A Survey of Attacks on Multivariate Cryptosystems

by

Adam Thomas Feldmann

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2005

©Adam Thomas Feldmann 2005

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis provides a survey of the attacks on multivariate cryptosystems. We begin by providing an outline of the general multivariate cryptosystem. Proceeding from there, we show that even with this level of detail, there are several attacks that are possible, including the method of Gröbner bases, the XL method, and the recently announced method of Dixon resultants. Less general attack techniques also exist, such as MinRank attacks and differential analysis. These attacks lack the universality of the first three mentioned. In order to explore these less general attacks further, more details are required, so we present four different multivariate cryptosystems. Then, we attack them, using the less general attacks of MinRank, differential analysis and even an attack specific to one system. This concludes our study of the attacks themselves, and we move on to note that not all routes of attack are promising. Specifically, quantum computing does not seem to be helpful beyond the quadratic speed-up of Grover's algorithm. We also note that not all multivariate cryptosystems have been successfully attacked as of the writing of this thesis. We conclude with the fact that multivariate cryptography is gaining more and more active study.

Acknowledgments

I would like to thank the following people for their help and insights. Thanks to Christopher Wolf and Xijin Tang, for their insights in personal correspondence. Scott Aaronson gave me a concise overview of quantum computing, for which I am grateful. Thanks to Jacques Patarin, for sending a modern version of one of his papers my way. Also, I would like to thank Arthur D. Chtcherba, for telling me what RSC stands for, and Nicolas T. Courtois, for answering a question about HFE security. I would like to thank my readers, Alfred Menezes and Douglas Stinson, for their comments. Last, but certainly not least, I would like to thank my advisor Edlyn Teske, whose support and advice made this thesis far better than I could have done alone.

Contents

List of Tables

Chapter 1

Introduction

Multivariate public key cryptosystems are generally traced back to Matsumoto and Imai's 1988 paper [MI88], which seems to be the one which popularized the idea. In it, Matsumoto and Imai introduced a public key encryption scheme called MIC*, which utilized an instance of what is known as the $\mathcal{MQ}$ problem for its security. The MIC* scheme was successfully broken by Patarin in a 1995 paper [Pat95, Pat05], who went on to produce his own $\mathcal{MQ}$-based public key encryption scheme, HFE, in a 1996 paper [Pat96b]. From there, research has continued, and quite a few schemes and attacks on them have been created [Din04, WP04, DG05, KS99, CKPS00, BWP04, FGS05, DGS$^+$05].

This thesis presents a survey of attacks on multivariate public key cryptosystems. For brevity, we will drop the "public key" portion of that term. It details attacks against the earliest multivariate cryptosystems [Pat95, Pat05, KS99], while also including the latest developments in multivariate cryptography, both in cryptosystems and attacks [TF05, FGS05, DG05, DGS$^+$05].

In order to do this survey of attacks efficiently, Chapter 2 provides the required level of setup to understand the remainder of the text. First, it introduces the problems upon which the security of multivariate public key cryptosystems relies. Then, it sets up a generic multivariate public key cryptosystem. The terminology and conventions of Chapter 2 will continue throughout the text, so even those already familiar with multivariate cryptosystems will want to peruse it. A reader who desires a more thorough grounding in multivariate cryptosystems before dealing with attacks may wish to skip ahead to Chapter 4, where four such systems are described in detail. For those people, it should be noted that Chapters 3 and 4 are independent of each other, and either may be read first without loss of understanding.

Once these preliminaries are taken care of, we move on to actual attacks. Chapter 3 is devoted to surveying the various techniques used to attack multivariate cryptosystems. We explain five different techniques used to attack multivariate cryptosystems. Each attack is presented so as to give the reader an understanding of how and why the attack works in addition to noting that it does work (although perhaps not so well as one might hope).

The first such attack technique is the method of Gröbner bases (see Section 3.1). In order to properly present this method, first the Gröbner basis is defined, which requires a bit of work. Next we show that a Gröbner basis can be calculated by showing that Buchberger's algorithm to compute a Gröbner basis successfully completes. Finally, we explain the value of a Gröbner basis in attacking multivariate cryptosystem. Thus, the reader need not peruse such tomes as [CLO92, CLO98] in order to understand the benefits and value of a Gröbner basis—the details important to a cryptanalyst attacking multivariate cryptosystems are contained in Section 3.1.

The second such attack technique is the XL method (see Section 3.2). Introduced by Courtois *et al.* [CKPS00], the XL method is presented in this thesis primarily for completeness. It is an attack technique that can be shown to be a kind of Gröbner basis attack. However, this was shown only very recently, by Ars *et al.* [AFI+04]. So that the interested reader can more easily understand papers written prior to [AFI+04], it makes sense to include the XL method.

Moving past the Gröbner basis we proceed to the method of Dixon resultants (see Section 3.3). The method of Dixon resultants was first applied to attacks on the AES cryptosystem by Tang and Feng [TF05]. Their AES attack works by representing the AES cryptosystem as a large system of multivariate polynomials, and attempting to find the common zeroes of this system. It has been adapted in this thesis in order to function directly as an attack on multivariate cryptosystems. In order to properly explain the attack, the concept of the Dixon resultant is introduced, and then the steps required to extend the Dixon resultant to be useful in a Dixon resultant-based attack on a multivariate cryptosystem. Finally, the DR algorithm of Tang and Feng [TF05] is presented, which uses the calculation of an extended Dixon resultant in order to attack a multivariate quadratic cryptosystem.

The last two attacks presented in Chapter 3 are somewhat different from the first three attacks. These last two attacks are the MinRank attack and the technique of differential analysis (see Sections 3.4 and 3.5, respectively). The MinRank attack technique is harder to quantify than an attack based on calculating a Gröbner basis or an extended Dixon resultant, because no MinRank-based attack exists that will work for any multivariate quadratic cryptosystem. What Section 3.4 does explain, however, is the idea behind such an attack. In Chapter 5, there are two different MinRank attacks presented, showing how even in different multivariate cryptosystems, a MinRank attack is possible. However, unlike the previous attacks, there is no simple algorithmic way to apply a MinRank-based attack to a given multivariate cryptosystem.

The last attack presented is the technique of differential analysis (see Section 3.5). This technique is also quite new, introduced by Fouque, Granboulan and Stern [FGS05] in a 2005 paper. They successfully applied it to several systems in that paper. It is presented in Chapter 3 as a technique that may work in many situations, but similar to the MinRank attack there is no known way to universally state what one must do to attack a multivariate cryptosystem with a differential analysis attack usefully. Later, the technique of differential analysis is applied to the multivariate cryptosystem PMI in Section 5.5 with efficacious

results.

Chapter 3 concludes with a note that not all attacks fall into one of the attack techniques listed in that chapter. Techniques exist which do not cleanly generalize to other systems and are therefore specific. One such attack is the attack of Patarin [Pat95, Pat05] on the MIC* multivariate cryptosystem, which is presented in Section 5.4.

This leads naturally to the question of what a multivariate cryptosystem actually looks like, so that a more specific attack may be attempted. Chapter 4 elucidates the general multivariate cryptosystem by providing four different examples of it—the STS family, basic HFE, MIC*, and Perturbed MIC* (or PMI). These example systems are presented in order to clarify several important points to the reader. First, no thesis involving multivariate cryptosystems would be complete without providing the reader with at least one functional multivariate cryptosystem as an example. Second, showing four cryptosystems helps the reader see that all multivariate cryptosystems have an essential sameness to them, as noted in Chapter 2—encryption is nearly identical between the systems, and decryption differs between the systems only in one place (see Section 2.3.2 for details). Third, each cryptosystem introduced is then used in Chapter 5 in order to explain an attack against that system.

The cryptosystems introduced in Chapter 4 run the gambit from old to new. The MIC* cryptosystem, due to Matsumoto and Imai [MI88], is one of the oldest proposed multivariate cryptosystems. Basic HFE was introduced by Patarin [Pat96b] as a way to fix the MIC* cryptosystem. The STS family is a generalization of cryptosystems proposed by Kasahara and Sakai [KS04b, KS04a]. Wolf, Braeken and Preneel [WBP04] introduced the STS family themselves in the very paper that attacked that family (and therefore also the schemes of Kasahara and Sakai). Finally, in 2004, the PMI cryptosystem was introduced by Ding [Din04].

Chapter 5 provides the answer to another question naturally arising from the end of Chapter 3—namely, "What does a specific attack against a multivariate cryptosystem look like?" One such attack is explained in detail in Section 5.4, as promised in the last section of Chapter 3. Chapter 5 has a larger purpose than explaining single specific attacks, however. Its goal is to provide concrete examples of attacks that could not have occured in Chapter 3 without obscuring the main point of the attack technique with a specific cryptosystem's details. Additionally, the attacks of Chapter 5 serve the purpose of further detailing the mechanics of the MinRank and differential analysis techniques of Chapter 3.

The first two attacks of Chapter 5 are attacks on the STS family and the Basic HFE system, as described in Chapter 4. The attack on STS, due to Wolf, Braeken and Preneel [WBP04] makes use of a MinRank attack. The Basic HFE attack, due to Kipnis and Shamir [KS99], and cleaned up slightly by Courtois [Cou01], also uses a MinRank attack. By viewing both attacks, it can easily be seen that a MinRank attack may take a very different form from one cryptosystem to the next.

The remaining two attacks are even more directly connected. The specific attack of Patarin [Pat95, Pat05] against the MIC* cryptosystem is presented

next. After this, a differential analysis attack against the PMI cryptosystem is presented. The PMI cryptosystem is related to the MIC* cryptosystem. Because of this connection, the differential analysis attack on the PMI cryptosystem incorporates the Patarin attack on MIC*. Additionally, it should be noted that the attack on PMI, due to Fouque, Granboulan and Stern [FGS05] is from a 2005 paper, and is therefore quite recent.

Chapter 6 is a step away from the main thrust of this thesis. While most of our effort has gone towards finding and elucidating successful attacks on multivariate cryptosystems, Chapter 6 introduces material about unsuccessful attacks. First and foremost, there is an explanation for a remarkable fact about multivariate cryptosystems—quantum computing does not seem to be able to decrypt a ciphertext encoded with a multivariate scheme in polynomial time. On the other hand, Shor's algorithm [Sho97] makes polynomial time decryption of RSA and elliptic curve-based cryptosystems on quantum computers a reality—provided such machines are ever constructed. Also, some systems that resist all known attacks are presented briefly.

Chapter 7 concludes the thesis with some suggestions for areas where further work would be useful in order to improve existing attacks, and also in order to create more secure multivariate cryptosystems. We note in Section 7.2 that multivariate cryptosystems are gaining more recognition and more study now than ever, judging by the number of recent papers versus the number of older papers in the field, and the speed at which the latest multivariate cryptosystems are created, attacked and patched.

Chapter 2

Multivariate Cryptography in Brief

Multivariate public key cryptography uses systems of multivariate quadratic equations as the public key and part of the private key. Thus, multivariate public key cryptosystems are in fact multivariate quadratic public key cryptosystems. We will usually omit the phrase "public key" for brevity. For the purposes of this paper, it will be assumed that all such equations occur over finite fields (although it is possible to define the problem in more generic rings). These systems are studied because of two problems that are believed to be hard. The first is called the $\mathcal{MQ}$ problem, where the $\mathcal{MQ}$ stands for multivariate quadratic. The second is called the Isomorphism of Polynomials problem (IP problem). $\mathcal{MQ}$-based encryption schemes are encryption schemes that rely on the difficulty of solving an apparently random instance of the $\mathcal{MQ}$ problem and the IP problem for their security.

2.1 The $\mathcal{MQ}$ Problem

The $\mathcal{MQ}$ problem over a finite field $\mathbb{F}_q$ (where q is a prime power) is finding a solution $x \in \mathbb{F}_q^n$ to a given system of m quadratic polynomial equations $y = (p_1, \ldots, p_m)$ over $\mathbb{F}_q$ in n indeterminates. That is, we wish to solve

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

for a given $y = (y_1, \ldots, y_m) \in \mathbb{F}_q^m$ and unknown $x = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_q^n$. True to the term quadratic, in the above system of equations, the polynomials p_i

have the form

$$p_i(x_1, \ldots, x_n) := \sum_{1 \le j \le k \le n} \gamma_{i,j,k} x_j x_k + \sum_{j=1}^{n} \beta_{i,j} x_j + \alpha_i$$

for $1 \le i \le m$; $1 \le j \le k \le n$ and $\alpha_i, \beta_{i,j}, \gamma_{i,j,k} \in \mathbb{F}_q$ (the constant, linear and quadratic coefficients, respectively). It has been shown that over a finite field, this problem is $\mathcal{NP}$-hard [PG97].

2.2 The Isomorphism of Polynomials problem

Given a pair of (not necessarily quadratic) polynomial vectors, $\mathcal{P}$ and $\mathcal{P}'$, where each vector is an m-tuple of polynomials over the same set of n indeterminates as in the previous section, the IP problem is to find a pair of affine transformations T and S such that

$$\mathcal{P} = T \circ \mathcal{P}' \circ S.$$

Thus, $T \in \text{Aff}(\mathbb{F}_q^m)$ and $S \in \text{Aff}(\mathbb{F}_q^n)$. They are therefore representable by the pairs of terms T_ℓ and T_c, and S_ℓ and S_c, respectively, where T_ℓ and S_ℓ are invertible $m \times m$ (resp. $n \times n$) matrices with elements from $\mathbb{F}_q$. Similarly, T_c and S_c are column vectors over $\mathbb{F}_q$ of lengths m and n respectively. Thus, $S(x)$, where $x = (x_1, x_2, \ldots, x_n)$ is given by $S(x) = S_\ell x + S_c$, viewing x as a column vector, and using standard matrix multiplication for $S_\ell x$ and standard matrix addition for the summation.

The difficulty of the IP problem is equivalent to solving for the unknowns of S and T using the entries of $\mathcal{P}$ and $\mathcal{P}'$ as constants. This is a problem quite similar to the $\mathcal{MQ}$ problem. Now, however, the unknown values are the values of T and S, while the values for $\mathcal{P}$ and $\mathcal{P}'$ are given. Solving such a problem reduces to solving a set of equations in the values of T and S of total degree one larger than the total degree of $\mathcal{P}'$. So, when solving problems in the situation of $\mathcal{MQ}$-based encryption schemes, the total degree of these equations is 3, in the unknown entries of T and S, namely, the unknown entries of the matrices T_ℓ, T_c, S_ℓ, and S_c. It can be shown that this problem is $\mathcal{NP}$-hard [PGC98].

Here is a simple example of an IP problem. Let $m = 1$ and let $n = 1$. Let $q = 3$, so we are working in $\mathbb{F}_3$. Let $\mathcal{P}' = x^2 + x$ and let $\mathcal{P} = x^2 + x + 1$. Then we wish to find T and S such that

$$x^2 + x + 1 = (T \circ \mathcal{P}' \circ S)(x).$$

Since we know that here $T(x) = t_1 x + t_2$, and $S(x) = s_1 x + s_2$, for indeterminates $t_1, t_2, s_1, s_2 \in \mathbb{F}_2$, we have the following equation to solve:

$$x^2 + x + 1 = t_1 \left((s_1 x + s_2)^2 + s_1 x + s_2 \right) + t_2.$$

The indeterminates can be determined by solving the following three equations, determined by setting the coefficients of $x^2, x, 1$ equal on each side of the previous

equation:

$$\begin{aligned} 1 &= s_1^2 t_1 \\ 1 &= 2s_1 s_2 t_1 + s_1 t_1 \\ 1 &= s_2^2 t_1 + s_2 t_1 + t_2. \end{aligned}$$

Note that these equations are of degree 3, one higher than the total degree of $\mathcal{P}'$, as above. There are two solutions. The first equation gives us the fact $t_1 = 1$ and $s_1 \neq 0$. If we choose $s_1 = 1$, then $s_2 = 0$ and $t_2 = 1$. If we choose $s_1 = 2$, then $s_2 = 1$ and $t_2 = 1$.

2.3 A Generic $\mathcal{MQ}$-based encryption scheme

Now that the basic problems underlying $\mathcal{MQ}$-based encryption schemes have been divulged, let us create a generic outline of how such a cryptosystem is constructed.

2.3.1 Key generation

The private key of a generic $\mathcal{MQ}$-based encryption scheme is an ordered set (T, $\mathcal{P}'$, S), following the same conventions as previous sections in this chapter. T and S are determined in the same way, so only the T case will be considered in detail. T is to be an invertible affine transformation on $\mathbb{F}_q^m$. Thus, by finding T_ℓ and T_c, an invertible $m \times m$ matrix and a column vector of length m respectively, T is totally determined. So, using a cryptographically secure pseudorandom number generator over $\mathbb{F}_q$, continue to determine random matrices in $\mathbb{F}_q^{m \times m}$ until one is found with nonzero determinant. This is an expected constant-time operation, and its output will be T_ℓ. Then choose a column vector of length m by choosing m elements from $\mathbb{F}_q$ with the random number generator. This determines T_c.

Determining $\mathcal{P}'$ is somewhat more difficult. $\mathcal{P}'$ is always a vector of quadratic polynomials. However, every $\mathcal{MQ}$-based encryption scheme embeds some sort of trapdoor into the choice of $\mathcal{P}'$, so that for a known output y of the function $\mathcal{P}'$, the preimage of $\mathcal{P}'$ is easily computable. It is simplest to choose an invertible $\mathcal{P}'$. By building a small amount of redundancy into messages sent (usually by concatenating part or all of a message's hash to the message proper), it is possible for non-invertible $\mathcal{P}'$ functions to be used as well. Generally, $\mathcal{P}'$ is chosen uniformly at random from some appropriate space of quadratic functions.

The public key $\mathcal{P}$ is given by

$$\mathcal{P} = T \circ \mathcal{P}' \circ S$$

where one notes that as $\mathcal{P}'$ is quadratic and the functions T and S are affine, the total degree of $\mathcal{P}$ is also 2.

