_{1,k}x_k)\beta_1 + \cdots + (\sum_{k=1}^n l_{n,k}x_k)\beta_n$.

  - Hence

$$\mathbf{x}^{q^\theta+1} = [x_1\beta_1 + \cdots + x_n\beta_n][(\sum_{k=1}^n l_{1,k}x_k)\beta_1 + \cdots + (\sum_{k=1}^n l_{n,k}x_k)\beta_j]$$

$$\mathbf{x}^{q^\theta+1} = \sum_{i,j=1}^{n} [x_i(\sum_{k=1}^{n} l_{j,k}x_k)]\beta_i\beta_j$$

Now for every $1 \leq i,j \leq n$, $\beta_i\beta_j$ is an element of $\mathbb{F}_q^n$, so there exists $r_{i,j,t} \in \mathbb{F}_q$ such that $\beta_i\beta_j = \sum_{t=1}^{n} r_{i,j,t}\beta_t$.

$$\mathbf{x}^{q^\theta+1} = \sum_{i,j=1}^{n} [x_i(\sum_{k=1}^{n} l_{j,k}x_k)](\sum_{t=1}^{n} r_{i,j,t}\beta_t)$$

– Which leads to the final equation:

$$\mathbf{x}^{q^\theta+1} = \sum_{t=1}^{n}[\sum_{i,k}^{n}(\sum_{j=1}^{n} l_{j,k}r_{i,j,t})x_ix_k]\beta_t \qquad (2.1)$$

$$\mathbf{x}^{q^\theta+1} = \sum_{t=1}^{n} P_t(x_1,\cdots,x_n)\beta_t \qquad (2.2)$$

Thus each coordinate P_t of $\pi^{-1}\mathbf{S}'\pi$ is a quadratic polynomial in $x_1,\cdots,x_n$.

- Now we show that $\mathbf{S}'$ is a bijection.
  Since $gcd(q^\theta+1, q^n-1) = 1$ there exist h, h' such that $h(q^\theta+1) + h'(q^n-1) = 1$.
  Hence $(\mathbf{S}'(\mathbf{x}))^h = \mathbf{x}^{h(q^\theta+1)} = \mathbf{x}^{1-h'(q^n-1)} = \mathbf{x}$, holds for $\mathbf{x} \in \mathbb{F}_{q^n}^*$, since $|\mathbb{F}_{q^n}^*| = q^n - 1$.

2.1 A simplified version of the Imai-Matsumoto Cryptosystem

We first present a simplified version of the Imai-Matsumoto scheme, for which the cryptanalysis is easier to understand.

2.1.1 Producing the key pair

1. Choose integers n, q, θ, such that q is a power of 2 and gcd $(q^n-1, q^\theta+1) = 1$. Construct the field $\mathbb{F}_q = \mathbb{F}_{2^l}$ (ie. choose an irreducible univariate polynomial Q of degree l over $\mathbb{F}_2$), then the field $\mathbb{F}_{q^n}$. Choose $(\beta_1,\cdots,\beta_n)$ a basis of $\mathbb{F}_{q^n}$ (for instance $(1, X, X^2, \cdots, X^{n-1})$).
   Choose random matrices $A, B \in M_n(\mathbb{F}_q)$, such that $det(A) \neq 0$ and $det(B) \neq 0$.

Compute the $l_{j,k}$ and $r_{i,j,t}$ from equation 2.1.

2. In the last section, the following equation was established:

$$\mathbf{S}^{'}(\mathbf{x}) = \mathbf{x}^{q^{\theta}+1} = \sum_{t=1}^{n}[\sum_{i,k}^{n}(\sum_{j=1}^{n} m_{j,k}r_{i,j,t})x_i x_k]\beta_t$$

However, in the cryptosystem, the public key is $B\pi^{-1}[\mathbf{S}^{'}(\pi(A\bar{x}))]$.
Using the last equation :
$B\pi(\pi^{-1}(A\bar{x}))^{q^{\theta}+1} = (\sum_{t=1}^{n} b_{l,t}[\sum_{u,v=1}^{n}(\sum_{i,k}^{n} a_{i,u}a_{k,v}(\sum_{j=1}^{n} m_{j,k}r_{i,j,t}))x_u x_v])_{1\leq l\leq n}$
we get the n formal polynomials:

$$(\sum_{t,u,v,i,j,k=1}^{n} b_{l,t}a_{i,u}a_{k,v}m_{j,k}r_{i,j,t}x_u x_v)_{1\leq l\leq n} = (P_l(x_1,\cdots,x_n))_{1\leq l\leq n} \quad (2.3)$$

3. Eventually, replacing $a_{i,j}, b_{i,j}, r_{i,j,t}, m_{i,j}$ with their chosen values gives rise to the public key.

Remark.
Note that the equation 2.3 is established once and for all, the only computations required when one wants to produce a new key pair, is to choose the new $a_{i,j}, b_{i,j}, r_{i,j,t}, m_{i,j}$ values.

2.1.2 Using the cryptosystem.

Encrypting a plaintext $\bar{m}$
$\bar{m} = (m_1,\cdots,m_n)$ with $m_i \in \mathbb{F}_q$. In the public key, replace the x_i with m_i to obtain a ciphertext vector $\bar{c} = (P_1(m_1,\cdots,m_n),\cdots,P_n(m_1,\cdots,m_n)) \in (\mathbb{F}_q)^n$.

Decrypting a ciphertext $\bar{c}$
$\bar{c} = (c_1,\cdots,c_n)$ with $c_i \in \mathbb{F}_q$. Compute $\bar{c^{'}} = B^{-1}\bar{c}$. Compute $\mathbf{m}^{'} = \mathbf{c}^{'h} \in \mathbb{F}_{q^n}$. *The exponent h is the inverse of $q^{\theta}+1$ in $\mathbb{Z}_{q^n-1}$. cf, last paragraph.* Compute $\bar{m} = A^{-1}\bar{m'}$.

2.2 Cryptanalysis of the (simplified version of) Imai-Matsumoto Cryptosystem

Patarin [45] found a weakness in the quadratic function $\mathbf{S}'$. Indeed, he found out that the plaintext/ciphertext pairs satisfy not only the quadratic equations given by the public key, but also some linear equations.
Once these linear relations (between each plaintext and its corresponding ciphertext) are known, they can be used to reduce the search space of the possible plaintext during a brute force attack of a given ciphertext.

2.2.1 General Strategy

In the scheme, the plaintext $\bar{m} = (m_1, \cdots, m_n)$ and ciphertext $\bar{c} = (c_1, \cdots, c_n)$ satisfy the following equation:

$$\bar{c} = B\pi^{-1}[\pi(A\bar{m})]^{q^\theta+1}$$

We introduce some convenient notations.
Let $\mathbf{c_B} = \pi(B^{-1}\bar{c}) \in \mathbb{F}_{q^n}$ be the new "ciphertext" and $\mathbf{m_A} = \pi(A\bar{m}) = m_{A1}\beta_1 + \cdots + m_{An}\beta_n \in \mathbb{F}_{q^n}$ be the new "plaintext".

$$\mathbf{c_B} = \mathbf{m_A}^{q^\theta+1} \tag{2.4}$$

Note that the left part of equation 2.4 is linear in the c_i's [1]*, and the right part is quadratic in the m_i's (as a product of two linear transformations).*

Patarin's trick is to linearize equation 2.4 by shifting one of the linear transformations in the m_i's to the left hand side of the equation.
Indeed, let us raise equation 2.4 to the $(q^\theta - 1)$-th power :

$$\mathbf{c_B}^{q^\theta-1} = \mathbf{m_A}^{q^{2\theta}-1}$$

Now multiply both sides with $\mathbf{c_B}\mathbf{m_A}$:

$$\mathbf{m_A}\mathbf{c_B}^{q^\theta} = \mathbf{m_A}^{q^{2\theta}}\mathbf{c_B} \tag{2.5}$$

$\pi^{-1}(\mathbf{m_A}^{q^{2\theta}})$ is linear in the $(m_A)_i$'s. Hence $(m_{A1}\beta_1 + \cdots + m_{An}\beta_n)^{q^{2\theta}} = m_{A1}\beta_1^{q^{2\theta}} + \cdots m_{An}\beta_n^{q^{2\theta}}$ and $\beta_i^{q^{2\theta}} = \sum_{k=1}^n r_{l,k}\beta_k$.
It follows that $\mathbf{m_A}^{q^{2\theta}} = \sum_{k=1}^n(\sum_{i=1}^n m_{Ai}r_{i,k})\beta_k$.

Each m_{Ai} is linear in the m_i, so $\pi^{-1}(\mathbf{m_A}^{q^{2\theta}})$ is also linear in the m_i's. The same argument applies to $\pi^{-1}(\mathbf{c_B}^{q^\theta})$.
Thus both sides of equation 2.5 are linear in both the m_i's and the c_i's.

[1] We consider the vectorized version of this equation

Hence, there exist coefficients $\alpha_{i,j,k} \in \mathbb{F}_q$ such that for every plaintext/ciphertext pair :

$$\sum_{1 \leq i < j \leq n} \alpha_{i,j,k} m_i c_j = 0 \quad \forall 1 \leq k \leq n \tag{2.6}$$

At the beginning, these n^3 coefficients $\alpha_{i,j,k}$ are not known, but anyone can use the public key and generate for a chosen plaintext $\bar{m} = (m_1, \cdots, m_n)$ the corresponding ciphertext $\bar{c} = (c_1, \cdots, c_n)$. Eq. 2.6 must hold for all $\bar{m}/\bar{c}$ pairs. Thus, if one compute many such pairs, the $\alpha_{i,j,k}$ can then be found using the Gaussian algorithm.
Eventually, we get the vectors $(\alpha_{1,1,k}, \cdots, \alpha_{n,n,k})$. Note that the number of these independent vectors is unknown.
Indeed, several equations of 2.6 can become linearly dependant, for a specific plaintext/ciphertext pair, this tends to decrease the number of linear independent equations.

Let L be the number of linear independant vectors $(\alpha_{1,1,k}, \cdots, \alpha_{n,n,k})$ found by the Gaussian algorithm.

Knowing the $(\alpha_{1,1,k}, \cdots, \alpha_{n,n,k})_{1 \leq k \leq L}$ means knowing linear relations between any ciphertext and its corresponding plaintext. Thus, given a ciphertext, replacing its value in eq. 2.6 leads to a linear system, which the plaintext must satisfy.
Note that this system may have less than L linearly independent equations. Because, when replacing the c_i, some of the equations may become linearly dependant. Let $\Lambda_{\bar{c}}$ denote the number of linear independent equations from eq. 2.6 for the ciphertext $\bar{c}$.

Eventually, one finds the plaintext accordingly:

- Construct a basis of the kernel of the matrix from eq. 2.6.
- Search the plaintext in this kernel. Do an exhaustive search among the $q^{n-\Lambda_{\bar{c}}}$ elements of the kernel.

The running time of this attack depends on the running time of the exhaustive search. So our next goal is to investigate how close to n can $\Lambda_{\bar{c}}$ be.

2.2.2 Evaluating $\Lambda_{\bar{c}}$

Recall that, given a ciphertext $\bar{c}$, the size of the kernel $n - \lambda_{\bar{c}}$ corresponds to the number of different plaintext $\mathbf{m_i}_A$, for which equation 2.5 holds:

$$\mathbf{m_i}_A \mathbf{c_B}^{q^\theta} = \mathbf{m_i}_A^{q^{2\theta}} \mathbf{c_B}$$

Moreover, this equation came from raising equation 2.4 ($\mathbf{c_B} = \mathbf{m_A}^{q^\theta+1}$) to the $(q^\theta - 1)$-th power, and then multipliying with $\mathbf{c_B}\mathbf{m_A}$.
Equation 2.4 corresponds to the invertible quadratic function $\mathbf{S}'$, hence for a given ciphertext $\mathbf{c_B}$ there is unique $\mathbf{m_A}$ which satisfies this equation. So the different solutions for $\mathbf{m_A}$ in equation 2.5 appeared during the process of raising eq. 2.5 to the $(q^\theta - 1)$-th power.

Suppose that for a given "ciphertext", there exist 2 different "plaintexts" $\mathbf{m_{0}}_A$ and $\mathbf{m_{1}}_A$. The following equations hold :

$$\mathbf{c_B}^{\mathbf{q^\theta - 1}} = \mathbf{m_0}_\mathbf{A}^{(q^\theta-1)(q^\theta+1)}$$
$$\mathbf{c_B}^{\mathbf{q^\theta - 1}} = \mathbf{m_1}_\mathbf{A}^{(q^\theta-1)(q^\theta+1)}$$

Therefore :

$$\mathbf{m_{0}}_A{}^{(q^\theta-1)(q^\theta+1)} = \mathbf{m_{1}}_A{}^{(q^\theta-1)(q^\theta+1)} \tag{2.7}$$

Recall that θ satisfies $gcd(q^\theta + 1, q^n - 1) = 1$, so there exists h, h' such that $h(q^\theta + 1) + h'(q^n - 1) = 1$.
Raising equation 2.8 to the power h leads to :

$$\mathbf{m_{0}}_A{}^{(q^\theta-1)} = \mathbf{m_{1}}_A{}^{(q^\theta-1)}$$

can be rewritten as:

$$\left(\frac{\mathbf{m_{0}}_A}{\mathbf{m_{1}}_A}\right)^{(q^\theta-1)} = 1 \tag{2.8}$$

Hence for every two possible "plaintexts" from equation 2.8, the quotient of those two "plaintexts" must be a $(q^\theta - 1)$ root of unity in $\mathbb{F}_{q^n}$.

Thus finding the maximum number of $(q^\theta - 1)$-root of unity in $\mathbb{F}_{q^n}$ leads to the maximum number of solutions of eq. 2.8 and to an upper bound for $n - \Lambda_{\bar{c}}$.

The next theorem is classical, a proof can be found in [36].

Theorem 2.2.1 *There are $gcd(q^\theta - 1, q^n - 1)$ $(q^\theta - 1)$-root of unity in $\mathbb{F}_{q^n}$. Moreover $gcd(q^\theta - 1, q^n - 1) = q^t - 1$, where $t = gcd(\theta, n)$.*

The theorem proves that there are (at most) $q^{gcd(\theta,n)} - 1$ possible (non-trivial) plaintext solutions to the equation 2.5. So $n - \Lambda_{\bar{c}} \leq gcd(\theta, n)$.

Now the question is : What is the biggest possible value for $gcd(\theta, n)$?

Lemma 3 *$n - \Lambda_{\bar{c}} \leq \frac{n}{3}$ and there exists $\bar{c}$ such that $n - \lambda_{\bar{c}} = \frac{n}{3}$.*

Proof.

- $gcd(\theta, n) = n$ is not possible.
  Otherwise $\theta = kn$ ($k \in \mathbb{Z}$) and then $\forall \mathbf{x}$, $\mathbf{x}^{q^\theta} = \mathbf{x}$ (because $(\mathbb{F}_{q^n}^*, \times)$ has order $q^n - 1$). So $\mathbf{S}'(\mathbf{x}) = \mathbf{x}^{q^\theta + 1} = \mathbf{x}^2$.
  Moreover q is a power of 2, so $\mathbb{F}_q$ has characteristic 2, thus $\mathbf{S}'(\mathbf{x}) = \mathbf{x}^2$ is a linear function, which is impossible.

- $gcd(\theta, n) = \frac{n}{2}$ is not possible.
  Otherwise $\theta = k_2 n + \frac{n}{2}$ for $k_2 \in \mathbb{Z}$.

  Let $d = gcd(q^\theta + 1, q^n - 1) = gcd(q^{k_2 n + \frac{n}{2}} + 1, q^n - 1)$.
  For all $k \in \mathbb{Z}$, $d = gcd(q^{k_2 n + \frac{n}{2}} + 1 - k(q^n - 1), q^n - 1)$.
  Setting $k = q^{(k_2 - 1)n + \frac{n}{2}}$ leads to $d = gcd(q^{(k_2-1)n+\frac{n}{2}} + 1, q^n - 1)$.
  Then setting $k = (k_2 - 2)n + \frac{n}{2}$ leads to $d = gcd(q^{(k_2-2)n+\frac{n}{2}} + 1, q^n - 1)$.
  Repeating this process we get $d = gcd(q^{\frac{n}{2}} + 1, q^n - 1)$.

  To conclude, notice that $d = gcd(q^{\frac{n}{2}} + 1, (q^{\frac{n}{2}})^2 - 1) = gcd(q^{\frac{n}{2}} + 1, (q^{\frac{n}{2}} - 1)(q^{\frac{n}{2}} + 1)) = (q^{\frac{n}{2}} + 1) > 1$. But θ and n are to be chosen so that $d = 1$ (cf. 2.0.5).

- $gcd(\theta, n) = \frac{n}{3}$ is possible.
  We prove that if q is even and n is an odd multiple of θ then $gcd(q^\theta + 1, q^n - 1) = 1$.
  Indeed if $n = k\theta$, the same trick as in the last paragraph gives :
  $gcd(q^\theta + 1, q^n - 1) = gcd(q^\theta + 1, (q^n - 1) \mod (q^\theta + 1))$.
  And $(q^n - 1) \mod (q^\theta + 1) = (q^\theta)^k - 1 \mod (q^\theta + 1) = ((q^\theta + 1) - 1)^k - 1 \mod (q^\theta + 1) = (-1)^k - 1 \mod (q^\theta + 1)$.

  So if k is odd and q is even : $gcd(q^\theta + 1, q^n - 1) = gcd(q^\theta + 1, -2) = 1$.
  This means that n and θ can be chosen in order to satisfy $gcd(\theta, n) = \frac{n}{3}$.

Conclusion. The last lemma implies that (for any parameters values) given a ciphertext, it is possible to recover the plaintext in time at most $O(q^{\frac{n}{3}})$. The previously best attack against this scheme was exhaustive search, which runs in time $O(q^n)$. Thus, to prevent it, Imai and Matsumoto chose q, n, such that $q^n \approx 2^{80}$. In order to prevent the new attack, one needs to have $q^{\frac{n}{3}} \approx 2^{80}$. Thus the values n and q have to be increased, which will make the

cryptosystem less efficient.
More importantly, $q^{\frac{n}{3}}$ is just an upper bound. In practice it is possible that for most of the ciphertexts the value $n - \lambda_{\bar{c}}$ is actually much smaller than $gcd(\theta, n)$. For these reasons the system has to be considered broken.

Other Attacks. There are other attacks against this scheme. In [45], the author describes another "linearization attack", very similar to the one presented here. The idea is to find another system of equations relating plaintext and ciphertext, which are linear in the plaintext.

From $\mathbf{c_B} = \mathbf{m_A}^{q^\theta+1}$, one can get $\mathbf{c_B}^h = \mathbf{m_A}$ (remember that h is the inverse of $(q^\theta + 1)$ modulo $(q^n - 1)$). This last equation is linear in the plaintext, as required. But is not exploitable because h is too big, there are too many terms to find during Gaussian reduction.
Indeed, the system obtained from $\mathbf{c_B}^h = \mathbf{m_A}$ is :

$$\sum_{\substack{t_1,\cdots,t_n \\ t_1+\cdots+t_n=h}} \prod_{i=1}^{n} {c_0}_i^{t_i} \alpha'_{t_1,\cdots,t_n,i} = \sum_{i=1}^{n} m_i \beta_i \qquad (2.9)$$

The number of unknowns (α') to find is roughly $h \approx q^n$ which is obviously out of reach.

This direct approach doesn't work. However, in some cases, the number of unknown α' may be less. For instance, if $h = \sum_{i=0}^{\approx \log q^n} h_i 2^i$, $h_i \in \{0, 1\}$ contains very few nonzero h_i. Then, $\mathbf{c_B}^h = \prod_{i=0}^{\approx \log q^n} (\mathbf{c_B}^{2^i})^{h_i}$, moreover, every $\mathbf{c_B}^{2^i}$ is linear in the "ciphertext", because $\mathbb{F}_q$ has characteristic 2. Thus the "ciphertext" terms appear in eq. 2.9 only with low degree, so the number of unknowns α' is also low.

Note that the method we will use to break the $2R$ cryptosystem also breaks the Imai-Matsumoto scheme. Actually, it is even more powerful than the attack presented here. Hence, it allows to find the private keys from the public ones, whereas Patarin's attack enables us to decrypt a given ciphertext, each time using an exhaustive search phase.

2.3 The (full) Imai-Matsumoto Cryptosystem

We will describe the cryptosystem as it was presented by its inventors in [28], then we will explain why the attack on the simplified version works also

on the full version.

To understand the modifications of the simplified scheme, which lead to the full version, it is convenient to have a look at the generic model of figure 1.1.
The simplified version follows the generic model. We refer to the different steps of this model, in order to explain the two modifications which lead to the full Imai-Matsumoto scheme.

- The first modification occurs in step 1 and 3. The linear transformations A, B are exchanged for affines transformations.
  Thus, additionnally to A, a (private) vector $C \in M_n(\mathbb{F}_q)$ has to be chosen. Step 1 becomes $\bar{x}' = A\bar{x} + C$. Step 3 is modified in the same way (with a private vector $D \in M_n(\mathbb{F}_q)$).

- The second modification takes place in step 2.
  Instead of the transformation $\mathbf{S}'(\mathbf{x}) = \mathbf{x}^{q^\theta+1}$, which takes place in an extension of degree n of $\mathbb{F}_q$, we write $n = n_1 + \cdots + n_d$ and use the d extensions $\mathbb{F}_{q^{n_1}}, \cdots, \mathbb{F}_{q^{n_d}}$. In each of these extensions, the function to be performed is $\mathbf{S_i}'(\mathbf{x}) = \mathbf{x}^{q^{\theta_i}+1}$ for a chosen θ_i (with $gcd(q^{\theta_i}+1, q^{n_i}-1) = 1$).
  Concretely, the vector $\mathbf{x}$ in $\mathbb{F}_{q^n}$ is split into d blocks of n_i components. Each of these blocks is regarded as an element in $\mathbb{F}_{q^{n_i}}$ and transformed using the quadratic function $\mathbf{S_i}'$. Once the $\mathbf{S_i}'$'s have been applied, the blocks are concatenated to form the n-dimensionnal input vector of step 3.

  The following picture describes step 2 in the full cryptosystem :

$$\bar{x}' = \underbrace{(\underbrace{\sum_{j=1}^{n} a_{1j}x_j}_{x'_1}, \underbrace{\sum_{j=1}^{n} a_{1j}x_j}_{x'_2}, \cdots, \underbrace{\sum_{j=1}^{n} a_{n_1 j}x_j}_{x'_{n_1}}}_{\bar{x}^*_1=(x'_1,\cdots,x'_{n_p})}, \quad \underbrace{\underbrace{\sum_{j=1}^{n} a_{n_{d-1}j}x_j}_{x'_1}, \cdots, \underbrace{\sum_{j=1}^{n} a_{n_d j}x_j}_{x'_{n_d}})}_{\bar{x}^*_d=(x'_1,\cdots,x'_{n_d})}$$

$$\Downarrow \qquad\qquad \Downarrow$$

$$\mathbf{S'_1}(\mathbf{x^*_1}) = \mathbf{x^*}_1^{q^{\theta_1}+1} \text{ over } \mathbb{F}_{q^{n_1}} \qquad \mathbf{S'_d}(\mathbf{x^*_d}) = \mathbf{x^*}_d^{q^{\theta_d}+1} \text{ over } \mathbb{F}_{q^{n_d}}$$

$$\Downarrow \qquad\qquad \Downarrow$$

$$n_1 \text{ Quad. Pol. in } x'_1, \cdots, x'_{n_1} \qquad n_2 \text{ Quad. Pol. in } x'_1, \cdots, x'_{n_d}$$

$$\Downarrow \qquad\qquad \Downarrow$$

$$(\textstyle\sum_{i,j=1}^{n} r_{1ij}x_ix_j, \cdots, \sum_{j=1}^{n} r_{n_1 j}x_ix_j) \qquad (\sum_{i,j=1}^{n} r_{n_1 ij}x_ix_j, \cdots, \sum_{i,j=1}^{n} r_{n_2 j}x_ix_j)$$

$$n_1 \text{ Quad. Pol in } x_1, \cdots, x_n \qquad n_d \text{ Quad. Pol in } x_1, \cdots, x_n$$

Comments on the Modifications.

The first modification is usual, it occurs in many multivariates schemes. Indeed, it is normal to think of replacing a linear function by an affine one, because it can not worsen the overall security, and might enhance it.
On the other hand, it enlarges the private key and, at least intuitively, doesn't make the problem (on which relies the security of the scheme) any harder. Actually, all the attacks of multivariate schemes work against both linear and affine versions

Concerning the second modification, it is not clear why the authors have decided to use different "small" fields instead of one "big" field.
The authors certainly didn't have the generic construction in mind. Nevertheless, their construction with small fields seems somewhat "artificial", whereas it seems very natural to work in one big field.
The authors might have thought to enhance the overall security of the scheme by introducing other private parameters (the n_i, θ_i), which "certainly" would have to be found in order to break the system. Overall, it is dubious to add a "layer" of security, which has nothing to do with the hard problem the cryptosystem is relying on.

2.4 Crpytanalysis of the (full) Imai-Matsumoto Cryptosystem

The attack of 2.2 works for the full version as well.
Indeed, the value $\mathbf{m_A}$ (resp. $\mathbf{c_B}$) from section 2.2.1 now satisfies $\mathbf{m_A} = \pi(A\bar{m} + C)$ (resp. $\mathbf{c_B} = \pi(B^{-1}(\bar{c} - D))$). So it is still linear in the plaintext (resp. the ciphertext), but with additional constant terms.

Instead of having the single equation 2.5 over the "big" field $\mathbb{F}_{q^n}$, we have now p equations:

$$\mathbf{m}^*_{\mathbf{A_i}} {\mathbf{c}^*_{\mathbf{B_i}}}^{q^{\theta_i}} = {\mathbf{m}^*_{\mathbf{A_i}}}^{q^{2\theta_i}} \mathbf{c}^*_{\mathbf{B_i}} \in \mathbb{F}_{q^{n_i}} \text{ for all } i,\ 1 \leq i \leq d$$

where $\bar{m^*_{A_i}} = (m_{A_{n_i+1}}, \cdots, m_{A_{n_{i+1}}})$ and $\bar{c^*_{B_i}} = (m_{B_{n_i+1}}, \cdots, m_{B_{n_{i+1}}})$.

Thus for all $i \in [1, p]$, the last equation gives rise to n_i equations of the same form as equation 2.6, plus some additional linear and constant values:

$$\sum_{1 \leq k < l \leq n} \alpha_{k,l,r} m_k c_l + \sum_{1 \leq k \leq n} \gamma_{k,r} m_k + \sum_{1 \leq k \leq n} \zeta_{k,r} c_k + d_r = 0 \ \forall r,\ n_i + 1 \leq r \leq n_{i+1}$$

These equations can be found in the very same way as equations 2.6. Therefor the attack is exactly the same as the one against the simplified scheme.

3

The HFE Cryptosystem

One year afer breaking the Imai-Matsumoto cryptosystem, Patarin developped several variants [46, 44] immune against the latter attack.

The most straightforward variant is to change the exponent in the quadratic function $\mathbf{S}'$. This led to the so-called Dragon schemes, which were quickly and cleverly cryptanalysed by Coppersmith and Patarin.