2.3.2 Encryption and decryption

Encryption is accomplished by taking a message x and embedding it into $\mathbb{F}_q^n$, possibly with the redundancy required by the choice of the key, and running it through the public key, forming $y = \mathcal{P}(x)$. The decryption process varies slightly with the particular trapdoor built into $\mathcal{P}'$, but the general procedure is to take a given ciphertext y, and convert it to the plaintext by using the secret key (T, $\mathcal{P}'$, S) to invert $\mathcal{P}$ over the message space, recalling the possibility of message redundancy to ensure this is possible. T is known to be invertible, so as a first step, we get

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

Then we find the set of preimages of $\mathcal{P}'$,

$$\mathcal{P}'^{-1}(T^{-1}(y)).$$

Then, each element $\tau \in \mathcal{P}'^{-1}(T^{-1}(y))$ is given as input to S^{-1}, which exists by definition of S. The exact redundancy specified by the encryption scheme will almost certainly occur in only one such preimage $S^{-1}(\tau)$. Remove the redundancy to retrieve the decrypted message in $\mathbb{F}_q^n$. In the special case of invertible $\mathcal{P}'$, one simply applies the three inverses T^{-1}, $\mathcal{P}'^{-1}$, and S^{-1} to the ciphertext y in order, then by recovering the plaintext $x = S^{-1} \circ \mathcal{P}'^{-1} \circ T^{-1}(y)$. Note that the only changing factor when encrypting or decrypting is the nature of the trapdoor built into $\mathcal{P}'$, which determines whether or not redundancy is required, and the exact technique used to find the preimage $S^{-1}(\tau)$.

2.3.3 Security

A fundamental requirement for security is that the private key be difficult to obtain from the public key. Therefore, the IP problem must be difficult. There exist families of $\mathcal{MQ}$-based schemes that have relatively few possible choices of private key value $\mathcal{P}'$ (for example, the Matsumoto and Imai C* scheme of [MI88]). Therefore, if the IP problem is easy, simply solving it over and over again for the possible $\mathcal{P}'$ values will eventually return the remainder of the private key, breaking the system.

Similarly, the $\mathcal{MQ}$ problem must also be kept difficult for security reasons. If the $\mathcal{MQ}$ problem is easily solved, then for any ciphertext y, the associated plaintext x can be efficiently computed by solving an instance of the $\mathcal{MQ}$ problem.

2.3.4 Efficiency

The efficiency of an $\mathcal{MQ}$-based scheme can be seen from multiple viewpoints. Courtois [Cou05] notes that for the $\mathcal{MQ}$-based scheme HFE (see Section 4.3 for details), the security of HFE for 80-bit messages can be made equivalent to the security of RSA for 512-bit messages—after all, the security parameters of HFE are the length of the message n (assuming that the base field is $q = 2$) and an

additional security parameter d (see Section 4.3.1 for the definition of d), while the RSA cryptosystem's security rests solely on its modulus (i.e., the message length). It is also noted that while the best attack on HFE operates in $O(e^{\ln^2 n})$ operations, where n is the length of the plaintexts, the number field sieve attack on RSA operates in

$$O(e^{(\frac{64}{9})^{1/3}(\log 2^n)^{1/3}(\log\log 2^n)^{2/3}})$$

(or, in L-function notation, $L_{2^n}(\frac{1}{3}, (\frac{64}{9})^{1/3})$) operations.

Let us attempt to compare security levels of RSA and HFE in the more usual way (as opposed to Courtois's note in the previous paragraph). So, for a given equivalent security level, we wish to determine the key sizes and encryption and decryption speeds of each system. Even given that the most powerful attack on HFE operates in $O(n^{10})$ base field operations, which comes from assuming that the estimates of Faugère [Fau03] hold for all n (a very generous assumption), we can choose, as suggested by Courtois [Cou05], $q = 2$, $n = 251$, and $d = 25$, and achieve an estimated security level of 2^{80} base field operations. Meanwhile, as a 1024-bit RSA key is equivalent to an 80-bit symmetric key, according to Kaliski [Kal03], we have roughly equivalent security for those parameter choices. We will now compare key sizes, encryption and decryption efficiency at this comparable security level.

The public key of RSA is a pair of integers. The public key of an $\mathcal{MQ}$-based scheme is a system of quadratic multivariate equations, stored as their associated set of coefficients. So, at an equivalent security level, a 1024-bit RSA system will use at most 2 kilobits for a public key—1024 bits for each number (and the encryption exponent can be chosen to be significantly shorter than 1024 bits). A multivariate cryptosystem stores $n(n-1)/2+n+1$ coefficients per equation, and has m equations. Note that in the case of HFE, $m = n$, and there are exactly as many equations as there are indeterminates. Each coefficient requires $\lceil \log_2(q) \rceil$ bits of storage. In the case of HFE, the recommendation is for $q = 2$, so we use $\approx \frac{n^3}{2}$ bits, or ≈ 7721 kilobits. RSA uses orders of magnitude less space for its public key at this security level.

The private key for RSA is approximately the same size as its public key—a 1024-bit RSA system will use at most 2 kilobits for a private key, then. The private key for an $\mathcal{MQ}$-based scheme consists of $\approx n^2$ field elements for each of T and S, as well as the storage required for $\mathcal{P}'$. In the case of HFE, this storage space required for storing $\mathcal{P}'$ is the same as the storage space required for one multivariate equation, so it is also $\approx n^2$. So, the total storage requirements for an HFE scheme are $3n^2$. At the same security level as RSA, that totals to 185 kilobits. Again, RSA uses orders of magnitude less space for the private key than HFE.

The efficiency of encryption in any $\mathcal{MQ}$-based scheme can easily be compared to that of RSA. To evaluate a randomly generated multivariate equation in n indeterminates requires $O(n^2)$ field multiplications and as many field additions. Thus, for m equations, a total of $O(mn^2)$ field operations are required. If the base field is $\mathbb{F}_2$ (which is recommended for HFE) and $m = n$ (which is

always true for HFE), then we have approximately $O(n^3)$ bit operations, or a number of bit operations cubic in the length of a single message. Encryption in RSA is effectively an $O(n^2)$ operation, as the encryption exponent is generally small. So, encryption is faster in RSA than in an $\mathcal{MQ}$-based scheme for equivalently-sized messages, as modular exponentiation is merely quadratic in the length of the modulus. Returning to the case of systems with equivalent security (such as this 251-bit HFE versus 1024-bit RSA), we can make a direct comparison. To send the same amount of data, the HFE system must send multiple messages, each message is encrypted in approximately 2^{24} bit operations with in this instance of HFE and approximately 2^{20} bit operations in RSA. These numbers are not too far apart, even if 8 or so messages (including space for redundancy requirements in HFE) are required to send the same data in the HFE instance. Unfortunately, HFE scales very poorly. In order to achieve the $\approx 2^{112}$ bit operations of security needed to reach the approximate security level of a 2048-bit RSA key (as noted in [Kal03]), we must use $n \approx 2048$ by our assumptions in this section, and the speed difference only increases, making RSA a better and better choice.

Decryption in the realm of RSA is approximately the same operation as encryption (a modular exponentiation), while decryption of an $\mathcal{MQ}$-based scheme can involve finding a set of pre-images for the trapdoor function, which can be quite costly. Courtois [Cou05] notes that HFE requires $O(n^4 d^2 \log(d))$ operations to do so, recalling that it has 2 security parameters n and d (see Section 4.3.1 for the definition of d). For our given HFE cryptosystem of $q = 2$, $n = 251$ and $d = 25$, we must compare $O(n^4 d^2 \log(d))$, the dominating cost of decryption in HFE to $O((4n)^3)$, which is the approximately the cost of decryption in 1024-bit RSA for $n = 251$. Since $4^3 = 64$, we expect that the cost $O(n^4 d^2 \log(d))$ will be much higher than $O((4n)^3)$. In general, as n increases, this will only get worse. Even if d is ignored in $O(n^4 d^2 \log(d))$, we have a decryption quartic in the message length compared to a decryption cubic in the message length. Again, as HFE scales very poorly, the 2048-bit RSA to 2048-bit HFE equivalent security level noted in the paragraph on encryption means that decryption is more than an order of magnitude more painful in the next step up, security-wise, for HFE.

In conclusion, HFE has some valid choices of security parameters, but it very quickly becomes too slow to use when RSA is available. Also, HFE uses copious amounts of memory for its public and private keys, while RSA uses a more reasonable amount. But it should be noted that there are parameter choices for HFE that are reasonably secure, as shown in this section. One might also wonder why HFE, or any $\mathcal{MQ}$-based encryption scheme is studied at all. See Section 7.1.2 for a brief defense of the schemes.

2.3.5 $\mathcal{MQ}$-Based signature schemes

It should be noted that there exist signature schemes that rely on the difficulty of the $\mathcal{MQ}$ problem and the IP problem for their security. These are denoted $\mathcal{MQ}$-based signature schemes. We do not discuss them in this thesis. The

interested reader may look to the taxinomy of Wolf and Preneel [WP04] for the workings of $\mathcal{MQ}$-based signature schemes.

Chapter 3

Attacking Multivariate Cryptosystems

The most basic kind of attack on any cryptosystem is to find a plaintext x for a given ciphertext y. In the case of multivariate cryptography, the intention is to make solving this problem equivalent to solving an instance of the $\mathcal{MQ}$ problem over a finite field. While this problem is $\mathcal{NP}$-hard, there are still techniques that are often helpful in practice.

This chapter presents five different attacks, and these attacks may be divided into two groups. The first three attacks of this chapter are based on Gröbner bases, the XL method and Dixon resultants, respectively (see Sections 3.1, 3.2, and 3.3 for details). These attacks all have one thing in common—they solve an $\mathcal{MQ}$ problem without using any information beyond the public key itself. So, they do not take advantage of an attacker having knowledge of the specific $\mathcal{MQ}$-based encryption scheme used. The last two attacks, MinRank attacks and Differential Analysis (see Sections 3.4 and 3.5 for details) require the use of such information, and are therefore less general, and not universally applicable.

3.1 Gröbner bases

The first attack method under consideration is the method of Gröbner bases. Buchberger [Buc65] both coined the term Gröbner basis (after his thesis advisor, Wolfgang Gröbner) and created the Buchberger algorithm [Buc65, vzGG03] to calculate them. Gröbner bases are used in a generic attack possible for any multivariate cryptographic scheme. The idea of the attack is to take a given ciphertext y and solve the following polynomial system

$$\mathcal{P}(x) - y = 0$$

for the plaintext x by forming a Gröbner basis of the ideal generated by the polynomials $\mathcal{P}(x) - y$. It shall be seen that the properties of the Gröbner basis calculated in this process should make finding x straightforward.

3.1.1 Defining a Gröbner basis

The basic definitions from this section were culled from [vzGG03]. Consider first the case of univariate polynomials. Let $\mathbb{F}$ be a field. When considering an ideal $I \subset \mathbb{F}[x]$, where x is a univariate indeterminate, the question, "Is the polynomial g contained in I?" is easy to answer. The technique normally used is simply polynomial division of $g(x)$ by the unique monic generating polynomial $i(x)$ of I; as $\mathbb{F}[x]$ is a euclidean domain, and therefore is also a principal ideal domain, so I is generated by a single element of $\mathbb{F}[x]$. It is clear that $i(x) \mid g(x)$ if and only if $g(x) \in I$. Gröbner bases are used to bring that same sort of simple containment test into the multivariate case. The multivariate problem is similar: given a field $\mathbb{F}$ and an ideal $I \subset \mathbb{F}[x_1, x_2, \ldots, x_n]$, is there a simple, division-like technique to determine whether or not a polynomial $g(x_1, x_2, \ldots, x_n)$ is contained in I?

There are several difficulties that must be overcome before such a containment test is possible. What is desired is a technique for dividing multivariate polynomials by other multivariate polynomials. To do this in a seemly way, it is necessary to use an ordering on the set of (monic) monomials. In the univariate case, the degree of the monomial was sufficient. Now a more complicated technique must be used.

Definition 3.1 (Monomial order) *A monomial order on $\mathbb{F}[x_1, x_2, \ldots, x_n]$ is a well-ordering $\prec$ of the monic monomials such that if α, β, and γ are all monic monomials, and $\alpha \prec \beta$, then $\alpha + \gamma \prec \beta + \gamma$.*

A monomial ordering such as this is satisfied by the degree of a monomial in the univariate case, and there are many examples of such monomial orderings in the multivariate case. To illustrate the concept, consider the following example, for monic monomials α and β as above.

Example 3.2 (Lexicographic order (LEX)) *Write any monic monomial α as $\alpha = x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}$. Then another such monomial $\beta = x_1^{b_1} x_2^{b_2} \cdots x_n^{b_n}$ is compared to α by forming the n-tuple $(a_1 - b_1, a_2 - b_2, \ldots, a_n - b_n)$. If the leftmost nonzero element of this n-tuple is negative, then $\alpha \prec_{LEX} \beta$.*

Note that this example defines two orderings. First, it defines $x_1 \prec x_2 \prec \cdots \prec x_n$. Then it defines the more obvious behavior on the powers of each indeterminate. Often when computing Gröbner Bases, a different monomial ordering is used.

Example 3.3 (Degree reversed lexicographic order (DRL)) *Again, let $\alpha = x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}$ and $\beta = x_1^{b_1} x_2^{b_2} \cdots x_n^{b_n}$ be monic monomials. Let $\deg(\alpha) = \sum_{i=1}^{n} a_i$ and $\deg(\beta) = \sum_{i=1}^{n} b_i$. Then $\alpha \prec_{DRL} \beta$ if $\deg(\alpha) < \deg(\beta)$, or $\deg(\alpha) = \deg(\beta)$, but the rightmost nonzero entry in $(a_1 - b_1, a_2 - b_2, \ldots, a_n - b_n)$ is positive.*

Any monomial order is sufficient to sort the terms of polynomials. Now any polynomial once again has well-defined leading terms with associated leading coefficients, defined by the maximal monomial in that polynomial. In the future, let

$\mathrm{LT}(f)$ represent the leading term of a polynomial f over some monomial ordering $\prec$. Also, let the multidegree of f, $\mathrm{MDEG}(f)$ be given by $a = (a_1, a_2, \ldots, a_n)$, where $\mathrm{LT}(f) = Cx_1^{a_1}x_2^{a_2}\cdots x_n^{a_n}$. The leading monomial of a polynomial is simply its leading term made monic, and is denoted $\mathrm{LM}(f) = x_1^{a_1}x_2^{a_2}\cdots x_n^{a_n}$.

With a monomial order $\prec$, it is possible to define a long division technique that is a generalization of long division with remainder in the univariate case. Recall that the univariate polynomial division algorithm relies on the leading term of a polynomial being well-defined, and that it proceeds along a monomial ordering. The multivariate polynomial division algorithm that follows uses the same convention. It is also made generic enough to deal with division of a single polynomial by multiple polynomials at once. This is desired behavior, as in general, an ideal $I \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ is not generated by a single element of $\mathbb{F}[x_1, x_2, \ldots, x_n]$, and therefore it becomes necessary to check a given polynomial against a set of polynomials when attempting to determine ideal containment. It should be noted that the Hilbert basis theorem ensures that this set is always finite.

The following algorithm is essentially identical to the one found in [vzGG03].

Algorithm 3.4 (Multivariate long division)

Input: Nonzero polynomials $f, f_1, f_2, \ldots, f_u \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, monomial order $\prec$.

Output: Polynomials $r, q_1, q_2, \ldots, q_u \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ such that $f = q_1f_1 + q_2f_2 + \cdots + q_uf_u + r$, and for each f_i for $1 \leq i \leq u$, the leading term (LT) of f_i does not divide any monomial term of r.

Let $r := q_1 := q_2 := \cdots := q_u := 0$.

Let $d := f$.

while $d \neq 0$ do

if there is an i such that $\mathrm{LT}(f_i)$ divides $\mathrm{LT}(d)$, then

choose the smallest such i, and let

$q_i \longleftarrow q_i + \frac{\mathrm{LT}(d)}{\mathrm{LT}(f_i)}$,

$d \longleftarrow d - \frac{\mathrm{LT}(d)}{\mathrm{LT}(f_i)}f_i$

else let

$r \longleftarrow r + \mathrm{LT}(p)$

$p \longleftarrow p - \mathrm{LT}(p)$

return $r, q_1, q_2, \ldots, q_u$

This algorithm allows one to divide a polynomial f by a set of polynomials $F = \{f_1, f_2, \ldots, f_u\}$, and returns the set of their quotients $q_1, q_2, \ldots, q_u$ and a remainder r, all uniquely defined in this algorithm. As other division algorithms are possible, the concept of a unique remainder (or even unique quotients) does

not exist in the multivariate case. Still, using this algorithm, we now have a well-defined remainder modulo a set of polynomials, and will be written as $f \bmod F$.

It is now possible to give the following definition of a Gröbner basis..

Definition 3.5 (Gröbner basis) *Let $I \subset \mathbb{F}[x_1, x_2, \ldots, x_n]$ be an ideal, $I = \langle i_1, i_2, \ldots, i_v \rangle$, where $i_1, i_2, \ldots, i_v \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ generate I. Let $\prec$ be a monomial ordering. Then $G = \{g_1, g_2, \ldots, g_k\} \subset I$ is a Gröbner basis of I with respect to $\prec$ if $\langle LT(G) \rangle = \langle LT(I) \rangle$, where $LT(G) = \{LT(g_1), LT(g_2), \ldots, LT(g_k)\}$.*

That is not the only possible way to define a Gröbner basis, however. It can be shown that the following definition is equivalent, and it is useful in showing other facts about Gröbner bases.

Definition 3.6 (Gröbner basis) *Let $I \subset \mathbb{F}[x_1, x_2, \ldots, x_n]$ be an ideal and let $\prec$ be a monomial ordering. Then $G = \{g_1, g_2, \ldots, g_k\} \subset \mathbb{F}[x_1, x_2, \ldots, x_n]$ is a Gröbner basis of I if for every $g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, the following is true: $g \in I$ if and only if g rem $G = 0$; that is, $g \in I$ if and only if the polynomial long division of g by the set G has a remainder of 0.*

It can be shown that for any given monomial ordering, every ideal $I \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ has a Gröbner basis, and that any such basis also generates the ideal I. Both facts follow from Hilbert's basis theorem. See [vzGG03] for details.