Another variant, or rather generalisation, is the so-called **HFE** (**H**idden **F**ield **E**quation) scheme [46]. This cryptosystem attracted considerable attention within the community.

In this chapter, **HFE** is presented in detail, followed by a failed attack by Kipnis and Shamir [35]. Our cryptanalysis of the **2R** scheme strongly relies on the ideas developped in this attack. Note also that the signature version of **HFE** , named **QUARTZ**, is a very attractive and (still) secure scheme in the field.

3.1 HFE: How does it work ?

Like the Imai-Matsumoto scheme, **HFE** follows the generic construction of figure 1.1. Thus only the quadratic invertible function S' has to be given.

Let q be a prime power, $\mathbb{F}_{q^n}$ is the finite field with q^n elements. $(\beta_1, \cdots, \beta_n)$ is a basis of $\mathbb{F}_{q^n}$ seen as a $\mathbb{F}_q$-vector space.

3.1.1 The function S'

$$
\begin{array}{lccl}
\mathbf{S}' : & \mathbb{F}_{q^n} & \longrightarrow & \mathbb{F}_{q^n} \\
& \mathbf{x} & \longrightarrow & \sum_{i,j} \mathbf{u_{i,j}} \mathbf{x}^{q^{\theta_{i,j}}+q^{\phi_{i,j}}} + \sum_i \mathbf{v_i} \mathbf{x}^{q^{\xi_i}} + \mathbf{w_0} \\
& & & \text{with } q^{\theta_{i,j}} + q^{\phi_{i,j}}, q^{\xi_i} \leq d
\end{array}
$$

Remarks.

- S' contains a sum of several quadratic terms (instead of a single one in Imai-Matsumoto). This protects **HFE** against the attack described in the last chapter.

- Unlike the Imai-Matsumoto's scheme, the function S' is not always a permutation of $\mathbb{F}_{q^n}$. However, the function S' is surjective on the set corresponding to the encrypted plaintexts.

- Let $\mathbf{p}$ be a polynomial of degree d (over $\mathbb{F}_{q^n}$) and let $\mathbf{y}$ be an element of $\mathbb{F}_{q^n}$, there exist polynomial time algorithms (in $d, \log q^n$) which compute $\mathbf{x}$, such that $\mathbf{p}(\mathbf{x}) = \mathbf{y}$. The most famous one is probably Berlekamp's algorithm whose complexity is of $O(nd \log d \log q)$ operations in the field. The asymptotically fastest algorithm is due to Von Zur Gathen and Shoup [33] with complexity $O(d^2 + d \log q^n)(\log d)^{O(1)}$ field operations.
  In practice, if d is very small, the algorithm finds the preimages very quickly. However, if the polynomial is "randomly chosen", the degree d of the polynomials is around q^n. For such a polynomial the algorithms previously mentioned need more than $O(q^n)$ field operations, which is out of reach [1].

- In the generic construction S' is public. In **HFE**, it is not clear whether or not the quadratic function should be publicly available, or part of the secret key.
  In the official draft, the authors suggested to keep S' secret because one does not "need" it to perform encryption. On the other hand, the scheme is supposed to rely on the **MQ** problem. It would be better to assume that the polynomial S' is known to the attackers, because the security "should" not rely on any other problem than **MQ**.

[1] Recall that q^n must be chosen to prevent exhaustive search attacks, so $q^n \geq 2^{80}$.

3.1.2 HFE: A complete description

We give the complete description of **HFE**, in the framework we devised in the chapter 1.

Keys of the cryptosystem

Public	*Private*
$n, q \in \mathbb{N}$,	Matrices $A, B \in M_n(\mathbb{F}_q)$ nonsingular
$\mathbf{S} \begin{cases} P_1(x_1, \cdots, x_n) \\ \vdots \\ P_n(x_1, \cdots, x_n) \end{cases}$	$\mathbf{S}'(\mathbf{x}) = \sum_{i,j} \mathbf{u_{i,j}} \mathbf{x}^{q^{\theta_{i,j}}+q^{\phi_{i,j}}} + \sum_i \mathbf{v_i} \mathbf{x}^{q^{\xi_i}} + \mathbf{w_0}$ with $q^{\theta_{i,j}} + q^{\phi_{i,j}}, q^{\xi_i} \leq d$

Encryption/Decryption:

Encryption
Let $\bar{m} = (m_1, \cdots, m_n) \in (\mathbb{F}_q)^n$ *be the plaintext.*   The ciphertext is $\bar{c} = (c_1, \cdots, c_n) = (P_1(m_1, \cdots, m_n), \cdots, P_n(m_1, \cdots, m_n)) = \mathbf{S}(\bar{m})$.

Decryption
$\bar{m} = (m_1, \cdots, m_n) = A^{-1}\pi^{-1}\mathbf{S}'^{-1}\pi B^{-1}\bar{c}$.

Practical Aspects.

As noticed above, the computation of a preimage of a polynomial (defined over a finite field) is not a hard task, as long as the degree d is low.
In practice, 2 questions arise :

1. First, the quadratic function is randomly chosen, so it may not be a permutation. An image may have several preimages. In this case, to

decrypt, one has to compute all the preimages, and see which of the corresponding plaintexts make sense.
According to the authors, in practice there will always be at most 2 or 3 preimages.

2. The second question is : "how fast can we compute the preimage in practice ?". To answer this, we need practical benchmarks to have an idea of the computational time needed to decrypt in "real life".
The following table is taken from [29], it shows the computational time to recover a preimage, depending on the degree "d" of the polynomial and the size of the field. It was obtained using the NTL library of Victor Shoup [50] (here $q = 2$):

(n, d)	$(80, 129)$	$(80, 257)$	$(80, 513)$	$(128, 129)$	$(128, 257)$
NTL (CPU time)	0.6 sec	2.5 sec	6.4 sec	1.25 sec	3.1 sec

Notice that if the degree is around 100, then decryption is "relatively" efficient, although less efficient than RSA. The first challenge proposed by Patarin (and broken by Faugere in 2003) was $q = 2$, $n = 80$ and $r = 96$.

An efficiency drawback appears. Unlike the Imai-Matsumoto cryptosystem, **HFE** is clearly less efficient than RSA for decryption. Nevertheless, the encryption stays much faster, and in many smart card applications, only encryption is done on the smart card, while decryption is done by a "normal" computer. Thus, despite the weak performances of decryption, **HFE** is still very attractive, because more "smart card friendly" than RSA.

3. Patarin proposed a padding scheme [46] which enables the decrypter to know (automatically) which pre-image of S' leads to the original plaintext. However, we will not detail padding schemes here because the attack in the next section works whether padding is used or not.

3.2 The Attack of Kipnis and Shamir

In 1999, Kipnis and Shamir [35] found an attack against **HFE**.
A clever rewriting process combined with a heuristical linearization technique allows to find the private key from the public one.

Patarin replied in a revised version of [46] that the attack was not fatal. On the one hand, one can choose the parameters so that the attack becomes exponential time. On the other hand, their attack was not able, in practice, to break **HFE** with "reasonable parameters (length)" (quote from [46]).

Surprisingly, in 2007, Jiang et al. [51] found a flaw in the analysis of the attack. It appears that nobody saw a problem in the method used by Kipnis and Shamir, which makes their attack run in exponential time.

However, in the meantime, the first **HFE** challenge had been solved with the Gröbner basis algorithm of Faugere and Joux [29]. Later, the same authors presented heuristic arguments why the Gröbner basis algorithm should run much faster on **HFE** instances than in general [37].

Currently, it is not known how to change the parameters size, to get security against improved Gröbner basis algorithms. The consequence is that very few people still have confidence in the security of this scheme.

The Kipnis-Shamir attack presented in this chapter may not break **HFE**. However, it severely weakens multivariate cryptography in general. Indeed, it shifts the difficulty of breaking the multivariate schemes from the **MQ** problem, to problems which are in many schemes much easier to solve. This is what will happen to the **2R** schemes.

3.2.1 Description of the attack

The attack proceeds in two steps. The first one is a rewriting phase which sheds new light on the **MQ** problem. Indeed, one proves that a system of polynomial equations in n variables over $\mathbb{F}_q$ is "equivalent" to a single univariate polynomial (of a special form) in an extension of degree n of $\mathbb{F}_q$.

In a second step, we look at **HFE** in this framework. We notice that the condition on S' to have a low degree, translates into a system of quadratic equations where the private key parameters are the unknowns.
The last phase is to use a linearization technique to find the solution of this system.

The important point is that, in this new quadratic system, there are many more equations than unknowns. Such a system is often called "overdefined" and is much easier to solve than **MQ** instances, in which there are as many equations as unknowns.

A small change (in HFE) to simplify the analysis of the attack

For clarity reasons, we first assume that the function $\mathbf{S}'$ does not contain linear and constant terms. Hence, from now on :

$$\mathbf{S}'(\mathbf{x}) = \sum_{i,j} \mathbf{u_{i,j}} \mathbf{x}^{q^{\theta_{i,j}}+q^{\phi_{i,j}}}$$

It is the quadratic part of $\mathbf{S}'$ which makes the **MQ** problem difficult. Intuitively, the linear and constant part neither decrease nor increase the overall security. Anyway, the attack applies, whether linear and constant terms are used or not.

3.2.2 First Step: Changing the Framework

From the beginning of this thesis, in the description of the cryptosystems, we alternate between transformations on elements of $(\mathbb{F}_q)^n$ and transformations in $\mathbb{F}_{q^n}$. This use of different frameworks makes it difficult to analyse and to attack the cryptosystems.
Kipnis and Shamir showed that **HFE** can be described in a unified framework, where all the transformations occur in $\mathbb{F}_{q^n}$.

The next two theorems are taken directly from [35].

Theorem 3.2.1 (Kipnis, Shamir 99) *Let M be a linear mapping from n-tuples to n-tuples of values in $\mathbb{F}_q$. Then there are coefficients $a_1, \cdots, a_n$ in $\mathbb{F}_{q^n}$ such that for any two n-tuples over $\mathbb{F}_q$, $(x_1, \cdots, x_n)$ (which represents $\mathbf{x} = \sum_{i=0}^{n-1} x_i\beta_i$ in $\mathbb{F}_{q^n}$) and $(y_1, \cdots, y_n)$ (which represents $\mathbf{y} = \sum_{i=0}^{n-1} y_i\beta_i$ in $\mathbb{F}_{q^n}$), $(y_1, \cdots, y_n) = M(x_1, \cdots, x_n)$ if and only if $\mathbf{y} = \sum_{i=1}^{n} a_i\mathbf{x}^{q^i}$.*

Proof

It is well known that $\mathbf{F}(\mathbf{x}) = \mathbf{x}^{q^i}$ is an $\mathbb{F}_q$-linear function in $\mathbb{F}_{q^n}$ (seen as a $\mathbb{F}_q$-vector space). Hence, every polynomial $P(\mathbf{x}) = \sum_{i=0}^{n-1} a_i\mathbf{x}^{q^i} \in \mathbb{F}_{q^n}[X]$ is also a $\mathbb{F}_q-$linear.
It follows that there exists a matrix M in $M_n(\mathbb{F}_q)$ such that for any two n-tuples (over $\mathbb{F}_q$) $(x_1, \cdots, x_n)$ (to which corresponds $\mathbf{x} = \sum_{i=1}^{n} x_i\beta_i$ in $\mathbb{F}_{q^n}$) and $(y_1, \cdots, y_n)$ (to which corresponds $\mathbf{y} = \sum_{i=1}^{n} y_i\beta_i$ in $\mathbb{F}_{q^n}$), $(y_1, \cdots, y_n) = M(x_1, \cdots, x_n)$ if $\mathbf{y} = \sum_{i=1}^{n} a_i\mathbf{x}^{q^i}$.

This means that each polynomial of the latter form is "equivalent" to a specific matrix, moreover distinct polynomials lead to distinct matrices.
There are q^{n^2} such polynomials and q^{n^2} different matrices, thus there is a bijection between the polynomial $\sum_{i=0}^{n-1} a_i\mathbf{x}^{q^i}$ and matrices in $M_n(\mathbb{F}_q)$.

Remark. Given a linear mapping $M = (m_{i,j})_{1\leq i,j\leq n}$, we can find the coefficients a_i in time roughly $O(n^5)$.

Let $\pi^{-1}a_i = (a_{i,1}, \cdots, a_{i,n}) \in \mathbb{F}_q^n$, then :

$$\pi(\sum_{i=0}^{n-1} a_i \mathbf{x}^{q^i}) = \begin{pmatrix} P_1((a_{i,j})_{1\leq i,j\leq n}, (x_i)_{1\leq i\leq n}) \\ \vdots \\ P_n((a_{i,j})_{1\leq i,j\leq n}, (x_i)_{1\leq i\leq n}) \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n m_{1,j}x_j \\ \vdots \\ \sum_{i=1}^n m_{n,j}x_j \end{pmatrix}$$

Where the P_i's are linear polynomial in both the $a_{i,j}$'s and the x_i's. We look at the P_i's as polynomials with only the x_i's as unknown, we group the x_i's terms together, this leads for each P_i to n linear equations between the $m_{i,j}$'s and the $a_{i,j}$'s. Using linear algebra algorithms we recover the $a_{i,j}$'s.

Now, we can state an even stronger result. Indeed, using the previous theorem, one can show that any system of n equations in n variables over $\mathbb{F}_q$, has an "equivalent" polynomial in $\mathbb{F}_{q^n}[X]$ of a special shape.

The proof of the next theorem also provides a method for building such an "equivalent" polynomial (in $\mathbb{F}_{q^n}[X]$).

Theorem 3.2.2 (Kipnis, Shamir 99) *Let $P_1(x_1, \cdots, x_n), \cdots, P_n(x_1, \cdots, x_n)$ be any set of n multivariate polynomials in n variables over $\mathbb{F}_q$. Then, there are coefficients $a_1, \cdots, a_{q^n}$ in $\mathbb{F}_{q^n}$ such that for any two n tuples $(x_1, \cdots, x_n)$ and $(y_1, \cdots, y_n)$ in $(\mathbb{F}_q)^n$, $y_j = P_j(x_1, \cdots, x_n)$ for all $1 \leq j \leq n$ if and only if $\mathbf{y} = \sum_{i=1}^{q^n} a_i \mathbf{x}^i$ (where $\mathbf{x} = \sum_{i=0}^{n-1} x_i\beta_i$ and $\mathbf{y} = \sum_{i=1}^{n} y_i\beta_i$ are the elements of $\mathbb{F}_{q^n}$ which correspond to the two vectors over $\mathbb{F}_q$).*

Proof The mapping $\overline{\phi_{1,i}} : (x_1, \cdots, x_n) \rightarrow (x_i, 0, \cdots, 0)$ is $\mathbb{F}_q$-linear. Thus, the first theorem shows that there exists a polynomial $P_{1,i} \in \mathbb{F}_{q^n}[X]$ so that $\overline{P_{1,i}(\mathbf{x})} = \overline{\phi_{1,i}}(\bar{x})$ [2] .
To represent the mapping $(x_1, \cdots, x_n) \rightarrow (\prod_{i=1}^k x_i^{c_i}, 0, \cdots, 0)$ we multiply the polynomials $P_{1,i}$ corresponding to each linear transformation.
The polynomial corresponding to the following mapping, $(x_1, \cdots, x_n) \rightarrow (0, \cdots, 0, \underbrace{\prod_{i=1}^k x_i^{c_i}}_{jthcomponent}, 0, \cdots, 0)$ is simply the product of all the polynomials $P_{1,i}^{c_i}$ $(i = 1, \cdots, n)$ multiplied with $\beta_1^{-1}\beta_j$.

This proof leads to the following lemma:

Lemma 4 *Let C be any collection of n homogeneous multivariate polynomials of degree d in n variables over $\mathbb{F}_q$. The only powers of x which can occur with non-zero coefficients in its univariate polynomial representation $G(x)$ over $\mathbb{F}_{q^n}$ are sums of exactly d (not necessarily distinct) powers of q*

[2] Remember that the notation $\overline{P_{1,i}(\mathbf{x})}$ stands for $\pi^{-1}P_{1,i}(\pi\bar{x})$

: $q^{i_1} + q^{i_2} + \cdots + q^{i_n}$. If d is a constant, then $G(x)$ is sparse [3] *, and its coefficients can be found in polynomial time.*

Remark. Given a system of n polynomials (in n variables) of maximum total degree d over $\mathbb{F}_q$, we can find the coefficients of the corresponding polynomial over $\mathbb{F}_{q^n}$ in polynomial time. Indeed, one can compute all the polynomials $P_{1,i}$ $(i = 1, \cdots, n)$ and then compute their product (as mentioned in the later proof). Another strategy is to use an interpolation technique based on sufficiently many input/output pairs.

3.2.3 Application of the theorems

According to the latter theorem, it is possible to find the polynomial $\mathbf{S}$ in $\mathbb{F}_{q^n}[X]$ so that $\pi^{-1}\bar{S}(\pi\mathbf{x}) = (P_1(x_1, \cdots, x_n), \cdots, P_n(x_1, \cdots, x_n))$ [4].

Moreover, there exist $\mathbf{P_A}$ (resp. $\mathbf{P_B}$) such that $\mathbf{P_A}(\mathbf{x}) = \sum_{i=0}^{n-1} \mathbf{a_i x}^{q^i} \in \mathbb{F}_{q^n}[X]$ (resp. $\mathbf{P_B}(\mathbf{x}) = \sum_{i=0}^{n-1} \mathbf{b_i x}^{q^i}$) verifying the following equation:

$$\mathbf{S}(\mathbf{x}) = \mathbf{P_B}(\mathbf{S}'(\mathbf{P_A}(\mathbf{x})))$$

We rewrite this equation in a more convenient way. If the private matrix B is nonsingular, according to theorem 3.2.1 the polynomial $\mathbf{P_B}$ is invertible and ${\mathbf{P_B}}^{-1} = \mathbf{P_{B^{-1}}}$.

Given $\mathbf{S}$, the attack of Kipnis and Shamir consists in recovering the polynomials $\mathbf{P_{B^{-1}}}$ and $\mathbf{P_A}$ such that :

$$\mathbf{P_{B^{-1}}}(\mathbf{S}(\mathbf{x})) = \mathbf{S}'(\mathbf{P_A}(\mathbf{x})) \qquad (3.1)$$

3.2.4 Second Step : Linearization Attack

The quadratic function $\mathbf{S}'$ is private, so the attacker doesn't know the right side of equation 3.1. However, he knows that $\mathbf{S}'$ has a degree d which is much smaller than q^n (the highest possible degree for any polynomial in $\mathbb{F}_{q^n}[X]$). How to exploit this property ?

Let $\mathbf{S}'(\mathbf{P_A}(\mathbf{x})) = \sum_{i,j} \mathbf{u_{i,j}} (\sum_{\mathbf{t=0}}^{\mathbf{n-1}} \mathbf{a_t x}^{\mathbf{q^t}})^{q^i+q^j}$

$\mathbf{S}'$ has very few quadratic terms (of the form $\mathbf{x}^{q^i+q^j}$), but the composition results in a polynomial $\mathbf{S}'(\mathbf{P_A}(\mathbf{x}))$ with many quadratic terms. Looking merely at the polynomial composition does not seem to help.

[3] Sparsity is a measure of the ratio between the number of non-zero terms and the degree of the polynomial. Roughly, a polynomial is sparse if only few terms are non-zero.

[4] Recall that the $P_1, \cdots P_n$ are the public polynomials.

Kipnis and Shamir noticed that the low degree of $\mathbf{S}'$ is exploitable, when the attacker looks at the polynomials as some non-standard quadratic forms.

Quadratic Forms.

Recall that a quadratic form over a field $\mathbb{F}$ is a homogeneous quadratic polynomial in n variables : $G(x_1, \cdots, x_n) = \sum_{1 \leq i,j \leq n} g_{i,j} x_i x_j$.
It is sometime convenient when working with quadratic form to use matrix notation. Given a polynomial $G(x_1, \cdots, x_n)$, there are (square) matrices $\underline{G} = (g_{i,j})$ such that $\bar{x}^t \underline{G} \bar{x} = G(x_1, \cdots, x_n)$.

The matrix $\underline{G}$ is not unique, for instance

$$G(x_1, x_2) = 4x_1^2 + 6x_1x_2 + x_2^2 = \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} 4 & 6 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} 4 & 3 \\ 3 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

If the field's characteristic is not 2, it is obvious that there is a unique symmetric $\underline{G}$: $\underline{G} = \begin{pmatrix} g_{1,1} & \frac{g_{1,2}+g_{2,1}}{2} & \cdots & \frac{g_{1,n}+g_{n,1}}{2} \\ \frac{g_{1,2}+g_{2,1}}{2} & \ddots & \cdots & \frac{g_{2,n}+g_{n,2}}{2} \\ \vdots & & \ddots & \vdots \\ \frac{g_{1,n}+g_{n,1}}{2} & \cdots & & g_{n,n} \end{pmatrix}$
In fields of characteristic 2, unicity can be obtained by requiring the matrix to be upper triangular [5].

Recall equation 3.1:

$$\mathbf{P_{B^{-1}}}(\mathbf{S}(\mathbf{x})) = \mathbf{S}'(\mathbf{P_A}(\mathbf{x}))$$

This equation can be seen as an equality between non standard quadratic form. Indeed, one writes $\mathbf{P_{B^{-1}}}(\mathbf{S}(\mathbf{x})) = \bar{x}^t(\underline{P_{B^{-1}} \circ S})\bar{x}$ where $\bar{x} = (\mathbf{x^{q^0}}, \mathbf{x^{q^1}}, \cdots, \mathbf{x^{q^{n-1}}})$ and $\underline{P_{B^{-1}} \circ S}$ is a matrix representation of the polynomial $\mathbf{P_{B^{-1}}}(\mathbf{S}(\mathbf{x}))$.
It is not a " true" quadratic form as defined below. However, the unicity of matrix representation seen in the last paragraph still holds.

Thus, equation 3.1 leads the following matrix equality [6] :

[5] Note that getting unicity by requiring the matrix to be triangular can also be used in characteristic $\neq 2$, but the averaging method is more "standard"

[6] When we require that both matrix are symmetric or upper triangular.

$$\underline{P_{B^{-1}} \circ S} = \underline{S' \circ P_A} \tag{3.2}$$

The fact that $\mathbf{S}'$ has a low degree can be exploited in equation 3.2, due to the specific shape of these matrices. The following theorem explicits these shapes. Recall that $\mathbf{P_A(x)} = \sum_{i=0}^{n-1} \mathbf{a_i} \mathbf{x}^{q^i} \in \mathbb{F}_{q^n}[X]$ and $\mathbf{P_{B^{-1}}(x)} = \sum_{k=0}^{n-1} \mathbf{p_{B^{-1},k}} \mathbf{x}^{q^k}$

Theorem 3.2.3 *The matrix of the quadratic form in $\bar{x}$ which represents the polynomial composition $\mathbf{P_{B^{-1}}(S(x))}$ is $\sum_{k=0}^{n-1} \mathbf{p_{B^{-1},k}} \underline{S_k^*}$ where $\underline{S_k^*}$ is obtained from the $n \times n$ matrix representation of $\mathbf{S}$ by raising each one of its entries to the power q^k in $\mathbb{F}_{q^n}$, and cyclically rotating forwards by k steps both the rows and the columns of the result. The matrix of the quadratic form in $\bar{x}$ which represents the polynomial composition $\mathbf{S}'(\mathbf{P_A(x)})$ is $\underline{P_A^*}^t \underline{S'} \underline{P_A^*}$ in which $\underline{P_A^*} = \left(\mathbf{p_{i,j}^*}\right)$ is an $n \times n$ matrix defined by $\mathbf{p_{i,j}^*} = (\mathbf{a_{j-i}})^{\mathbf{q^i}}$, where $j - i$ is computed modulo n.*

Proof

The polynomial representation of $\mathbf{P_{B^{-1}}(x)}$ is $\sum_{k=0}^{n-1} \mathbf{p_{B^{-1},k}} \mathbf{x}^{q^k}$ and the polynomial representation of $\mathbf{S(x)}$ is $\sum_{i,j=0}^{n-1} \mathbf{s_{i,j}} \mathbf{x}^{q^i+q^j}$. Their polynomial composition can be evaluated using the fact that raising sums to the power q^i is a linear operation:

$$\mathbf{P_{B^{-1}}(S(x))} = \sum_{k=0}^{n-1} \mathbf{p_{B^{-1},k}} (\sum_{i,j=0}^{n-1} \mathbf{s_{i,j}} \mathbf{x}^{q^i+q^j})^{q^k} = \sum_{k=0}^{n-1} \mathbf{p_{B^{-1},k}} (\sum_{i,j=0}^{n-1} (\mathbf{s_{i,j}})^{q^k} \mathbf{x}^{q^{i+k}+q^{j+k}})$$

The exponents of q can be reduced modulo n since $\mathbf{x}^{q^n} = \mathbf{x}$. From now on, let all the indices in the sum be computed modulo n.

$$\sum_{k=0}^{n-1} \mathbf{p_{B^{-1},k}} (\sum_{i,j=0}^{n-1} (\mathbf{s_{i,j}})^{q^k} \mathbf{x}^{q^{i+k}+q^{j+k}}) = \sum_{k=0}^{n-1} \mathbf{p_{B^{-1},k}} (\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (\mathbf{s_{i-k,j-k}})^{q^k} \mathbf{x}^{q^i+q^j})$$

The matrix (of the quadratic form) representation of this polynomial in terms of $\bar{x}$ is:

$$\underline{P_{B^{-1}} \circ S} = \sum_{k=0}^{n-1} \mathbf{p_{B^{-1},k}} \underline{S_k^*}, \text{ the } (i,j)\text{-th entry of } \underline{S_k^*} \text{ is } (\mathbf{s_{i-k,j-k}})^{q^k}$$

The proof of the other composition is similar:

$$\mathbf{S}'(\mathbf{P_A(x)}) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\mathbf{u_{i,j}}(\sum_{\mathbf{t=0}}^{\mathbf{n}}\mathbf{a_t x^{q^t}})^{q^i+q^j} = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\mathbf{u_{i,j}}(\sum_{\mathbf{t=0}}^{\mathbf{n}}\mathbf{a_t x^{q^t}})^{q^i}(\sum_{\mathbf{v=0}}^{\mathbf{n}}\mathbf{a_v x^{q^v}})^{q^j}$$

Again we use linearity and cyclic index shifting to evaluate $\mathbf{S}'(\mathbf{P_A(x)})$ as :

$$\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\mathbf{u_{i,j}}(\sum_{\mathbf{t=0}}^{\mathbf{n}}\mathbf{a_t^{q^i}x^{q^{t+i}}})(\sum_{\mathbf{v=0}}^{\mathbf{n}}\mathbf{a_v^{q^j}x^{q^{v+j}}})$$

After rearranging we get:

$$\mathbf{S}'(\mathbf{P_A(x)}) = \sum_{t=0}^{n-1}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\sum_{v=0}^{n-1}\mathbf{x}^{q^u}\mathbf{a_{t-i}^{q^i}u_{i,j}a_{v-j}^{q^j}x}^{q^v} = \bar{x}^t \underline{P_A^*}^t \underline{S'}\underline{P_A^*}\bar{x}$$

where $\underline{P_A^*}$ is the matrix specified in the theorem. □

Important Remark.

As already mentioned, the matrix representation for quadratic form is not unique in general, but is unique as long as one requires the matrix to be triangular or symmetric. Here $\underline{S_k^*}$, $\underline{P_A^*}$ and $\underline{S'}$ are the unique triangular (or symmetric) representation of $\mathbf{S_k^*}$, $\mathbf{P_A^*}$ and $\mathbf{S}'$. In order to translate the polynomial equation 3.2 to a matrix equation, we have to verify that $\sum_{k=0}^{n-1}\mathbf{p_{B^{-1},k}}\underline{S_k^*}$ and $\underline{P_A^*}^t\underline{S'}\underline{P_A^*}$ are also triangular (or symmetric) matrices.

- If $\underline{S_k^*}$, $\underline{S'}$ and $\underline{P_A^*}$ are upper triangular, then both $\sum_{k=0}^{n-1}\mathbf{p_{B^{-1},k}}\underline{S_k^*}$ and $\underline{P_A^*}^t\underline{S'}\underline{P_A^*}$ are also triangular.
- If the matrices are symmetrics [7]:
  - If $\underline{S'}$ is the symmetric matrix corresponding to the "quadratic form" $\mathbf{S}'$, the matrices $\underline{S_k^*}$ are also symmetric [8], so $\sum_{k=0}^{n-1}\mathbf{p_{B^{-1},k}}\underline{S_k^*}$ is also symmetric.
  - Let $\underline{S'} = (\mathbf{s'_{ij}})$ and $\underline{P_A^*} = (\mathbf{p_{ij}})$.
    Then $\underline{S'}\underline{P_A^*} = (\mathbf{c_{ij}})$ and $c_{ij} = \sum_{k=1}^{n}\mathbf{s'_{ik}p_{kj}}$.

    $\underline{P_A^*}^t\underline{S'}\underline{P_A^*} = (\mathbf{d_{ij}})$ and $d_{ij} = \sum_{t=1}^{n}\mathbf{p_{ti}c_{tj}} = \sum_{t=1}^{n}\mathbf{p_{ti}}(\sum_{k=1}^{n}\mathbf{s'_{tk}p_{kj}}) = \sum_{t=1}^{n}\sum_{k=1}^{n}\mathbf{p_{ti}s'_{tk}p_{kj}}$, so $d_{ij} = d_{ji}$.

[7] which is the one the authors suggested

[8] Rotating k rows and columns does not change the symmetric property.

Both matrices given by the theorem are symmetric, thus the quadratic form equality translates to the equality of the two matrices.

3.2.5 Recovering The private matrix B

The MinRank Problem

From theorem 3.2.2 we know that it is possible to compute (in polynomial time) $\underline{S_k^*}$ for $k \in [0, n-1]$, such that :

$$\sum_{k=0}^{n-1} \mathbf{p_{B^{-1},k}} \underline{S_k^*} = \underline{P_A^*}^t \underline{S'} \underline{P_A^*} \tag{3.3}$$

for some $\mathbf{p_{B^{-1},k}}$ in $\mathbb{F}_{q^n}$.

The right side of equation 3.3 is unknown to the attacker, but he still knows something about $\underline{S'}$. It is the matrix corresponding to the "quadratic form" $\mathbf{S'}(\mathbf{x}) = \sum_{i,j} \mathbf{u_{i,j}} \mathbf{x}^{q^{\theta_{i,j}}+q^{\phi_{i,j}}}$ which has a low degree.

Low degree means also few non zero terms, so the matrix $\underline{S'}$ contains a lot of zero columns. More precisely, if $\theta_{i,j}, \phi_{i,j} \leq r$, the matrix representation $\underline{S'}$ has its non-zero elements only in the upper left block of size $r \times r$. Indeed, the (i,j)-th element of $\underline{S'}$ is the coefficient corresonding to the term $\mathbf{x}^{\mathbf{q^i}+\mathbf{q^j}}$ of the polynomial $\mathbf{S'}(\mathbf{x})$.

Moreover, $rank(\underline{P_A^*}^t \underline{S'} \underline{P_A^*}) \leq rank(\underline{S'})$, thus $rank(\underline{P_A^*}^t \underline{S'} \underline{P_A^*}) \leq r$.

As a conclusion, the coefficient $\mathbf{p_{B^{-1},k}}$ can be computed (therefore also the private matrix B, according to theorem 3.2.1) if we find an algorithm to solve the following problem :

The MinRank Problem

- *Given n square matrices of dimension n: $\underline{S_0^*}, \cdots, \underline{S_{n-1}^*}$ defined over $\mathbb{F}_{q^n}$ and an integer r .*
- *find $p_0, \cdots, p_{n-1}$ in $\mathbb{F}_{q^n}$ such that $\sum_{i=0}^{n-1} p_i \underline{S_i^*}$ has rank equal or less than r.*

To the best of our knowledge, the MinRank problem first appeared in '93. Coppersmith et al. [9, 10] solved an instance of this problem to attack the

birational permutation signature schemes. A generalized version [34] of this problem was introduced in 1996 and was proved NP-hard.
In ′01 N. Courtois published an authentication scheme [13] based on this problem.

Several different methods exist to solve the MinRank problem, depending on the size of the different parameters.
If r is "small" the problem may not be hard. Separately, Coppersmith et al. [9, 10] and Kipnis et al. developped heuristic algorithms to solve it.

First, Coppersmith algorithm will be sketched. Then, the method of Kipnis and Shamir will be detailed.

Remark.
Notice that we suppose that given $\underline{S_0^*}, \cdots, \underline{S_{n-1}^*}$, r is so small that $\mathbf{p_{B^{-1},1}}, \cdots, \mathbf{p_{B^{-1},n}}$ is the only solution to the problem. This heuristic seems intuitively correct because the $\underline{S_i^*}$'s are somehow random and so should have rank n. The linear combinations of the $\underline{S_i^*}$'s should also have rank n or close to n.

Coppersmith Algorithm

The sum $\sum_{i=0}^{n-1} p_i \underline{S_i^*}$ can be seen as one matrix containing the variables p_i. If this matrix has rank less than r, every submatrix of dimension $(r+1)\times(r+1)$ must have a vanishing determinant. There are $\binom{n}{r+1}^2$ such submatrices, leading to the same amount of polynomials of degree $(r+1)$ in $\mathbb{F}_{q^n}[p_1, \cdots, p_n]$.

Moreover, each of these polynomials contains $\frac{n^{r+1}}{(r+1)!}$ different terms, thus by replacing in each polynomial each term $p_1^{i_1} \cdots p_n^{i_n}$ by an unknown $x_{i_1,\cdots,i_n}$, we get a system of $\binom{n}{r+1}^2$ linear equations in $\frac{n^{r+1}}{(r+1)!}$ variables.

Hopefully $\frac{n^{r+1}}{(r+1)!}$ of the $\binom{n}{r+1}^2$ equations are linearly independent, then solving the system requires roughly $n^{3r+o(1)}$ steps.

Once the linear system (with unknowns the $x_{i_1,\cdots,i_n}$) has been solved, it is easy to compute the solution in term of the orginal unknowns $p_1, \cdots, p_n$.

Notice that the time complexity is polynomial for fixed r. In practice, it can work as long as r is small.

Kipnis-Shamir Algorithm

This algorithm works in the same way the previous one does. It finds many non-linear equations, and uses an heuristic argument to linearize them and to obtain a square linear system. The solutions are found by inverting this system.

- *Finding the (overdefined) non-linear system.*

  It is possible to derive a system of quadratic equations from the rank condition.

  As above, consider the sum $\sum_{i=0}^{n-1} p_i \underline{S_i^*}$ as a square matrix containing n variables (the p_i) whose rank must be less than or equal to r.
  The kernel of this matrix [9] is a vector space of dimension at least $(n-r)$.
  It follows that there exists $(n-r)$ vectors $\bar{x_i} = (x_{i_1}, \cdots, x_{i_n}) \in (\mathbb{F}_{q^n})^n$ whose multiplication with the matrix gives the zero vector.

  In other words, $\bar{x_j}(\sum_{i=0}^{n-1} p_i \underline{S_i^*}) = (0)^n$ for all j in $[1, n-r]$. Thus we get $n(n-r)$ quadratic equations in $n(n-r)+n$ variables (the x_{i_j} with $(i,j) \in [1, n-r] \times [1, n]$) over $\mathbb{F}_{q^n}$.
  If r is small the number of equations is $O(n^2)$, roughly the same as the number of variables. This quadratic system is not yet overdefined. Fortunately there is an easy argument to turn it into an overdefined system.

  The vectors $\bar{x_i}$'s which form the kernel vector space of dimension $(n-r)$ are contained in the ambiant space of dimension n.
  A basis of this kernel can be written as a $(n-r) \times n$ matrix.

$$\begin{matrix} \bar{x}_1 \\ \vdots \\ \bar{x}_{n-r} \end{matrix} \quad \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & & \ddots & \vdots \\ x_{n-r,1} & \cdots & & x_{n-r,n} \end{pmatrix}$$

  This is not a square matrix. We can multiply it by an invertible transformation leading to another basis $(\bar{x_1}', \cdots, \bar{x_n}')$ with the following shape :

$$\begin{matrix} \bar{x_1}' \\ \vdots \\ \bar{x_{n-r}}' \end{matrix} \quad \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & x'_{1,n-r} & \cdots & x'_{1,n} \\ 0 & 1 & 0 & \cdots & 0 & x'_{2,n-r} & \cdots & x'_{2,n} \\ \vdots & & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & 1 & x'_{n-r,n-r} & \cdots & x'_{n-r,n} \end{pmatrix}$$

[9] $\bar{x}$ such that $\bar{x}(\sum_{i=0}^{n-1} p_i \underline{S_i^*}) = 0$.

In this last matrix, there are only $r \times (n-r)$ unknowns.
On the other hand, $\bar{x_j}'(\sum_{i=0}^{n-1} p_i \underline{S_i^*}) = 0$ holds.

Thus we get a system of $n \times (n-r)$ quadratic equations and $r \times (n-r)$ unknowns.
If $r \ll n$ this system is overdefined.

- *The relinearization technique.*

In this section, we are interested in solving an overdefined system of quadratic equations in some finite field.
For greater generality, let us say that this system has m variables, and many more equations, for instance ϵm^2 with $\epsilon > 0$. Kipnis and Shamir gave an algorithm [35] to solve this system, when ϵ is not too small.

First, if $\epsilon \gtrsim \frac{1}{2}$ it is possible to solve the problem. Indeed, set new variables $y_{ij} = x_i x_j$ where $i \geq j$. The equations become linear in the $\approx \frac{m^2}{2}$ variables y_{ij}. When $\epsilon \gtrsim \frac{1}{2}$, there are more equations than variables, so this can be solved using linear algebra.

Remark. This method is only heuristic because we assume that all (or most) of the linearized equations are linearly independent. This should be the case as long as the equations are sufficiently "random".
Moreover, we assume that if the number of equations is (much) larger than $\frac{m^2}{2}$ there will only be very few parasitic solutions for the $y_{i,j}$. Parasitic solutions are solutions for $y_{i,j}$ which do not translate into solutions for the x_i's. In [11], the authors have conducted extensive experiments in the same context, and this heuristic always turned out to be right.

Essentially, if $\epsilon < \frac{1}{2}$ we set the new variables y_{ij}, but there are fewer (linear) equations than variables. The set of solutions is a vector space of dimension roughly [10] $(\frac{1}{2} - \epsilon)m^2$.

It is easy to find a basis $(b_1, \cdots, b_{(\frac{1}{2}-\epsilon)m^2})$ for this space. Such a basis allows to reduce the number of unknowns. Indeed, the solution we are looking for is somewhere in the previously computed kernel. It corresponds to a vector of the form $(y_{11}, \cdots, y_{mm}) = \sum_{i=1}^{(\frac{1}{2}-\epsilon)m^2} z_i b_i$ where the vectors b_i are known but the coefficients $z_i \in \mathbb{F}_q$ are unknowns.
In every equation we replace the y_{ij} with this linear combination, this

[10] Here again, we assumed that most of the equations are linearly independent.

decreases the number of variables from $O(\frac{m^2}{2})$ to $O((\frac{1}{2}-\epsilon)m^2)$.

Notice that the linearization step also produces new equations :

$(x_a x_b)(x_c x_d) = (x_a x_c)(x_b x_d) = (x_a x_d)(x_c x_b) \Rightarrow y_{ab}y_{cd} = y_{ac}y_{bd} = y_{ad}y_{bc}$

For each sorted list (a, b, c, d) we get 2 equations (for simplicity we neglect the case where a, b, c, d are not distinct). There are $\approx \frac{m^4}{12}$ new quadratic equations in y_{ij}, leading to the same amount of quadratic equations in the z_i.
These quadratic equations (in the $(\frac{1}{2}-\epsilon)m^2$ variables z_k) are linearized again to get $\frac{m^4}{12}$ linear equations in $\frac{((\frac{1}{2}-\epsilon)m^2)^2}{2}$ variables.
This system is solvable if there are more equations than variables, so if $\frac{m^4}{12} \geq \frac{((\frac{1}{2}-\epsilon)m^2)^2}{2}$ which corresponds to $\epsilon \gtrsim 0.1$.

Therefore, if $\epsilon \gtrsim 0.1$, it is possible to solve the problem in polynomial time. For a detailed and precise analysis of the algorithm, see [40]. Of course, the method presented above can be generalised in many ways [35]. In [12] it is shown that the heuristical argument works well "in practice", moreover, they conjectured that one can solve the system for every $\epsilon > 0$ in time $n^{O(\frac{1}{\sqrt{\epsilon}})}$. Thus, even for ϵ smaller than 0.1 this type of method might work.

Let us apply the previous method to the concrete instance of the MinRank problem exhibited by the attack.
The number of equations is $n(n-r)$ and the number of variables is $r(n-r)+n$. In this case, $\epsilon = \frac{1}{1+\frac{1}{r}-\frac{r}{n}} - \frac{1}{1+r-\frac{r^2}{n}}$. If r is small and $r \ll n$ then $\epsilon > 0.1$, the method works. Moreover if r is a constant in regard to n , it works in time polynomial in n.

As a conclusion, assuming that the heuristic is correct, the attacker can recover the $\mathbf{p_{B^{-1},k}}$ from equation 3.3. Therefore, he can recover the private matrix B.

On the correctness of the heuristic. The heuristic used here is very general. They work for sufficiently random system of equations, that was experimented in [12].

3.2.6 Recovering The (private) A Matrix

Having recovered the matrix B, the matrix $\sum_{k=0}^{n-1} \mathbf{p_{B^{-1},k}} \underline{S_k^*}$ is also fully known. The problem is now to recover the matrix $\underline{P_A^*} = (\mathbf{P^*_{A_{i,j}}})_{1 \leq i,j \leq n}$ from

:

$$\sum_{k=0}^{n-1} \mathbf{p_{B^{-1},k}} \underline{S_k^*} = \underline{P_A^{*t}}\,\underline{S'}\,\underline{P_A^*}$$

Which leads to the matrix A.

As mentioned before, if the polynomial $\mathbf{S'}$ has degree r, the matrix $\underline{S'}$ has a rank at most r.

First, suppose that $\underline{S'}$ has rank exactly r and that the matrix $\underline{P_A^*}$ is full rank, ie n.

Now the matrix $\underline{P_A^{*t}}\,\underline{S'}\,\underline{P_A^*}$ is known, so one can compute its (left) kernel, whose dimension is $n-r$.
Besides, $\underline{P_A^*}$ is invertible, so this kernel is also the kernel of $\underline{P_A^{*t}}\,\underline{S'}$. Let $(\mathbf{b_1}, \cdots, \mathbf{b_{n-r}})$ in $(\mathbb{F}_{q^n})^{(n-r)}$ be a basis of the computed kernel. It follows that $\mathbf{b_1}\underline{P_A^{*t}}, \cdots, \mathbf{b_{n-r}}\underline{P_A^{*t}}$ is a basis of $Kern(\underline{S'})$.

Moreover, all vectors of $Kern(\underline{S'})$ have the following form $(\underbrace{0, \cdots, 0}_{r}, *, \cdots, *)$.
Indeed, if some of the first coordinates were not zero, that would means that a linear combination of the first r rows of $\underline{S'}$ leads to the zero vectors. Contradicting the fact that $\underline{S'}$ has rank r and is of the following form :

$$\underline{S'} = \begin{pmatrix} \mathbf{u_{1,1}} & \cdots & \cdots & \mathbf{u_{1,r}} & 0 & \cdots & 0 \\ 0 & \mathbf{u_{2,2}} & \cdots & \mathbf{u_{2,r}} & \vdots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & \mathbf{u_{r,r}} & 0 & \cdots & 0 \\ 0 & \ldots & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & & & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & \cdots & 0 & 0 & \ldots & 0 \end{pmatrix} \in (\mathbb{F}_{q^n})^{n \times n}$$

This implies that $\mathbf{b_i}\underline{P_A^{*t}} = (\overbrace{0, \cdots, 0}^{r}, \cdots)$ holds for all $i \in [1, n-r]$. This provides $r(n-r)$ linear equations (over $\mathbb{F}_{q^n}$) in the n^2 unknown coefficients $\mathbf{p_{A^*_{i,j}}}$ of $\underline{P_A^{*t}}$.
From theorem 3.2.3, $\mathbf{p_{A^*_{i,j}}} = \mathbf{a_{j-i}}^{q^i}$ (where the $\mathbf{a_{j-i}}$ are the coefficient of $\mathbf{P_A(x)}$) which reduces the number of unknowns to n, but these equations in the $\mathbf{a_{j-i}}$'s are not linear.

However, due to the form of the $\mathbf{p^*_{i,j}}(= \mathbf{a_{j-i}}^{q^i})$ it is possible to get linear equations, considering the latter as equations over $\mathbb{F}_q$ instead of $\mathbb{F}_{q^n}$. $\mathbf{a_t} = \sum_{j=1}^n a_{t,j}\beta_j$, and $\mathbf{a_t}^{q^i} = \sum_{j=1}^n a_{t,j}\beta_j^{q^i} = \sum_{j=1}^n a_{t,j}(\sum_{r=1}^n l_{r,i,j}\beta_r)$ for

which the component along every basis element is linear in the $a_{t,j}$.

Therefore, the condition $\mathbf{x_i}\underline{P_A^*}^t = (\overbrace{0,\cdots,0}^{r},\cdots)$ provides $nr(n-r)$ linear equations over $\mathbb{F}_q$ in n^2 unknowns (the $a_{i,j}$). For any r the system is greatly overdefined. Thus one can invert it, [11], and recover the private matrix A.

In the case where $\underline{S'}$ (resp. $\underline{P_A^*}$) does not have rank r (resp. n), the kernel of the matrix is bigger which leads to more equations. On the other hand the condition $\mathbf{x_i}\underline{P_A^*}^t = (\overbrace{0,\cdots,0}^{r},\cdots)$ may not always hold.
Anyway, they are many more equations than unknowns. It is always possible to dismiss some of the equations. Moreover, the rank of $\underline{S'}$ (resp. $\underline{P_A^*}$) should be close to r (resp. n).

3.2.7 Recovering the (private) Polynomial $\mathbf{S'}$

Once the matrices A and B have been found, it is easy to find the Polynomial $\mathbf{S'(x)}$. Indeed, we now have $\mathbf{P_{B^{-1}(x)}}$, $\mathbf{P_A(x)}$, $\mathbf{S(x)}$, and equation 3.1 holds :

$$\mathbf{P_{B^{-1}}(S(x)) = S'(P_A(x))}$$

The left part of the latter equation is a polynomial which is fully known. On the right side we have the following composition :

$$\mathbf{S'(P_A(x))} = \sum_{u,v} \mathbf{s'_{u,v}} (\sum_{i=0}^{n-1} \mathbf{a_i x^{q^i}})^u (\sum_{i=0}^{n-1} \mathbf{a_i x^{q^i}})^v$$

where the $\mathbf{s'_{u,v}}$'s are unknowns. If we expand this last expression, we see that each term $\mathbf{x}^{q^t}\mathbf{x}^{q^r}$ has a linear combination of the $\mathbf{s'_{u,v}}$ as coefficients. Moreover this linear combination is equal to the (beknown) coefficient of the same term $\mathbf{x}^{q^t}\mathbf{x}^{q^r}$ in the left part of equation 3.1. This way we get $\frac{n(n+1)}{2}$ linear equations, which is enough to find the few $\mathbf{s'_{u,v}}$ [12].

3.2.8 Conclusion

Attacking the non simplified HFE scheme

Recall that in 3.2.1 a simplified version of **HFE** was introduced in which the function $\mathbf{S'}$ contains only quadratic terms and no longer linear and constant

[11] Notice that here again we assume that there won't be too much linear dependant equations

[12] Remember that the polynomial $\mathbf{S'(x)}$ has a very small degree, so very few non zero terms, see section 3.1.2.

terms as specified in the (general) scheme (see 3.1.2).

The attack presented above works on this simplified version of **HFE**, where $\mathbf{S}'$ contains no linear and constant terms.

This "simplified version" was introduced only for clarity reasons. It is easy to transpose the attack to the general version.

Let $\mathbf{S}'(\mathbf{x}) = \sum_{i,j} \mathbf{u_{i,j}} \mathbf{x}^{q^{\theta_{i,j}}+q^{\phi_{i,j}}} + \sum_i \mathbf{v_i} \mathbf{x}^{q^{\xi_i}} + \mathbf{w_0}$.

The public key is the sum of terms coming from the quadratic part of $\mathbf{S}'$ and other coming from the linear and constant part. Moreover these parts can be identified in the public key because of the linearity of π^{-1} and B :

$$\begin{aligned} \bar{c} &= B\pi^{-1}[\mathbf{S}'(\pi(A\bar{m}))] = B\pi^{-1}[\mathbf{S}'_{quadratic}(\pi(A\bar{m}))] + B\pi^{-1}[\mathbf{S}'_{lin+const}(\pi(A\bar{m}))] \\ B\pi^{-1}[\mathbf{S}'(\pi(A\bar{m}))] &= \bar{c}_{quadratic} + \bar{c}_{lin+const} \end{aligned}$$

The quadratic terms in the public key comes from the term $B\pi^{-1}[\mathbf{S}'_{quadratic}(\pi(A\bar{x}))]$ which is exactly the simplified version of **HFE** studied so far.

Considering only the quadratic part to be the public key, allows to recover A, B (and $\mathbf{S}'_{quadratic}$).

Once A and B have been found, $\mathbf{S}' = \mathbf{S}'_{quadratic} + \mathbf{S}'_{lin+const}$ can be computed by identification from $\mathbf{P_{B^{-1}}}(\mathbf{S}(\mathbf{x})) = \mathbf{S}'(\mathbf{P_A}(\mathbf{x}))$.

Time complexity of the attack

First, the attacker uses theorem 3.2.2 and the univariate polynomial over $\mathbb{F}_{q^n}$ which corresponds to $\mathbf{S}'$.
Then he finds B via the re-linearization method. Finally, he recovers A by solving a linear system.
The time complexity of the attack is the sum of the time complexity of all these steps.

- According to lemma 4 (section 3.2.2), changing the framework from a system of quadratic multivariate polynomials in a small field to a univariate polynomial in a big field, has a polynomial complexity.
- Recovering the matrix B is polynomial time (in n) as long as r is constant. It has been discussed when the Kipnis-Shamir algorithm is used. It is also true if the coppersmith algorithm is used instead. On the other hand, if $r = O(n)$ (or even $r = O(\log n)$) the time complexity becomes exponential.

- The matrix A is computed by a simple Gaussian reduction of a system of polynomial size in n, its complexity is thus polynomial.

3.3 Why the Kipnis-Shamir Attack does not work

In 2007, a hole has been found in the Kipnis-Shamir attack [51]. It is very surprising that such a long time was needed to discover the problem. It is actually quite frightening considering that **HFE** was for many years among the few "extensively studied" cryptosystems.

In their attack, Kipnis and Shamir assumed the equivalence between the recovery of the private matrix B and the solutions $\mathbf{p}_{\mathbf{B^{-1}},\mathbf{k}}{}_{1\leq k\leq n}$ of the MinRank problem which arise from :

$$\sum_{k=0}^{n-1} \mathbf{p}_{\mathbf{B^{-1}},\mathbf{k}} \underline{S_k^*} = \underline{P_A^*}^t \underline{S'} \underline{P_A^*}$$

They argued that, for any other coefficient than the $\mathbf{p}_{\mathbf{B^{-1}},\mathbf{k}}$, the matrix formed by the linear combination of the $\underline{S_k^*}$ is of rank n (or close to n) with very high probability, thus the $\mathbf{p}_{\mathbf{B^{-1}},\mathbf{k}}$'s form the only solution.
Under this assumption, they showed that the solution can be found by solving an overdefined system of quadratic equations.

This whole method works only if there is a unique (or a very small number of) solution to the **MQ** instance which appears in the attack. Indeed, all these solutions are also solutions of the last linear system produced at the end of the re-linearization method. The attacker has to go through the whole kernel of this system to find the right $\mathbf{p}_{\mathbf{B^{-1}},\mathbf{k}}{}_{1\leq k\leq n}$. Moreover, the more solutions of **MQ** instance, the more parasitic solutions may appear during the re-linearization.

In [51], the authors showed that the instances of the MinRank problem which appear in the attack have many different solutions. Moreover, they seem to create more parasitic solutions during the re-linearization process. As a consequence, the linearization process does not work in practice.

The re-linearization method does not work in the present case. But there exist other methods to solve systems of quadratic equations, these methods may take advantage of the fact that the system is overdefined.

The authors of [51] tried to solve the overdefined quadratic system (from the MinRank problem) with the help of Gröbner basis algorithms. Their conclusion is that it has exponential complexity.

The next theorem states that there are many different solutions of the MinRank instance which appears during the attack.

The idea behind is that, if
$\mathbf{S}'(\mathbf{P_A}(\mathbf{x})) = \bar{x}^t \underline{P_A^{*t}} \underline{S'} \underline{P_A^*} \bar{x}$ and $\underline{P_A^{*t}} \underline{S'} \underline{P_A^*}$ has rank at most r, then the matrix corresponding to $\mathbf{S}'(\mathbf{P_A}(\mathbf{x}))^{q^i}$ should have the same rank (at least, not a bigger one). Indeed, putting the polynomial to the power q^i does not add more terms to the polynomial.

Theorem 3.3.1 *Let the notation $\underline{P_{B^{-1}}}$, S, $\underline{S'}$, P_A, $\underline{S_k^*}$ be defined as before (sec. 3.2.4); Let $(\mathbf{p_0}, \cdots, \mathbf{p_{n-1}})$ be a solution of the MinRank problem (ie. such that $\sum_{i=0}^{n-1} \mathbf{p_i} \underline{S_i^*}$ has rank equal or less than r) with rank r_1. Then, for every $l \in [1, n]$, the rank of the matrix $\sum_{i=0}^{n-1} \mathbf{p_{i-l}}^{q^l} \underline{S_i^*}$ is r_1 (where the $i-l$ is computed modulo n).*

Proof
Raise both sides of the following equation to the power q^l:

$$\mathbf{P_{B^{-1}}}(\mathbf{S}(\mathbf{x})) = \mathbf{S}'(\mathbf{P_A}(\mathbf{x}))$$

- On the left side :
  $(\mathbf{P_{B^{-1}}}(\mathbf{S}(\mathbf{x})))^{q^l} = \sum_{k=0}^{n-1} \mathbf{p_{B^{-1},k}}^{q^l} (\sum_{i,j=0}^{n-1} \mathbf{s_{i,j}} \mathbf{x}^{q^i+q^j})^{q^{k+l}}$
  $= \sum_{k=0}^{n-1} \mathbf{p_{B^{-1},k}}^{q^l} (\sum_{i,j=0}^{n-1} \mathbf{s_{i-l-k,j-l-k}}^{q^{k+l}} \mathbf{x}^{q^i+q^j})$

  The matrix of the quadratic form is:
  $\underline{(P_{B^{-1}} \circ S)^{q^l}} = \sum_{k=0}^{n-1} \mathbf{p_{B^{-1},k}}^{q^l} \underline{S_{k+l}^*} = \sum_{i=0}^{n-1} \mathbf{p_{B^{-1},i-l}}^{q^l} \underline{S_i^*}$.