3.1.2 Computing Gröbner bases

The definition of a Gröbner basis fails to account for the process of computing one. Several algorithms exist, including Buchberger's, F_4 and F_5, with the latter two due to work by Faugère (see [Fau99, Fau02]). The F_4 and F_5 algorithms apply linear algebra ideas to Buchberger's algorithm in order to save effort, as noted in [JKJMR05] . Buchberger's algorithm itself is widely implemented, and will be presented here.

By the second definition of a Gröbner basis, every element of an ideal I is a linear combination of elements of a Gröbner basis of I. Therefore, one way to decide whether or not a given set $G \subset I$ is a Gröbner basis of I is to check this condition for various polynomials, and add more elements to G as necessary, as a Gröbner basis is not necessarily unique, and not necessarily minimal.

Buchberger's algorithm employs just such a technique. It relies on calculating large numbers of so-called syzygy polynomials, or S-polynomials (as they are more commonly known). An S-polynomial of two multivariate polynomials f and g is given by finding the minimum monomials m_1 and m_2 such that in $m_1 f - m_2 g$ the leading terms of $m_1 f$ and $m_2 g$ cancel. More precisely,

Definition 3.7 (S-polynomial) *Let $f, g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$. Let $a = MDEG(f)$, and let $b = MDEG(f)$. Then let $c = (\max(a_1, b_1), \max(a_2, b_2), \ldots, \max(a_n, b_n))$. Then the S-polynomial of f and g is given by*

$$S(f, g) = \frac{x^c}{LT(f)} f - \frac{x^c}{LT(g)} g.$$

S-polynomials are easy to compute, and have an interesting property—any time it is possible to cancel terms, the cancellation can be written so as to involve S-polynomials. This fact is the basis of Buchberger's algorithm. In order to prove that Buchberger's algorithm works, and that therefore computing Gröbner bases is possible, a couple of theorems are necessary. These will be proved in detail, along with a proof of the correctness of the algorithm itself, in an effort to demonstrate the ideas involved in computing Gröbner bases, and to justify their use in this attack on multivariate cryptosystems by convincing the reader of their computability.

Theorem 3.8 *Let $G = \{g_1, g_2, \ldots, g_u\} \subset \mathbb{F}[x_1, x_2, \ldots, x_n]$. Let $\alpha_1, \alpha_2, \ldots, \alpha_u \in \mathbb{N}^n$ be multidegrees of monomials, and let $c_1, c_2, \ldots, c_u \in \mathbb{F}^*$. Then let*

$$f = \sum_{i=1}^{u} c_i x^{\alpha_i} g_i,$$

with a $\delta \in \mathbb{N}^n$ defined with the α_i for $1 \leq i \leq u$ so that $\delta = \alpha_i + MDEG(g_i)$, and $MDEG(f) \prec \delta$. Note that this condition implies cancellation occurs on the leading term of f. Let $\gamma_{ij} \in \mathbb{N}^n$ be defined by $x^{\gamma_{ij}} = lcm(LM(g_i), LM(g_j))$. Then $x^{\gamma_{ij}}$ divides x^δ, and there are $c_{ij} \in \mathbb{F}$ such that

$$f = \sum_{1 \leq i < j \leq u} c_{ij} x^{\delta - \gamma_{ij}} S(g_i, g_j)$$

and for all $1 \leq i < j \leq u$, $MDEG(\delta - \gamma_{ij} S(g_i, g_j)) \prec \delta$, using $\prec$ here to denote the ordering monomials would have given those two elements of $\mathbb{N}^n$ as their multidegrees.

PROOF (from [vzGG03]): Without loss of generality, assume that all of the g_i are monic (if not, then use the c_i to make it so.). Then for all the g_i, the leading monomial is the same as the leading term. So, as $x^\delta = x^{\alpha_i} \mathrm{LM}(g_i)$ and $x^\delta = x^{\alpha_j} \mathrm{LM}(g_j)$, for all $1 \leq i < j \leq u$, x^δ is a common multiple of $\mathrm{LM}(g_i)$ and $\mathrm{LM}(g_j)$. So by definition, $x^{\gamma_{ij}}$ divides x^δ.

Now note that as

$$S(g_i, g_j) = \frac{x^{\gamma_{ij}}}{\mathrm{LT}(g_i)} g_i - \frac{x^{\gamma_{ij}}}{\mathrm{LT}(g_j)} g_j$$

it is clear that $\mathrm{MDEG}(S(g_i, g_j)) \preccurlyeq \gamma_{ij}$. As the leading terms cancel,

$$\mathrm{MDEG}(x^{\delta - \gamma_{ij}} S(g_i, g_j)) = \delta - \gamma_{ij} + \mathrm{MDEG}(S(g_i, g_j)) \prec \delta - \gamma_{ij} + \gamma_{ij} = \delta$$

and thus $\mathrm{MDEG}(x^{\delta - \gamma_{ij}} S(g_i, g_j)) \prec \delta$.

All that remains to be proven is that

$$f = \sum_{1 \leq i < j \leq u} c_{ij} x^{\delta - \gamma_{ij}} S(g_i, g_j)$$

and this can be shown by induction on u. The statement is trivially true for $u = 1$. Then for a given u, all sets G with $u-1$ or fewer elements make this theorem true. Consider a set G with u elements. Then define a $g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ by

$$\begin{aligned} g &= f - c_1 x^{\gamma_{12}} S(g_1, g_2) \\ &= c_1 x^{\alpha_1} g_1 + c_2 x^{\alpha_2} g_2 + \sum_{i=3}^{u} c_i x^{\alpha_i} g_i - c_1 x^{\gamma_{12}} \left(\frac{x^{\gamma_{12}}}{\mathrm{LT}(g_1)} g_1 - \frac{x^{\gamma_{12}}}{\mathrm{LT}(g_2)} g_2) \right) \\ &= c_1 \left(x^{\alpha_1} - x^{\delta - \mathrm{MDEG}(g_1)} \right) g_1 + \left(c_2 x^{\alpha_2} + c_1 x^{\delta - \mathrm{MDEG}(g_2)} \right) g_2 + \sum_{i=3}^{u} c_i x^{\alpha_i} g_i \\ &= (c_1 + c_2) x^{alpha_2} g_2 + \sum_{i=3}^{u} c_i x^{\alpha_i} g_i. \end{aligned}$$

The last lines utilize the fact that $\delta - \mathrm{MDEG}(g_i) = \alpha_i$ for $1 \leq i \leq u$. Then note that the function g has the same form as the function f, but the set G' that g uses has either $u-1$ elements, if $c_1 \neq c_2$, or $u-2$ elements if $c_1 = c_2$. So, by the inductive hypothesis, this statement is true for g.

Then $g = \sum_{2 \leq i < j \leq u} c_{ij} x^{\delta - \gamma_{ij}} S(g_i, g_j)$. By rewriting in terms of f, we have

$$f = g + c_1 x^{\gamma_{12}} S(g_1, g_2)$$

which fulfills the requirements of the theorem in the case of u elements in G. $\diamond$

Theorem 3.9 *Let $G = \{g_1, g_2, \ldots, g_u\} \subset \mathbb{F}[x_1, x_2, \ldots, x_n]$ generate an ideal $\langle G \rangle$. Then G is a Gröbner basis for the monomial ordering $\prec$ if and only if for every $f, g \in G$, $S(f, g) \bmod G = 0$*

PROOF ([CLO92]): It follows from the second definition of a Gröbner basis that if G is in fact a Gröbner basis of $\langle G \rangle$, then it is also a generating set for $\langle G \rangle$, and for every $f, g \in G$, $S(f, g) \bmod G = 0$, as $S(f, g) \in \langle G \rangle$. Therefore it suffices to show that if for every $g_i, g_j \in G$, $S(g_i, g_j) \bmod G = 0$, then G is a Gröbner basis. This is done by taking some $f \in \langle G \rangle$, and showing that $\mathrm{LT}(f) \in \langle \mathrm{LT}(g_1), \mathrm{LT}(g_2), \ldots, \mathrm{LT}(g_u) \rangle$.

By definition of $\langle G \rangle$, one can write f as

$$f = \sum_{i=1}^{u} q_i g_i$$

Let $m_i = \mathrm{MDEG}(q_i g_i)$ for $1 \leq i \leq u$. If $\mathrm{MDEG}(f) = \delta$, where $\delta = \max_{\prec}\{m_i : 1 \leq i \leq u\}$, and δ is chosen so that it is minimal, then it is clear that $\mathrm{LM}(f) = \mathrm{LM}(q_i g_i) = \mathrm{LM}(q_i)\mathrm{LM}(g_i)$ for at least one i. Since this calculation is being done in a field, this leads to the conclusion that $\mathrm{LT}(f) = C\mathrm{LT}(q_i)\mathrm{LT}(g_i)$ for some $C \in \mathbb{F}$. Then $\mathrm{LT}(f) \in \langle \mathrm{LT}(g_1), \mathrm{LT}(g_2), \ldots, \mathrm{LT}(g_u) \rangle$.

Now suppose for contradiction that $\mathrm{MDEG}(f) \prec \delta$, using the monomial ordering to order $\mathbb{N}^n$, and again chosing δ so that it is minimal with f in the

required form. Then a contradiction is derived in the following way:

$$\begin{aligned} f &= \sum_{m_i=\delta} q_i g_i + \sum_{m_i<\delta} q_i g_i \\ &= \sum_{m_i=\delta} \mathrm{LT}(q_i) g_i + \sum_{m_i=\delta} (q_i - \mathrm{LT}(q_i)) g_i + \sum_{m_i<\delta} q_i g_i \end{aligned}$$

Since the second and third sums on the second line all have multidegree $< \delta$, it follows that the first sum must also have multidegree $< \delta$. Then this first sum satisfies exactly the conditions of theorem 3.8—each term has multidegree δ, but their summation has a strictly smaller multidegree. Then by those same conditions, it is possible to represent the expression in this way

$$\sum_{m_i=\delta} \mathrm{LT}(q_i) g_i = \sum_{1 \le j < k \le u} c_{jk} x^{\delta - \gamma_{jk}} S(g_j, g_k)$$

where as before the $c_{jk} \in \mathbb{F}$ and $x^{\gamma_{jk}} = \mathrm{lcm}(\mathrm{LM}(g_j), \mathrm{LM}(g_k))$.

Now use the previous theorem. By that theorem,

$$S(g_j, g_k) = \sum_{i=1}^{u} a_{ijk} g_i$$

where $a_{ijk} \in \mathbb{F}[x_1, x_2, \ldots, x_n]$. By the division algorithm, $\mathrm{MDEG}(a_{ijk} g_k) \preccurlyeq \mathrm{MDEG}(S(g_j, g_k))$, for all i, j, k. Multiply on both sides by $x^{\delta - \gamma_{jk}}$ to get

$$x^{\delta - \gamma_{jk}} S(g_j, g_k) = \sum_{i=1}^{u} b_{ijk} g_i$$

where $b_{ijk} = x^{\delta - \gamma_{jk}} a_{ijk}$. Then

$$\mathrm{MDEG}(b_{ijk}) \preccurlyeq \mathrm{MDEG}(x^{\delta - \gamma_{jk}} S(g_j, g_k)) \prec \delta.$$

Substituting back into the original expression for $S(g_j, g_k)$ leads to

$$\begin{aligned} \sum_{m_i=\delta} \mathrm{LT}(q_i) g_i &= \sum_{j,k} c_{jk} x^{\delta - \gamma_{jk}} S(g_j, g_k) \\ &= \sum_{j,k} c_{jk} \left(\sum_i b_{ijk} g_i \right) = \sum_i \left(\sum_{j,k} c_{jk} b_{ijk} \right) g_i. \end{aligned}$$

Then rewrite the last sum as $\sum_i \bar{q}_i g_i$, and note that because the c_{jk} are constants, it is still true that $\mathrm{MDEG}(\bar{q}_i g_i) \prec \delta$.

Finally, substitute $\sum_i \bar{q}_i g_i$ for $\sum_{m_i=\delta} \mathrm{LT}(q_i) g_i$ and note that

$$f = \sum_{m_i=\delta} \bar{q}_i g_i + \sum_{m_i=\delta} (q_i - \mathrm{LT}(q_i)) g_i + \sum_{m_i<\delta} q_i g_i$$

is now expressed as a polynomial combination of terms, all of which have multidegree $\prec \delta$. This contradicts the minimality of δ. $\diamondsuit$

This leads naturally to a simple algorithm for finding a Gröbner basis. The following algorithm is due to Buchberger and is reprinted from [CLO92]. Many refinements to it are possible.

Algorithm 3.10

Input: $G = \{g_1, g_2, \ldots, g_s\} \subset \mathbb{F}[x_1, x_2, \ldots, x_n]$

Output: A Gröbner basis F for $I = \langle G \rangle$

$F \longleftarrow G$

DO

$F' \longleftarrow F$

FOR each pair $\{p, q\}$, $p \neq q$, $p, q \in F'$ DO

$S \longleftarrow S(p, q) \mathrm{mod} F'$

IF $S \neq 0$ THEN $F \longleftarrow F \cup \{S\}$

WHILE $F \neq F'$

In layman's terms, the algorithm calculates S-polynomials for a given generating set G and adds new polynomials to the set G whenever an S-polynomial is found that does not have a remainder of 0 modulo G. It remains to show that this algorithm is correct, and that it terminates in a finite period of time. Note that the termination condition is precisely the condition of Theorem 3.9, so if this algorithm terminates, it will output a Gröbner basis for the expanded set F.

PROOF ([CLO92]): Two facts must be shown. First, it must be shown that $F \subset I$, so that the Gröbner basis generated is clearly a basis of the original ideal I. Second, it must be shown that the algorithm itself terminates in a finite number of steps. It is clear that before the algorithm begins, $F \subset I$. Then it is sufficient to show that each added element S is contained in I. As $p, q \in F' \subset I$, therefore $S(p, q) \in I$. Combine this with the fact that the long division is by $F' \subset I$, and it becomes clear that the remainder must also be an element of the ideal I by additive closure. So it is true that $F \subset I$.

The other fact to be shown is that the algorithm terminates. If $F \neq F'$, then by the algorithm, $F' \subsetneq F$. This implies $\langle \mathrm{LT}(F') \rangle \subsetneq \langle \mathrm{LT}(F) \rangle$ in the following way. Let r be a non-zero remainder of an S-polynomial that was divided byF'. Then it is clear that r cannot be divisible by the leading terms of elements of F', and therefore $\mathrm{LT}(r) \notin \langle \mathrm{LT}(F') \rangle$. However, $\mathrm{LT}(r) \in \langle \mathrm{LT}(F) \rangle$, as it was explicitly added to F.

This leads to an ascending chain of ideals $\langle \mathrm{LT}(F) \rangle$ of $\mathbb{F}[x_1, x_2, \ldots, x_n]$. As it is true that the ideals of $\mathbb{F}[x_1, x_2, \ldots, x_n]$ satisfy the ascending chain condition, it is true that after a finite number of ideals in this chain, they stop growing, and $\langle \mathrm{LT}(F') \rangle = \langle \mathrm{LT}(F) \rangle$. The previous paragraph shows that this implies $F = F'$. Thus, the algorithm terminates in a finite number of steps. $\diamondsuit$

Buchberger's algorithm is a useful starting point. Faugère's F_4 and F_5 algorithms both flow from Buchberger's algorithm, with modifications designed to reduce the number of times an S-polynomial is calculated that reduces to 0 over the current set of polynomials. In fact, in many cases, it is noted in [Fau02] that no "useless" pairs (those S-polynomials that reduce to 0) are generated at all. Generally-speaking, F_4 is at least an order of magnitude faster than Buchberger's Algorithm, and F_5 is at least an order of magnitude faster than F_4, as noted experimentally in [Fau02].

Still, the problem of computing a Gröbner basis is not necessarily solvable in any reasonable time-frame, as it can be shown that the problem of finding a Gröbner basis is $\mathcal{EXPSPACE}$-complete.

Definition 3.11 ($\mathcal{EXPSPACE}$ and $\mathcal{EXPSPACE}$-complete) *A decision problem DP solved by an algorithm A is in the class $\mathcal{EXPSPACE}$ if A requires $2^{k^{O(1)}}$ space to run for inputs of length k. DP is $\mathcal{EXPSPACE}$-complete if in addition, every decision problem DP' in $\mathcal{EXPSPACE}$ can be reduced to DP in polynomial time.*

Typically speaking, an $\mathcal{EXPSPACE}$-complete problem uses, for at least some inputs of length k, doubly-exponential time $2^{2^{O(k)}}$, as noted in [vzGG03, Section 21.7]. Thus, in the worst case, computing a Gröbner basis is impractical.

3.1.3 The elimination theorem and Gröbner basis conversion

One of the cornerstones, if not the cornerstone, of Gröbner basis attacks is the elimination theorem. This theorem states that Gröbner bases constructed using certain monomial orderings essentially solve the problem of finding the plaintext x for the ciphertext y. Specifically, the elimination theorem states that such a Gröbner basis has subsets $G_0, G_1, \ldots, G_{n-1}$ so that in each subset G_k, the only indeterminates that occur in the polynomials of G_k are $x_{k+1}, x_{k+2}, \ldots, x_n$. Then it becomes possible to solve for the indeterminates $x_n, x_{n-1}, \ldots, x_1$ in sequence, creating a set of possible solutions from which the "correct" solution should be easy to find.

First, in order to state the elimination theorem, the elimination ideal must be defined.

Definition 3.12 (Elimination ideal) *Given an ideal $I = \langle f_1, f_2, \ldots, f_s\rangle \subset \mathbb{F}[x_1, x_2, \ldots, x_n]$, the kth elimination ideal I_k is the ideal of $\mathbb{F}[x_{k+1}, x_{k+2}, \ldots, x_n]$ defined by*

$$I_k = I \cap \mathbb{F}[x_{k+1}, x_{k+2}, \ldots, x_n]$$

Using this definition, it is possible to state the elimination theorem itself.