- On the right side :

  $(\mathbf{S}'(\mathbf{P_A}(\mathbf{x})))^{q^l} = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \mathbf{u_{i,j}}^{q^l} (\sum_{\mathbf{t=0}}^{\mathbf{n-1}} \mathbf{a_t} \mathbf{x}^{\mathbf{q^t}})^{q^{i+l}+q^{j+l}}$

  $(\mathbf{S}'(\mathbf{P_A}(\mathbf{x})))^{q^l} = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \mathbf{u_{i,j}}^{q^l} (\sum_{\mathbf{t=0}}^{\mathbf{n-1}} \mathbf{a_t}^{\mathbf{q^{i+l}}} \mathbf{x}^{\mathbf{q^{t+i+l}}}) (\sum_{\mathbf{v=0}}^{\mathbf{n-1}} \mathbf{a_v}^{\mathbf{q^{j+l}}} \mathbf{x}^{\mathbf{q^{v+j+l}}})$

  $(\mathbf{S}'(\mathbf{P_A}(\mathbf{x})))^{q^l} = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \mathbf{u_{i,j}}^{q^l} (\sum_{\mathbf{t'=0}}^{\mathbf{n-1}} \mathbf{a_{t'-i-l}}^{\mathbf{q^{i+l}}} \mathbf{x}^{\mathbf{q^{t'}}}) (\sum_{\mathbf{v'=0}}^{\mathbf{n-1}} \mathbf{a_{v'-j-l}}^{\mathbf{q^{j+l}}} \mathbf{x}^{\mathbf{q^{v'}}})$

  So the matrix of the quadratic form is :
  $\underline{(S' \circ P_A)^{q^l}} = \underline{P_A^{*'t}} \underline{S''} \underline{P_A^{*'}}$

in which $\underline{P_A^{*'}} = \left(\mathbf{a_{n-l-i+j}}^{q^{l+i}}\right)_{0\leq i,j\leq n-1}$ and $\underline{S''} = \left(\mathbf{u_{i,j}}^{q^l}\right)_{0\leq i,j\leq n-1}$

Recall that theorem 3.2.3 shows that $\underline{S' \circ P_A} = \underline{P_A^*}^t \underline{S'} \underline{P_A^*}$ *where* $\underline{S'} = (\mathbf{u_{i,j}})_{0\leq i,j\leq n-1}$ *and* $\underline{P_A^*} = \left(\mathbf{a_{j-i}}^{\mathbf{q^i}}\right)_{0\leq i,j\leq n-1}$.

We want to argue that the rank of $\underline{(S' \circ P_A)}$ is the same as the rank of $\underline{(S' \circ P_A)^{q^l}}$. We show that the rank of $\underline{P_A^*}$ and $\underline{P_A^{*'}}$ is the same, as well as the rank of $\underline{S'}$ and $\underline{S''}$.

- Suppose that the rank of $\underline{S'}$ and $\underline{S''}$ is not the same. Say $\underline{S'}$ has a smaller rank r_2. There exists $\alpha_0, \cdots, \alpha_n \in \mathbb{F}_{q^n}$ with at most r_2 nonzero values, such that for all i in $[0, n-1]$ $\sum_{j=0}^{n-1} \alpha_j u_{i,j} = 0$. it follows that $\forall l$ $(\sum_{j=0}^{n-1} \alpha_j u_{i,j})^{q^l} = \sum_{j=0}^{n-1} \alpha_j^{q^l} u_{i,j}^{q^l} = 0$.
  This means that there is a linear dependant family of r_2 columns in $\underline{S''}$, in contradiction with the assumption.
  The assumption $rank(\underline{S'}) > rank(\underline{S''})$ is similar.
  The same idea applies if we suppose that $\underline{S'}$ has a bigger rank than $\underline{S''}$.
- $\underline{P_A^{*'}} = \left(\mathbf{a_{n-l-i+j}}^{q^{l+i}}\right)_{0\leq i,j\leq n-1}$ can be seen as the matrix $\underline{P_A^*} = \left(\mathbf{a_{j-i}}^{\mathbf{q^i}}\right)_{0\leq i,j\leq n-1}$ whose columns have been rotated each of $n-l$ positions. Obviously, $\underline{P_A^{*'}}$ and $\underline{P_A^*}$ have the same rank.

As a conclusion, the matrix $\sum_{i=0}^{n-1} \mathbf{p_{B^{-1},i-l}}^{q^l} \underline{S_i^*}$ has rank r_1 and thus $(\mathbf{p_{B^{-1},i-l}})_{1\leq i\leq n-1}$ is also a solution to the MinRank problem. □

Last theorem states that if $(\mathbf{p_0}, \cdots, \mathbf{p_{n-1}})$ is a solution of the MinRank problem, then so is $(\mathbf{p_{n-l}}^{q^l}, \cdots, \mathbf{p_{n-l-1}}^{q^l})$ for $l \in [1, n-1]$. Now, to be rigourous, we prove that these $n-1$ solutions are different, at least with overwhelming probability.

Lemma 5 *Let the matrix* B^{-1} *be a randomly chosen transformation over* $(\mathbb{F}_q)^n$ *and* $(\mathbf{p_0}, \cdots, \mathbf{p_{n-1}})$ *a solution of the MinRank problem corresponding to* B^{-1} *(as in the last theorem). Set* $(\alpha_0^l, \cdots, \alpha_{n-1}^l) = (\mathbf{p_{n-l}}^{q^l}, \cdots, \mathbf{p_{n-l-1}}^{q^l})$, $0 \leq l \leq n-1$. *Then*

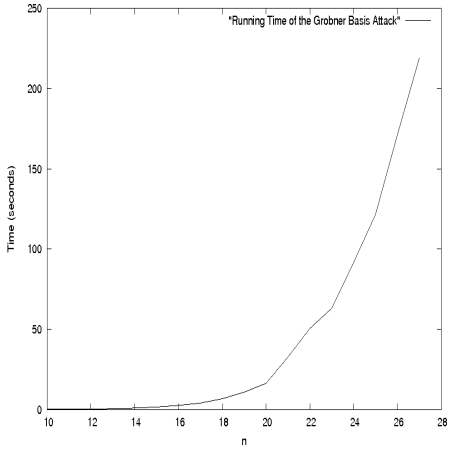
$$Prob(\alpha_i^j = \alpha_i^k : j \neq k, 0 \leq i, j, k \leq n-1) \leq O(n^2 q^{-n})$$

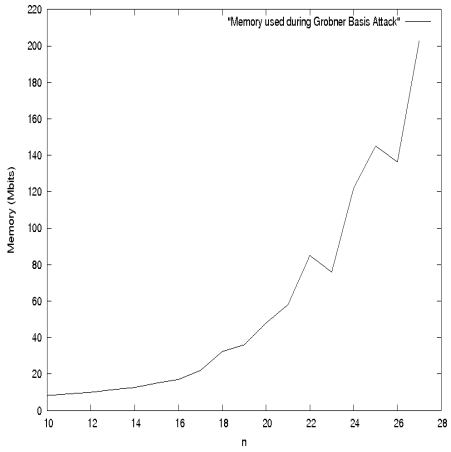
Proof.
Since B^{-1} is randomly chosen (meaning every entry is unformly and independently chosen in $\mathbb{F}_q$), the vector $(\mathbf{p_0}, \cdots, \mathbf{p_{n-1}})$ is also random in $(\mathbb{F}_{q^n})^n$.

Hence: $Prob[\alpha_i^j = \alpha_i^k : j \neq k, 0 \leq i, j, k \leq n-1] \leq 1-(1-nq^{-n})^n$ and $\leq 1-(1-nq^{-n})^n) \leq O(n^2q^{-n})$. $\square$

The last theorem proved that there are least n different solutions to the MinRank problem instance which appears in the Attack of Kipnis Shamir. Jiang and Ding implemented the Kipnis-Shamir attack on **HFE**. Already for small values of r $(r = 2, 3)$ and n $(n = 5, \cdots, 10)$, the attack did not work.

In [51], the authors also tried to solve the overdefined quadratic system, with the help of the Gröbner bases algorithm developped by Faugere. The following figures show the time and memory consumption of this algorithm.





As can be seen on the figure, the running time as well as the memory needed by the attack using Gröbner basis is exponential in the security parameter n.

3.4 Conclusion

In this chapter the **HFE** cryptosystem has been presented as well as an attack from Kipnis and Shamir. The attack does not work as well as claimed, and is certainly not practicable for the parameter values recommended by the author of the cipher.
Thus, strictly speaking, breaking **HFE** is still an open problem [13]. However, another powerful attack against this cryptosystem exists, it relies on Gröbner basis. Faugere et al. were able to break the first challenge with it. They also gave evidences that Gröbner basis algorithms perform better in this case than in general. More precisely, that its complexity may be rather quasipolynomial [14]than exponential.

Our feeling is that those two attacks, even if they don't theoretically break **HFE**, give low confidence in its security. This view is shared by most experts in this field.
On the other hand, attacking **HFE** is still worthwhile. Indeed, as mentioned in the introduction, **QUARTZ**, the very interesting and still unbroken system, is the signature version of **HFE**.

In the next chapter the **2R** cryptosystem will be detailled along with some cryptanalytical work. Particularly, one cryptanalysis which will be explained strongly relies on Kipnis and Shamir attack of **HFE**.

[13] By that we means to find a polynomial time algorithm which break it.
[14] by quasipolynomial they mean subexponential complexity but where the hidden constant are smaller than in the case of factoring algorithms, see [37].

4

The Double Round Quadratic Cryptosystem

One year after **HFE**, Patarin et al. published two new proposals of multivariate cryptographic schemes [27, 26] .

Even though no attacks were known against **HFE** at that time, its decryption process is much slower than in the Imai-Matsumoto cryptosystem. Thus people were still looking for more efficient Imai-like cryptosystems. Patarin's two proposals tried to fulfill these wishes.

In the first part of [27, 26], a slight modification of the Imai cryptosystem, named D^*, is proposed. Unfortunately, the modified system is weak. Patarin then proposed to re-enforce the security by using some ideas taken from private key cryptography.
The first idea is to repeat the public key creation process several times. These new rounds may improve security, as they do in private key cryptography. A second idea is to replace the quadratic function by a Sbox, which is a Boolean function, describe as a matrix, commonly used in private key cryptosystem such as DES or AES.

From these new ideas, two cryptosystems were created :

- The first one is a variant of the Imai scheme, which follows the general frame to construct multivariate schemes, but with two rounds in the creation of the public key. This scheme was named $\mathbf{D}^{**}$ in [26], we name it "Double Round Quadratic Cryptosystem", which is the name used in [36] to refer to $\mathbf{D}^{**}$.
- A second scheme, which didn't really had a name, contains Sbox to replace quadratic functions and is also two rounds. Eli Biham gave a cryptanalysis of this scheme [5].

In this chapter, we will focus on the "Double Round Quadratic Cryptosystem"

The first section is dedicated to D^* and its cryptanalysis. This attack is interesting for two reasons :

- On the one hand, because it motivates the introduction of a second round in the key creation to block this attack.
- On the other hand, because the cryptanalysis of the two round version will use this attack.

The next section describes the "Double Round Quadratic Cryptosystem". The third section contains a cryptanalysis due Ding-Feng et al. The last section is devoted to our cryptanalysis of this same scheme.

4.1 The D* Cryptosystem

In [26], Patarin et al. aimed to create a secure cryptosystem as efficient as the one of Imai-Matsumoto. They proposed $\mathbf{D}^*$ and in the very same article gave a cryptanalysis of it. Although this system was instantaneously broken, it led to the Double-Round Quadratic cryptosystem, which has resisted several years.

4.1.1 D*: How does it works ?

$\mathbf{D}^*$ strictly follows the framework introduced in 1.1. One has only to specify the quadratic function being used in the creation of the public key.

The Function S*

The quadratic function is :

$$\begin{array}{rccl} \mathbf{S}' : & \mathbb{F}_{p^n} & \longrightarrow & \mathbb{F}_{p^n}, \quad p = 3 \mod 4 \\ & \mathbf{x} & \longrightarrow & \mathbf{x}^2 \end{array}$$

Remarks.

- $\mathbf{S}'$ is not a bijection, because the preimage of $\mathbf{x_0}^2$ can either be $\mathbf{x_0}$ or $-\mathbf{x_0}$. Recall that no other roots are possible. Indeed a degree d polynomial, defined over a field, has at most d roots in this field.
- Given a way to know whether the solution we are looking for is $\mathbf{x_0}$ or $-\mathbf{x_0}$, $\mathbf{S}'$ can easily be "inverted", indeed:

$$(\mathbf{x_0}^2)^{\frac{p^n+1}{4}} = \mathbf{x_0}^{\frac{p^n-1}{2}}\mathbf{x_0} = \pm\mathbf{x_0}$$

1. For all $\mathbf{x_0} \in \mathbb{F}_{p^n}$, ${\mathbf{x_0}}^{p^n-1} = 1$, so ${\mathbf{x_0}}^{\frac{p^n-1}{2}} = \pm 1$.
2. As p satisfies: $p = 3 \mod 4$, $\frac{p^n+1}{4} \in \mathbb{Z}$, thus one can compute $({\mathbf{x_0}}^2)^{\frac{p^n+1}{4}}$.

- A specially designed padding scheme can help recognize which of $\mathbf{x_0}$ or $-\mathbf{x_0}$ is the pre-image we are looking for. Thus, the quadratic function can be made bijective, and the cryptosystem can work.

- Many such padding schemes can be found. For instance, the following one will do the job.
  Let $\mathcal{M}$ be the set of authorised plaintexts :

  $$\mathcal{M} = \{\bar{m} = (m_1, \cdots, m_n) \in (\mathbb{F}_p)^n, (\exists k, 1 \leq k \leq n, 1 \leq m_k \leq \frac{p-1}{2}) \text{ and } (\forall i < k, m_i = 0)\}$$

  Obviously either $\mathbf{m} \in \mathcal{M}$ or $-\mathbf{m} \in \mathcal{M}$. Hence, if $\bar{\mathbf{m}} = (0, \cdots, 0, m_k, m_{k+1}, \cdots, m_n)$ and $m_k \leq \frac{p-1}{2}$, then $-\bar{\mathbf{m}} = (0, \cdots, 0, -m_k, -m_{k+1}, \cdots, -m_n)$ und $-m_k = n - m_q \geq \frac{p-1}{2}$.

  If the padding scheme ensures that the plaintexts are in $\mathcal{M}$, there will not be any ambiguity while decrypting the ciphertext. Indeed, inverting $\mathbf{S}'$ gives $\mathbf{x_0}$ or $-\mathbf{x_0}$ (where $\mathbf{x_0}$ is the root which leads to the correct plaintext). The last step in decryption is applying the linear fonction A. Thus we get either $A\bar{x_0} = \bar{m}$ or $A(-\bar{x_0}) = -A\bar{x_0} = -\bar{m}$. It is easy to determine which is the correct one.

  A possibility to implement this padding in practice is to set messages in block of length $n-1$, and to add one block at the beginning with a constant value less than $\frac{p-1}{2}$. For instance $m = (1, m_1, \cdots, m_{n-1}) \in (\mathbb{F}_q)^n$.

4.1.2 D*: A complete description

Keys of the cryptosystem

Public	*Private*
$n, p \in \mathbb{N}$, $p = 3 \mod 4$	Matrices $A, B \in M_n(\mathbb{F}_q)$ nonsingular
$\mathbf{S} \begin{cases} P_1(x_1, \cdots, x_n) \\ \vdots \\ P_n(x_1, \cdots, x_n) \end{cases}$	$\mathbf{S}'(\mathbf{x}) = \mathbf{x}^2$

Fundamental equation of the cryptosystem

$$\mathbf{S}(\mathbf{x}) = \pi A \pi^{-1} \mathbf{S}'(\pi B \pi^{-1} \mathbf{x})$$

Encryption/Decryption:

Encryption

Let $\bar{m} = (m_1, \cdots, m_n) \in (\mathbb{F}_q)^n$ *be the (already padded) plaintext.*

The ciphertext is
$\bar{c} = (c_1, \cdots, c_n) = (P_1(m_1, \cdots, m_n), \cdots, P_n(m_1, \cdots, m_n)) = \mathbf{S}(\bar{m})$.

Decryption

$\bar{m} = (m_1, \cdots, m_n) = A^{-1}\pi^{-1}[\pi(B^{-1}\bar{c})^{\frac{q^n+1}{4}}]$.

Practical Aspects.

Patarin suggested in [26] the parameters $p = 251$, $n = 9$.
It easy to see that the complexity of encryption is $O(n^3(\log p)^2)$. A subtle yet easy analysis can show that the decryption complexity is $O(n^2(\log p + \log n)(\log p)^2)$.

4.1.3 Cryptanalysis of D*

The attack described here is not only very elegant, but paves the way for differential cryptanalysis attack in multivariate cryptography. This type of attack culminated with the devastating cryptanalysis of Dubois against **SFLASH**.

First, to facilitate the presentation, let us introduce some new notations. Let a and b be the following functions:

$$\begin{array}{rccc} a: & (\mathbb{F}_p)^n & \longrightarrow & \mathbb{F}_{p^n} \\ & \bar{x} & \longrightarrow & \pi(A\bar{x}) \end{array}$$

$$\begin{array}{rccc} b: & \mathbb{F}_{p^n} & \longrightarrow & (\mathbb{F}_p)^n \\ & \mathbf{x} & \longrightarrow & B\bar{x} \end{array}$$

Encryption can be described as follows:

$$F(m) = \bar{c} = b(a(\bar{m})^2)$$

The decryption corresponds to :

$$F^{-1}(c) = \bar{m} = a^{-1}(b^{-1}(\bar{c})^{\frac{q^n+1}{4}}) = a^{-1}((\underbrace{b^{-1}(\bar{c}).b^{-1}(\bar{c})\cdots b^{-1}(\bar{c})}_{\frac{q^n+1}{4} times}) \qquad (4.1)$$

Equation 4.1 means that there exists a linear equality between the plaintext and the terms $\prod_{1\leq i_1,\cdots,i_{\frac{q^n+1}{4}}\leq n} c_{i_1}\cdots c_{i_{\frac{q^n+1}{4}}}$. Unfortunately, this linear relation is useless to the attacker because the number of terms that it contains is roughly q^n which is at least superior to 2^{80} to prevent brute force attack. [1]

The first idea of the attack is to get rid of the quadratic part by linearizing the system, computing some sort of differential. The public quadratic functions consist only of squaring, the first idea that comes to mind should be to compute $F(m+m') - F(m-m')$.

Using the linearity of a and b, one gets:
$F(m+m') = b(a(m+m')^2) = b(a(m)^2) + b(a(m')^2) + 2b(a(m).a(m'))$
$F(m-m') = b(a(m)^2) + b(a(m')^2) - 2b(a(m).a(m'))$

$$\text{Thus} \qquad \Phi(m,m') = \frac{F(m+m') - F(m-m')}{4} = b(a(m).a(m'))$$

The function Φ is easily computable, symmetric and bilinear.

Having a bilinear function, we look for classical computable quantities which are connected to it.
For instance the pair (D,E) of linear function in the following set :

$$V = \{(E,D) : (\mathcal{L}((\mathbb{F}_q)^n), \mathcal{L}((\mathbb{F}_q)^n)) \ / \ E(\Phi(m,m')) = \Phi(D(m),m')\}$$

V is a vector space.

Notice that if $D_\lambda(\bar{x}) = a^{-1}(\lambda a(\bar{x}))$ for any λ in $\mathbb{F}_{q^n}$:
$\Phi(D(m),m') = b(\lambda a(m)a(m')) = b(a(a^{-1}(\lambda a(m)))a(m')) = b(\lambda a(m)a(m')) =$

[1] Notice though that computing an element to the power $\frac{q^n+1}{4}$ is easy, for instance using the square and multiply method which runs roughly in $O(n \log q)$.

$E_\lambda(\Phi(m, m^{'}))$ where $E_\lambda(\bar{x}) = b(\lambda.b^{-1}(\bar{x}))$.

For all $\lambda \in \mathbb{F}_{q^n}$ the pairs $(a^{-1}(\lambda a(\bar{x})), b(\lambda.b^{-1}(\bar{x})))$ are in V.
Therefore V has dimension at least n, because all the pairs $(a^{-1}(\beta_i a(\bar{x})), b(\beta_i.b^{-1}(\bar{x})))$ [2] are linearly independent, as a and b are linear of rank n.

Moreover, the constraints on E, D being quite strong, we can hope that element in V of other form are marginal. Thus in practice, V should have dimension exactly (or close to) n. This fact has been confirmed experimentally according to the authors of [44].

Now, suppose that for any λ, we can compute the function D_λ and E_λ. [3] There is a simple yet clever idea using E_λ to efficiently compute equation 4.1.

Indeed, given a ciphertext $\bar{c_0}$, if we could compute:

$$\bar{c_1} = E_{b^{-1}(\bar{c_0})}(\bar{c_0}) = b(b^{-1}(\bar{c_0}).b^{-1}(\bar{c_0}))$$

Equation 4.1 becomes :

$$F^{-1}(c) = \bar{m} = a^{-1}((b^{-1}(\bar{c_1}).\underbrace{b^{-1}(\bar{c_0}) \cdots b^{-1}(\bar{c_0})}_{\frac{q^n+1}{4}-2 \text{ times}})$$

Then we compute c_2 as :

$$\bar{c_2} = E_{b^{-1}(\bar{c_1})}(\bar{c_1}) = b(b^{-1}(\bar{c_1}).b^{-1}(\bar{c_1})) = b(b^{-1}(\bar{c_0}).b^{-1}(\bar{c_0})b^{-1}(\bar{c_0}).b^{-1}(\bar{c_0}))$$

And 4.1 becomes :

$$F^{-1}(c) = \bar{m} = a^{-1}((b^{-1}(\bar{c_2}).\underbrace{b^{-1}(\bar{c_0}) \cdots b^{-1}(\bar{c_0})}_{\frac{q^n+1}{4}-4 \text{ times}})$$

Repeat this process i times :

$$\bar{c_i} = E_{b^{-1}(\bar{c_{i-1}})}(\bar{c_{i-1}}) = b(b^{-1}(\bar{c_{i-1}}).b^{-1}(\bar{c_{i-1}})) = b(\underbrace{b^{-1}(\bar{c_0}) \cdots (b^{-1}(\bar{c_0}))}_{2^i \text{ times}})$$

4.1 becomes :

$$F^{-1}(c) = \bar{m} = a^{-1}((b^{-1}(\bar{c_i}).\underbrace{b^{-1}(\bar{c_0}) \cdots b^{-1}(\bar{c_0})}_{\frac{q^n+1}{4}-2^i \text{ times}})$$

[2] Recall that $(\beta_1, \cdots, \beta_n)$ is a basis of F_{q^n}.
[3] withouth necessarily knowing the actual value of λ

If we can also compute $D_{b^{-1}(\bar{c_i})}(\bar{c_j})$ for $j \leq i$:

$$F^{-1}(c) = \bar{m} = a^{-1}((\underbrace{b^{-1}(\bar{c_i}).b^{-1}(\bar{c_{i'}})}_{b^{-1}(D_{b(c_i)}(c_{i'}))} . \underbrace{b^{-1}(\bar{c_0}) \cdots b^{-1}(\bar{c_0})}_{\frac{q^n+1}{4}-2^i-2^{i'} \text{ times}})$$

Thus:

$$F^{-1}(c) = \bar{m} = a^{-1}(b^{-1}(\bar{c_{i''}}). \underbrace{b^{-1}(\bar{c_0}) \cdots b^{-1}(\bar{c_0})}_{\frac{q^n+1}{4}-2^i-2^{i'} \text{ times}})$$

Applying carefully the iteration, one can end up after i_o iterations with the following:

$$F^{-1}(c) = \bar{m} = a^{-1}((b^{-1}(\bar{c_{i_o}})) \tag{4.2}$$

In equation 4.2, $\bar{c_{i_0}}$ is known, and a^{-1}, b^{-1} are unknown. However, a and b are linear functions, so there is a matrix M such that $\bar{m} = M\bar{c_{i_0}}$. Using multiple plaintext/ciphertext pairs, this matrix can be recovered.

To summerize, we assumed that we were able, for a given $\bar{c}$, to compute $E_{b^{-1}(c)}(\bar{x})$ for every $\bar{x}$ in $(\mathbb{F}_q)^n$. Under this assumption it is possible to mount a polynomial time attack which breaks the scheme.

More precisely, using a polynomial number of plaintexts m_0, the attacker can compute the corresponding ciphertexts c_0 (with the public key). For each different c_0 he can compute the corresponding c_{i_0}.
This polynomial number of couples (m_0, c_{i_0}) enables the attacker to discover the matrix M. Once M is known, the attacker can decrypt any ciphertext by computing the corresponding c_{i_0} and multiplying this vector with M.

The question which remains is : how to compute $E_{b^{-1}(c)}(\bar{x})$ for any given c and $\bar{x}$?

1. First, we know that V is a vector space, and E_λ (resp. D_λ) is a linear function, so there exists a matrix M_{E_λ} (resp. M_{D_λ}) so that $E_\lambda(\bar{x}) = M_{E_\lambda}\bar{x}$ (resp. $D_\lambda(\bar{x}) = M_{D_\lambda}\bar{x}$) . Finding E_λ and D_λ amount to finding these matrix.

   $\Phi(\bar{m}, \bar{m'})$ is known so the equality $M_{E_\lambda}\Phi(\bar{m}, \bar{m'}) = \Phi(M_{D_\lambda}\bar{m}, \bar{m'})$ provides us with $O(n^3)$ linear equations [4] that the $2n^2$ unknowns (the

[4] Each of the n coordinates of this vectorial equation must have each of its $O(n^2)$ terms which match.

matrix coefficients) must satifsfy. We know that the set of solutions is a vector space of dimension (at least) n. Thus, out of the $O(n^3)$ equations, a maximum of $2n^2 - n$ can be linearly independent.

Note that by rearranging the unknowns, the space of solutions can be seen as a kernel of a given matrix, which can be efficiently computed.

In the following we assume that V contains only pair of the form $(a^{-1}(\beta_i a(\bar{x})), b(\beta_i.b^{-1}(\bar{x})))$, and so is of dimension n. Any basis $(E_i, D_i)_{1\leq i\leq n}$ of such vector space has the form $(a^{-1}(\beta_i a(\bar{x})), b(\beta_i.b^{-1}(\bar{x})))_{1\leq i\leq n}$ where $(\beta_1, \cdots, \beta_n)$ is a basis of $\mathbb{F}_{q^n}$.