Theorem 3.13 (Elimination theorem) *Let G be a Gröbner basis of an ideal $I = \langle f_1, f_2, \ldots, f_s\rangle \subset \mathbb{F}[x_1, x_2, \ldots, x_n]$ with respect to the lexicographical monomial ordering. Then the set*

$$G_k = G \cap \mathbb{F}[x_{k+1}, x_{k+2}, \ldots, x_n]$$

is a Gröbner Basis of the kth elimination ideal I_k with respect to the lexicographical monomial ordering.

This statement is proven in various books, including [CLO92]. Such a proof will not be repeated here. Note that this theorem uses explicitly the lexicographical monomial ordering, while the previous work on Gröbner bases done here is monomial ordering agnostic. In fact, it is also true that the lexicographical ordering is generally inferior to the degree reversed lexicographical ordering when computing Gröbner bases (as mentioned in [CLO98, 2.4]). Then in order to make efficient use of the elimination theorem, Gröbner basis conversion is necessary. Then a Gröbner basis can be constructed efficiently, converted to lexicographical ordering, and the elimination theorem put to use. As [CLO98] notes, it is faster in general to compute the DRL-Gröbner basis and convert it to LEX than it is to compute the LEX-Gröbner basis in the first place.

The Faugère-Gianni-Lazard-Mora algorithm is suggested here as one "black-box" technique for performing this conversion. It may be found in [CLO98, 2.3].

3.1.4 Attacks using Gröbner bases

It is now possible to outline a generic attack on a multivariate cryptosystem using Gröbner bases. Recall that this attack involves solving a known system of equations $\mathcal{P}(x) = y$ for x. The attacker finds a set of possible solutions $\mathcal{X}$ to the equations $\mathcal{P}(x) = y$, or, equivalently,

$$\begin{aligned} p_1(x_1, x_2, \ldots, x_n) - y_1 &= 0 \\ p_2(x_1, x_2, \ldots, x_n) - y_2 &= 0 \\ &\vdots \\ p_m(x_1, x_2, \ldots, x_n) - y_m &= 0 \end{aligned}$$

by computing a Gröbner basis G for the ideal $I = \langle p_1 - y_1, p_2 - y_2, \ldots, p_m - y_m \rangle$. For reasons of efficiency, it is usually constructed using the degree reversed lexicographical ordering, and then converted to the lexicographical ordering using Gröbner basis conversion.

This set G is useful for the following reason. Once G is computed, note that any x that is a root of all the polynomials of G must also be a root of all polynomials of $\mathcal{P} - y$, and vice-versa. One proves this by noting that as G and $\mathcal{P} - y$ both generate I, any polynomial of G may be rewritten as a polynomial sum of the elements of the other set $\mathcal{P}$. If x is a root of every element of $\mathcal{P}$, then by this, any element of G has a root at x. The same logic works for x a root of G. So, every common root of G is a common root of $\mathcal{P}$, and vice-versa.

Therefore, because the elimination theorem holds for G, the attacker can solve a sequence of sets of univariate equations $G_n, G_{n-1}, \ldots, G_0 = G$, where for each G_k, the values of the indeterminates $x_{k+1}, x_{k+2}, \ldots, x_n$ have been computed using $G_n, G_{n-1}, \ldots, G_{k+1}$. The attacker factors $n + 1$ sets of univariate polynomials to carry out the attack. Even the Berlekamp algorithm [vzGG03,

Section 14.8] is capable of factoring a given polynomial in $O(d^3)$ field operations, where d is the degree of the polynomial to be factored. The algorithm branches at each possible value for x_i, for $i = 1, 2, \ldots, n$, creating a set of possible solutions $\mathcal{X}$. Note that this will not find all possible solutions to the set of common roots of G, as to do so requires an appropriate splitting field for $\mathbb{F}$, but the roots that cannot be found by this method are unwanted anyways, as they cannot be valid plaintexts. So, the set $\mathcal{X}$ is the set of all possible common roots of G, and therefore all possible common roots of $\mathcal{P}$. This is precisely the set of possible solutions desired.

Finally, it is easy to determine the correct value of x to use, given the set $\mathcal{X}$, regardless of the implementation of the multivariate cryptosystem. In a multivariate cryptosystem in which $\mathcal{P}$ is invertible, $\mathcal{X}$ consists of exactly one element. In a multivariate cryptosystem in which $\mathcal{P}$ is not invertible, the correct choice of $x \in \mathcal{X}$ is the one that follows the redundancy condition of that particular multivariate cryptosystem (e.g., an x that has the form $x = m||H(m)$ for some hash function H).

This attack is not meant to be seen as "the" Gröbner basis attack. It is possible to note variants thereof. For example, any Gröbner basis G has a minimal Gröbner basis G' as a subset (one that has no polynomial g such that $\mathrm{LT}(g) \in \langle \mathrm{LT}(G - g) \rangle$, and all polynomials are monic). In any minimal Gröbner basis G that results from an attack on an invertible $\mathcal{P}$, all elements of G have degree 1, and there are enough of them to fully determine x. Naturally, as forming a minimal Gröbner basis from a given Gröbner basis is straightforward, in this situation, one does so, saving a basis conversion and essentially all of the remaining math (the correct value of x can be read off the Gröbner basis instead, as noted in [JKJMR05]). In fact, in general forming a minimal Gröbner basis is easy enough to do (it is simply a sequence of attempted monomial divisions by other monomials, which can be viewed as a subtraction of their multidegrees) that it is a way to reduce the number of computations in later steps. In [FJ03], still further optimizations are noted for the case $\mathbb{F} = \mathbb{F}_2$ and the system $\mathcal{P}(x)$ is invertible.

The main cost of a Gröbner basis attack is the computation of the Gröbner basis. Even using the most advanced algorithm to date, F_5, there are limits. In 2003, a variant of F_5 designed to be used over $\mathbb{F}_2$ was used to to defeat the first HFE Challenge (proposed for the HFE cryptosystem in [Pat96b], available at [Pat96a]). Faugère and Joux [FJ03] noted that this challenge is broken, and that it was a system of 80 equations of degree 2 in 80 indeterminates. To demonstrate the superiority of the F_5 algorithm, Faugère and Joux [FJ03] presented Table 3.1 to note that F_5 is a significantly faster technique to compute a Gröbner basis. However, it is quite difficult to estimate in general the cost of computing Gröbner bases in these problem spaces, so exact measurements of what can and cannot be done are rare.

In Section 6.2.1, it is noted that Gröbner basis attacks on the HFE cryptosystem are often highly effective, and run in polynomial time. More detailed information about the efficacy of Gröbner basis attacks on this particular system will be noted there.

Table 3.1: The number of equations of degree 2 various algorithm implementations can handle in the given time frames (on a PC PIII 1 GHz) [FJ03].

Algorithm	Buchberger				F_4	F_5
System	Maple	Magma	Macaulay	Singular	FGb	FGb
CPU Time < 10m	12	17	18	19	22	**35**
CPU Time < 2h	14	19	20	21	28	**45**

3.2 The XL method

The eXtended Linearization (aka XL) algorithm, first proposed by Courtois *et al.* [CKPS00], is designed to solve the same sort of multivariate system as the one solved using a Gröbner Basis attack. That is, when given a ciphertext $y = (y_1, y_2, \ldots, y_m)$, it attempts to solve the following polynomial system

$$\mathcal{P}(x) - y = 0$$

for the plaintext $x = (x_1, x_2, \ldots, x_n)$. The XL algorithm attempts to discover the indeterminates $x_1, x_2, \ldots, x_n$ one at a time, and recursively calls itself on the new, simplified system on each success. At this level of detail, it bears a striking resemblence to the techniques of the Gröbner basis method.

Unfortunately, this resemblence is significantly more than skin-deep. The XL method is now known to be a slower version of F_4, as shown by Ars *et al.* [AFI$^+$04]. Therefore, there is little point in using it. F_4 itself has been supplanted by F_5 in terms of speed, so at this time there is really no reason to consider the XL algorithm in its present state as a useful technique.

However, this discovery is quite recent. Papers from 2000 to 2004 all mentioned XL as its own avenue of attack on a cryptosystem. Only very recently have papers begun to note that it is unnecessary to seperately consider the efficacy of XL and Gröbner bases attacks. In a 2004 paper, Ding [Din04] makes no mention of the fact the XL method is a disguised Göbner basis technique. However, in a 2005 paper, Ding and Schmidt [DS05] note the new connection, as well as the fact that the XL algorithm may now be ignored, as it is merely a slower implementation of the Gröbner basis attack than the state of the art.

Thus, the XL algorithm is covered here for a pair of reasons. First, it has some historical significance, in that for several years, many papers referenced the idea. Second, it is interesting in its own right, as a slightly different take on some of the same ground covered by a standard Gröbner basis technique.

3.2.1 Defining the XL algorithm

In [CKPS00], the XL algorithm is explicitly set up to solve the polynomial system $\mathcal{P}(x) - y = 0$ for the plaintext x, given the ciphertext y. It is implicitly noted in [CKPS00] and explicitly noted in [AFI$^+$04] that one assumption of the XL algorithm is that the x being sought is unique–that is, there is only

one solution $x = (x_1, x_2, \ldots, x_n)$ over $\mathbb{F}_q^n$. Without this assumption, claims of efficiency for the XL algorithm become quite dubious.

In order to define the algorithm itself, first some notation is necessary.

Let $\mathcal{X}^k$ denote the set of all monic monomials of degree k in $x = (x_1, x_2, \ldots, x_n)$. That is,

$$\mathcal{X}^k = \left\{ \prod_{i=1}^{n} x_i^{\alpha_i} \mid \alpha_i \in \mathbb{N}_0, \sum_{i=1}^{n} \alpha_i = k \right\}.$$

Let $\mathcal{X}^k \times \mathcal{P}$ denote the set of all polynomials formed by multiplying each polynomial of the public key p_i, for $1 \leq i \leq m$, by each element of $\mathcal{X}^k$. That is,

$$\mathcal{X}^k \times \mathcal{P} = \left\{ fp \mid f \in \mathcal{X}^k, p \in \mathcal{P} \right\}.$$

So, if $m = 2$, $n = 2$ and $k = 2$, where m is the number of polynomial equations in $\mathcal{P}$ and n is the number of indeterminates, then there are a total of 6 polynomials formed (3 possible monic monomials x_1x_2, x_1^2 and x_2^2 times 2 polynomials p_1 and p_2).

Now, let $D \in \mathbb{N}$. Consider the set of polynomials

$$\bigcup_{k=1}^{\infty} \mathcal{X}^k \times \mathcal{P}.$$

Since every element of $\mathcal{P}$ has degree 2, the subset of these polynomials of total degree $\leq D$ is given by

$$\bigcup_{k=0}^{D-2} \mathcal{X}^k \times \mathcal{P}.$$

Let $\mathcal{I}_D$ be the ideal of $\mathbb{F}_q[x_1, x_2, \ldots, x_n]$ generated by such polynomials. So, $\mathcal{I}_D$ is the vector space over $\mathbb{F}_q[x_1, x_2, \ldots, x_n]$ generated by $\bigcup_{k=0}^{D-2} \mathcal{X}^k \times \mathcal{P}$. If $D = 2$, then the ideal is generated by $\mathcal{P}$ itself and is refered to as $\mathcal{I}$. For all $D \geq 0$, it is true that $\mathcal{I}_D \subset \mathcal{I}$.

The creators of the XL method [CKPS00] note that the idea behind the technique is to move from $\mathcal{I}$ to $\mathcal{I}_D$ in the hope that for some D not too large the equations will become simpler to solve than the equations of $\mathcal{I}$.

Finally, recall that when performing multivariate long division, some kind of monomial ordering is required. Monomial orderings were defined in Definition 3.1 on page 14.

We are now ready to define the XL algorithm (definition follows [CKPS00]).

Definition 3.14 (XL Algorithm) *For a positive integer $D \geq 2$, execute the following steps:*

*1. **Multiply:** Generate all the products $\mathcal{X}^k \times \mathcal{P}$ for $k \leq D - 2$.*

*2. **Linearize:** Consider each monomial term in the x_i of degree $\leq D$ as a new indeterminate. Substitute these new indeterminates into the products obtained in 1, creating a system of linear equations. Perform Gaussian elimination on these linear equations, using a monomial ordering that eliminates all the terms containing one indeterminate (say, x_1) last.*

3. ***Solve:*** *Assuming that step 2 yields at least one univariate equation in the powers of x_1, solve this equation over the finite field (e.g., with Berlekamp's algorithm).*

4. ***Repeat:*** *Simplify the equations and repeat the process to find the values of the other indeterminates.*

It should be clear from how it is defined that the XL algorithm is not designed to return a set of possible values for x, and therefore, x should be unique for efficient use of the algorithm.

The original paper [CKPS00] failed to demonstrate the efficacy of this attack, by failing to prove that the XL algorithm terminates. It is shown in [AFI+04] that the XL algorithm not only terminates, it is in fact a redundant variant of the F_4 Gröbner basis algorithm.

Diem [Die04] notes that this algorithm has no graceful exit behavior for problem instances that have multiple solutions or no solutions over their given finite field, although such behavior is easy enough to implement–simply allow those cases to end in failure at step 3, rather than assuming success.

Ars *et al.* [AFI+04] note that the algorithm fixes D. Clearly, it is implausible to believe that D will be initialized to an appropriate value all of the time. Therefore, it will be necessary to start with a small value of D, such as $D = 2$, and increase it at each failed step (failure being possible due to the first fix). There are various ways to increase D, and they will be mentioned in the next section.

Finally, [AFI+04] notes that if the XL algorithm terminates successfully, it does so with a lexicographical ordering.

3.2.2 Attack methods using the XL algorithm

In carrying out an attack on an $\mathcal{MQ}$-based encryption scheme using the XL algorithm, an attacker seeks to solve a system

$$\mathcal{P}(x) - y = 0$$

for the unique plaintext x, where $\mathcal{P}$ and y are known. Using the XL algorithm basically takes care of all the details. The only choice remaining to the attacker is how the XL algorithm implementation will increase D. In [AFI+04], four seperate ways of doing so are suggested. They are reprinted here, edited to begin with $D = 2$ rather than $D = 1$, as using $D = 1$ leads to polynomial fractions and is therefore nonsensical in this context. So, in all these cases, begin with $D = 2$ and let $\mathcal{A} = \mathcal{P} - y$ be the system we desire to solve, and repeat until a solution is found.

1. Do the XL algorithm on $\mathcal{A}$. If the solution cannot be obtained, set $D := D + 1$. Repeat until solved.

2. Iterate the "multiply" and "linearize" steps for $\mathcal{A}$ by adding new equations obtained by "linearize" to $\mathcal{A}$. If the resulting system is not solveable, then return to the original set $\mathcal{A}$, set $D := D + 1$. Repeat until solved.

3. Do the XL algorithm on $\mathcal{A}$. On failure, set $D := D + 1$, replace $\mathcal{A}$ by the resulting system of the "linearize" step of the most recent XL. Repeat until solved.

4. Iterate the "multiply" and "linearize" steps for $\mathcal{A}$ by adding new equations obtained by "linearize" to $\mathcal{A}$. If the resulting system $\mathcal{A}'$ is not solveable, then replace $\mathcal{A}$ by $\mathcal{A}'$ and $D := D + 1$. Repeat until solved.

It is further noted in [AFI+04] that the second pair of techniques can result in a smaller final D than the first pair.

For any method of incrementing D, the XL algorithm will eventually terminate with an answer, assuming y really is a ciphertext to a system that has an invertible function $\mathcal{P}$. This answer x will be the plaintext. As noted previously, however, XL algorithm-based attacks are no longer considered useful at this time.

3.3 Dixon resultants

The method of Dixon resultants was recently introduced by Tang and Feng [TF05]. Like Gröbner basis techniques, Dixon resultants do not depend on the particular $\mathcal{MQ}$-based encryption scheme used. They are a generic technique that works for any instance of an $\mathcal{MQ}$ problem. From a cryptanalysis standpoint, they have the advantage of an easily determined complexity. Early results also indicate that they are significantly faster than Faugère's F_4 method of Gröbner basis calculation [TF05], although there are no published comparisons to Faugère's faster F_5 algorithm.

A Dixon resultant-based method uses a necessary condition for finding a common affine zero x of a system of equations $\mathcal{P}$ in order to solve, given a ciphertext y, the following polynomial system

$$\mathcal{P}(x) - y = 0$$

for the plaintext x.

3.3.1 The Dixon polynomial, Dixon matrix and Dixon resultant

Some preliminary definitions are required to define the Dixon resultant. First, a multivariate polynomial p with n indeterminates $x_1, x_2, \ldots, x_n$ is called a *generic ndegree* polynomial if one can write p as

$$p(x_1, x_2, \ldots, x_n) = \sum_{i_1=1}^{k_1} \sum_{i_2=1}^{k_2} \cdots \sum_{i_n=1}^{k_n} a_{i_1,i_2,\ldots i_n} x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}$$

for some positive integers $k_1, k_2, \ldots k_n$. The salient feature of such a polynomial is that each of its monomial terms incorporates all the indeterminates (i.e., note

that $i_1, i_2, \ldots, i_n \geq 1$). Let F be a set of $n+1$ generic ndegree polynomials in n indeterminates $x_1, x_2, \ldots, x_n$. We write $F = \{f_1, f_2, \ldots, f_{n+1}\}$. Next, take the following determinant:

$$\Delta(x_1, x_2, \ldots, x_n, \alpha_1, \alpha_2, \ldots, \alpha_n) =$$

$$\det \begin{bmatrix} f_1(x_1, x_2, \ldots, x_n) & f_2(x_1, x_2, \ldots, x_n) & \ldots & f_{n+1}(x_1, x_2, \ldots, x_n) \\ f_1(\alpha_1, x_2, \ldots, x_n) & f_2(\alpha_1, x_2, \ldots, x_n) & \ldots & f_{n+1}(\alpha_1, x_2, \ldots, x_n) \\ f_1(\alpha_1, \alpha_2, \ldots, x_n) & f_2(\alpha_1, \alpha_2, \ldots, x_n) & \ldots & f_{n+1}(\alpha_1, \alpha_2, \ldots, x_n) \\ \vdots & \vdots & & \vdots \\ f_1(\alpha_1, \alpha_2, \ldots, \alpha_n) & f_2(\alpha_1, \alpha_2, \ldots, \alpha_n) & \ldots & f_{n+1}(\alpha_1, \alpha_2, \ldots, \alpha_n) \end{bmatrix}. \quad (3.1)$$

Here, $\alpha_1, \alpha_2, \ldots, \alpha_n$ are new indeterminates. Then the *Dixon polynomial* δ of F is given by

$$\delta(x_1, x_2, \ldots, x_n, \alpha_1, \alpha_2, \ldots, \alpha_n) = \frac{\Delta(x_1, x_2, \ldots, x_n, \alpha_1, \alpha_2, \ldots, \alpha_n)}{(x_1 - \alpha_1)(x_2 - \alpha_2) \cdots (x_n - \alpha_n)}.$$

Note that δ is still a polynomial, as for each x_i, replacing it by α_i in Δ leads to two identical rows in the matrix of (3.1), namely the ith row and the $(i+1)$th row. Thus, the determinant of this matrix becomes zero, and division by $(x_i - \alpha_i)$ corresponds to removing a root of Δ.