So far, we can compute a basis $(E_i, D_i)_{1\leq i\leq n}$, but we cannot yet compute $E_\lambda = \sum_{i=1}^n t_i E_i$ for a given λ in $\mathbb{F}_{q^n}$. We don't know from which basis $(\beta_1, \cdots, \beta_n)$ of V the $(E_i, D_i)_{1\leq i\leq n}$ are built. [5], thus we don't know how to choose the t_i's.

2. Now assume we have a basis $(E_i, D_i)_{1\leq i\leq n}$ of V, it means we have n pairs (M_{D_i}, M_{E_i}), let us look for a way to compute the t_i's, for a given λ, so that $E_\lambda = \sum_{i=1}^n t_i E_i$.

   - We want to learn the following function :

     $$\begin{array}{rccl} f: & (\mathbb{F}_p)^n & \longrightarrow & (\mathbb{F}_p)^n \\ & \bar{\lambda} & \longrightarrow & (t_1, \cdots, t_n) \textit{ such that } E_\lambda(\bar{x}) = \sum_{i=1}^n t_i E_i(\bar{x}) \end{array}$$

     Fortunately, the function f is linear. Therefore, finding f amount to find a matrix M_f such that $E_\lambda(\bar{x}) = \sum_{i=1}^n (M_f\bar{\lambda})_i E_i(\bar{x})$ where $(M_f\bar{\lambda})_i$ denotes the i-th component of the vector $(M_f\bar{\lambda})_i$.

   - In order to identify M_f we need to find another equality matching with $E_\lambda(\mathbf{x}) = b(\lambda.b^{-1}(\bar{x}))$ for any x and λ. We do not know if it is possible to find such an equality in the general case. But remember that to perform the attack, we only need to be able to compute $E_{b^{-1}(\bar{x})}$ for any $\bar{x}$.
     The key point is to notice that :
     $E_{b^{-1}(\bar{x})}(\bar{x}') = b(b^{-1}(\bar{x}).b^{-1}(\bar{x}')) = b(b^{-1}(\bar{x}').b^{-1}(\bar{x})) = E_{b^{-1}(\bar{x}')}(\bar{x})$
     Let us re-write these two equalities in term of matrices

     $$M_{E_{b^{-1}(\bar{x})}}\bar{x'} = \sum_{i=1}^n (M_f\overline{b^{-1}(\bar{x})})_i M_{E_i}(\bar{x'}) = M_{E_{b^{-1}(\bar{x}')}}\bar{x} = \sum_{i=1}^n (M_f\overline{b^{-1}(\bar{x}')})_i M_{E_i}(\bar{x})$$

[5] We also don't know how the indices match.

$b(\mathbf{x}) = B\bar{x}$ is bijective, thus $\overline{b^{-1}(\bar{x})} = B^{-1}\bar{x}$.
Replace $\overline{b^{-1}}$ in the last equality :

$$\sum_{i=1}^{n}(M_f B^{-1}\bar{x})_i M_{E_i}(\bar{x'}) = \sum_{i=1}^{n}(M_f B^{-1}\bar{x}')_i M_{E_i}(\bar{x})$$

The last equation holds for all $\bar{x}$ and $\bar{x'}$, and the unknowns are M_f and B_{-1}. The two matrices appear only as a product, it can be seen as a unique unknown matrice $M_{fB^{-1}}$ satisfying:

$$\sum_{i=1}^{n}(M_{fB^{-1}}\bar{x})_i M_{E_i}(\bar{x'}) = \sum_{i=1}^{n}(M_{fB^{-1}}\bar{x}')_i M_{E_i}(\bar{x}) \qquad (4.3)$$

Choosing $\bar{x}$ (resp. $\bar{x'}$) as the r-th (resp. t-th) canonical basis vector (for all r, t in $[1, n]$) provides a linear system of $O(n^3)$ equations [6], in the n^2 unknowns coefficients. The matrix $M_{fB^{-1}}$ can be recovered.

After this last step, the attacker is now able to compute for any $\bar{x}$, $\bar{y}$:

$$E_{B^{-1}\bar{x}}(\bar{y}) = \sum_{i=1}^{n}(M_{fB^{-1}}\bar{x})_i E_i(\bar{y})$$

This enables the attacker to decrypt any ciphertext he wants.

4.2 The Double-Round Quadratic Cryptosystem

In the very same article in which they cryptanalysed D^*, the authors proposed a modification which, hopefully, prevents the previous attack. They first named the new scheme D^{**}, it was later renamed the "Double-Round Quadratic Cryptosystem".

The idea of the modification is that the public system of equations is produced by twice computing the process which leads to the public key in D^*.
The public key is obtained after two "rounds" instead of one.

[6] For each r, t the vector equation holds, ie it provides n equations over $\mathbb{F}_q$.

4.2.1 Description of the Double-Round Quadratic Cipher

Keys of the cryptosystem

Public	*Private*
$n, p \in \mathbb{N}$, $p = 3 \mod 4$ $\mathbf{S}\begin{cases} P_1(x_1, \cdots, x_n) \\ \vdots \\ P_n(x_1, \cdots, x_n) \end{cases}$	Matrices $A, B, C \in M_n(\mathbb{F}_q)$ nonsingular

Encryption/Decryption:

Encryption
Let $\bar{m} = (m_1, \cdots, m_n) \in (\mathbb{F}_q)^n$ *be the (already padded) plaintext.*   The ciphertext is : $\bar{c} = (c_1, \cdots, c_n) = (P_1(m_1, \cdots, m_n), \cdots, P_n(m_1, \cdots, m_n)) = \mathbf{S}(\bar{m})$.

Decryption
$\bar{m} = (m_1, \cdots, m_n) = A^{-1}\pi^{-1}[\pi B^{-1}\pi^{-1}[\pi(C^{-1}\bar{c})]^{\frac{q^n+1}{4}}]^{\frac{q^n+1}{4}}$.

Creation of the public key

$\bar{x} = (x_1, \cdots, x_n)$

$\boxed{\bar{x}' = A.\bar{x}}$ — **linear** with matrix A (Private)

$\bar{x}' = (\sum_{j=1}^{n} a_{1j}x_j, \cdots, \sum_{j=1}^{n} a_{nj}x_j)$

$\Downarrow$

$\boxed{\bar{x}'' = \mathbf{x}'^{\mathbf{2}}}$ — **nonlinear** inversible (Public)
quadratic

$\bar{x}'' = (\sum_{i,j=1}^{n} r_{1ij}x_ix_j, \cdots, \sum_{i,j=1}^{n} r_{nij}x_ix_j)$

$\Downarrow$

$\boxed{\bar{y} = B.\bar{x}''}$ — **linear** with matrix B (Private)

$\bar{y} = (\sum_{i,j=1}^{n} s_{1ij}x_ix_j, \cdots, \sum_{i,j=1}^{n} s_{nij}x_ix_j)$

$\Downarrow$

$\boxed{\bar{y}' = \mathbf{y}^{\mathbf{2}}}$ — **nonlinear** inversible (Public)
quadratic

$\bar{y}' = (\sum_{i,j=1}^{n} u_{1ij}x_ix_j, \cdots, \sum_{i,j=1}^{n} u_{nij}x_ix_j)$

$\Downarrow$

$\boxed{\bar{y}'' = C.\bar{y}'}$ — **linear** with matrix C (Private)

$\bar{y}'' = (\sum_{i,j=1}^{n} v_{1ij}x_ix_j, \cdots, \sum_{i,j=1}^{n} v_{nij}x_ix_j)$

$\Downarrow$

$$\begin{aligned} y_1'' &= P_1(x_1, \cdots, x_n) \\ &\vdots \\ y_n'' &= P_n(x_1, \cdots, x_n) \end{aligned}$$ **Public**

Abbildung 4.1: Construction of the $2R$ public key

Practical Aspects.
Because two rounds are applied, the public key consists of a system of polynomials of degree 4, which makes it bigger than in the other multivariate schemes. However a small computation with the recommended parameter ($q = 251$,$n = 9$) shows that the public key is around 6 Kbits whereas it is around 1 to 2 Kbits for RSA.

On the other hand, the authors claim that a careful study shows that the secret key computation amounts to less than 50 multiplications of numbers of lengths less than 72 bits. For a standard RSA key it requires roughly 1000 multiplications with numbers of 1000 bits .

Hence, even if the Double-Round Quadratic cipher has a bigger key size than the other multi-variate cryptosystems it stays very competetive in front of RSA which remains the most widely used cryptosystem.

4.3 An attack of the Double-Round Quadratic cipher from Crypto'99

At the Crypto'99 conference, Y. Ding-Feng, L. Kwok-Yan and D. Zong-Duo presented an attack [17] which allows, under various heuristics, to recover the private parameters.

Their aim is to find the composition of the two quadratic polynomials which leads to the public system of polynomials of degree 4.

If we write $g(x_1, \cdots, x_n)$ for the quadratic mapping (from $((\mathbb{F}_q)^n$ to $(\mathbb{F}_q)^n)$ which corresponds to $\bar{y}$ in figure 4.1.
$g(x_1, \cdots, x_n) = (g_1(x_1, \cdots, x_n), \cdots, g_n(x_1, \cdots, x_n))$.
We have $g(x_1, \cdots, x_n) = (\sum_{i,j=1}^{n} s_{1ij} x_i x_j, \cdots, \sum_{j=1}^{n} s_{nj} x_i x_j)$.

Let f be the quadratic mapping (from $((\mathbb{F}_q)^n$ to $(\mathbb{F}_q)^n)$ which corresponds to a square transformation in $\mathbb{F}_{q^n}$ followed by a linear transformation with matrix C. Formally $f(\bar{y}_1, \cdots, \bar{y}_n) = C\pi^{-1}(\mathbf{y}^2)$.

The public key corresponds to the following composition :

$$f \circ g = (f_1(g_1, \cdots, g_n), \cdots, f_n(g_1, \cdots, g_n))$$

- First, notice that if f and g are known to the attacker, he can break the system, applying the one round attack which was discussed in 4.1.3 to both f and g. Using this attack, he can compute $f^{-1}(\bar{c})$ and $g^{-1}(\bar{c})$ for all $\bar{c}$. Then, given a ciphertext, he first decrypts it with f^{-1}, the

decrypted text corresponds to the ciphertext output by g, it is also decrypted, leading to the plaintext of $f \circ g$.

- Now, suppose that the attacker only knows the quadratic system of equations $g = (g_1, \cdots, g_n)$, it is easy to deduce from it the quadratic system $f = (\sum_{i,j=1}^{n} f_{1ij} x_i x_j, \cdots, \sum_{i,j=1}^{n} f_{nij} x_i x_j)$. Indeed, he computes $f \circ g = (\sum_{i,j=1}^{n} f_{1ij} g_i g_j, \cdots, \sum_{i,j=1}^{n} f_{nij} g_i g_j)$ and compares it with the public key $h = (\sum_{i,j,k,l=1}^{n} h_{1ijkl} x_i x_j x_k x_l, \cdots, \sum_{i,j,k,l=1}^{n} h_{nijkl} x_i x_j x_k x_l)$. He finds a system of $O(n^5)$ linear equations that the $O(n^3)$ values f_{kij} must satisfy. Solving this system allows to recover f.

- Notice that for this to work, it is not necessary to "exactly" know $g(x_1, \cdots, x_n) = (g_1(x_1, \cdots, x_n), \cdots, g_n(x_1, \cdots, x_n)) = C\pi^{-1}[\pi(A\bar{x})^2]$. A linear combination of the g_i would be sufficient.
  ie a vector $\begin{pmatrix} g_1' \\ \vdots \\ g_n' \end{pmatrix} = M \begin{pmatrix} g_1 \\ \vdots \\ g_n \end{pmatrix}$
  Indeed, such a linear combination translates to a matrix multiplication $g'(x_1, \cdots, x_n) = MC\pi^{-1}[\pi(A\bar{x})^2]$ for a matrix M. If the attacker gets this modified g, he can still find a f' by identification, and in this case f' will verify: $f(\bar{y}_1, \cdots, \bar{y}_n) = C\pi^{-1}[\pi(M^{-1}\bar{y})^2]$.
  The attack on each round still works, with any such f', g' instead of f, and g.

Therefore, the goal of the attack is to find such a linear combination of the g_i's, or equivalently, to find an element in the linear class of g, defined as follows :

$$\{m \circ g | \text{ for all linear bijection } m\}$$

Remarks.
A first assumption has been made.
The decomposition of the degree 4 polynomial, in two quadratic transformations in the linear class of g (and of f [7]) has to be unique. Otherwise the hidden quadratic transformation in f and g may not be the squaring function.
In this case, the attack on the one round version seen in the previous section would not work any more.

4.3.1 First step

The main idea of the attack is that it may be possible to gain knowledge of the linear class of g, while looking at the partial derivatives of the public

[7] The linear class of f is $\{f \circ m | \text{ for all linear bijection } m\}$.

system of polynomials.

If $g_i(x_1, \cdots, x_n) = \sum_{s,t=1}^{n} g_{s,t,i} x_s x_t$ and $f_i(x_1, \cdots, x_n) = \sum_{u,v=1}^{n} f_{u,v,i} x_u x_v$, the partial derivatives of $f \circ g$ give linear combinations of the $x_j g_i(x_1, \cdots, x_n)$'s. To see this clearly, we take an example in which f_i has only one term. Let $f_{i_0}(x_1, \cdots, x_n) = f_{u_0,v_0,i} x_{u_0} x_{v_0}$.

$$f_{i_0}(g_1, \cdots, g_n) = f_{u_0,v_0,i} g_{u_0}(x_1, \cdots, x_n) g_{v_0}(x_1, \cdots, x_n)$$

$$\begin{aligned}\frac{\partial f_{i_0}(g_1, \cdots, g_n)}{\partial x_1} =& f_{u_0,v_0,i} \frac{\partial g_{u_0}(x_1, \cdots, x_n)}{\partial x_1} g_{v_0}(x_1, \cdots, x_n) \\ &+ f_{u_0,v_0,i} \frac{\partial g_{v_0}(x_1, \cdots, x_n)}{\partial x_1} g_{u_0}(x_1, \cdots, x_n)\end{aligned}$$

$$\begin{aligned}\frac{\partial f_{i_0}(g_1, \cdots, g_n)}{\partial x_1} =& f_{u_0,v_0,i} (\sum_{s=1}^{n} g_{s,1,u_0} x_s) g_{v_0}(x_1, \cdots, x_n) \\ &+ f_{u_0,v_0,i} (\sum_{s=1}^{n} g_{s,1,u_0} x_s) g_{u_0}(x_1, \cdots, x_n)\end{aligned}$$

$$\frac{\partial f_{i_0}(g_1, \cdots, g_n)}{\partial x_1} = \sum_{s=1}^{n} x_s (\textit{ linear combination of } g_{u_0} \textit{ and } g_{v_0})$$

This can be generalised for every i, k and f :

$$\frac{\partial f_i(g_1, \cdots, g_n)}{\partial x_k} = \sum_{s=1}^{n} x_s (\textit{ linear combination of } g_1, \cdots, g_n) \qquad (4.4)$$

Recall that we are looking for any element in $\{m \circ g |$ for all linear bijections $m\}$. Which, in terms of vector space, means to have any basis of the $\mathbb{F}_q$-linear space :

$$W = span(g_1, \cdots, g_n)$$

The partial derivatives do not lead to linear combinations of the g_i's. In equation 4.4, the g_i's are present, but hidden behind the x_i's. Someone who computes the derivative, sees a cubic equation in which he cannot distinguish the x_i's coming from the g_i's and the other.

On the other hand it is reasonable to think that sufficiently many partial derivatives will eventually lead to a basis of the following vector space:

$$V = \sum_{i=1}^{n} x_i W \tag{4.5}$$

V has dimension at most n^2 (because W has dimension at most n) and there are n^2 partial derivatives, this means n^2 elements of V.

If we suppose that each of the n^2 partial derivatives are random elements in V, the probability that they form a basis of V is $\prod_{i=1}^{n-1}(1-\frac{1}{q^i})$ [8], which is roughly $(1-\frac{1}{q})$ when q is big.

If $dim(V) < n^2$ a basis is found in the following way : write each partial derivative as a vector with $\binom{n}{3}$ coordinates which correspond to the coefficients of all the possible terms $x_i x_j x_k$. Then, run the Gaussian algorithm on the matrix to find a linear independent family.

Having found a basis for V, the next question which arises to fulfill the attack is:
is it possible to get a basis of W out of a basis of V ?

4.3.2 Last step

Notice that if a quadratic polynomial r is in W, then for every linear space $\mathcal{L}$ generated by $x_1, \cdots, x_n$ the polynomial $r\mathcal{L}$ will be in V. It may be possible that the converse also holds.

Formally, for any subspace $\mathcal{L}'$ of the linear space $\mathcal{L}$ generated by $x_1, \cdots, x_n$, let

$$\big(V : \mathcal{L}'\big) = \{r \in \mathbb{F}_q[x_1, \cdots, x_n] : r\mathcal{L}' \subseteq V\} \tag{4.6}$$

- First, notice that $(V : \mathcal{L}')$ is easy to compute for any $\mathcal{L}'$.
  Indeed, every cubic polynomial can be represented via a vector in $(\mathbb{F}_q)^{\frac{n(n-1)(n-2)}{6}}$ [9]. Let $r_{i,j}$ be the coefficients of the quadratic polynomial r, then $r\mathcal{L}'$ is represented via a vector of dimension $\frac{n(n-1)(n-2)}{6}$ which contains linear combinations of the $r_{i,j}$.
  $r\mathcal{L}' \subseteq V$ means that the vector $r\mathcal{L}'$ is equal to a linear combination of the basis vectors of V, which contains at most n^2 elements.
  Considering the coefficients of the linear combination as unknowns, this system has $\binom{n}{3}$ linear equations and $\binom{n}{2}+n^2$ unknowns. Therefore it is possible to find a basis of $(V : \mathcal{L}')$ with linear algebra.

[8] because having already a familiy of i linear independent elements, the probability that the $i+1$th is linear independent of the first i element is $\frac{q^n-q^i}{q^n}$.

[9] where each coordinate corresponds to the coefficient of a cubic term.

- It is obvious that $W \subseteq (V : \mathcal{L}')$. Furthermore, the authors of the attack conjectured that $(V : \mathcal{L}') \subseteq W$. Under this hypothesis, for any $\mathcal{L}'$:

$$(V : \mathcal{L}') = W$$

Thus, the computed basis of $(V : \mathcal{L}')$ is also a basis of $W = vect(g_1, \cdots, g_n)$ which is an element of the linear class of g we are looking for. As explained in the previous paragraphs, once a basis for $(g_1, \cdots, g_n)$ has been found, it is possible to eventually break the system. This concludes the attack.

About the conjecture

The whole attack is based on the supposed equality of a given vector space, with another, easy computable, space. The authors could not prove this equality but tried to give a "mathematical explanation" why the conjecture should work.
This argumentation contains mistakes which are explained in the following. The conjecture might or might not hold. According to us, none of the given theoretical explanations leads to rather one or the other possibility.

Conjecture. *Notations and assumptions as above, then for a randomly chosen W, the probability ρ that $(V : \mathcal{L}) = W$ is very close to 1 when $n > 2$.*

The argumentation is based on some lemma from linear algebra. In their articles the authors stated the lemma without any proof or reference. We detail them, because, according to us, the second lemma is false.

- **Lemma 6** *The number of subspaces of dimension k in a space of dimension $n > k$ is :*

$$\mu(k, n) = \prod_{i=0}^{k} (q^n - q^i) / \prod_{i=0}^{k} (q^k - q^i) \approx q^{(n-k)k}$$

Proof. Let us first count the number of possible choices to construct a k-dimensional subspace basis. For the first vector the choice is among $q^n - 1$ vectors. For the second there are :
$q^n - 1 -$"All the linear combinations of the first vector" $= q^n - 1 - (q - 1)$.
More generally, the choice for the ith vector is among :
$q^n - 1 -$ "All the linear combinations of the first $i - 1$ th vectors" $= q^n - 1 - (q^{i-1} - 1)$. As a conclusion there are $\prod_{i=0}^{r-1}(q^n - q^i)$ possibilities to choose the basis of a r-dimensional vector space.

However, different basis can lead to the same vector space, we count them. Let us write $b_1, \cdots, b_k$ the r vectors that have been picked in the selection process. As the first vector of our equivalent space we can choose any linear combination of the $b_1, \cdots, b_k$, there are thus $q^k - 1$ possibilities. the second vector can be any vector from $span(b_1, \cdots, b_k)$ which is not a linear combination of the first chosen vector. There are $q^k - q$ possibilities. Obviously the ith chosen vector can be any vector of the $q^k - 1$ which is not a linear combination of the $i-1$th already chosen (there are $q^{i-1}-1$ such vectors). There are $\prod_{i=0}^{k-1}(q^k - q^i)$ different ways to build an equivalent subspace. Thus they are $\prod_{i=0}^{k}(q^n - q^i)/\prod_{i=0}^{k}(q^k - q^i)$ different subspaces of dimension k in a space of dimension n.

Lemma 7 *The probability that the intersection of two random subspaces S_1 and S_2 of dimension n_1, n_2 in a space of dimension n has dimension $n_1 + n_2 - n + \delta$ for a $\delta \geq 0$ is*

$$\mu(n - n_2 - \delta, 2n - n_1 - n_2 - \delta)\mu(n - n_1 - \delta, 2n - n_1 - n_2 - \delta)$$
$$\mu(n_1 + n_2 - n + \delta, n)\mu(n_1, n)\mu(n_2, n) \approx q^{-\delta(n_1+n_2-n+\delta)} \quad (4.7)$$

Thus with high probability the intersection has dimension $n_1 + n_2 - n$.

Equation 4.7 is incorrect, we will explain why.
First, for clarity reasons, write $u = n_1 + n_2 - n + \delta$. The equation 4.7 becomes:

$$\mu(n_1 - u, n - u)\mu(n_2 - u, n - u)$$
$$\mu(u, n)\mu(n_1, n)\mu(n_2, n) \quad (4.8)$$

It corresponds to the probability that a random subspace S_1 and a random subspace S_2 have an intersection of dimension u, according to the authors of the attack.

Let us try to understand where this equation comes from.
Let U be a fixed subspace of dimension u, the probability that a random space of dimension n_1 contains U is :

$$\frac{\text{nb of space of dimension } n_1 \text{containing } U}{\text{nb of space of dimension } n_1} =$$
$$\frac{\text{nb of way to complete } U \text{ in a } n_1 \text{dimension space}}{\text{nb of space of dimension } n_1} = \frac{\mu(n_1 - u, n - u)}{\mu(n_1, n)} \quad (4.9)$$

The authors put the equations together in the following fashion:

The probability that the intersection is of dimension u is

$$\begin{aligned} &= \sum_{\text{all space of dim } u} Pr(\text{ the random space } S_1 \text{ contains U}) \\ &\qquad \times Pr(\text{ the random space } S_2 \text{ contains U}) \\ &= \mu(u,n)\mu(n_1-u,n-u)\mu(n_2-u,n-u)\mu(n_1,n)\mu(n_2,n) \end{aligned} \tag{4.10}$$

What is computed is the probability that the intersection has dimension u or higher.

To get the probability that the intersection has dimension exactly u, we have to take care that for the given subspaces U and S_1 (where U is contained in S_1) the random subspace S_2 of dimension n_2 does not intersect S_1 except for U. The probability for this event is $\frac{\mu(n_2-u,n-n_1)}{\mu(n_2,n)}$.

Thus the probability that the intersection of S_1 and S_2 has dimension u is:

$$\mu(u,n)\mu(n_1-u,n-u)\mu(n_2-u,n-n_1)\mu(n_1,n)\mu(n_2,n)$$

In the last equation it is difficult to give a nice estimate of the dimension which has the highest probability. Neither to say if it is actually close to n_1+n_2-n the estimation given by the authors. On the other hand if n_1, n_2 are very close to n, the intersection will also have a dimension close to n, so the estimate of the authors may work in this precise case.

Lemmas 6 and 7 help the authors to argument that $(V:\mathcal{L}) = W$. More specifically, they want to argue that the dimension of the vector space $(V:\mathcal{L})/W$ is very small.

Their argumentation is done in 3 steps:

- First notice that L is the vector space generated by the $X_1, \cdots, X_n$ so $(V:\mathcal{L}) = \cap_i (V:X_i)$ and thus $(V:\mathcal{L})/W = \cap_i((V:X_i)/W)$.
- Think of the $(V:X_i)/W$ as random independent variables. The dimension of their intersection relies on the dimension of each $(V:X_i)/W$ in an independent manner.

- To estimate the dimension of $(V : X_i)/W$, they notice that each element of this space has the form $\frac{\sum_{j \neq i} X_j w_j}{X_i} + w_i$.
  Let $(g_1, \cdots, g_n)$ be a basis of W, we can re-write the element in the following way $\frac{\sum_j g_j F_j}{X_i}$ where F_j are linear form in $X_1, \cdots, X_n$.

  Moreover $\sum g_i(X_1, \cdots, X_i, 0, X_{i+1}, X_n)F_i = 0$ [10], so the F_i are in the kernel of the following linear map :

  $$\sigma : (F_1, \cdots, F_n) \longrightarrow \sum g_i(X_1, \cdots, X_i, 0, X_{i+1}, \cdots, X_n)F_i$$

  It follows that $dim((V : X_i)/W) \leq dim(ker(\sigma))$. Now if we look at σ as a random linear mapping (between space of dimensions $n(n-1)$ and $(n-1)n(n+1)/6)$ this kernel should be of very small dimension [11].

  As a conclusion $dim((V : X_i)/W)$ should also be very small, concluding the argumentation.

The biggest problem in this argumentation is that not everything can be random [12] at the same time, as the authors claim. If the vector space $(V : X_i)/W$ is randomly chosen in the bigger space of dimension $\frac{n(n+1)}{2} - n$, its dimension tends to be $\frac{n(n+1)}{2} - n$. Now, this random space (as any other random space) can be seen as the kernel of some linear application, but saying that it is produced as the kernel of some random linear function, leads to the fact that this space has a very small dimension [13]. In other words, generating a random vector space via random basis elements, doesn't give the same kind of space as generating it via the kernel of some random function.

As a matter of fact, we could use the same argumentation as the authors, but stop after the second steps. We could say that $(V : X_i)/W$ should behave like a random space which means it has dimension close to the one of the ambient space, so $\cap_i((V : X_i)/W) = (V : \mathcal{L})/W$ has also a rather big dimension which shows that $(V : \mathcal{L}) \neq W$.

4.3.3 Conclusion

This attack contains very interesting ideas. It is the first time that some kind of differential equations are used in multi-variate cryptography. Later,

[10] Recall that each element el of $(V : X_i)/W$ satisfie $X_i el = \sum_j X_j w_j = \sum_j g_j F_j(X_1, \cdots, X_n)$

[11] Because the dimension of the image space is bigger than the dimension of the pre-image space

[12] Here random is not clearly defined, it is to be understood as some kind of uniform distribution.

[13] It is well known that "random" linear function tend to have small kernel.

Dubois and Stern pushed these ideas further to break $SFLASH$.

On the other hand, their method has no proof. They claimed that experiments confirmed their conjecture, but it is very hard to understand why the conjecture actually works, and what the precise heuristics are.