Now we have a polynomial δ in indeterminates $x_1, x_2, \ldots, x_n, \alpha_1, \alpha_2, \ldots, \alpha_n$. If the original set of polynomials F has a common zero, then note that the determinant in (3.1), upon which the Dixon polynomial is based, has a row of all zeroes (the top row) when evaluated at that common zero. Thus, when $x_1, x_2, \ldots, x_n$ correspond to a common zero of F, the Dixon polynomial evaluated at $(x_1, x_2, \ldots, x_n)$ is zero, regardless of the values of $\alpha_1, \alpha_2, \ldots, \alpha_n$. With this in mind, we consider δ to be a polynomial in $\alpha_1, \alpha_2, \ldots, \alpha_n$. Thus, its coefficients are polynomials in $x_1, x_2, \ldots, x_n$. Let ϵ' be the vector given by fixing an ordering to the set of all monomials in $\alpha_1, \alpha_2, \ldots, \alpha_n$ that appear as terms of δ. Fix an ordering of the monomials in $x_1, x_2, \ldots, x_n$ that appear in at least one coefficient of one element of ϵ'. Let V' be the column vector given by

$$V' = (v'_1, v'_2, \ldots, v'_{s_2})^T,$$

where there are s_2 such monomials, and each v'_i, for $1 \leq i \leq s_2$ is given by

$$v'_i = x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}, \text{ where } x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n} \text{ is a term contained in some element of } \epsilon'.$$

When we refer to V', we refer to a vector of monomials. This allows us to set up a homogeneous system of equations to express ϵ' in terms of the elements of V', represented as a matrix D with entries from the base field $\mathbb{F}_q$. If $x_1, x_2, \ldots, x_n$ is a common zero of the original system, then we have

$$\epsilon' = DV' = (0\ 0\ 0\ 0\ 0\ \ldots\ 0)^T.$$

Kapur, Saxena and Yang [KSY94] note that if F is a set of generic ndegree polynomials, then $|\epsilon'| = |V'|$. Let $s = |V'|$, and note that D is a square $s \times s$ matrix. Now we replace each element of V' by a new indeterminate v_i. Let $V = (v_1, v_2, \ldots, v_s)^T$ be the result of this change of indeterminates. Then we can form an equivalent ϵ from ϵ' by forming the system

$$\epsilon = DV.$$

Now, if $x_1, x_2, \ldots, x_n$ is a common zero of the original system, we still have a homogeneous linear system of equations

$$\epsilon = DV = (0\ 0\ 0\ 0\ 0\ \ldots\ 0)^T$$

in the indeterminates $V = (v_1, v_2, \ldots, v_s)^T$. We call the matrix D the *Dixon matrix*, and we call $\det(D)$ the *Dixon resultant*.

The benefits of calculating the Dixon resultant are as follows. First, note that if $x = (x_1, x_2, \ldots, x_n)$ is a common root of the polynomials of F, then the Dixon polynomial is zero, and thus the Dixon matrix has rank 0. So, the Dixon resultant also vanishes. Thus we see that the vanishing of the Dixon resultant is a necessary condition for a given x to be a common root of the set F.

Of course, this is not the whole story. These Dixon terms are all defined for a set of polynomials F that are generic ndegree polynomials. However, the polynomials used in an $\mathcal{MQ}$-based encryption scheme are generally not of this form (in fact, they are essentially never in this form). Therefore, in order to deal with the situation of $F = \mathcal{P}(x) - y$, where $\mathcal{P}$ is a public key for an $\mathcal{MQ}$-based encryption scheme and y is a known ciphertext of that scheme, more work is required. In fact, if the condition on the set of polynomials F that it be a set of generic ndegree polynomials is dropped, it is no longer certain that the Dixon matrix will be square, as noted by Kapur, Saxena and Yang [KSY94]. Thus, the Dixon resultant may not even exist. Somehow, the concept must be extended before it remains suitable for use in the case of general polynomials.

When dealing with general polynomials, everything stated in this section about the Dixon polynomial and Dixon matrix remains true, save that the Dixon matrix may not be square, and thus the Dixon resultant may not exist. In particular, it is still a necessary condition for an affine zero x to exist that the Dixon polynomial vanish at x. In the remainder of this chapter, the terms Dixon polynomial and Dixon matrix will be used to refer to the case of general polynomials.

3.3.2 The KSY Dixon matrix and the extended Dixon resultant

The failure of a standard Dixon resultant to be useful in the general case, which is also the case of an $\mathcal{MQ}$-based encryption scheme, led to an extension of the Dixon resultant by Kapur, Saxena and Yang [KSY94]. Their technique can handle the general case, subject to a condition that they noted always held in their experiments, the Rank Submatrix Construction Criteria, or RSC Criteria

(see Theorem 3.16). They created a new matrix, dubbed by Tang and Feng [TF05] the KSY Dixon matrix with a determinant referred to as the extended Dixon resultant. Tang and Feng [TF05], who worked with extended Dixon resultants, noted as well that the RSC Criteria always held in their experiments.

The technique of Kapur, Saxena and Yang [KSY94] takes the Dixon matrix D, as computed normally, and, given that the rank of D is r, finds an $r \times r$ submatrix of D that is also of rank r. This new matrix is called the KSY Dixon matrix, and its determinant is the extended Dixon resultant. It is presented here with the following change: now, all calculations take place using a finite field $\mathbb{F}_q$ as the base field, rather than $\mathbb{Q}$.

In order to work with the Dixon matrix, a few labels and definitions are necessary, as noted by Kapur, Saxena and Yang [KSY94]. A Dixon matrix D is an $s_1 \times s_2$ matrix, where s_1 and s_2 are not necessarily equal. Let the columns of D be represented as $m_1, m_2, \ldots, m_{s_2}$. Then let $monom(m_i) = v_i$. Let C be a set of constraints on the indeterminates $x_1, x_2, \ldots, x_n$ of the form

$$x_{i_1} \neq 0 \wedge x_{i_2} \neq 0 \wedge \ldots \wedge x_{i_k} \neq 0, \tag{3.2}$$

for some $1 \leq i_1, i_2, \ldots, i_k \leq n$. Finally, let $nvcol(C)$ denote the set of all columns m_i such that $C \rightarrow monom(m_i) \neq 0$.

Let G be a set of $n+1$ polynomials, $G \subset \overline{\mathbb{F}_q}[x_1, x_2, \ldots, x_n]$, where $\overline{\mathbb{F}_q}$ is the algebraic closure of $\mathbb{F}$. Let N be the Dixon matrix of G, and let

$$\begin{aligned} \mathcal{N}_1 \quad = \quad & \{X \mid X \text{ is an } s_1 \times (s_2 - 1) \text{ submatrix of } N \text{ obtained by} \\ & \text{deleting a column which belongs to } nvcol(C) \text{ from } N\}. \end{aligned} \tag{3.3}$$

The following lemma and its proof are as [KSY94].

Lemma 3.15 *Let $G \subset \overline{\mathbb{F}_q}[x_1, x_2, \ldots, x_n]$ be a set of $n+1$ polynomials with associated Dixon matrix N. Let C be a set of constraints on the indeterminates $x_1, x_2, \ldots, x_n$ of form (3.2). If G has a common affine zero which satisfies C, then*

$$\textit{For every } X \in \mathcal{N}_1, \ \textit{rank}(X) = \ \textit{rank}(N).$$

Proof: Let $(c_1, c_2, \ldots, c_n)$ be a common zero of G. Then as each indeterminate of V corresponds to a monomial of $x_1, x_2, \ldots, x_n$, this common zero leads to a solution to the system of homogeneous linear equations $\epsilon = NV = 0$, denoted $C_1, C_2, \ldots, C_{s_2}$. Then for the column vectors $m_1, m_2, \ldots, m_{s_2}$ of N, we have

$$C_1 m_1 + C_2 m_2 + \ldots + C_i m_i + \ldots + C_{s_2} m_{s_2} = 0.$$

Let $X \in \mathcal{N}_1$. Then as X was obtained from N by removing some column vector m_i belonging to $nvcol(C)$, we know that $monom(m_i)$ does not vanish for any solution satisfying C. Thus, $C_i \neq 0$. Then we may divide by C_i, and by rearranging terms we get

$$m_i = \frac{-C_1}{C_i} m_1 + \frac{-C_2}{C_i} m_2 + \ldots + \frac{-C_{i-1}}{C_i} m_{i-1} + \frac{-C_{i+1}}{C_i} m_{i+1} + \ldots + \frac{-C_{s_2}}{C_i} m_{s_2}.$$

Therefore, as the column vector m_i can be expressed as a linear combination of the other column vectors, removing it does not decrease the rank of the matrix X. So, $\text{rank}(X) = \text{rank}(N)$, as desired.$\diamondsuit$

This lemma leads to a testable necessary condition for an affine zero to exist in the case of general polynomials, and calls for more notation, again following the work of Kapur, Saxena and Yang [KSY94]. Let $\mathcal{K} = \mathbb{F}_q[a_1, a_2, \ldots, a_k]$, for indeterminates $a_1, a_2, \ldots, a_k$. Let H be a set of $n+1$ polynomials. However, this time, let $H \subset \mathcal{K}[x_1, x_2, \ldots, x_n] = \mathbb{F}_q[x_1, x_2, \ldots, x_n, a_1, a_2, \ldots, a_k]$, with Dixon matrix D having entries in $\mathcal{K}$. Let C be a set of constraints of form (3.2). Let $r = \text{rank}(D)$. Let

$$\begin{aligned} \mathcal{D}_1 &= \{X \mid X \text{ is an } s_1 \times (s_2 - 1) \text{ submatrix of } D \text{ obtained by} \\ &\quad \text{deleting a column which belongs to } nvcol(C) \text{ from } D\}. \end{aligned}$$

Now, let $\phi : [a_1, a_2, \ldots, a_k] \longrightarrow \overline{\mathbb{F}_q}$ be an assignment of values to the indeterminates $a_1, a_2, \ldots a_k$ from $\overline{\mathbb{F}_q}$. Define $\phi(H) \subset \overline{\mathbb{F}_q}[x_1, x_2, \ldots, x_n]$ to be the set H after applying the indeterminate assignment of ϕ. Similarly, for a matrix M with entries in $\mathcal{K}$, we define a matrix $\phi(M)$ with entries in $\overline{\mathbb{F}_q}$ by performing ϕ on each entry, and then removing all zero rows and columns that arise due to the assignment. We use this definition to get $\phi(D)$ and $\phi(\mathcal{D}_1)$. Then let

$$\mathcal{R} = \{Y \mid Y \text{ is an } r \times r \text{ nonsingular submatrix of } D\}.$$

Since D has rank r, $\mathcal{R}$ is nonempty, and of course for every element $Y \in \mathcal{R}$ we know that $\det(Y) \neq 0$. Thus, Kapur, Saxena and Yang [KSY94] state and prove the following theorem.

Theorem 3.16 (RSC Criteria) *Let H, D, $\mathcal{D}_1$, and ϕ be as defined in the previous paragraph, and let C be as in (3.2). Assume that $\phi(H)$ has a common affine zero $x_1, x_2, \ldots, x_n$ which satisfies C. Then, if there exists $X \in \mathcal{D}_1$ such that rank$(X) <$ rank(D), then for all $Y \in \mathcal{R}$, we have that* $\det(\phi(Y)) = 0$.

Proof: Let $G = \phi(H)$. Then let $N = \phi(D)$, and note that N is the Dixon matrix of G. Similarly, note that $\mathcal{N}_1$, as defined in (3.4), is given by $\mathcal{N}_1 = \phi(\mathcal{D}_1) - N$. So, by Lemma 3.15, if G has a common zero which satisfies C, then for every $X \in \mathcal{D}_1$, we have

$$\text{rank}(\phi(D)) = \text{rank}(\phi(X)).$$

Of course, the indeterminate assignment of ϕ cannot increase the rank of a matrix, so we also know that for every $X \in \mathcal{D}_1$,

$$\text{rank}(\phi(X)) \leq \text{rank}(X).$$

The last piece of information is to note that if the initial condition is true, we have some matrix $X_1 \in \mathcal{D}_1$ such that

$$\text{rank}(X_1) < \text{rank}(D).$$

Putting these statements together, we see that for X_1,

$$\text{rank}(\phi(D)) = \text{rank}(\phi(X_1)) \leq \text{rank}(X_1) < \text{rank}(D) = r.$$

In other words, $\text{rank}(\phi(D)) < r$. So, all submatrices of $\phi(D)$ must have a rank $< r$. Every element $Y \in \mathcal{R}$, after the indeterminate assignment is performed, has less than full rank r. So, every such matrix, after indeterminate assignment, is singular. Therefore, $\det(\phi(Y)) = 0$, as desired.$_\diamondsuit$

Note that the condition "if there exists $X \in \mathcal{D}_1$ such that $\text{rank}(X) < \text{rank}(D)$" of Theorem 3.16 is the *RSC Criteria* (Rank Submatrix Construction Criteria). For any set of polynomials F with a Dixon matrix that satisfies the RSC Criteria, any element $Y \in \mathcal{R}$ is considered the *KSY Dixon matrix*, and $\det(Y)$ is the *extended Dixon resultant*.

3.3.3 The DR algorithm

Tang and Feng [TF05] created an algorithm that uses the extended Dixon resultant to solve, given a public key $\mathcal{P}$ of an $\mathcal{MQ}$-based encryption scheme and ciphertext y, the polynomial system

$$\mathcal{P}(x) - y = 0$$

for the plaintext x. Note that if RSC Criteria is not met, this algorithm will fail.

The algorithm DR, as described by Tang and Feng [TF05], assumes that there are n indeterminates and n polynomial equations. So, here, the number of polynomial equations m is set to $m = n$.

Algorithm 3.17 (DR Algorithm)
Input: *A system $H(x)$ of multivariate quadratic equations with n indeterminates $x_1, x_2, \ldots, x_n$ over a finite field $\mathbb{F}_q$. Note that $H(x) = \mathcal{P}(x) - y = 0$ in terms of the encryption scheme.*

Output: *At least one solution of the input system.*

Step 1: *Compute the Dixon matrix $D(x_n)$ for H, with the entries of D taken from $\mathbb{F}_q[x_n]$.*

Step 2: *Choose a $v \in \mathbb{F}_q$ uniformly randomly.*

Step 3: *Let $D' = D(v)$.*

Step 4: *Perform Gaussian Elimination on D'. Call the result of this E. Let $r = rank(E)$.*

Step 5: *Let $(s_1, s_2) := (NumRows(E), NumColumns(E))$. If $s_1 = s_2 = r$, then let $Y = D$, and goto step 8.*

Step 6: *For $1 \leq i \leq s_2$ do:*
Delete the ith column of E; call the result E_i.
If $rank(E_i) < r$, then goto step 7.
End loop.
Goto step 2.

Step 7: *Perform Gaussian elimination on D to choose the columns needed to construct an $r \times r$ submatrix of D with rank r. Perform Gaussian elimination*

again to choose the rows needed to construct said submatrix. Call the submatrix of D formed by these rows and columns Y.

Step 8: *Calculate* $\det(Y)$.

Step 9: *Solve* $\det(Y)$ *for its roots (e.g., with Berlekamp's algorithm[vzGG03, Section 14.8]). Let $\mathcal{S}$ be the set of roots found.*

Step 10: *For each root $w \in \mathcal{S}$, substitute it into Y. Then solve the linear system $Y(w)$ for its indeterminates $v_1, v_2, \ldots v_r$. Let $\mathcal{U}$ be the set of r-tuples discovered in this way.*

Step 11: *If Step 10 failed to find a valid solution to Y, let $\mathcal{S} = \{0, v\}$, and rerun Step 10.*

Step 12: *For each r-tuple of $[v_1, v_2, \ldots v_r] \in \mathcal{U}$, use the indeterminates $v_1, v_2, \ldots v_r$ to determine the indeterminates $x_1, x_2, \ldots, x_{n-1}$. Store the set of $\{n-1\}$-tuples found this way in a set $\mathcal{W}$*

Step 13: *Return the subset of $\mathcal{W}$ that actually correspond to common roots of H.*

Proof that Algorithm 3.17 terminates and outputs the desired result:

First, we must assume that the RSC Criteria will hold for the Dixon Matrix D given by H. If not, then this algorithm may fail. Kapur, Saxena and Yang [KSY94] noted, and so did Tang and Feng[TF05] note that in all their simulations, the RSC Criteria always held. Neither group offered an explanation for why this RSC Criteria holds so often.

Step 1 merely computes a Dixon matrix. The random element chosen in Step 2 is equivalent to the function ϕ of Theorem 3.16. Steps 3 through 6 check the RSC Criteria of the same theorem using an implicitly chosen set of constraints $C = \{x_1 \neq 0 \wedge \cdots x_n \neq 0\}$. It is assumed that the RSC Criteria will hold. Note that the equality of Step 5 occurs only if the original Dixon matrix D was a square matrix of full rank r. If the loop of Step 6 completes without being broken to go to Step 7, then the choice of $x_n = v$ was bad. The choice of v can only be bad if $x_n = v$ is part of a solution $x_1, x_2, \ldots, x_{n-1}, v$ to the system $\mathcal{P}(x) - y = 0$. The solution is to choose a different v and repeat the process. Step 7 computes a matrix Y as in Theorem 3.16, which is the KSY Dixon matrix. Step 8 computes the KSY Dixon resultant. Step 9 is used to determine for which values of x_n the KSY Dixon resultant is 0. Then we know that the function ϕ must evaluate x_n to one of the values found in Step 9 in order for Theorm 3.16 to hold. Step 10 attempts to solve the linear system given by the KSY Dixon matrix for some of the indeterminates $v_1, v_2, \ldots, v_{s_2}$. Here, without loss of generality, we call them $v_1, v_2, \ldots, v_r$.

Step 10, similar to Step 6, can fail. It can fail if an affine zero exists only for $x_n = 0$, which the constraints C forbid, and therefore do not check. It can also fail if an affine zero exists for $x_n = v$. As we assume that an affine zero exists and that the RSC Criteria holds, if Step 10 fails to find a necessary condition for an affine zero, then one of these last two values must provide such a necessary condition. The answer here is to retry Step 10 using the last two possibilities for x_n, namely $0, v$, as indicated by Step 11.

Step 12 uses the indeterminates $v_1, v_2, \ldots v_r$ to determine the indeterminates

$x_1, x_2, \ldots, x_{n-1}$ by recalling that each v_i is a monomial in $x_1, x_2, \ldots, x_{n-1}$. In fact, it is likely that some of the v_i are equivalent to monomials x_j, allowing one to read off the indeterminates $x_1, x_2, \ldots, x_{n-1}$ from the indeterminates $v_1, v_2, \ldots v_r$.

Step 13 simply checks all the potential common roots found.

Note that only Step 12 is inexact. It is possible that that the column corresponding to the monomial $1 = x_1^0 x_2^0 \cdots$ in the Dixon matrx D is a linear combination of the other columns, in which case it is not possible to fully define the solution set of H, as there will be no constant term to "start" a fully defined solution using the KSY Dixon matrix K. It is also possible that the indeterminates $v_1, v_2, \ldots v_r$ are not equivalent to monomials proper for computing the entirety of $x_1, x_2, \ldots, x_{n-1}$. In this case, the remaining terms can be chosen using a brute force search.$_\diamond$

3.3.4 Runtime of the DR algorithm

Tang and Feng [TF05] note that their algorithm is normally dominated by Gaussian elimination on the Dixon matrix, which is an $s_1 \times s_2$ matrix, in Step 4. They note that even if $x_n = v$ is part of a common root $(x_1, x_2, \ldots, x_n)$ of F, only a small finite number of Gaussian eliminations are required, in the case of their testing, 3 was usually the number necessary. So, the algorithm's overall complexity is based on the complexity of performing Gaussian eliminations on a matrix of size $s_1 \times s_2$. Then it is necessary to estimate s_1 and s_2. Tang and Feng [TF05] note that each of s_1 and s_2 are bounded above by $\frac{(2n)!}{n!(n+1)!}$. They note that if $f(x) = \frac{(2x)!}{x!(x+1)!}$, then as $\frac{f(x+1)}{f(x)} = 2\frac{2x+1}{x+2}$, we have

$$\lim_{x \to \infty} \frac{f(x+1)}{f(x)} = 4.$$

As $2\frac{2x+1}{x+2} \leq 4$ for all positive x, we have 4^n as upper bound on s_1 and s_2 for all n. Therefore, the DR algorithm has a complexity of $O(4^{n\omega})$, where Gaussian elimination on an $n \times n$ matrix is assumed to have complexity $O(n^\omega)$. For example, one possible value is $\omega = 3$. In fact, for $22 \leq n \leq 80$, it is possible to estimate an upper bound on s_1 and s_2 of 3.5^n, and even smaller values as n approaches 1.

Finally, Tang and Feng [TF05] note that their technique is effective by generating various small, nonsparse systems of equations and comparing the runtime of the DR algorithm with the Buchberger algorithm in Maple 9.5 and Faugère's F_4 algorithm in Singular 3.0. The conclusion that they draw is that the DR algorithm is significantly faster, often by orders of magnitude, than the other algorithms used. These experiments are by no means conclusive (they are all done over $\mathbb{F}_{127}$ and for systems of no more than 8 equations and indeterminates), but they do merit further review.

One aspect of Tang and Feng's work not used here is that Tang and Feng [TF05] claim that for certain sparse systems, the running time of the DR algorithm is polynomial in n, the number of equations and indeterminates. Generic

$\mathcal{MQ}$-based encryption schemes do not necessarily involve sparse systems at all, so such a claim is less useful than one might hope from a cryptanalysis point of view.

3.4 MinRank-based attacks

Solving instances of the MinRank problem is another technique employed in some attacks on $\mathcal{MQ}$-based encryption schemes. Typically speaking, such attacks aim to break apart the public key $\mathcal{P} = T \circ \mathcal{P}' \circ S$ into the private key $(T, \mathcal{P}', S)$. However, they are usually tied rather tightly to the underlying cryptosystem, making them hard to describe given the current generic example. Still, since MinRank is used in more than one attack, and on quite different cryptosystems, it makes sense to think of it as a somewhat more generic technique than some others.

The MinRank problem is a deceptively simple problem in linear algebra. It is restated here from [GC00].

Definition 3.18 (MinRank Problem) *Let r be an integer and let $\mathbb{F}$ be a field. Then the MinRank(r) problem is the following: given a set $\{M_1, M_2, \ldots, M_\ell\} \in \mathbb{F}^{n \times n}$, find a non-zero ℓ-tuple $(\lambda_1, \lambda_2, \ldots, \lambda_\ell) \in \mathbb{F}^\ell$ such that Rank $\left(\sum_{i=1}^{\ell} \lambda_i M_i\right) \leq r$.*

The MinRank problem usually arises when considering the public key $\mathcal{P} = (p_1, p_2, \ldots, p_m)$. One can consider the quadratic part of each p_i to be given by a matrix multiplication of the form $x^t M_i x$, where $x = (x_1, x_2, \ldots, x_n)$, and $M \in \mathbb{F}^{n \times n}$. By viewing the public key in this way, it is possible to consider various MinRank problems, some of which are useful in specific attacks.

The idea behind a MinRank-based attack is to substitute one $\mathcal{NP}$-hard problem, the $\mathcal{MQ}$ problem, for another one, the MinRank(r) problem [SFB96]. The hope is that it is faster to solve an instance of the MinRank(r) problem and use that solution to determine the private key of an $\mathcal{MQ}$-based encryption scheme than it would be to directly solve an instance of an $\mathcal{MQ}$-based encryption scheme for its private key.

3.4.1 Goals of MinRank attacks

The goal of a MinRank-based attack is to learn the private key $(T, \mathcal{P}', S)$, starting with the function T. Once this is known, the attack proceeds to uncover the quadratic coefficients of $\mathcal{P}'$ and the function S. Once these parts of the private key are known, the linear and constant coefficients of $\mathcal{P}'$ can be discovered using gaussian elimination. The details of some such attacks will be provided later, in Section 5.2.

3.4.2 Solving MinRank(r)

The MinRank(r) problem is shown in [SFB96] to be $\mathcal{NP}$-hard. Of course, this does not preclude specific instances of the problem being much easier to solve. For example, it is clear that if $r > n$, the MinRank(r) problem is totally trivial. Solving a MinRank(r) problem also varies with the system from which it arises. Typically, either a heuristic is given to solve an overly-constrained (many more equations than indeterminates) $\mathcal{MQ}$ problem in expected polynomial time (thus solving the MinRank(r) problem, where the matrices are quadratic coefficients of a set of polynomials), or some effort is made to show that a process of repeated guessing and checking is sufficient to mount an expeditious attack. More information will follow when details of such attacks are discussed in Section 5.2.

3.5 Differential cryptanalysis

Differential cryptanalysis was introduced by Fouque, Granboulan and Stern [FGS05] to attack the PMI $\mathcal{MQ}$-based encryption scheme. The PMI cryptosystem was introduced by Ding [Din04] (The PMI cryptosystem is also described here in Section 4.5, with the associated attack from [FGS05] described in Section 5.5.). The technique of differential cryptanalysis is also used in [FGS05] to attack the MIC* cryptosystem (which is described in Section 4.4), so the technique is not unique to the PMI cryptosystem.

The differential cryptanalysis techniques of [FGS05] are designed to create practical decryption functions. That is, for any given ciphertext y, its associated plaintext x will be computed using a calculated function rather than resolving a system for each ciphertext y, as in a Gröbner basis attack. The primary tools used to determine such a function are the differential and the bilinear function.

3.5.1 Differentials and bilinear functions

As noted in [FGS05], the following definitions are useful when dealing with differential cryptanalysis attacks on $\mathcal{MQ}$-based schemes. First and foremost, let $\mathcal{G} : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^m$ be any function, and let $k \in \mathbb{F}_q^n$. Then, the *differential* of $\mathcal{G}$ with respect to k is $d\mathcal{G}_k(x)$, which is given by

$$d\mathcal{G}_k(x) = \mathcal{G}(x + k) - \mathcal{G}(x).$$

If $\mathcal{G}$ is quadratic, as it is if it is the public key of an $\mathcal{MQ}$-based encryption scheme, then $d\mathcal{G}_k$ is an affine function. In this case, it is possible to go one step further and compute the linear part of the differential, $L_{\mathcal{G},k}(x)$, given by

$$L_{\mathcal{G},k}(x) = d\mathcal{G}_k(x) - d\mathcal{G}_k(0).$$

Then it is also true that $L_{\mathcal{G},k}(x)$ is linear in the term k, as $L_{\mathcal{G},k}(x) = \mathcal{G}_k(x + k) - \mathcal{G}_k(x) - \mathcal{G}_k(k) + \mathcal{G}_k(0)$. In this case, $L_{\mathcal{G},k}(x)$ is a bilinear function $B_{\mathcal{G}}(x, k)$, where

$$B_{\mathcal{G}}(x, k) = L_{\mathcal{G},k}(x).$$

3.5.2 Attack methods using differential cryptanalysis

The idea of attack methods using differential cryptanalysis is to use differentials of quadratic functions $\mathcal{P}$ to get bilinear functions $L_{\mathcal{P},k}(x) = B_{\mathcal{P}}(x,k)$. The goal is to find clever ways to take these differentials, and clever ways to manipulate the bilinear maps in order to generate a bilinear system in terms of the plaintexts x and the ciphertexts y. The main attack of Fouque, Granboulan and Stern [FGS05], which does just that, is reproduced in Section 5.5 in this work.

3.6 Specific attacks

Sometimes a given $\mathcal{MQ}$-based encryption scheme has a design flaw allowing an attack on that particular system of a type not previously mentioned. It is rare that the attack methods used in such an attack lend themselves to use in other attacks on other $\mathcal{MQ}$-based encryption schemes. A specific attack against a particular $\mathcal{MQ}$-based encryption scheme will be explained in detail in Section 5.4.

Chapter 4

Selected $\mathcal{MQ}$-Cryptosystems

4.1 Preliminaries

Often, a family of $\mathcal{MQ}$-based encryption schemes uses a public key function $\mathcal{P}$ that is not a bijection on its image. Because of this, it is necessary to create some redundancy in the domain of the function $\mathcal{P}$. In these instances, rather than create our plaintext $x = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_q^n$ by a simple embedding of the original message M into $\mathbb{F}_q^n$, something slightly more complicated occurs.

In these cases, the plaintext x is generated from the message M in the following way. First, let H be a cryptographically secure hash function, with output at least 64 bits. Patarin [Pat96b] recommends that at least 64 bits of redundancy be used to ensure that the messages encrypted can be decrypted, so embed the original text M concatenated with the first 64 bits of $H(M)$ into $\mathbb{F}_q^n$, to form our plaintext x.

A side effect of this fact is the following–decryption techniques will create a set of candidates. The correctly deciphered plaintext x will be the one with the property $x = M||H(M)$.

4.2 The STS family

The Step-wise Triangular Schemes (STS) family was introduced by Wolf, Braeken and Preneel [WBP04]. They are a somewhat artificial family, introduced solely to showcase a new attack on a pair of schemes proposed by Kasahara and Sakai [KS04b, KS04a]. The STS family itself contains the Triangle Plus-Minus (TPM) family, another artificial family of cryptosystems introduced to showcase an attack, this time by Goubin and Courtois [GC00]. As mentioned in Chapter 2 the trapdoor technique used is the primary piece of information required to understand the unique features of an $\mathcal{MQ}$-based encryption system's encryption, de-

cryption, and key generation techniques. Here, the trapdoor technique is to create a "triangle" out of the indeterminates in the equations $\mathcal{P}' = (p'_1, p'_2, \ldots, p'_m)$ that make up the private key, where p'_1 uses $r+1$ indeterminates, and each following p_i uses a superset of the indeterminates that were in the previous p_{i-1}, until all n indeterminates are in use. A totally triangular shape would use n equations, have $r = 0$, and introduce a single additional indeterminate at each private equation.

4.2.1 Key generation

The STS family uses a number of parameters. First, a scheme from the STS family uses a field of q elements for all operations, $\mathbb{F}_q$. The triplet $(T, \mathcal{P}', S)$ making up the private key uses m equations and n indeterminates. There are L layers or steps in an STS scheme.

$$
\begin{array}{ll}
\text{Step 1} & \left\{ \begin{array}{lcl} p'_1 & = & g_1(x_1, x_2, \ldots, x_{a_1}) \\ \vdots & \vdots & \\ p'_{m_1} & = & g_{m_1}(x_1, x_2, \ldots, x_{a_1}) \end{array} \right. \\
\text{Step 2} & \left\{ \begin{array}{lcl} p'_{m_1+1} & = & g_{m_1+1}(x_1, x_2, \ldots, x_{a_1}, x_{a_1+1}, \ldots, x_{a_1+a_2}) \\ \vdots & \vdots & \\ p'_{m_1+m_2} & = & g_{m_1+m_2}(x_1, x_2, \ldots, x_{a_1}, x_{a_1+1}, \ldots, x_{a_1+a_2}) \end{array} \right. \\
\vdots & \\
\text{Step } L & \left\{ \begin{array}{lcl} p'_{m-m_L+1} & = & g_{m-m_L+1}(x_1, \ldots, x_n) \\ \vdots & \vdots & \\ p'_m & = & g_m(x_1, \ldots, x_n). \end{array} \right.
\end{array}
$$

The positive integers $a_1, a_2, \ldots, a_L$ are the number of new indeterminates introduced at each step—they are the width of each step i, for $1 \leq i \leq L$. The positive integers $m_1, m_2, \ldots, m_L$ are the number of polynomials used for each step—they are the height of each step i. It is the case that $a_1 + \cdots + a_L = n$ and $m_1 + \cdots + m_L = m$, as one might expect. In terms of the metaphor, the stairs go all the way down.

The TPM family is a special case of the STS family in the following way: in the TPM family,

$$a_2 = a_3 = \cdots = a_L = m_1 = m_2 = \cdots = m_{L-1} = 1.$$

The remaining values, a_1 and m_L, represent the number of indeterminates used in the initial equation, and the number of equations that utilize all n indeterminates, respectively. The idea is that in the TPM family, the perfect triangle shape has had some $(a_1 - 1)$ equations subtracted from it and some $(m_L - 1)$ additional equations using all the indeterminates added to it. Goubin and Courtois [GC00] noted that so long as $a_1 \leq m_L$, the probability this system of equations is not injective is negligible.

The private key is computed in the following way. First, the affine functions T and S are computed in the usual way, as described in Section 2.3.1. Second, for each of the private polynomials p'_i, the polynomial g_i will be a random quadratic polynomial using the first $a_1 + \cdots + a_j$ indeterminates, where j is the smallest positive integer such that $i \leq a_1 + \cdots + a_j$. So,

$$p'_i = g_i(x_1, \ldots, x_{a_1+\cdots+a_j}).$$

This g_i is determined by choosing the coefficients using a uniformly random number generator over the finite field $\mathbb{F}_q$. The public key $\mathcal{P}$ is determined as usual, $\mathcal{P} = T \circ \mathcal{P}' \circ S$.

4.2.2 Encryption

In an STS system, encryption is accomplished by embedding a message M into $\mathbb{F}_q^n$, forming a plaintext x, using the technique described in Section 4.1 for $\mathcal{MQ}$-based encryption schemes that may not be bijective. The public key $\mathcal{P}$ is then used to compute $\mathcal{P}(x)$, which is sent out as the ciphertext y.

4.2.3 Decryption

Decryption is a step-wise process. The recipient begins with a ciphertext y, where

$$y = \mathcal{P}(x) = T \circ \mathcal{P}' \circ S(x).$$

First, the recipient applies T^{-1} to this ciphertext, forming

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

After this, the recipient performs a sequence of exhaustive searches on each "step" of the private set of polynomial equations $\mathcal{P}'$. At the first step, an exhaustive search of the a_1 indeterminates requires $O(q^{a_1})$ indeterminate assignments. Once the a_1-tuples satisfying the first m_1 equations are found, there is a recursive loop, as for each valid set of choices for $x_1, x_2, \ldots, x_{a_1}$, a new system is created by assigning the indeterminates $x_1, x_2, \ldots, x_{a_1}$ each set of possible values. Then the next step can be solved, using the same technique. After all L steps, there will be a set of valid x values. The correct x value will be the one of the form $x = M||H(M)$ (see Section 4.1).

Note that these exhaustive searches are bounded in speed by the values $a_1, a_2, \ldots, a_L$, so it is imperative that no step be too wide, or the stairs will take too long to climb down, and the system will be unusably slow as an encryption scheme.