In the next section we show another heuristic method to break the **2R** cryptosystem. The advantage of this new method is that it is more general, and the heuristic is easier to understand and to agree with.

4.4 Another Cryptanalysis of the 2-Round Quadratic Cipher

In this Chapter we present our contribution [47] to the field.

Our attack does not rely on new or unconventional "tricks". Rather, we show that some of the ideas developped for the attack of **HFE** (see chapter 3) apply to the $2R$ scheme. The attack we are able to mount is heuristic, like all other attacks in the field. However, contrary to the attack of Feng-Yan-Duo, the heuristic we use is very general and "well defined", it inspires more confidence.
More than that, we feel that our contribution is to show that the way multivariate scheme are presented so far, may not be the best, in that it makes things look harder than they really are.

We use another, more appropriate, formalism to present the $2R$ scheme. In this new formalism attacking the scheme is an easy task. More importantly it will be clear that this formalism allows to break most of the cryptosystem in the field, and particularly, all the schemes introduced in the thesis so far.

4.4.1 The Kipnis-Shamir Formalism

Here we mention the theorems and lemmas that we will use, which were proved in section 3.2.2.

Theorem 4.4.1 (Kipnis, Shamir 99) *Let M be a linear mapping from n-tuples to n-tuples of values in $\mathbb{F}_q$. Then there are coefficients $a_1, \cdots, a_n$ in $\mathbb{F}_{q^n}$ such that for any two n-tuples over $\mathbb{F}_q$, $(x_1, \cdots, x_n)$ (which represents $\mathbf{x} = \sum_{i=1}^{n} x_i \beta_i$ in $\mathbb{F}_{q^n}$) and $(y_1, \cdots, y_n)$ (which represents $\mathbf{y} = \sum_{i=1}^{n} y_i \beta_i$ in $\mathbb{F}_{q^n}$), $(y_1, \cdots, y_n) = M(x_1, \cdots, x_n)$ if and only if $\mathbf{y} = \sum_{i=1}^{n} a_i \mathbf{x}^{q^i}$.*

Theorem 4.4.2 (Kipnis, Shamir 99) *Let $P_1(x_1, \cdots, x_n), \cdots, P_n(x_1, \cdots, x_n)$ be any set of n multivariate polynomials in n variables over $\mathbb{F}_q$. Then, there are coefficients $a_1, \cdots, a_{q^n}$ in $\mathbb{F}_{q^n}$ such that for any two n tuples $(x_1, \cdots, x_n)$*

and $(y_1, \cdots, y_n)$ *in* $(\mathbb{F}_q)^n$, $y_j = P_j(x_1, \cdots, x_n)$ *for all* $1 \leq j \leq n$ *if and only if* $\mathbf{y} = \sum_{i=1}^{q^n} a_i \mathbf{x}^i$ *(where* $\mathbf{x} = \sum_{i=1}^{n} x_i \beta_i$ *and* $\mathbf{y} = \sum_{i=1}^{n} y_i \beta_i$ *are the elements of* $\mathbb{F}_{q^n}$ *which correspond to the two vectors over* $\mathbb{F}_q$*).*

Lemma 8 *Let* C *be any collection of* n *homogeneous multivariate polynomials of degree* d *in* n *variables over* $\mathbb{F}_q$*. Then, the only powers of* x *which can occur with non-zero coefficients in its univariate polynomial representation* $G(x)$ *over* $\mathbb{F}_{q^n}$ *are sums of exactly* d *(not necessarily distinct) powers of* q *:* $q^{i_1} + q^{i_2} + \cdots + q^{i_n}$*. If* d *is a constant, then* $G(x)$ *is sparse, and its coefficients can be found in polynomial time.*

4.4.2 Relinearization Technique

Recall that in section 3.2.5, we discussed a method to solve a system of ϵm^2 quadratic equations and m variables when ϵ is at least superior to 0.1.

4.4.3 Cryptanalysis

In this section we prove the following theorem :

Theorem 4.4.3 *Under the heuristic of the relinearization technique (from 3.2.5) it is possible to recover the private key (A, B and C) of the 2R cryptosystem in polynomial time.*

Recovering the C matrix

Theorems 5 and 6 enable us to present the Double-Round Quadratic cryptosystem in a unified framework.
The private values (the matrix A, B and C) are now the polynomials P_A, P_B, P_C in $\mathbb{F}_{q^n}[X]$.
$P_A = \sum_{i=0}^{n-1} a_i \mathbf{x}^{q^i}$, $P_B = \sum_{i=0}^{n-1} b_i \mathbf{x}^{q^i}$, $P_C = \sum_{i=0}^{n-1} c_i \mathbf{x}^{q^i}$.

The public system of equations is also represented by a polynomial $P_{public}(\mathbf{x})$ in $\mathbb{F}_{q^n}[X]$ and satisfies :

$$P_C(P_B(P_A(\mathbf{x})^2)^2) = P_{public}(\mathbf{x}) \tag{4.11}$$

Moreover, we see from the shape of P_A, P_B, P_c that P_{public} has the form.

$$P_{public} = \sum_{0 \leq i_1 \leq i_2 \leq i_3 \leq i_4 \leq n-1} p_{i_1,i_2,i_3,i_4} \mathbf{x}^{q^{i_1}+q^{i_2}+q^{i_3}+q^{i_4}}$$

The shape of the public polynomial in $\mathbb{F}_{q^n}[X]$ can also be viewed as a direct consequence of lemma 8.
The coefficients p_{i_1,i_2,i_3,i_4} can be found by the constructive method presented

in the proof of this theorem.

As P_C is the polynomial corresponding to the linear transformation with matrix C, it follows that $(P_C)^{-1}$ corresponds to the linear transformation with matrix C^{-1}. From the first theorem we know that $(P_C)^{-1}$ is of the form:

$$(P_C)^{-1} = \sum_{t=0}^{n-1} c_t^{'} \mathbf{x}^{q^t}$$

We can write the equation (4.11) in the following way:

$$P_B(P_A(\mathbf{x})^2)^2 = (P_C)^{-1}(P_{public}(\mathbf{x})) = \sum_{\mathbf{t=0}}^{\mathbf{n-1}} \sum_{\mathbf{0 \le i_1 \le i_2 \le i_3 \le i_4 \le n-1}} \mathbf{c_t^{'} p_{i_1,i_2,i_3,i_4}^{q^t} x^{q^{i_1+t}+q^{i_2+t}+q^{i_3+t}+q^{i_4+t}}} \tag{4.12}$$

Moreover $P_A(\mathbf{x}) = \sum_{i=0}^{n-1} a_i \mathbf{x}^{q^i} \Rightarrow (P_A(\mathbf{x}))^2 = \sum_{0 \le i \le j \le n}^{n-1} a_{i,j}^{'} \mathbf{x}^{q^i+q^j}$
so $P_B(P_A(\mathbf{x})^2) = \sum_{k=0}^{n-1} b_k [\sum_{0 \le i \le j \le n}^{n-1} a_{i,j}^{'} \mathbf{x}^{q^i+q^j}]^{q^k}$

It follows that there exists $b_{i,j}$ in $\mathbb{F}_{q^n}$ so that $P_B(P_A(\mathbf{x})^2)$ satisfies:

$$P_B(P_A(\mathbf{x})^2) = \sum_{0 \le i_1 \le i_2 \le n-1} b_{i_1,i_2} \mathbf{x}^{q^{i_1}+q^{i_2}} \tag{4.13}$$

Which means that the following equation must hold:

$$(\sum_{0 \le i_1 \le i_2 \le n-1} b_{i_1 i_2} \mathbf{x}^{q^{i_1}+q^{i_2}})^2 = \sum_{t=0}^{n-1} \sum_{0 \le i_1,i_2,i_3,i_4 \le n-1} c_t^{'} p_{i_1,i_2,i_3,i_4}^{q^t} \mathbf{x}^{q^{i_1+t}+q^{i_2+t}+q^{i_3+t}+q^{i_4+t}} \tag{4.14}$$

We get an equality between two polynomials, which leads to as many equations as the number of different terms in the polynomials. All the terms of the form $\mathbf{x}^{q^{i_1}+q^{i_2}+q^{i_3}+q^{i_4}}$ with $0 \le i_1, i_2, i_3, i_4 \le n-1$ are present.
And clearly if (i_1, i_2, i_3, i_4) (with $i_1 \le i_2 \le i_3 \le i_4$), and $(i_1^{'}, i_2^{'}, i_3^{'}, i_4^{'})$ (with $i_1^{'} \le i_2^{'} \le i_3^{'} \le i_4^{'}$) are different, then $q^{i_1}+q^{i_2}+q^{i_3}+q^{i_4} \neq q^{i_1^{'}}+q^{i_2^{'}}+q^{i_3^{'}}+q^{i_4^{'}}$. So the number of different terms is $\binom{n+3}{4} = \frac{1}{24}n^4 + o(n^4)$, and the number of unknowns (the b_{ij} and $c_k^{'}$) is $\frac{n(n+1)}{2} + n$.

We obtain another system of quadratic equations, but instead of having n equations in n unknowns (over $\mathbb{F}_q$) there are $\approx \frac{1}{24}n^4$ equations, in $\approx \frac{n^2}{2}$ variables (over $\mathbb{F}_{q^n}$). Using the relinearization technique, we solve this system, because $\epsilon \approx \frac{4}{24} = \frac{1}{6} > 0.1$. We recover the matrix $C^{'}$ via the coefficients $c_i^{'}$, and we can compute $C = (C^{'})^{-1}$.

Remarks.

- It makes sense to look at the asymptotical values for n, because it is the security parameter. n is the parameter to be increased if one wants to keep the overall security as regards to the increase of computational power to perform attacks.
  Hence the system is theoretically broken, if it is broken for $n \to \infty$.
  In practice, the proposed values were $q = 251$ and $n = 9$, an exact computation leads to $\epsilon = 0.17$ which means that the system is also practically solvable for any values of n (as ϵ increases with n).

- Notice also that contrary to the attack of Kipnis and Shamir, here, there is a unique solution to the system of quadratic equations. The problem which appeared in the cryptanalysis of **HFE**, and detailed in 3.3, does not occur in our case.

Recovering the B and A matrices

The coefficients $b_{i,j}$ which were found in the previous paragraph lead us to the polynomials $Q(\mathbf{x})$ with:
$Q(\mathbf{x}) = P_B(P_A(\mathbf{x})^2) = \sum_{0 \leq i_1 \leq i_2 \leq n-1} b_{i_1 i_2} \mathbf{x}^{q^{i_1}+q^{i_2}}$.

Now we can use exactly the same method as above to find the matrices B and A.

We know that $P_B(\mathbf{x}) = \sum_{i=0}^{n-1} b_i \mathbf{x}^{q^i}$, Hence $P_B^{-1}(\mathbf{x}) = \sum_{i=0}^{n-1} b'_i \mathbf{x}^{q^i}$.
$P_A(\mathbf{x})^2 = P_B^{-1}(Q(\mathbf{x}))$.
Hence, we get (remember $P_A(\mathbf{x}) = \sum_{i=0}^{n-1} a_i \mathbf{x}^{q^i}$):

$$(\sum_{i=0}^{n-1} a_i \mathbf{x}^{q^i})^2 = \sum_{i=0}^{n-1} b_i (Q(\mathbf{x}))^{q^i}$$

We have a quadratic system of $2n$ variables and $\binom{n+2}{2} = \frac{n^2}{2} + o(n^2)$ equations over $\mathbb{F}_{q^n}$. In this case $\epsilon = \frac{1}{8} > 0.1$, we find the variables (so also the matrices A and B) with the relinearization technique.
As for the recovering of C, the attack works also in practice, as for $n = 9$ we find $\epsilon = 0.17$.

The Affine Case

It is common in Multivariate Cryptography that the private transformations are chosen to be affine (and not linear), because it does not cost more (at least asymptotically), and may enhance the security of the scheme. Instead of only using the matrices A, B and C, we would also have vectors A', B'

and C' to make these three transformations affine.

It is easy to see that even when the transformations are affine, the same cryptanalysis would work. Indeed, in the Shamir-Kipnis formalism the only change would be to add a constant term to P_A, P_B and P_C.
So we would have $P_A(\mathbf{x}) = \sum_{i=0}^{n-1} a_i \mathbf{x}^{q^i} + a_c$, $P_B(\mathbf{x}) = \sum_{i=0}^{n-1} b_i \mathbf{x}^{q^i} + b_c$ and $P_C(\mathbf{x}) = \sum_{i=0}^{n-1} c_i \mathbf{x}^{q^i} + c_c$.

If we use the same technique as above, the number of unknowns and the number of equations changes. Obviously, we have 3 more unknowns (in F_{q^n}) in the affine case. On the other hand we have many more equations to be completed. In the linear case each equation corresponded to one monom of the form $\mathbf{x}^{q^{i_1}+q^{i_2}+q^{i_3}+q^{i_4}}$, whereas in the affine case we also have to take into account all the monoms of the form $\mathbf{x}^{q^{i_1}+q^{i_2}+q^{i_3}}$,$\mathbf{x}^{q^{i_1}+q^{i_2}}$, $\mathbf{x}^{q^{i_1}}$ and the constant term.
Overall, there are many new equations and only 3 new variables, so the same technique (as in the linear case) will also work.

Complexity Analysis

The relinearization technique has polynomial time complexity. We use it for quadratic systems of $O(n^2)$ variables, where n is our security parameter. Hence, our attack is clearly polynomial time, so the system is theoretically broken.
Let us take a deeper look at the actual complexity. It is known that one can solve a linear system of dimension m over a finite field using Coppersmith/Winograd method in $O(m^{2.4})$. The relinearization seeks to solve a system of dimension roughly m^4 so an overall complexity of $O(m^{10})$.
In the cryptanalysis we use the relinearization with $m = n^2$, hence having a complexity of roughly $O(n^{20})$. The proposed value was $n = 9$, the attack is practically feasible.

Teil II

Commitment Range Protocols

5
Introduction

In this (short) second part, we present our work, still unpublished, in the area of zero-knowledge proof of knowledge.

Basically, a *"zero-knowledge proof of knowledge"* is a protocol between a person called the *prover*, and another one called the *verifier*. By this protocol, the *prover* proves to the *verifier* that he knows a secret value, without revealing it.
The term *"zero-knowledge"* means that the verifier does not gain any information about the secret value during the protocol, although, in the end, he is convinced of the prover's knowledge of this value.

These proofs appear extremely useful. They are irreplaceable elements to enable a secure use of cryptographic protocols.
Many cryptographic protocols use such proofs.
Among some applications : identification, authentication, multi-party computation, electronic voting, electronic cash systems, publicly verifiable secret sharing...

Our contribution to this field is an improved commitment range proof.

A commitment range proof is a *zero-knowledge proof of knowledge*, where the prover does not only convince the verifier that he knows a secret value, but also that this value (e.g number) belongs to a certain interval I.

At the end of the protocol, the verifier learnt nothing about the secret value although he is convinced that the prover knows it, and that it lies in the interval I.

Organisation of the chapter.
First, we introduce the necessary concepts, definitions and protocols that are to be used. Then we concentrate on the most efficient commitment range proof known so far. Finally, we improve this proof substantially.

5.1 Introduction to Zero-Knowledge Proof of Knowledge

5.1.1 Definition

A proof of knowledge is a protocol between a prover (also called Alice) who claims to know a secret value x, and a verifier (also called Bob) who does not know x.
The goal is for the prover to convince the verifier that he knows x without revealing it.
To be secure, such a protocol must prevent the existence of cheating provers and verifiers. A prover cheats if he does not know x but succeed in convincing the verifier that he knows it. A verifier can cheat in that he uses the protocol to get x or any information about x such as its parity or an interval in which lies x or more generally any (mathematical) statement about x.

In order to be a *zero-knowledge proof of knowledge*, the protocol must have the following properties :

1. *Completeness.* If the prover has x, then by the protocol the verifier gets convinced that x is known to the prover, we say that the verifier accept the proof.

2. *Soundness.* If the prover does not know x, then, at the end of the protocol, the verifier does not accept.

3. *Zero-knowledge.* During the protocol, the verifier learns nothing about x.

Remarks.

- The *completeness* and *soundness* properties basically say that the verifier is convinced if and only if the prover has x (is not a cheater).

- The last property ensures that neither the verifier nor anyone who has access to the exchanged data can learn extra information about the secret x.

- These 3 properties are often relaxed in various ways in order to stick with the protocols used in practice. For instance, soundness can be defined in different ways. On the one hand, soundness property can traduce the fact that a verifier never accepts if the prover is a cheater (and does not know x). On the other hand, soundness definition could allow the verifier to have a negligible probability of accepting while the prover is cheating. This change in the definition does not affect the essence of the soundness property.

Several accurate mathematical definitions exist for these properties, but they are not relevant in this thesis.

5.1.2 The discrete logarithm problem

In *zero-knowledge proof of knowledge* the prover proves to the verifier that he knows the solution of an instance of a hard problem.

In this paragraph we formally define the discrete logarithm problem.

Discrete Logarithm Problem in $(\mathbb{Z}_n)$

- *Given* $n \in \mathbb{Z}$ *and* $g, h \in \mathbb{Z}_n^*$ *with* $h \in < g >$.
- *find* x *such that* $h = g^x \mod n$

Remarks.

- *The discrete logaritm problem is hard to solve.*
  If a prover wants to make a zero-knowledge proof of knowledge that he has a discrete logarithm, the most basic assumption is that the discrete logarithm problem is hard.
  It lies in the complexity class $NP \cap CoNP$ which allegedly contains no NP-hard problem, and indeed there exist sub-exponential algorithms which solve the discrete logarithm problem. However for sufficiently large N (several hundreds bits), no practical algorithm can solve the problem in a reasonable amount of time.
  More importantly, this problem is among the two most studied problems in cryptography (the other being the factorization problem). We can be pretty confident that there will not be a major breakthrough which allows to efficiently solve it, at least until quantum computers become a reality.

- When using the discrete logarithm problem, the prover chooses x, n and g, he computes g and $h = g^x \mod n$. Then he makes n, g, h public to the prover. Finally, he makes a (zero-knowledge) proof that he knows x.

- Note that such a proof identifies the prover uniquely, indeed he is the only one who knows x. Nobody else can compute x, either from the public parameter h, g, n (because the problem is hard to solve) or by using extra information gain during the protocol (because of the zero-knowledge property). Thus nobody else can make a proof of knowledge of x on this precise instance, for this reason such proofs are used for electronic identification.

5.1.3 The Strong RSA Problem

Some proofs of knowledge rely not only on the discrete logarithm problem, but also on the so called "Strong RSA Problem".

Strong RSA Problem

- *Given a RSA-modulus n of unknown factorization and $v \in \mathbb{Z}_n^*$.*
- *find $u \in \mathbb{Z}_n^*$ and $e > 1$ so that $v = u^e \mod n$.*

For the proofs based on the latter problem, the following assumption is made. **Strong RSA Assumption :**
There is a probabilistic polynomial time Algorithm A which on input $1^{|n|}$ outputs $v \in G$. All probabilistic polynomial time algorithms have negligible probability of solving the Strong RSA Problem on the latter instance.

Remarks.

- The famous RSA problem asks to find the e-th root of a given element (in $\mathbb{Z}_n^*$) for a specific e, whereas in the strong RSA problem any root is a solution.
  The strong RSA problem may be easier to solve than the RSA problem, however it was introduced more than 10 years ago in [4] and used in many cryptographic constructions since then. There are still no better algorithms to solve the strong RSA problem than the RSA problem. Thus one can be confident of the difficulty of this problem.

5.1.4 A proof of knowledge of a discrete logarithm in a group of known order

In this subsection, we present a concrete example of proof of knowledge (based on the discrete logarithm problem) which was introduced in the seminal work of Schnorr [48]. All the proofs which will follow are based on this one. Note that we speak of *proof of knowledge* instead of *zero-knowledge proof of knowledge* because the next protocol is not known to be zero-knowledge. Still, in practice, no information about the secret are leaked during the execution of the protocol, this will be explained later.

Note that there exist *zero-knowledge proof of knowledge* based on numerous other problems, but most of the advanced cryptographic protocol uses the discrete logarithm problem, because of its efficiency and attractive mathematical properties.

In Schnorr's proof the secret value is a number x, the public ones are g and h such that $h = g^x$. All the operations occur in the subgroup $< g >$ of $\mathbb{Z}_n^*$ of prime order p.

Alice (the prover) knows $p, n \in \mathbb{Z}$, $h, g \in \mathbb{Z}_p^*$ and $x < p$.
Bob (the verifier) knows p, n, h, g.

Schnorr's Proof of knowledge of a discrete logarithm

Alice		Bob
Chooses w uniformly in $[1, p]$ $a = g^w$	$\xrightarrow{a}$	
	$\xleftarrow{c}$	Chooses c uniformly in $\{0,1\}^k$
Computes $r = w + x \cdot c \mod p$	$\xrightarrow{r}$	Verifies that $g^r = a \cdot h^c$ If so, Bob is convinced.

We sketch that this protocol is *complete* and *sound*.

- *Completeness.* If Alice has x she can compute the correct r. Bob is convinced.

- *Soundness.* First, notice that Alice can cheat with probability at least 2^{-k}. Indeed, without knowing x, she first makes a guess at the value c_0 that Bob will send her. Then she chooses any r_0 and computes $a_0 = g^{r_0} h^{-c_0}$. Finally she starts the protocol and sends a_0 to Bob. If Bob sends c_0, she replies with r_0 and Bob is convinced because the equation is satisfied.

  On the other hand, Alice cannot cheat with probability greater than 2.2^{-k}. Otherwise it is possible to show that for some a she can answer at least two different challenges sent by Bob.
  Suppose she sends a and can reply correctly to both c_0 and c_1. It means she can compute r_0 (in reply to c_0) so that $g^{r_0} = a \cdot h^{c_0}$, and she can compute r_1 (in reply to c_1) such that $g^{r_1} = a \cdot h^{c_1}$. Let us divide the two equations, $g^{r_1 - r_0} = h^{c_1 - c_0}$, moreover $c_0 \neq c_1 \mod p$ so $c_1 - c_0$ is invertible modulo the prime p. Thus $g^{\frac{r_1 - r_0}{c_1 - c_0}} = h = g^x$ so $x = \frac{r_1 - r_0}{c_1 - c_0} \mod p$.
  After Alice has answered these two challenges c_0, c_1 sent by Bob, she

can easily compute the secret value x.

In order to turn the last paragraph into a full proof, the remaining problem is to estimate the time spent by Alice until she is asked to answer to such two challenges.
Formally, it is possible to prove that if Alice can cheat with probability $\epsilon > 2^{-k+1}$, after running the protocol $\frac{4}{\epsilon}$ times (on average), she is able to compute x the discrete logarithm of h.

We assumed at the beginning that the discrete logarithm problem is hard to solve. According to the last theorem, if there is no algorithm which solve the discrete logarithm in less than 2^{80} steps, then there is no cheater with probability bigger than $4.cst.2^{-80}$ where cst is the time needed to run the protocol one time.

As a conclusion, the soundness property holds.

- *Zero-Knowledge.* To prove the zero-knowledge property of a protocol in general, we show that Bob is able to produce (in polynomial time) on his own the same triplets (a, c, r) as the ones he would exchange with Alice if he runs the proof.
  Thus by running the protocol, Bob learns nothing more than what he was able to learn on his own, using only the public key.

  Note that the triplets are randomized variables, based on the distribution that Alice and Bob respectively chose to produce both a and c. So saying that Bob is able to produce the same triplets actually means that he is able to produce triplets with the same distibution.
  It is important to notice that Bob could behave according to the plan and choose c uniformly in $\{0,1\}^k$, but he could also be dishonest and choose c with another distribution. Indeed, if it was possible for him to learn something about the secret x by changing the way he produces c, he would certainly use this opportunity.
  The distribution of c, chosen by Bob, has incidence on the one of r, so in order to prove the Zero-Knowledge property, Bob must be able to produce, on his own, valid (a, c, r) triplets, and peculiarly for any possible distribution of c in $\{0,1\}^k$.
  Notice also that Alice has no interest in being dishonest and not to choose w with uniform distribution.

  We show that Schnorr's protocol is Zero-Knowledge if Bob is honest, this property is also known as *honest verifier zero-knowledge*.
  Bob simulates valid triples as follows : he chooses uniformly $r \in [1, p]$

and $c \in \{0,1\}^k$, then computes $a = g^r h^{-c}$ (which clearly has uniform distribution in $< g >$). Obviously the triplet (a, c, r) is valid and has the same distribution as in a normal execution of the protocol, in which Bob picked uniformly distributed c.
Besides this proof, it is still unknown if Schnorr's protocol is Zero-Knowledge also when Bob is dishonest (and the interval $[0, 2^t]$ is big enough, meaning it has a size which is exponential in the input length). This is due to the fact that in the real protocol, Bob could choose c according to the value of a that he has just received from Alice. We don't know if Bob can simulates this behavior, because in the method used above to simulate valid triples he must choose c before having a.

Honest verifier zero-knowledge might sound like a very weak security property. Indeed, proving security only when the participants are honest is not very satisfying. However, in practice, the protocol is used in a non-interactive fashion in which the value c is not chosen by Bob. Thus Bob cannot be dishonest on this matter, the honest verifier zero-knowledge property shows that in this case Bob cannot learn anything about the secret.

5.1.5 Non-interactive zero-knowledge proofs : The Fiat-Shamir paradigm

In Schnorr's protocol, Alice and Bob must both be online, in order to run the proof. In practice, Alice and Bob are two computers in a network, and Bob asks Alice to run the proof in order to identify herself. Thus Alice must be always turned on, moreover, each identification requires Alice non-negligeable computations payload.

In practice, there is a way to overcome this problem, introduced by Fiat and Shamir [2]. Remember that in the soundness proof of section 5.1.4, it was established, that if Alice can foresee the challenge c before having chosen a, she can cheat. On the other hand, if she cannot foresee c she cannot cheat. Therefore, what is important is to prevent Alice to know the challenge before she chooses a w. In Schnorr's protocol, this happens because c is chosen by Bob after Alice has sent him a.
Fiat and Shamir proposed a way for Alice to compute herself the challenge c but without possibly foreseeing it before w was chosen.

Let us suppose that Alice has a hash function $\mathcal{H}$ as an element of the public key. In the proof, instead of receiving c, she computes it herself as $c = \mathcal{H}(g^w)$. If the hash function has the desired properties, Alice cannot predict c or choose it before computing $c = \mathcal{H}(g^w)$. Thus she can not cheat. She then publishes the proof anywhere as the following sequence :

$$n, p, \mathcal{H}, h, g, (a, r)$$

Notice that Alice does not need anyone else to run the protocol which is then non interactive. Moreover, she does not need to run it several times. In practice, she publishes the result of the protocol, and anyone who wants to be convinced that Alice knows x takes (a, r) and verifies that $a = g^r h^{-\mathcal{H}(a)}$. The published set of values is then really a "proof", in the mathematical sense, that Alice knows x.

The Fiat-Shamir paradigm works as long as the hash function has some "ideal" security properties. For such a hash function, called a "random oracle" one can prove that the non-interactive proof is a "zero knowledge proof of knowledge". Notice, however, that the hash used in practice are not "random oracles" but only assumed to be ones.