4.2.4 Suggested Parameters

Wolf, Braeken and Preneel's STS family was created to generalize an attack on the TPM family of Goubin and Courtois [WBP04, GC00]. Both families are quite general. A pair of schemes RSSE(2)PKC and RSE(2)PKC introduced by

Kasahara and Sakai [KS04b, KS04a] are shown by Wolf *et al.* [WBP04] to be instances of the STS family, and use the following parameters. They use the base field $\mathbb{F}_2$. Both set a value r such that

$$r = a_1 = \cdots = a_L = m_1 = \cdots m_L.$$

This greatly simplifies much of the notation, and also makes it highly unlikely [GC00], although not impossible, that at any given step, there will be more than one possible solution.

Each of these systems is susceptible to the STS attacks of Wolf *et al.* [WBP04], so they are by no means secure. One particular challenge involved an instance with $r = 5$, $n = 100$, which, by the rules for r just listed, means that $m = 100$ and $L = 20$. It was beaten by Wolf *et al.* [WBP04] in a matter of hours using an AMD Athlon XP 2000+.

Similarly, TPM schemes are not considered secure. The main scheme considered by Goubin and Courtois [GC00] was the TTM cryptosystem, which uses the following parameters:

$$q = 256, \quad n = 64, \quad m = 100, \quad a_1 = 2, \quad m_L = 38,$$

recalling that as a TPM scheme, all steps consist of 1 equation with one new indeterminate, except the first step, which introduces $a_1 = 2$ new indeterminates and 1 new equation in this case and the last step, which introduces 1 new indeterminate and m_L new equations. Goubin and Courtois [GC00] note that they have an attack on TTM that uses $O(2^{52})$ elementary operations.

Details of attacks on STS and TPM are given in Section 5.2.

4.3 Basic HFE

The Hidden Field Equations (HFE) family of $\mathcal{MQ}$-based encryption schemes includes a number of variants. Basic HFE, which will be explained here, forms the basis for the rest. HFE was introduced by Patarin [Pat96b], as an extension of the ideas of Matsumoto and Imai [MI88]. However, it is sufficiently different from the MIC* system of [MI88] to be considered its own system. MIC* itself is described in Section 4.4. The trapdoor technique used in Basic HFE is to use a certain kind of univariate equation in $\mathbb{F}_{q^n}$ as the basis of the private key element $\mathcal{P}'$. In order to efficiently do this, we use a bijection from the n-tuples of elements of $\mathbb{F}_q$ to $\mathbb{F}_{q^n}$. These bijections are easily found, as they correspond to the monic irreducible polynomials of degree n in $\mathbb{F}_q[z]$, for a univariate indeterminate z. Unfortunately, the HFE system requires redundancy in the plaintexts it uses, as the function $\mathcal{P}'$ is rarely invertible.

4.3.1 Key generation

In a basic HFE encryption scheme, there are as many indeterminates as there are equations, so $n = m$. The private key of an HFE system can be considered,

as always, a triple (T, $\mathcal{P}'$, S). T and S are formed in the usual way (see Section 2.3.1). Let ϕ be an isomorphism

$$\phi : \mathbb{F}_q^n \longrightarrow \mathbb{F}_{q^n}.$$

ϕ can be constructed by forming uniformly random monic univariate polynomials $g(z)$ of degree n over $\mathbb{F}_q$ until an irreducible one is found. This irreducible polynomial $g(z)$ defines the isomorphism ϕ in the usual way–

$$\begin{array}{rccc} \phi: & \mathbb{F}_q^n & \longrightarrow & \mathbb{F}_q[x]/\left(g(x)\right) \\ & (a_0, \ldots, a_{n-1}) & \longmapsto & \sum_{i=0}^{n-1} a_i x^i. \end{array}$$

Since $g(z)$ generates a maximal ideal of $\mathbb{F}_q[z]$, the commutative ring $\mathbb{F}_q[z]/\langle g(z)\rangle$ is a field, and that field has q^n elements. Let $\mathcal{Q}$ be a univariate polynomial over $\mathbb{F}_{q^n}$, defined in the following way

$$\mathcal{Q}(z) = \sum_{\substack{a,b\in\mathbb{Z}_{\geq 0} \\ q^a+q^b\leq d}} \alpha_{a,b} z^{q^a+q^b} + \sum_{\substack{c\in\mathbb{Z}_{\geq 0} \\ q^c\leq d}} \beta_c z^{q^c} + \gamma.$$

The coefficents of $\mathcal{Q}$ are chosen uniformly randomly, with a maximum degree d, which, amongst other things, prevents polynomials of infinite degree. Kipnis and Shamir suggest [KS99] a maximum value of $d = 8192$. In order to see that $Q(z)$ is in fact a quadratic function, first recall that in $\mathbb{F}_{q^n}$, the function $g(z) = z^q$ is a *linear function*:

$$g(z_1 + z_2) = g(z_1) + g(z_2) \text{ and } g(k \cdot z_1) = kg(z_1) \text{ for } z_1, z_2 \in \mathbb{F}_{q^n}, k \in \mathbb{F}_q.$$

Therefore, z^{q^c} is a composition of linear functions, and thus also linear. Now a small theorem is requred.

Theorem 4.1 *Let f be a linear function over a finite field $\mathbb{F}_{q^n}$. Then f is either the zero function or a bijection; furthermore, f can be written as $f(x) = cx$ for some constant $c \in \mathbb{F}_{q^n}$.*

Proof: First note that f is a linear transformation over $\mathbb{F}_{q^n}$ seen as a vector space. Therefore, it has a kernel of dimension 0 or dimension 1. If $\dim(\ker(f)) = 1$, then f is the zero function. If $\dim(\ker(f)) = 0$, then f is a bijection. To see that $f(x) = cx$, for some constant $c \in \mathbb{F}_{q^n}$, let $c = f(1)$. Then the function $f(x) - cx$ is also linear, as $f(x)$ and cx are linear functions, and the sums and differences of linear functions are linear. Therefore, since the function $f(x) - cx$ has at least 2 roots $x = 1$ and $x = 0$, we have that $\dim(\ker(f(x) - cx)) = 1$, and $f(x) - cx$ is the zero function. So $f(x) = cx$, as desired.$\diamond$

Theorem 4.1 implies that the product of 2 linear univariate functions is a quadratic function in the usual sense; if f and g are linear functions, then by Theorem 4.1, they may be written as $f(x) = cx$ and $g(x) = dx$, respectively. So their product is a quadratic function $f(x)g(x) = cdx^2$. Therefore, the terms of the form $z^{q^a+q^b} = z^{q^a} z^{q^b}$ are a product of linear functions of z, so they are

quadratic functions of z. Thus, basic HFE is an $\mathcal{MQ}$-based scheme. Then the private key element $\mathcal{P}'$ may be constructed from these components, forming $\mathcal{P}' = \phi^{-1} \circ \mathcal{Q} \circ \phi$. However, this is not actually done. Instead of computing $\mathcal{P}'$ and storing it, the polynomial $\mathcal{Q}$ and the invertible function ϕ are stored instead. $\mathcal{Q}$ is stored using its coefficients, while ϕ is stored as an invertible matrix, since it is a linear transformation. The public key $\mathcal{P}$ is given by $\mathcal{P} = T \circ \mathcal{P}' \circ S$, as usual, noting that in this case, $\mathcal{P}'$ itself is never directly computed—it is merely a placeholder for $\phi^{-1} \circ \mathcal{Q} \circ \phi$.

Note that the term hidden field equations comes from the fact that "hidden" inside the public key polynomial system is a univariate polynomial system over an extension field of the field used in the public key itself.

4.3.2 Encryption

In an HFE system, encryption is accomplished by embedding a message M into $\mathbb{F}_q^n$, forming a plaintext x, using the technique described in Section 4.1 for $\mathcal{MQ}$-based encryption schemes that may not be bijective. The public key $\mathcal{P}$ is then used to compute $\mathcal{P}(x)$, which is sent out as the ciphertext y.

4.3.3 Decryption

Let $y = \mathcal{P}(x)$ be the ciphertext for some unknown plaintext x. The intended recipient of the ciphertext begins applying the private key in reverse order, as

$$y = (T \circ \phi^{-1} \circ \mathcal{Q} \circ \phi \circ S)(x).$$

T and ϕ^{-1} are invertible, and represented by matrices, so not only does $\phi \circ T^{-1}$ exist, it can be computed with simple matrix manipulations. Therefore,

$$\phi \circ T^{-1}(y) = \mathcal{Q} \circ \phi \circ S(x),$$

is easily computed. Now, however, we are working with a univariate polynomial in x of degree $\leq d$. Many techniques exist to find the roots of such polynomials, such as Berlekamp's algorithm [vzGG03, 14.8]. Let $\mathcal{R}$ denote the set of all solutions x to

$$0 = (\mathcal{Q} \circ \phi \circ S)(x) - \phi \circ T^{-1}(y).$$

Note that only roots in the field $\mathbb{F}_{q^n}$ itself are found by Berlekamp's algorithm, and only those roots are desired. Note also that finding the roots of a polynomial is bounded in difficulty by the degree of that polynomial. This is the reason that Kipnis and Shamir [KS99] suggested a maximum value of $d = 8192$, as for larger values root-finding algorithms perform too slowly. The set $\mathcal{R}$ is the set of all possible preimages that could lead to y, so

$$\mathcal{R} = \left\{ z \in \mathbb{F}_{q^n} \mid 0 = (\mathcal{Q}(z) - \phi \circ T^{-1}(y) \right\}.$$

Now construct the set $S^{-1}(\phi^{-1}(\mathcal{R}))$ from $\mathcal{R}$. For this set, we have

$$S^{-1}(\phi^{-1}(\mathcal{R})) = \left\{ x \in \mathbb{F}_q^n \mid 0 = (\mathcal{Q} \circ \phi \circ S)(x) - \phi \circ T^{-1}(y) \right\}.$$

Thus, it is exactly the set of possible plaintexts x that could encrypt to the ciphertext y. The correct plaintext is the element of this set having the property listed in Section 4.1.

4.3.4 Suggested parameters

While attacks against basic HFE do exist (see Section 5.3 for details), they are subexponential, and certain parameter choices push one outside their bounds. For starters, Patarin [Pat96b] suggested using plaintexts at least 128 bits long to avoid brute force searching. Courtois [Cou05] recommended using $q = 2$. For n, the number of equations and indeteriminates, he recommended that $n \geq 127$, in order to avoid attacks that threaten small values of n. He also recommended that n be prime, to avoid the possibility of attacking a system through subfields. Such attacks have been reported against $\mathcal{MQ}$-based signature schemes that use coefficients from proper subfields, as noted by Wolf and Preneel [WP04]. As for d, the degree of the hidden field equation $\mathcal{Q}$, provided $n \geq 127$, he notes $25 \leq d \leq 33$ is sufficient to resist all known attacks. Courtois also gave an example of choices that lead to a system unassailable by current attacks, namely $q = 2$, $n = 251$ and $d = 25$. For a more detailed look at the security of HFE, see Section 6.2.1.

4.3.5 Toy example

Here is a toy example of HFE key generation, encryption and decryption. We use $q = 2, m = n = 3, d = 3$ as parameters. We begin selecting our private key by fixing ϕ,

$$\phi : (x_1, x_2, x_3) \longmapsto x_1 + x_2\alpha + x_3\alpha^2,$$

where α is a root of $v^3 + v + 1$. We generate a random univariate function $\mathcal{Q}(z) = z^3 + z^2$ in $\mathbb{F}_8$. Next, we generate two random affine transformations T and S. Here, T is given by:

$$T(x) = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}.$$

Similarly, S is given by:

$$S(x) = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}.$$

This gives us a private HFE key. We generate the public key $\mathcal{P}$ by composing these functions $T \circ \phi^{-1} \circ \mathcal{Q} \circ \phi \circ S$. Thus, $\mathcal{P}$ is given by:

$$\begin{aligned} y_1 &= x_1x_2 + x_2x_3 + x_1 + x_2 + x_3 \\ y_2 &= x_1x_2 + x_1x_3 + x_2x_3 + x_2 + x_3 \\ y_3 &= x_1x_3 + 1. \end{aligned}$$

In this example, we still need a level of redundancy in order to determine correct answers versus incorrect answers. So, all plaintexts will consist of two bits of information followed by a parity bit. Therefore, in order to send 11, we encode it as $x = (1, 1, 0)$, then encrypt it by using x as the input to the public key. The result of this encryption is our ciphertext $y = (1, 0, 1)$.

Now we will attempt to decrypt our ciphertext $y = (1, 0, 1)$. To do this, we need the functions ϕ^{-1}, T^{-1} and S^{-1}. For this example, ϕ^{-1} is trivial if we work with $\mathbb{F}_8$ using the polynomial basis $1, \alpha, \alpha^2$. We have T^{-1} given by:

$$T^{-1}(x) = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} x + \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

Similarly, S^{-1} is given by:

$$S^{-1}(x) = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

Now we can continue. We begin by taking $T^{-1}(y)$, which is:

$$T^{-1}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

This maps to $0 \in \mathbb{F}_8$ after applying ϕ. So, now we have to solve the equation

$$z^3 + z^2 = 0$$

for its roots. By inspection, $z = 0$ is a double root and $z = 1$ is the other root. Apply ϕ^{-1} to each of these and we have $(0, 0, 0)$ and $(1, 0, 0)$, respectively. Then apply S^{-1}, and we see that

$$S^{-1} : (0, 0, 0) \longmapsto (1, 0, 0).$$

A simple parity check shows that $(1, 0, 0)$ is not the plaintext. However,

$$S^{-1} : (1, 0, 0) \longmapsto (1, 1, 0)$$

passes the parity check, so we decrypt our original ciphertext $y = (1, 0, 1)$ to $x = (1, 1, 0)$.

4.4 MIC*

Introduced by Matsumoto and Imai [MI88], the MIC* cryptosystem can also be seen as a special case of the basic HFE cryptosystem family, although it should be noted that MIC* came first. Because the nature of the MIC* system makes encryption and decryption significantly different from HFE encryption and decryption, it is listed seperately. An MIC* system requires that the field $\mathbb{F}_q$ is characteristic 2, so $q = 2^k$. This is a requirement that HFE does not have. MIC* systems retain the HFE assumption that $n = m$.

4.4.1 Key generation

The private key of an MIC* encryption scheme is found in a way quite similar to that of the key of basic HFE. As usual, T and S are determined in the same way as any other $\mathcal{MQ}$-based encryption scheme. The function ϕ is determined exactly as in Section 4.3.1, and the private key element $\mathcal{P}'$ is again created using ϕ and a univariate polynomial Q over the extension field $\mathbb{F}_{q^n}$. In the case of the MIC* scheme, however, the hidden field equation Q is much simpler, having the form

$$Q(z) = z^{q^\ell+1}$$

where $\ell \in [1, n-1]$ is uniformly randomly chosen until

$$\gcd(q^\ell + 1, q^n - 1) = 1.$$

Note that Q is quadratic, as it is the product of two *linear functions* in z, namely z^{q^ℓ} and z, where a *linear function* is a function g such that

$$g(z_1 + z_2) = g(z_1) + g(z_2) \text{ and } g(k \cdot z_1) = kg(z_1) \text{ for } z_1, z_2 \in \mathbb{F}_{q^n}, k \in \mathbb{F}_q.$$

This is shown exactly as in Section 4.3.1, using Theorem 4.1 in the same way, as the MIC* family is strictly contained within the HFE family. Thus, this scheme is $\mathcal{MQ}$-based. Now, just like in the case of the HFE system (see Section 4.3.1), the private key is stored as (T, ϕ, Q, S), where the usual middle term $\mathcal{P}'$ is given by $\mathcal{P}' = \phi^{-1} \circ \mathcal{Q} \circ \phi$. The public key is given by $\mathcal{P} = T \circ \mathcal{P}' \circ S$. As in the HFE system, the set of polynomial equations $\mathcal{P}'$ are merely a shorthand for $\phi^{-1} \circ \mathcal{Q} \circ \phi$ in this system.

4.4.2 Encryption

In the MIC* system, encryption is accomplished by embedding a message m into $\mathbb{F}_q^n$, forming a plaintext x. The public key $\mathcal{P}$ is then used to compute $\mathcal{P}(x)$.

4.4.3 Decryption

In the MIC* system, decryption is much more straightforward than in basic HFE systems. The choice of ℓ in generating the private key polynomial Q implies that by Fermat's Little Theorem, there exists an $h \in [1, q^n - 1]$ such that $h(q^\ell + 1) \equiv 1 \pmod{q^n - 1}$. Then, $Q^{-1}(z) = z^h$, as Q is invertible. Note that $\gcd(q^\ell + 1, q^n - 1) = 1$ is the requirement that forces fields of even characteristic to be used in MIC*. In a field of odd characteristic, $\gcd(q^\ell + 1, q^n - 1) \geq 2$ and thus $\mathcal{Q}$ is not necessarily invertible.

Let $y = \mathcal{P}(x)$ be a received ciphertext. The legitimate recipient of this ciphertext simply applies the inverses of each section of the private key in turn, moving from

$$y = (T \circ \phi^{-1} \circ \mathcal{Q} \circ \phi \circ S)(x)$$

to

$$(S^{-1} \circ \phi^{-1} \circ Q^{-1} \circ \phi \circ T^{-1})(y) = x.$$

4.4.4 Suggested parameters

There is an extremely effective attack on MIC* due to Patarin [Pat95]. Because of this, there are no real suggested parameters for this $\mathcal{MQ}$-based encryption scheme. As a historical footnote, Matsumoto and Imai [MI88] created an implementation of MIC* for $q = 2^8$ and $n = 32$.

4.5 Perturbed MIC*

Perturbed MIC* is a variant of MIC* introduced by Ding [Din04]. It is usually abbreviated to PMI. The PMI system attempts to increase the complexity of the private key term $\mathcal{P}'$ in order to increase security, using a system $\mathcal{A}$ of r arbitrary quadratic equations over $\mathbb{F}_q$, with the assumption that $r << n$.