In practice, all the proofs of knowledge, and particularly the ones which will be described later, are used in a non-interactive fashion.

However, in this thesis, we will always describe them as interactive, because it is easier to understand. Anyway, turning these interactive proofs non-interactive is obvious.

5.1.6 A proof of knowledge in a group of unknown order

For some applications, such as commitment range proof, it is necessary that the order of the group in which the discrete logarithm proof is made remains unbeknown to the prover.

In practice, one uses the group $\mathbb{Z}_N$ where N is the product of two big primes unbeknown to both the prover and the verifier. The prover is also given an element g of "high order". Different techniques exist to provide the prover and the verifier with such parameters, the most common is to ask a trusted third party to generate them.

In such a setting the prover chooses a private x and computes the public value $h = g^x \mod n$. Next, we show how the prover can make a proof of knowledge of x. This proof is similar to the one in groups of known order, however its security relies on the Strong RSA assumption.

We suppose that both the prover and the verifier know the bitlength of the group order of $< g >$, noted l_g. Moreover we use security parameters $\epsilon > 1$ and $k > 1$.

Protocol		
Alice *(prover)* Knows n, g, h, l_g, x		Bob *(verifier)* Knows n, g, h, l_g
Chooses w uniformly in $\{0,1\}^{\epsilon(l_g+k)}$ $a = g^w$	$\xrightarrow{a}$	
	$\xleftarrow{c}$	Chooses c uniformly in $\{0,1\}^k$
Computes $r = w + x \cdot c$	$\xrightarrow{r}$	Verifies $g^r = a \cdot h^c$

- *Completeness.* It is clear that if Alice knows x and runs the protocol properly, then Bob can verify the validity of the equation.

- *Soundness.* The proof is similar to the case where the group order is known, however the Strong RSA assumption is needed.
  The sketch of proof is as follows. If for a value of w Alice can answer correctly to two different values c and $c^{'}$ (leading to r and $r^{'}$), then $g^{r-r^{'}} = h^{c-c^{'}}$. Let $d = gcd(r-r^{'}, c-c^{'})$, there exist (t,s) so that $t\frac{c-c^{'}}{d} + s\frac{r-r^{'}}{d} = 1$, then $(g^t h^s)^{\frac{c-c^{'}}{d}} = g^{1-s\frac{r-r^{'}}{d}}(h^{c-c^{'}})^{\frac{s}{d}} = g^{1-s\frac{r-r^{'}}{d}}(g^{r-r^{'}})^{\frac{s}{d}} = g$.
  If $d < c-c^{'}$, then $\frac{c-c^{'}}{d} \in \mathbb{Z}$ and $\frac{c-c^{'}}{d} > 1$, which contradicts the Strong-RSA assumption. Therefore $d = c-c^{'}$ and it follows that $c-c^{'}$ divides $r-r^{'}$, so the secret $x = \frac{r-r^{'}}{c-c^{'}}$ can be computed, which solve the discrete logarithm problem.

- *Honest Verifier Zero-Knowledge.* As in the case where the group order is known, if Bob behaves as expected in the protocol then we have Zero-Knowledge property.
  A simulator is created in the same way as in the previous proof. Note however that, as the group order is not known, there is no modular reduction when computing r, which clearly leaks information about the secret x. For this reason, the coefficient w has to be chosen in a bigger interval (whose size is fixed by the new variable ϵ) in order to hide x correctly.
  The bitlength of the group order is known, so it is possible to fix ϵ so that g^w is close enough to uniform distribution. See [30] for details.

Remarks.
The protocol will be turned non-interactive. There is no need for stronger

zero-knowledge property, since when using the Fiat-Shamir heuristic, it is assumed that we are in the honest verifier case.

5.1.7 Fujisaki-Okamoto Commitment Scheme

Because of the zero-knowledge property, no one can learn anything new about the secret x (in the last two proofs of knowledge) by seeing the proof.
It does not mean that it is impossible to learn something about x. More precisely, the public key itself h might leak information about x. For instance it is well known that given $h = g^x \mod p$ the parity of x is computable in polynomial time (via the computation of $h^{\frac{p-1}{2}} \mod p$). It means that in Schnorr's proof, anyone can learn the parity of the secret x.

In order for the identification scheme to be secure, it is then very important that neither the proof (of knowledge) nor the public parameters leak information about the secret.

In the case of the discrete logarithm problem, if the group order is known, as we mentioned above, the public key leaks at least the parity of x. If the group order is unknown, and more specifically if the parameters are chosen as in section 5.1.6, we don't know if the public parameter $h = g^x \mod n$ leaks information on x or not.

To avoid the possible security problem which might occur by using $h = g^x \mod n$ as the public parameters, it is possible to use the so-called Fujisaki-Okamoto commitment scheme [19] instead.

In practice, in such scheme, Alice and Bob are given g_1 and g_2 by the third party which has alread given them the modulus n. Moreover $g_2 \in < g_1 >$ and Alice does not know the discrete logarithm of g_2 in basis g_1.
Then Alice chooses in addition to x a secret (random) r and computes $E(x, r) = g_1^x g_2^r \mod n$. The public parameter is then $E(x, r)$.
In this setting it can be proved that without the factorisation of n, and the value of $log_{g_1}(g_2)$, it is not possible to compute x_1, x_2, r_1, r_2 (where $x_1 \neq x_2$) so that $E(x_1, r_1) = E(x_2, r_2)$, thus the prover can not pretend to know another secret.
Fujisaki and Okamoto also showed that $E(x, r)$ reveals no information about x.

It is not difficult to adapt all the proofs of knowledge a la Schnorr in order to use the Fujisaki-Okamoto (private/public) key pair. In such a proof, we are sure that no one can learn anything about the secret at all, this is the one used in practice.

However, as in the case of non-interactive proofs, proof of knowledge of a

discrete logarithm are easier to present and to understand when the public key is $h = g^x \mod n$, instead of the Fujisaki Okamoto commitment. Thus, in what follows we stick to $h = g^x \mod n$ as the public parameter.
Anyway, it is not difficult to change all the coming proofs in order for them to use the Fujisaki Okamoto commitment.

5.1.8 Proof of knowledge that two discrete logarithms are equal

We briefly remind the protocol used by Alice to prove that she knows x, the discrete logarithm of u (with $u = g^x$) in basis g, and that it is equal to the discrete logarithm of v in basis h ($v = h^x$).
This kind of proof first appeared in [15], for group of known order.
In our case, it is crucial that Alice does not know the group order. How to build such a proof is detailed in [30], here we simply give a sketch of it.

As in the previous proof, the bitlength l_g of the group order is known. $\epsilon > 1$ and $k > 1$ are security parameters.

Protocol		
Alice *(prover)* Knows g, u, v, h, x		Bob *(verifier)* Knows g, u, v, h
Chooses w uniformly in $\{0,1\}^{\epsilon(l_g+k)}$ $(a, b) = (g^w, h^w)$	$\xrightarrow{(a,b)}$	
	$\xleftarrow{c}$	Chooses c uniformly in $\{0,1\}^k$
Computes $r = w + x \cdot c$	$\xrightarrow{r}$	Verifies $g^r = a \cdot u^c$ and $h^r = b \cdot v^c$

- Note that Bob uses the same r to check both equations, in the last step. This proves that x is the discrete logarithm of both u (in basis g) and v (in basis h).
- The completeness, soundness and zero-knowledge properties are proved in the same way as in the proof of knowledge of section 5.1.6.

Characteristics of the proof

Alice needs to do 2 exponentations, and Bob needs 4. Moreover, if we use standard security values, the modulus is a 1024 bit number and the security parameters of the proof are set so that the probability of a cheater to succeed

is 2^{-80}. With such values, the length of the non-interactive version of this proof is a few thousand bits, according to [6].

5.1.9 Proof that a discrete logarithm is a square

Suppose that Alice knows x so that $y = g^{x^2}$, and Bob knows y and u (such that $u = g^x$). Alice wants to prove to Bob that y hides the square of the discrete logarithm of u.
Using the latter protocol, she can prove that $log_u(y) = log_g(u)$. This convinces Bob that y hides a square (in basis g) of the discrete logarithm of u.
The characteristics of this proof are identical (number of exponantiations and length) to the characteristics of the proof of equality of discrete logarithms.

5.2 Commitment Range Proof

Identification relies on zero-knowledge proof of knowledge of a secret number, being able to prove at the same time that this number lies in a certain interval, is a very general and natural problem. Such proofs are called Commitment Range Proof or also Range Bounded Commitment.

So far, they have been used in a large variety of protocols, such as electronic cash systems [1], group signatures ([22] or [23]), and some zero-knowledge proofs [19].

This section begins with a first efficient commitment range proof [1], called CFT proof [1]. Then we construct the most efficient proof to date, due to Boudot [6].
Finally we show how to modify Boudot's proof to get a shorter, more efficient and more general proof.

5.2.1 CFT Proof

In this section we present a first proof showing that a discrete logarithm lies in an interval.
This proof enables Alice, who has an integer x in $[0, b]$, to prove to Bob (who knows only $h = g^x$) that x lies in $[-\delta b, \delta b]$. δ is called the expansion rate.
On the one hand this proof is very cheap, it is very short and requires only 3 modular exponentiations. On the other hand, as we will see in the next paragraphs, Alice can only prove (formally) that x is in $[-\delta b, \delta b]$. For standard parameter values, δ is 2^{120}.

[1] named after its authors : Chan, Frankel and Tsiounis.

So Alice can only prove the containment of x in a much wider interval.

We stress again that, to make the proof work, Alice must know $x \in [0, b]$, but Bob can only be convinced that $|x|$ is not superior to δb.

As in the last section settings, we suppose that both the prover and the verifier know the bitlength l_g of the group order. The protocol is similar to all discrete logarithm protocols, such as the one presented in section 5.1.8. Indeed, Alice will prove that she knows the discrete logarithm of $h = g^x$.
First, she chooses w in an interval and sends $u = g^w$ to Bob. Then Bob sends a challenge c to Alice who will reply with $r = w + xc$. The only additional requirement (compared to the standard proof) is that Bob checks the range of r.
The idea is that if Bob looks at the value of r, he can know something on the interval containing x. If the value of r is "small" then x should also be "small", on the other hand if x is "large" so should be r.
Note that if the group order was known by Alice, she could cheat and reduce r modulo the group order before sending it. Then, the interval containing r would give false information on the interval in which x lies.

Protocol		
Alice *(prover)* Knows $g, 0 < x \leq b, h(= g^x), \delta$		Bob *(verifier)* Knows g, h, δ, b
Chooses w random in $[0, \delta b]$ Computes $a = g^w$	$\xrightarrow{a}$	
	$\xleftarrow{c}$	Chooses c at random in $[0, 2^t]$, satisfying $t \leq l_g$
Computes $r = w + x \cdot c$	$\xrightarrow{r}$	Verifies $g^r h^{-c} = a$ and that r is in $[2^t b, \delta b]$

The idea of the proof is the following :

- This is a proof a la Schnorr, so in order for Bob to accept, Alice must know x. However, we have to prove that Alice can not cheat in regard to the interval in which x lies.

- On the one hand if Alice is dishonest and $x > \delta b$, once w has been chosen (to whatever value, even negative as Alice is dishonest) there is at most one possible value for c in $[0, 2^t]$ so that $r(= w + xc) \leq \delta b$.
  Indeed, for two different challenges c_0, c_1 then $r_0(= w + xc_0)$ and $r_1(= w + xc_1)$ verify $|r_1 - r_0| > \delta b$. So if Alice is dishonest and $|x| > \delta b$

then with probability bigger than $1 - \frac{1}{2^t}$ we have $r < 0$ or $r > \delta b$.

- On the other hand, w is there to hide the value of x (remember that there is no modular reduction of r), so should be chosen in an interval at least as large as the one in which x could be (from the adversary point of view).
  Secondly, we want to have $r \leq \delta b$, but as w has to be chosen in the same interval ($[0, \delta b]$), we will have $r(= w + xc) \leq \delta b$ with high probability only if x is actually much smaller than δb (actually in $[0, b]$).
  A precise computation of the later condition will determine which value has to be taken for δ, so that with high probability $r \leq \delta b$.

We are now ready to translate the last arguments into a formal proof that the latter protocol is an *honest verifier zero-knowledge proof of knowledge* of a discrete logarithm in an interval.

- **Completeness.** It is clear that if Alice knows x, she can compute $r = w + cx$ and Bob will verify that $g^r h^{-c} = a$. Now if x $[0, b]$ we have to show that $r \in [2^t b, \delta b]$ with high probability. Indeed for a given $x \in [0, b]$, let $P = Pr_{w,c}\{w + xc < \delta b\}$. Clearly $P > Pr_w\{w < \delta b - 2^t b\} = \frac{\delta b - 2^t b}{\delta b} = 1 - \frac{2^t}{\delta}$. Hence taking $\delta = 2^{40+t}$ we have $P > 1 - 2^{-40}$.
  The event $r < 2^t b$ clearly happen with very low probability. Indeed the probability on w and c so that $r = w + xc > 2^t b$ is less than the probability that $r > 2^t b$. And $Pr\{r < 2^t b\} < \frac{2^t b}{\delta b} = 2^{-40}$.
  As a conclusion the protocol is complete.

- **Soundness.** It is well known that when Bob checks the equality $g^r = a \cdot h^c$, Alice must know the secret x, so we already know that Alice cannot be dishonest on this.
  Can she be dishonest as regard to the interval containing x ? Let's suppose that r is in $[0, \delta b]$ and Alice is dishonest (ie $|x| > \delta b$), she can convince Bob with a probability at most 2^{-t}. Indeed, if $|x| > \delta b$, once w has been fixed by Alice, Bob will send the challenge $c \in [0, 2^t]$, and there is only one value of c so that $w + x \cdot c \in [0, \delta b]$.
  Hence by taking $t = 80$, we achieve soundness property. This imposes $\delta = 2^{120}$.

- **Honest Verifier Zero-Knowledge** We have to show that the exchanged values can be simulated, if Bob is honest. This is indeed possible when $\delta = 2^{120}$ and $t = 80$.

  1. As $\delta = 2^{120}$ it makes sense to say that $\delta b \gg 2^{l_g}$, because, a random x has a bitlength l_g (and so has b). Therefore u is uniformly

distributed among the group elements.

2. If Bob is honest, he chooses c uniformly in his interval.
3. For a fixed $x \in [0, b]$, when w ($\in [0, \delta b]$) and c ($\in [0, 2^t]$) are uniformly distributed, and r is in $[0, \delta b]$, r is clearly uniformly distributed in $[2^t b, \delta b]$. The distribution of r in $[0, 2^t b]$, on the contrary, is not uniform and depends on x. Thus this event can not be simulated, this is precisely why the verifier check that $r \in [2^t b, \delta b]$ and not $r \in [0, \delta b]$.

   Moreover when r is uniformly distributed in $[cb, \delta b]$, as (at least for a random x) $\delta b - cb \gg 2^{l_g}$, the value g^r will be uniformly distributed among group elements.

As a conclusion, one can choose c uniformly in $[0, 2^t]$, r uniformly in $[2^t b, \delta b]$ and compute a as $a = g^r h^{-c}$, As g^r is uniformly distributed among the group elements, so is a. The triple (r, c, a) is valid and has the same distribution as during a real execution of the protocol.

Non-interactive version

We show how to turn the proof non-interactive using a secure hash function which outputs t bit strings, t being the same security parameter as in the interactive proof.

- Bob chooses w randomly and computes $c = H(g^w)$ and $r = w + xc$. Finally he sends (c, r).
- Alice verifies that $r < \delta b$ and $c = H(g^r \cdot h^{-c})$.

In conclusion, this protocol enables Alice to prove to Bob that $x \in [-2^{120} b, 2^{120} b]$.

Characteristics of the proof

Alice has to compute one exponentiation, whereas Bob needs two. The proof made out of a t-bit number, and of a $|\delta b|$-bit number.

5.2.2 Boudot's Method

In this section, we present the commitment range proof given by Boudot in [6].
It enables Alice who has $h(= g^x)$ and x, to prove to Bob that she knows the discrete logarithm of h and that $a < x < b$.

To prove that $x < b$, Boudot shows that $x' = b - x$ is positive. The same method is used to show that $a < x$.

Conceptually, the proof that $x' = b - x$ is positive is done in two steps :

- first we show (using [1]) that x' lies in $[q^2 - \Delta, q^2 + \Delta]$ for a certain square number q^2 and integer Δ.
- The second idea is to use a scaling trick, to reduce the size of the interval (still centered around q^2) by reducing Δ. In the end, it is proved that x' lies in $[q^2 - \Delta, q^2 + \Delta]$ with $\Delta < 1$, this is an interval of length less than 2 centered around a square, therefore $x' > 0$.

Apply the CFT proof on smaller numbers.
Alice wants to prove that $x'(= b - x)$ is positive. She decomposes $x' = q^2 + \mu$, then she sends $v(= g^\mu)$ and $u(= g^{q^2})$ to Bob, along with a proof that u hides a square (in basis g) and a CFT proof on v.

Bob can compute $g^{x'}$ from g^x, he checks that $uv = g^{x'}$ which convinces him that $x' = \log_g(u) + \log_g(v)$.
From the proof that v hides a square Bob is convinced that $\log_g(v) = q^2$ for an unknown q, and from the CFT proof he is convinced that $-\Delta < \log_g(u) < \Delta$. Thus he knows that $q^2 - \Delta < x' < q^2 + \Delta$. Bob is then convinced that x belongs to a certain interval, centered around a square number.

In practice the protocol is the following :

1. Alice computes $q = \lfloor \sqrt{x'} \rfloor$ and μ so that $x' = q^2 + \mu$.
2. Alice sends Bob $u = g^{q^2}$ and $v = g^\mu$ along with a proof that $\log_g(u)$ is a square.
3. Bob verifies that $g^{x'} = u \cdot v$ and is convinced that $\log_g(z)$ hides a square.

- Instead of proving that x' lies in an interval, Alice will prove to Bob that μ lies in an interval (which is smaller).
- We know that $\mu = x' - (\lfloor \sqrt{x'} \rfloor)^2$, hence $\mu \leq 2\sqrt{x'} = 2\sqrt{b - x} < 2\sqrt{b - a}$.
- With the CFT proof, Alice is able to prove to Bob that $v = g^\mu$ is a commitment of an integer in the interval $[-2^{120}\sqrt{b - a}, 2^{120}\sqrt{b - a}]$.
- So Alice can prove to Bob that $x' \in [q^2 - 2^{120}\sqrt{b - a}, q^2 + 2^{120}\sqrt{b - a}]$ for a certain q, unknown to him.

Scaling trick.

Using the latter proof with square decomposition, one can notice two things :

- The value of q^2 is roughly proportional to $x^{'}$.
- The value of the expansion rate δ is roughly proportional only to $\sqrt{x'}$.

This leads to the following idea.
Let $x^{''} = 2^T x^{'}$ for some T which will be fixed later. There exist $q^{'}$ and $\mu^{'}$ such that $x^{''} = q^{'2} + \mu^{'}$.
Alice can prove to Bob that $x^{''} \in [q^{'2} - \Delta^{'}, q^{'2} + \Delta^{'}]$ where $q^{'} = \lfloor\sqrt{2^T x^{'}}\rfloor$ and $\Delta^{'} = 2^{120}\sqrt{2^T(b-a)} = 2^{T/2}\Delta$.
This is equivalent to $x^{'} \in [\frac{q^{'2}}{2^T} - \frac{\Delta^{'}}{2^T}, \frac{q^{'2}}{2^T} + \frac{\Delta^{'}}{2^T}]$.
The factor $\frac{\Delta^{'}}{2^T} = \frac{2^{120}\sqrt{b-a}}{2^{T/2}}$ can be fixed to any arbitrary values (less than 1).

If T is chosen such that $\frac{\Delta^{'}}{2^T} < 1$ then the proof that $2^T x^{'} \in [q^{'2} - \Delta^{'}, q^{'2} + \Delta^{'}]$

proves that $x^{'} \in [\frac{q^2}{2^T} - \frac{\Delta^{'}}{2^T}, \frac{q^2}{2^T} + \frac{\Delta^{'}}{2^T}]$ (with $\frac{\Delta^{'}}{2^T} \leq 1$), consequently x is a positive (or null) number.

How to choose the parameter T

We want to have $\frac{\Delta^{'}}{2^T} = \frac{2^{120}\sqrt{b-a}}{2^{T/2}} = 1$, it implies $T = 240 + \log_2(b-a)$.

Boudot Scheme

Figure 5.1 represents Boudot's proof that $b - x$ is positive.

Schematically it is performed in 3 phases, even though in real life all are sent at the same time.

5.2.3 Characteristics of the proof

In the latter protocol, Alice must compute 6 exponentiations (1 for CFT proof, 2 in order to prove that a value hides a square, and the computation of g^q, g^{q^2} and g^μ), whereas Bob has 7 exponentiations to perform.
To prove both $x > a$ and $x < b$, the proof of figure 5.1 has to be performed twice, requiring a total of 26 exponentiations. Actually, only 25 exponentiations are really needed, because Bob can scale x (to $2^T x$) once, instead of scaling $x^{'}$ for both $x > a$ and $x < b$.

Protocol		
Alice Knows $z = g^{x'}, x' = 2^T x, a, b, T$		Bob Knows z, a, b, T
Computes $T = 240 + \frac{1}{2}\log_2(b-a)$		Computes $T = 240 + \frac{1}{2}\log_2(b-a)$ and $y = z^{2^T}$
Finds q, μ so that $q = \lfloor\sqrt{2^T x'}\rfloor$, $\mu = x' - q^2$	$\xrightarrow{u=g^{q^2}, v=g^{\mu}, ProofD}$) *where* $ProofD$ *is a proof that* $\log_g(v)$ *is a square*	Verifies that $z = u \cdot v$ and is convinced by $ProofD$
Makes the proof that $\mu \in [-2^{120+\frac{T}{2}}\sqrt{b-a}, 2^{120+\frac{T}{2}}\sqrt{(}b-a)]$ (cf. 5.2.1)	$\xrightarrow{Proof}$	Is convinced by $Proof$.

Abbildung 5.1: Boudot's proof that $x < b$

The length of the proof is the sum of the length of :

- Twice the length of the proof that a discrete log is a square.
- Twice the length of the CFT proof.
- The length of 4 numbers in Z_n .

5.2.4 Conclusion

Compared to the CFT method, Boudot's proof allows to prove exact containment of the committed number in an interval, but at the cost of more computation on both sides. The final proof is also longer, due to the other proof (than CFT) that has to be done, and that the whole scheme has to be performed twice.

We also stress that all the proofs have been presented using simple exponentiation as commitment. In practice, one uses the Fujisaki-Okamoto ([19]) commitment scheme instead, the computational (and length) cost will roughly double.

Even if in Boudot's proof the number of exponentiations to be performed is a few dozen, in most discrete logarithm based protocols this number is much smaller (generally only a few exponantiations are needed). This is a reason why some authors, like [8] still prefer to use CFT proof, in environments where efficiency is crucial.

Remarks. It is also obvious that the length (and computation) of the method is proportional to the size of the interval. Hence the larger the interval, the more computations.
Actually, this is problematic if one wants to prove that the committed number is in an interval $[a, +\infty[$, because the scaling factor T goes to infinity.

5.3 New Scheme

Introduction We present an improvement to Boudot's proof. The improvement factor is about $\frac{1}{4}$ both in term of computation and length of the proof. Boudot's proof roughly needs 5 proofs of knowledge a la Schnorr, whereas our proof requires about 3. We get very close to the limit (which may or may not be reached) of committment range proof with only one proof of knowledge.
Moreover this (new) proof is a general proof that two discrete logarithms have different signs, and hence can be used in a more general setting than interval containment.

In section 5.2.1, we have presented the CFT proof, in which Alice, for $x \in [0, b]$ can produce very fast (3 modular exponentiations), a very short proof that $x \in [-\delta b, \delta b]$ where $\delta = 2^{120}$. In section 5.2.2, we showed how Boudot uses this CFT proof, to prove exact containment of x in $[a, b]$.
This proof is roughly 9 times more costly (than CFT). To show $x \in [a, b]$, Alice proves that $x > a$ by showing that $(x-a)2^T$ is in an interval of length 2^{T+1} centered around a positive number. This last assertion is done with the help of a CFT proof and a proof that two discrete logarithms are equal. Once $x > a$ has been proved, Alice does the same for $x < b$.
Our proof is more efficient, because instead of proving that $x > a$ and $x < b$, we show that $a < x < b$ in one single step.

5.3.1 Idea

1. Notice that proving that $a < x < b$ is equivalent to proving that $(x-a)$ and $(x-b)$ have different signs.
2. Showing that $X_a = x - a$ and $X_b = x - b$ have different signs, is equivalent to showing that their product is a negative number.

5.3.2 Protocol

- Alice knows $X_a = x - a > 0$ and $X_b = x - b < 0$, she computes $z = g^{X_a X_b}$, along with a proof $\mathbf{P_1}$ that the discrete logarithm of z in basis g^{X_a} is the same as the discrete logarithm of g^{X_b} in basis g.

- Alice produces a proof that the discrete logarithm of z in basis g is a negative number, she can do it the same way as Boudot, e.g she decomposes $X_a X_b = -(q^2 + r)$ and does a proof that a discrete logarithm is a square, and a CFT proof.

- Bob verifies the three proofs, and is convinced that $X_a X_b$ is a negative number which is equivalent to $x \in [a, b]$.

5.3.3 Characteristics of the proof

If we look at the scheme, we notice that Alice has to produce two proofs of equality of discrete logarithm, and one CFT proof, so 3 proof a la Schnorr whereas Boudot's proof requires 4.

The length (and computational cost) of the new proof is roughly twice smaller than in Boudot's proof.

By a careful counting, we can see that Alice must do 5 exponentiations while Bob must perform 8, this means that overall 13 exponentiations have to be done.

Remark. It makes sense to compare the number of exponentiation between our proof and the one in [6], because in both proofs the exponents to compute have roughly the same size, which is directly proportional to the size of the interval.

5.4 Proof that $x \notin [a, b]$

There can be cases where we need to prove that x is not in an interval.

Boudot's proof does not allows to do that directly, however it is possible to prove that $x \in]-\infty, a]$ (with Boudot's proof) or $x \in [b, \infty[$, using standard OR proof (see [16] for instance).
The final proof length is twice as long as the original of Boudot.

On the other hand, it is very easy to use our scheme to prove that $x \notin [a, b]$. This is equivalent to proving that $(x - a)$ and $(x - b)$ have the same sign. So, in order to convince Bob, Alice can prove that $(x - a)$ and $(b - x)$ have different signs. This proof has the same length and same computational cost as the proof that $x \in [a, b]$.

5.5 Commitment Range Proof in Groups of known order.

The schemes presented so far prove the containment of the committed number in an interval only under the condition that the group order (along with the order of the basis element) is unknown.
An interesting question (which is also very relevant in practice) is to build such containment proofs for group of known order.
It is indeed possible to build such a proof, with roughly 2 proof of knowledge (roughly twice as when the order is unknown).

The key idea is to transfer the commited value from a group with known order to a group of unkown order. Once this has been done, we use our method (or any other commitment range proof) to prove the containment of the committed value in the interval.

Let's say an element g_1 generates a group $G_1 =< g_1 >$ of known order. The value $h = g_1^x$ is public, while x is kept private.
Now suppose Alice has also N of unknown (to her) factorisation along with an element g of "big" unknown (to her) order producing the group $G =< g >$. Alice can compute $y = g^x$ and run the proof of section 5.1.8 to show that $\log_{g_1}(h) = log_g(y)$. Clearly the security proof still holds even if one of the group has a known order. Hence, at the end, Bob is convinced that the secret has been transferred from one group to another. Thereafter, Alice can perform the commitment range proof on x in the group G.

We see that for this method to work, it is crucial that Alice is given a group G of unknown order.

There are mainly three approaches to provide Alice with this values:

1. Alice can be given the number N (of unknown factorization) along with an element g (of big unknown order) by any trust party. It is often the case in advanced cryptographic protocol that each user (or a trust party) compute such public values in advance.

2. Alice and Bob can use one of the existing schemes to produce a shared RSA module whose factorization is unknown to Bob and Alice, see [25] and [14].

3. Alice can produce N and g herself in a way that Bob is convinced that she neither knows the factorisation of N nor the order of g. This will be the case if she chooses N "randomly", to prevent her from lying we can ask that the computation makes use of hashing function. In this way Bob is convinced that Alice has not produced N by multiplying together the (then known) prime factors.
   We argue that if N is big enough it won't be smooth with very high

probability, hence Alice won't be able to factor it, moreover 2 is an element with big order in $\mathbb{Z}_n$ with high probability. These claims are well known in cryptographie and formal (asymptotic) proof can be found in [7].

Literaturverzeichnis

[1] Y. Tsiounis A. Chan, Y. Frankel. Easy Come-Easy Go Divisible Cash . *Proceeding of Eurocrypt'98*. LNCS 1403 (p561-574) .

[2] A. Shamir A. Fiat. How to prove yourself: Practical solutions to identification and signature problems . *Proc of Crypto 86*.

[3] A. Shamir A. Kipnis. Cryptanalysis of the oil and vinegar signature scheme. *LNCS 1462*. Proc of Eurocrypt 1998.

[4] Niko Bari'c and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. *LNCS 1233*. Proc. of Eurocrypt '97.

[5] E. Biham. Cryptanalysis of Patarin 2-Round Public Key System with S Boxes (2R) . *LNCS 1807*, 2000.

[6] F. Boudot. Efficient proofs that a committed number lies in an interval . *Proc of Eurocrypt 2000*.

[7] I. Shparlinski C. Pomerance. Smooth Orders and Cryptographic Applications . . *ANTS 2002*.

[8] S. Canard. Signatures de Groupe, Variantes et Applications . PhD Thesis, University of Caen, 2003. p 77. .

[9] Don Coppersmith, Jacques Stern, and Serge Vaudenay. Attacks on the birational permutation signature schemes. *Lecture Notes in Computer Science*, 773, 1994.

[10] Don Coppersmith, Jacques Stern, and Serge Vaudenay. The security of the birational permutation signature schemes. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 10(3):207–221, Summer 1997.

[11] N. Courtois, L. Goubin, and J. Patarin. SFLASH, a fast asymmetric signature scheme . 2003. available at http://eprint.iacr.org/2003/211/.

[12] N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations. *LNCS 1807*. Proc of Eurocrypt 2000.

[13] Nicolas T. Courtois. Efficient zero-knowledge authentication based on a linear algebra problem MinRank. *Lecture Notes in Computer Science*, 2248:402+, 2001.

[14] M. Franklin D. Boneh. Efficient generation of shared rsa keys. *LNCS* 1462. Crypto '97.

[15] T. P. Perdersen D. Chaum. Wallet databases with observers . *Proc of Crypto 92.*

[16] I. Damgard. On Σ-Protocols . lecture on Cryptologic Protocol Theory. See Damgard webpage .

[17] Y. Din-Feng, L. K-Yan, and D. Zong-Duo. Cryptanalysis of $2R$ Schemes. *LNCS 1666.* Proc of Crypto 1999.

[18] V. Dubois, P. Fouque, A. Shamir, and J. Stern. Practical Cryptanalysis of SFLASH. *LNCS 1807.* Proc of Crypto 2007.

[19] T. Okamoto E. Fujisaki. Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations . *Proceedings of Crypto'97.* LNCS 740 (p89-195) .

[20] J-C. Faugere. A new efficient Algorithm for computing Gröbner bases without reduction to zero (F5) . *http://www-calfor.lip6.fr/ jcf/Paper/@papers/f5.pdf.* Proc of ISSAC 02.

[21] J-C. Faugere. A new efficient Algorithm for computing Gröbner (F4) . Journal of Pure and Applied Algebra 139 (1999).

[22] B. de Medeiros G. Ateniese. Efficient Group Signatures without Trapdoors . *Proc of Asiacrypt 2003.*

[23] M. Joye G. Tsudik. G. Ateniese, J. Camenish. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme . *Proc of Crypto 2000.* LNCS, vol. 1880. .

[24] C. Jefferson G. Bard, N. Courtois. Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers . *http://eprint.iacr.org.* eprint 2007.

[25] J. Stern G. Poupard. Generation of shared rsa keys by 2 parties. *LNCS* 1514. Asiacrypt '98.

[26] L. Goubin and J. Patarin. Asymmetric Cryptography with S-Boxes . *LNCS 1807.* Proc of ICICS 1997.

[27] L. Goubin and J. Patarin. Asymmetric Cryptography with S-Boxes . *LNCS 1807.* Proc of ICICS 1997.

[28] H. Imai and T. Matsumoto. Algebraic Methods for Constructing Asymmetric Cryptosystems. *LNCS 229*, 1986.

[29] A. Joux J-C. Faugere. Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems using Gröbner Bases. *LNCS 2729.* Proc of Crypto 2003.

[30] M. Michels J. Camenish. A Group Signature Scheme Based on an RSA-Variant . Tech. Report, available at *http : //www.zurich.ibm.com/ jca/*. .

[31] D. Schmidt J. Ding, J. Gower. Zhuang-Zi: A new Algorithm for Solving Multivariate Polynomial Equations over a Finite Field . *http://eprint.iacr.org/2006/038.pdf*. eprint 2006.

[32] N. Courtois J. Patarin, L. Goubin. C^*_{-+} and HM: Variations around two schemes of T. Matsumoto and H. Imai. *LNCS 1514*. Proc of Asiacrypt 1998.

[33] V. Shoup J. Von Zur Gathen. Computing frobenius maps and factoring polynomials. *ACM Press, 1992*. Proc of 24th Annual ACM Symp. Theory of Comput.

[34] JO. Shallit JF. Buss, GS. Frandsen. The computational complexity of some problems of linear algebra. pages 451–462, 1997.

[35] A. Kipnis and A. Shamir. Cyptanalysis of the HFE Public Key Cryptosystem. *LNCS 1666*. Proc of Crypto 1999.

[36] N. Koblitz. Algebraic Aspects of Cryptography. *Collection. Algorithms and Computations in Mathematics*, 1996. Springer Verlag.

[37] J. Stern L. Granboulan, A. Joux. Inverting hfe is quasipolynomial. *LNCS 4117*. Proc of Crypto '06.

[38] H. Imai M. Sugita, M. Kawazoe. Relation between XL and Gröbner bases Algorithms. *http://eprint.iacr.org*. IACR preprint (2004).

[39] B. Mishra. Algorithmic algebra. *Springer (2003)*.

[40] T. Moh. The Method of Relinearization of Kipnis and Shamir and its Applications to TTM. 1999. available at http://citeseer.ist.psu.edu/371723.html.

[41] J. Pieprzyk N. Courtois. Cryptanalysis of Block Ciphers with Overdefinied Systems of Equations. *LNCS 2501*. Asiacrypt 2002.

[42] J-C.Faugere P. Loidreau, G. Ars. Comparison of xl and gröbner bases algorithms over finite field. *http://ensta.fr/ loidreau/CCA/ars.pdf*. Slide presentation.

[43] P. Paillier. Public-key cryptosystems based on discrete logarithms residues. *LNCS 1592*. Proc of Eurocrypt 1999.

[44] J. Patarin. Asymmetric cryptography with a hidden monomial. *LNCS 1109*. Proc of Crypto 1996.

[45] J. Patarin. Cryptanalysis of the matsumoto and imai public key scheme of eurocrypt '88. *LNCS 963*. Proc of Crypto 1995.

[46] J. Patarin. Hidden field equations (hfe) and isomorphisms of polynomials (ip):two new families of asymmetric algortihms. *LNCS 1070*. Proc of Eurocrypt 1996.

[47] A. Scemama. A Cryptanalysis of the Double-Round Quadratic Cryptosystem. *LNCS 4817*. Proc of ICISC 2007.

[48] C. P. Schnorr. Efficient Signature Generation for Smart Cards. *Journal of Cryptology, 1991,239-252*.

[49] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *Siam vol. 1484*. Proc of Siam 1997.

[50] V. Shoup. NTL 5.3.1, a library for doing Number Theory, 2003 . http://www.shoup.net/ntl .

[51] L. Hu X. Jiang, J. Ding. Kipnis-Shamir's Attack on HFE Revisited. *http://mirror.cr.yp.to/eprint.iacr.org/2007/203.pdf*. eprint.

[52] Y. Feng X. Tang. A new Efficient Algorithm for Solving Systems of Multivariate Polynomial Equations . *http://eprint.iacr.org*. eprint 2005.

Teil III

Zusammenfassung

Diese Arbeit widmet sich der ausführlichen Beschreibung zweier Ergebnisse, die ich während meiner Zeit in Frankfurt gefunden habe. Im Anhang A und B werden die Ergebnisse aus folgenden Artikeln zusammengefasst :

- Der erste Artikel stammt aus dem Bereich den "Multivariate Kryptographie". Er liefert eine Kryptanalyse eines Kryptosystems. Mein Artikel "A Cryptanalysis of the Double-Round Quadratic Cryptosystem" [47] wurde in der Konferenz **ICISC'07** veröffentlicht.

- Der zweite Teil der Arbeit ist ein Beitrag zum Bereich "Zero-Knowledge Proof of Knowledge". In dem noch nicht veröffentlichten Artikel "Improved Commitment Range Proof" verbessere ich die Effizienz des "Commitment Range Proofs".

A

Eine Kryptanalyse des 2R Kryptoschemas

Seit der Erfindung asymmetricher Kryptographie, wurde versucht, Kryptoschemata zu entwickeln, deren Sicherheit nicht auf der Faktorisierung ganzer Zahlen oder dem diskreten Logarithmus Problem beruht, sondern auf anderen schwierigen Problemen.
Als 1995 gezeigt wurde, dass die Faktorisierung und der diskrete Logarithmus mit Hilfe von Quantencomputern einfach zu berechnen sind, wurden solche Alternativen immer dringender.
Die "Multivariate Cryptography" versucht Kryptoschemata zu entwickeln, die auf dem folgenden NP-harten Problem beruhen :

- *Gegeben ein endlicher Körper* $\mathbb{F}_q$, $n, m \in \mathbb{Z}$ *und Polynome* $P_i \in \mathbb{F}_q[X_1, \cdots, X_n]$ *mit* $1 \leq i \leq m$, *von totalem Grad größer als* 2.

- *Finde* $(x_1, \cdots, x_n)$ *in* $(\mathbb{F}_q)^n$, *so dass :*

$$\begin{cases} P_1(x_1, \cdots, x_n) & = 0 \\ \vdots & \vdots \\ P_m(x_1, \cdots, x_n) & = 0 \end{cases}$$

Das $2R$ Kryptoschema

Das $2R$ Kryptoschemata, auch "Double-Round Quadratic Cryptosystem" gennant, wurde 1997 von Patarin erfunden [26]. Es funktioniert wie folgt :

- *Privat* : Drei invertierbare Matrizen A, B, C in $(\mathbb{F}_p)^{n\times n}$.
- *Öffentlich :*
  - Die Primzahl p mit $p = 3 \mod 4$, und $n \in \mathbb{N}$.
  - Eine "invertierbare" quadratische Funktion
    $$\begin{array}{rccc} \mathbf{f}: & \mathbb{F}_{p^n} & \longrightarrow & \mathbb{F}_{p^n} \\ & \mathbf{x} & \longrightarrow & \mathbf{x}^2 \end{array}$$
  - Polynome $P_1, \cdots, P_n$ in $\mathbb{F}_p[x_1, \cdots, x_n]$, so dass :
    $$(P_1, \cdots, P_n) = \pi^{-1}\mathbf{f}(\pi(B\pi^{-1}\mathbf{f}(\pi(A(x_1, \cdots, x_n))))$$ (A.1)

  Dabei ist π der (Vektorraum) Isomorphismus :
  $$\begin{array}{rccc} \pi: & (\mathbb{F}_p)^n & \longrightarrow & \mathbb{F}_{p^n} \\ & \bar{x} = (x_1, \cdots, x_n) & \longrightarrow & \mathbf{x} = \sum_{i=1}^n x_i\beta_i \end{array}$$
  $(\beta_1, \cdots, \beta_n)$ ist eine Basis von $\mathbb{F}_{p^n}$ als $\mathbb{F}_p$-Vektor Raum.
- *Verschlüsselung :* Sei $\bar{m} = (m_1, \cdots, m_n) \in (\mathbb{F}_p)^n$ ein Klartext. Der zugehörige Ziffertext ist :
  $\bar{c} = (P_1(m_1, \cdots, m_n), \cdots, P_n(m_1, \cdots, m_n))$.
- *Entschlüsselung :* Mit dem privaten Schlüssel berechnet man :
  $\bar{m} = A^{-1}\pi^{-1}\mathbf{f^{-1}}(\pi B^{-1}\pi^{-1}\mathbf{f^{-1}}(\pi(C^{-1}\bar{c})))$

Abbildung A.1: 2R Kryptoschema

Bemerkungen.

- Die zweimalige Quadrierung wurde eingeführt, in der Hoffnung, dass dies die Sicherheit früherer Kryptosystem mit einmaliger Quadrierung verbessert.
- Die P_i sind Polynome von totalem Grad 4 in den Variablen $x_1, \cdots, x_n$. Denn, die Funktion S' ist die Projektion von $\mathbf{f}$ über $(\mathbb{F}_p)^n$, sie ist quadratisch in den Variablen $x_1, \cdots, x_n$.

  $S'(\bar{x}) = \pi^{-1}(\mathbf{f}(\pi(\bar{x})))$ oder anders geschrieben,
  $$\begin{array}{rccc} S': & (\mathbb{F}_p)^n & \longrightarrow & (\mathbb{F}_p)^n \\ & (x_1, \cdots, x_n) & \longrightarrow & \sum_{i,j=1}^n s_{i,j}x_ix_j \end{array}$$
  Die $s_{i,j}$ sind aus den β_i leicht zu berechnen.

- Die Funktion $\mathbf{f}(\mathbf{x}) = \mathbf{x}^2$ ist kein Isomorphismus von $\mathbb{F}_{p^n}$. Aber es gilt : $\mathbf{f}^{-1}(\mathbf{x}) = \pm\mathbf{x}^{\frac{p^n+1}{4}}$, wegen $(\mathbf{x}^2)^{\frac{p^n+1}{4}} = \mathbf{x}^{\frac{p^n-1}{2}}\mathbf{x} = \pm\mathbf{x}$. Eine geeignete Formatierung der Nachrichten $\bar{m}$ erlaubt es, bei der Entschlüsselung das richtige Vorzeichen der Wurzel zu wählen.

- In solchen Kryptosystemen, kann ein Angreifer einen Ziffertext $\bar{c}$ entziffern, indem er das Polynomiale Gleichungsystem
  $\bar{c} = (P_1(x_1, \cdots, x_n), \cdots, P_n(x_1, \cdots, x_n))$ löst. Dieses Problem ist für beliebige Polynome NP-hart.
  Im Fall von Polynomen der Form wie in A.1, kann die Schwierigkeit des Problems nur vermutet werden.
  Diese Situation ist ähnlich bei RSA wo die Äquivalenz "RSA zu brechen" und die Zerlegung des RSA-modulus ist nicht bewiesen.

Eine Kryptanalyse von $2R$

1999 wurde eine erste heuristische Kryptanalyse veröffentlicht [17]. Wir beschreiben in unserer Arbeit diese Kryptanalyse ausführlich. Insbesondere, beschreiben wir die "exotische" Heuristisk die benutzt wird, und wir erklären, dass einige Argumente unklar sind. Unsere Analyse ist klarer, sie benutzt nur eine ausgewiesene Heuristik.

Unsere Kryptanalyse läuft wie folgt :

1. Aus der Gleichung A.1 kann man bemerken, dass die öffentliche Polynome $P_1, \cdots, P_n$ durch alternierende Transformationen zwischen $(\mathbb{F}_p)^n$ und $\mathbb{F}_{p^n}$ entstehen.
   Diese Alternierung löst unnötige Schwierigkeiten aus. Wir harmonisieren die Beschreibung des Kryptosystems mit Hilfe eines Theorems aus [35] :

   Theorem A.0.1 (Kipnis, Shamir 99) *Sei $M : (\mathbb{F}_p)^n \longrightarrow (\mathbb{F}_p)^n$ eine lineare Funktion. Es existieren $a_1, \cdots, a_n$ in $\mathbb{F}_{p^n}$, so dass $\bar{y} = M\bar{x}$ genau dann wenn $\mathbf{y} = \sum_{i=1}^{n} a_i \mathbf{x}^{p^i}$, für $\mathbf{x} = \pi(\bar{x})$ und $\mathbf{y} = \pi(\bar{y})$.*

   Das Theorem sagt, dass P_A, P_B, P_C in $\mathbb{F}_{p^n}[X]$ existiern, mit $P_A = \sum_{i=0}^{n-1} a_i \mathbf{x}^{q^i}$, $P_B = \sum_{i=0}^{n-1} b_i \mathbf{x}^{q^i}$, $P_C = \sum_{i=0}^{n-1} c_i \mathbf{x}^{q^i}$, $a_i, b_i, c_i \in \mathbb{F}_{p^n}$.
   So dass die folgende Gleichung gilt [1] :

$$P_C(P_B(P_A(\mathbf{x})^2)^2) = \pi(P_1(\pi^{-1}\mathbf{x}), \cdots, P_n(\pi^{-1}\mathbf{x}))) \qquad \text{A.2}$$

[1] Wir ersetzen in Gleichung A.1, $A\bar{x}$ (resp. $B\bar{x}$ und $C\bar{x}$) durch $\pi^{-1}P_A(\pi\bar{x})$ (resp. $\pi^{-1}P_B(\pi\bar{x})$ und $\pi^{-1}P_C(\pi\bar{x})$) .

Zudem zeigt der Beweis des letzten Theorems folgendes : gegeben P_A (resp. die Matrix A) ist es möglich die Matrix A (resp. das Polynom P_A) in polynomieller Zeit zu berechnen. Das selbe gilt auch für B und C.

Wir kryptanalysieren $2R$ dadurch, dass wir die Polynome P_A, P_B, P_C in polynomieller Zeit berechnen.

2. Die rechte Seite der Gleichung A.2 ist der öffentliche Schlüssel. Es gibt $p_{i_1,i_2,i_3,i_4} \in \mathbb{F}_{p^n}$ so dass :

$$\pi(P_1(\pi^{-1}\mathbf{x}), \cdots, P_n(\pi^{-1}\mathbf{x})) = \sum_{0 \leq i_1 \leq i_2 \leq i_3 \leq i_4 \leq n-1} p_{i_1,i_2,i_3,i_4} \mathbf{x}^{p^{i_1}+p^{i_2}+p^{i_3}+p^{i_4}}$$

Der Angreifer kann die p_{i_1,i_2,i_3,i_4} in polynomieller Zeit berechnen.

Wir wandeln die Gleichung A.2 um :

$$P_B(P_A(\mathbf{x})^2)^2 = (P_C)^{-1}(P_{public}(\mathbf{x})) = \sum_{t=0}^{n-1} \sum_{0 \leq i_1 \leq i_2 \leq i_3 \leq i_4 \leq n-1} c_t^{'} p_{i_1,i_2,i_3,i_4}^{p^t} \mathbf{x}^{p^{i_1+t}+p^{i_2+t}+p^{i_3+t}+p^{i_4+t}} \quad \text{A.3}$$

wobei $(P_C)^{-1} = \sum_{t=0}^{n-1} c_t^{'} \mathbf{x}^{p^t}$ (Folge des Theorems).

Aus der obigen Form für P_A, P_B folgt die Existenz von $b_{i,j}$ in $\mathbb{F}_{p^n}$, so dass :

$$P_B(P_A(\mathbf{x})^2) = \sum_{0 \leq i_1 \leq i_2 \leq n-1} b_{i_1,i_2} \mathbf{x}^{p^{i_1}+p^{i_2}} \quad \text{A.4}$$

Also gilt die folgende Gleichung :

$$\left(\sum_{0 \leq i_1 \leq i_2 \leq n-1} b_{i_1 i_2} \mathbf{x}^{p^{i_1}+p^{i_2}}\right)^2 = \sum_{t=0}^{n-1} \sum_{0 \leq i_1,i_2,i_3,i_4 \leq n-1} c_t^{'} p_{i_1,i_2,i_3,i_4}^{p^t} \mathbf{x}^{p^{i_1+t}+p^{i_2+t}+p^{i_3+t}+p^{i_4+t}} \quad \text{A.5}$$

3. Die Gleichung A.5 drückt die Gleichheit zweier Polynome in der Variablen $\mathbf{x}$ (mit den $b_{i,j}$ und $c_t^{'}$ als Unbekannten) aus. Das bedeutet, dass alle Monome $\mathbf{x}^{p^{i_1+t}+p^{i_2+t}+p^{i_3+t}+p^{i_4+t}}$ links und rechts gleich sind.
Das liefert dem Angreifer ein System von $O(n^4)$ quadratischen Gleichungen über $\mathbb{F}_{p^n}$ in $O(n^2)$ Unbekannten ($b_{i,j}$ und $c_t^{'}$).

Wichtig ist, dass es viel mehr quadratische Gleichungen als Unbekannte gibt.

Es wurde unter standard Heuristik bewiesen [35], dass es möglich ist, ein System von ϵm^2 quadratischen Gleichungen und m Unbekannten (in polynomieller Zeit) zu lösen, solange ϵ größer als 0.1 ist.

In unserem Fall zeigt eine präsize Berechnung, dass $\epsilon = 0,17$.
Das heisst, der Angreifer kann die $b_{i,j}$ und $c_t^{'}$ berechnen. Danach kann er aus den $c_t^{'}$ die Koeffizienten c_i des Polynoms P_C berechnen und daraus die Matrix C.

4. Mit der Matrix C kann man nach derselben Methode die Matrizen A und B berechnen, und somit den privaten Schlüssel rekonstruiren.

B

Eine verbesserte "Commitment Range Proof"

Ein *"zero-knowledge proof of knowledge"* ist ein Protokoll zwischen einem Prover und einem Verifier. Durch das Protokoll, zeigt der Prover dem Verifier, dass er einen geheimen Wert kennt, ohne Informationen über diesen Wert preiszugeben.

Die bekanntesten und am meisten benutzten *"zero-knowledge proof of knowledge"* beruhen auf dem diskreten Logarithmus Problem. Typischerweise, wählt der Prover eine grosse Primzahl q, und ein Element g in $\mathbb{Z}_q^*$ mit großer Ordnung. Dann, wählt der Prover ein x und berechnet $h = g^x \mod q$. Somit ist x der diskrete Logarithmus von h zur Basis g.[1]
Der Prover übermittelt dem Verifier g, h, q und beginnt das Protokoll. Am Ende, ist der Verifier überzeugt, dass der Prover x kennt.

Ein solches Protokoll ermöglicht dem Verifier sich elektronisch zu identifizieren. Denn, das Problem aus gegebenen h, g, q, x zu finden ist schwierig. Ferner liefern die im Protokoll übertragene Daten keine Informationen über x.

In manchen kryptographischen Protokollen (electronic cash systems, group signature, verifiable secret sharing, und andere zero-knowledge proofs) muss der Prover nicht nur beweisen, dass er den diskreten Logarithmus x kennt, sondern auch, dass x in einen bestimmten Intervall $[a, b]$ liegt.
Es soll also der Verifier davon überzeugt werden, dass der Prover x kennt, und dass $x \in [a, b]$, ohne weitere Informationen über x preiszugeben. Solche Protokolle heißen "Commitment Rang Proof" oder "Range Bounded Commitment".

Der bis jetzt effizienteste "Commitment Range Proof", stammt von Boudot [6].

[1] Im diskreten Logarithmus Problem sucht man zu gegebenen h, g, q eine Lösung $x \in \mathbb{Z}$ der Gleichung $h = g^x \mod q$

In Boudots Methode, berechnet der Verifier $g^{x-a} \mod q$ (er ist in der Lage dies zu machen, da er g^x, a, q kennt).
Der Prover überzeugt ihn, dass $x > a$ indem er zeigt, dass der diskrete Logarithmus von g^{x-a} (zur Basis g) positiv ist.
Danach, zeigt der Prover dem Verifier mit der selben Methode, dass $x < b$.
Diese Methode erfordet, dass die Ordnung von $< g >$ dem Prover unbekannt bleibt. Es existieren mehrere Methoden, um diese Voraussetzung zu erfüllen.

Unsere neue Methode basiert auf der von Boudot. Anstatt zuerst $x > a$ und danach $x < b$ zu beweisen, zeigen wir in einem Schritt, dass $x \in [a, b]$. Daraus folgt eine verbesserte Effizienz, angesichts der Anzahl der Operationen (Multiplikationen in $\mathbb{Z}_q$) im Protokoll und der Größe der ausgetauschten Zahlen.

Die Ideen, die wir benutzen sind folgenden : sei $a < b$,

- Es gilt $x > a$ und $x < b$ genau dann, wenn die Vorzeichen von $x - a$ und $x - b$ verschieden sind.
- Die Vorzeichen von $x - a$ und $x - b$ sind verschieden genau dann, wenn das Produkt $(x - a)(x - b)$ negativ ist.

Nun also zu beweisen, dass $a < x < b$ gilt, beweist der Prover, dass $(x - a)(x - b) < 0$.

Dazu, schickt der Prover dem Verifier $h' = g^{-(x-a)(x-b)} \mod q$. Mit Boudots Methode beweist er, dass der diskrete Logarithmus zur Basis g von $g^{-(x-a)(x-b)} \mod q$ positiv ist.

Abschließend muß der Prover dem Verifier zeigen, dass der gesendete Wert h' wirklich $g^{-(x-a)(x-b)} \mod q$ entspricht, denn, der Verifier kennt nur $g_a = g^{x-a} \mod q$ und $g_b = g^{x-b} \mod q$, und kann nicht selbst $g^{-(x-a)(x-b)} \mod q$ berechnen. Zudem darf er nicht einfach dem Prover vertrauen, dass h' korrekt ist.

Dazu, muss der Prover dem Verifier überzeugen, dass die folgende Gleichung gilt :

$$-\log_{g_a}(h') = \log_g(g_b)$$

Ein Protokoll für einen solchen Beweis existiert und wurde schon in [30] veröffentlicht.

Nach präzisen Berechnungen stellen wir fest, dass die neue Methode ungefähr 25% effizienter ist, als die von Boudot. Zudem kann man mit diesem

Protokoll beweisen, dass x nicht in einem Intervall liegt, was mit Boudots Methode nicht möglich ist.