4.5.1 Key generation

Key generation in perturbed MIC* is a strict superset of the process described in Section 4.4.1. The private key becomes still more complicated, as the function $\mathcal{P}'$ is modified using an additional pair of functions. The first function is a linear map $\pi : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^r$. π can be constructed randomly by taking random $r \times n$ matrices until one of rank r is found. The second function is a system $\mathcal{A}$ of n arbitrary quadratic equations in r indeterminates over $\mathbb{F}_q$. Choosing uniformly random coefficients for each term of the n polynomials $(a_1, a_2, \ldots, a_n)$ of $\mathcal{A}$ determines $\mathcal{A}$.

The private key is now $(T, \phi, \mathcal{Q}, \mathcal{A}, \pi, S)$, where ϕ and $\mathcal{Q}$ are constructed and stored as in Section 4.4.1. That is, the usual term $\mathcal{P}'$ in the private key is not stored directly. Rather, since ϕ, $\mathcal{Q}$ and $\mathcal{A}$ are already being stored, and

$$\mathcal{P}' = \phi^{-1} \circ \mathcal{Q} \circ \phi + \mathcal{A} \circ \pi,$$

there is no need to calculate $\mathcal{P}'$ explicitly. The public key, as usual, is $\mathcal{P} = T \circ \mathcal{P}' \circ S$. Note that as in Section 4.4.1, the term $\mathcal{P}'$ is not stored; it is a shorthand for the function compositions and sums used in the PMI system.

4.5.2 Encryption

In a PMI system, encryption is accomplished by embedding a message M into $\mathbb{F}_q^n$, forming a plaintext x, using the technique described in Section 4.1 for $\mathcal{MQ}$-based encryption schemes that may not be bijective. The public key $\mathcal{P}$ is then used to compute $\mathcal{P}(x)$, which is sent out as the ciphertext y.

4.5.3 Decryption

Decryption is somewhat more painful than in MIC* itself. The system $\mathcal{A}$ is an arbitrary system of quadratic equations, so in general it is not invertible. Therefore, typically $\mathcal{A}$ is stored as a set of points P, where

$$P = \left\{(\lambda, \Lambda) \mid \lambda \in \mathrm{Im}(\mathcal{A}), \Lambda = \mathcal{A}^{-1}(\lambda)\right\}.$$

The decryption process is now fairly straightforward. Let $y = \mathcal{P}(x)$ be a received ciphertext. The legitimate recipient of this ciphertext begins with

$$y = (T \circ (\phi^{-1} \circ \mathcal{Q} \circ \phi + \mathcal{A} \circ \pi) \circ S)(x).$$

First, the recipient computes

$$T^{-1}(y) = ((\phi^{-1} \circ \mathcal{Q} \circ \phi + \mathcal{A} \circ \pi) \circ S)(x).$$

Note that this can be rewritten as

$$T^{-1}(y) = (\phi^{-1} \circ \mathcal{Q} \circ \phi \circ S + \mathcal{A} \circ \pi \circ S)(x).$$

This leads to the next step, a brute force search of the points P. For each pair $(\lambda, \Lambda) \in P$, assume that

$$T^{-1}(y) = (\phi^{-1} \circ \mathcal{Q} \circ \phi \circ S)(x) + \lambda.$$

Now it is sufficient to compute

$$(\phi^{-1} \circ \mathcal{Q}^{-1} \circ \phi)(T^{-1}(y) - \lambda) = S(x).$$

If $(\pi \circ S)(x) \in \Lambda$, then compute x from $S(x)$ as usual, and add the computed x to a set of candidate solutions C. If $(\pi \circ S)(x) \notin \Lambda$, then ignore this λ and continue. Once each element of P is considered, there will be a set of possible solutions. Use the standard technique set forth in Section 4.1 to find the plaintext x.

4.5.4 Suggested parameters

Standard PMI can be attacked. This is shown in Section 5.5. Ding [Din04] suggests that a practical implementation of the PMI system be as follows: choose $q = 2$, $n = 136$, and $r = 6$. Let the function $Q(z) = z^{2^{5\times 8}+1}$. He believed that the attack complexity of his given system was on the order of 2^{100} operations. However, the attack of Fouque, Granboulan and Stern [FGS05] indicates a security level on the order of 2^{49} operations. Ding and Gower [DG05] later recommended that in general, $r \geq 6$, based on computer simulations. For $r = 6$, they suggest $n = 100$. The attack of Fouque, Granboulan and Stern is exponential in r and polynomial in n, so basic PMI is considered broken. It has been patched, however, as noted in Section 6.2.3.

Chapter 5

Attacking TPM, HFE, MIC* and PMI

5.1 Preliminaries

Sometimes, in the course of attacking an $\mathcal{MQ}$-based encryption scheme, it is useful to know that T and S, the affine transformations that make up part of the private key, are in fact linear transformations. It is often possible to show this directly by a simple trick, due to Wolf, Braeken and Preneel [WBP04]. Recall that T and S are defined as $T(x) = T_\ell x + T_c$ and $S(x) = S_\ell x + S_c$, where $T_\ell \in \mathbb{F}_q^{m \times m}$ is an invertible matrix and $T_c \in \mathbb{F}_q^m$ is a column vector, and $S_\ell \in \mathbb{F}_q^{n \times n}$ is an invertible matrix and $S_c \in \mathbb{F}_q^n$ is a column vector. Let $\mathcal{P}$ be the public key of a generic $\mathcal{MQ}$-based encryption scheme, with private key $(T, \mathcal{P}', S)$. It is a simple matter to rewrite T and S in a more favorable way. First, view T_ℓ and S_ℓ as linear operators, and let Id denote the identity operator. Then let

$$S(x) = (Id + S_c) \circ (S_\ell x).$$

In a similar way, let

$$T(x) = T_\ell \circ (Id + T_\ell^{-1} T_c)(x).$$

Now, instead of writing $\mathcal{P} = T \circ \mathcal{P}' \circ S$, we consider

$$\mathcal{P}(x) = (T_\ell \circ (Id + T_\ell^{-1} T_c)) \circ \mathcal{P}' \circ ((Id + S_c) \circ S_\ell)(x).$$

Now, as function composition is associative, this can be restated as:

$$\mathcal{P}(x) = T_\ell \circ ((Id + T_\ell^{-1} T_c) \circ \mathcal{P}' \circ (Id + S_c)) \circ S_\ell(x).$$

Now, let $\mathcal{U}$ be this new internal function, so

$$\mathcal{U} = (Id + T_\ell^{-1} T_c) \circ \mathcal{P}' \circ (Id + S_c).$$

If this new system (T_ℓ, $\mathcal{U}$, S_ℓ) still satisfies the trapdoor condition inherent to the particular $\mathcal{MQ}$-based encryption scheme in question using the new middle

function $\mathcal{U}$, then an attacker may assume that T and S are linear. Wolf, Braeken and Preneel [WBP04] note that this result holds for STS encryption schemes (described here in Section 4.2), and that it has been shown to hold for HFE encryption schemes (described here in Section 4.3).

5.2 Attacking STS

The following attack is due to Wolf, Braeken and Preneel [WBP04].

Recall that the STS cryptosystem was first discussed in Section 4.2. This attack uses the MinRank attack ideas of Section 3.4. The values of note in an STS cryptosystem are the order q of the base field, the number n of indeterminates, the number m of equations, and the number L of steps. For simplicity, this attack assumes all steps have equal height and width r. STS schemes with steps of this sort are refered to as regular STS schemes. In regular STS schemes, $n = m$, and $L = n/r$. Note that the term layer is equivalent to the term step, and they will be used interchangeably in this section.

The flow of this attack will be as follows. First, there will be some ground work about matrices, and demonstrating that a chain of kernels exists. Then, using the fact there is a chain of kernels, an attack of Wolf, Braeken and Preneel [WBP04] will be presented. This particular attack recovers a system $(\tilde{T}, \tilde{\mathcal{P}}', \tilde{S})$ that functions equivalently to the original private key $(T, \mathcal{P}', S)$. Finally, this attack will use the facts pointed out in Section 5.1 to assume that the private key $(T, \mathcal{P}', S)$ uses linear functions T and S rather than affine functions. Note finally that this attack is one of several variants offered by Wolf, Braeken and Preneel [WBP04]. There are variants that do not recover a complete equivalent private key, and that solve for $\tilde{T}$ slightly differently. The variants exist because for certain parameter choices, they are asymptotically faster than the one presented here, but do not significantly stray from the ideas of this variant. Thus, they are omitted.

5.2.1 Matrices and MinRank

It is necessary, when dealing with MinRank attacks, to deal with matrices. Here, those matrices come from the quadratic coefficients of the public key $\mathcal{P}$ polynomials $p_i, 1 \leq i \leq m$. Let $x \in \mathbb{F}_q^n$. For $1 \leq i \leq m$, the quadratic coefficients of p_i can be represented with a matrix $M_i \in \mathbb{F}_q^{n \times n}$. The quadratic terms themselves can be constructed by taking $x^t M_i x$, where ^{t} denotes the transpose. These matrices M_i are all public, as the public key is public. In the same way, the quadratic coefficients for the private key polynomials $\mathcal{P}'$, $p_i', 1 \leq i \leq m$ can be represented with a matrix $A_i \in \mathbb{F}_q^{n \times n}$. The quadratic terms themselves are then constructed by taking $x^t A_i x$. These matrices A_i are all secret.

In order for MinRank attacks to work correctly, such matrices must always be uniquely represented. If q is odd, then this is always possible using symmetric matrices. Any quadratic polynomial γ can have its quadratic terms represented

by a symmetric matrix Γ in the following way, demonstrated using a toy example with 3 indeterminates:

$$\Gamma = \begin{pmatrix} \gamma_{1,1} & \frac{\gamma_{1,2}}{2} & \frac{\gamma_{1,3}}{2} \\ \frac{\gamma_{1,2}}{2} & \gamma_{2,2} & \frac{\gamma_{2,3}}{2} \\ \frac{\gamma_{1,3}}{2} & \frac{\gamma_{2,3}}{2} & \gamma_{3,3} \end{pmatrix}.$$

This corresponds to a system

$$x^t \Gamma x = \gamma_{1,1}x_1x_1 + \gamma_{1,2}x_1x_2 + \gamma_{1,3}x_1x_3 + \gamma_{2,2}x_2x_2 + \gamma_{2,3}x_2x_3 + \gamma_{3,3}x_3x_3.$$

Note that this requires a division by 2. A different technique must be used to deal with the case of q even.

If q is even, then let γ be any quadratic polynomial, and, similar to before, let Γ be a matrix representation of the quadratic coefficients of γ such that $x^t\Gamma x$ is the quadratic part of γ. Using Gaussian elimination, reduce Γ to a lower triangular matrix L. Then, according to Wolf, Braeken and Preneel [WBP04], the symmetric matrix $L + L^t$ is also capable of fulfilling the role of Γ and is symmetric by construction. This particular result can be traced back to Don Coppersmith, according to Wolf, Braeken and Preneel, who note that Kipnis and Shamir [KS99] attribute the statement to Coppersmith, and that this is the first time to their knowledge that the statement was reported. Note also that the diagonal is necessarily all 0. This is good. If q is even, then the function x_i^2 is a linear function. Thus, it is always possible to uniquely represent the quadratic parts of polynomials with symmetric matrices.

5.2.2 Chain of kernels

Now we wish to show that there exists a chain of kernels, one per step. This chain of kernels is the key to attacking the STS system. First, note that the secret matrices A_i have common kernels. For each layer ℓ, for $1 \leq \ell \leq L$ all of the matrices $A_{(\ell-1)r+1}, \ldots A_{\ell r}$ are rank $r\ell$, and use exactly the first rl indeterminates of $x \in \mathbb{F}_q^n$. There are common kernels for each matrix in the same layer as well. For each A_i sharing the same layer ℓ, the kernel is given by

$$\ker'_\ell = \left\{ a' \in \mathbb{F}_q^n \mid a'_1 = \cdots = a'_{r\ell} = 0 \right\}.$$

This gives a chain of kernels

$$\ker'_L \subset \cdots \subset \ker'_1.$$

Now we seek to determine something about the kernels of the public matrices M_i from this fact. Note first that these kernels are always hidden behind the linear transformation S, which is a mere change of indeterminates. For each private polynomial p'_i, we can take the composition of that function and the change of indeterminates S and instead let $\hat{p}_i := p'_i \circ S$. Then we obtain a set of matrices $\hat{A}_i$ from this,

$$\hat{A}_i := S^t A_i S.$$

S is invertible, so $\text{rank}(\hat{A}_i) = \text{rank}(A_i)$. Also, the kernels of $\hat{A}_i$ for all $\hat{A}_i$ sharing the same layer—that is, for a fixed ℓ, $1 \leq \ell \leq L$, and for $(\ell - 1) < i \leq r\ell$—are given by

$$\ker_\ell = \left\{ a'S^{-1} \mid a' \in \mathbb{F}_q^n \text{ and } a'_1 = \cdots = a'_{r\ell} = 0 \right\}.$$

Since there is a chain of kernels for the $\ker'_\ell$, there is a chain of kernels

$$\ker_L \subset \cdots \subset \ker_1 .$$

It is now possible to write down M_i entirely in terms of these $\hat{A}_i$ and T. First, let $T = (\tau_{i,j})_{1 \leq i,j \leq m}$. Then as $\mathcal{P} = T \circ \mathcal{P}' \circ S$, each M_i can be written as

$$M_i = \sum_{j=1}^{m} \tau_{i,j} \left(S^t A_i S \right) = \sum_{j=1}^{m} \tau_{i,j} \hat{A}_i.$$

Finally, as

$$T^{-1} \circ \mathcal{P} = \mathcal{P}' \circ S,$$

finding T^{-1} is reduced to finding a linear combination of the public matrices M_i with a specific rank. The chain of kernels $\ker_L \subset \cdots \subset \ker_1$ gives this property.

5.2.3 Finding $\tilde{T}$

This attack attempts to guess row vectors of T^{-1}. The reasoning behind this begins with the vector spaces

$$J_\ell = \left\{ b'T^{-1} \mid b' \in \mathbb{F}_q^m \text{ and } b'_{\ell r+1} = \cdots = b'_m = 0 \right\} \text{ for } 1 \leq \ell \leq L.$$

It is clear that these also form a descending chain of subspaces, and that each J_ℓ has rank $m - \ell r$. Thus, if $v \in_R J_{\ell+1}$ is chosen randomly, there is a probability of q^{-r} that $v \in J_\ell$ as well. Now we need a method to determine whether or not $v \in J_\ell$. We will consider a method called *matrixCheck* that returns true if and only if

$$\text{rank} \left(\sum_{i=1}^{m} v_i M_i \right) \leq \ell r.$$

Definition 5.1 (matrixCheck) *The algorithm matrixCheck is defined as follows:*

matrixCheck$(m, n, r, \ell, M_1, M_2, \ldots, M_m, v_1, v_2, \ldots, v_m)$

Input: A set of matrices M_i and a set of constants v_i, where $1 \leq i \leq m$. Also, m, n, r, ℓ as above.

Output: TRUE, if $v \in J_\ell$ or FALSE if $v \notin J_\ell$.

Let $M = \sum_{i=1}^{m} v_i M_i$.

If rank$(M) \leq \ell r$, then return TRUE

Else return FALSE

To see that this method actually works, note that instead of considering $v \in J_{\ell+1}$, we may write v as $b'T^{-1}$, by definition of $J_{\ell+1}$. Then instead of computing $\sum_{i=1}^{m} v_i M_i$, note that the T^{-1} and T cancel, so instead we consider a matrix M given by:

$$M := \sum_{i=1}^{m} b'_i \hat{A}_i = \sum_{i=1}^{m} b'_i \left(S^t A_i S\right) = S^t \left(\sum_{i=1}^{m} b'_i A_i\right) S.$$

Then it is clear from the ranks of the secret polynomials A_i that

$$\text{rank}(M) \leq \ell r \text{ if and only if } v \in J_\ell.$$

Since M is the same matrix as the one defined in the definition of *matrixCheck*, this procedure works as advertised.

This leads to a simple algorithm. Given the matrices M_i, we can compute each vector space J_ℓ from $L-1$ down to 1, as $J_L = \mathbb{F}_q^m$. We do this using $O(mq^r)$ randomly chosen $v \in J_{\ell+1}$ per layer. Each v requires one use of *matrixCheck*, which requires $O(n^3)$ operations, as it requires a rank computation of a matrix $M \in \mathbb{F}_q^{n \times n}$. Finally, there are L layers. So the total time to compute this chain of kernels is $O(mn^3 Lq^r)$.

Now that the chain of kernels J_ℓ, for $1 \leq \ell \leq L$ is known, we can compute $\tilde{T}$ by first computing $\tilde{T}^{-1}$. We compute this value in the following way. Assuming $J_0 = \emptyset$, Let

$$\tilde{\mathcal{J}}_{\ell+1} := J_{\ell+1} - J_\ell, \text{ for } 0 \leq \ell \leq L-1.$$

Then each $\tilde{\mathcal{J}}_{\ell+1}$ has rank r. Pick a basis $B_{\ell+1}$ for each $\tilde{\mathcal{J}}_{\ell+1}$. Define $\tilde{T}^{-1}$ row by row by setting rows 1 through r as the basis vectors of B_1, and in general rows $\ell r + 1$ through $(\ell+1)r$ as the basis vectors of B_ℓ.

Finally, invert $\tilde{T}^{-1}$ and return $\tilde{T}$. Wolf, Braeken and Preneel note [WBP04] that in general this is not T itself, and that the rows of $\tilde{T}$ are linear combinations of rows of T within the same layer. The computational difficulty of the steps after the chain of kernels is computed is negligible, so the overall cost of calculating $\tilde{T}$ is $O(mn^3 Lq^r)$.

5.2.4 Computing $\tilde{\mathcal{P}}'$ and $\tilde{S}$

The first step of the attack is over. We have a replacement for T—$\tilde{T}$. The next step is to calculate $\tilde{S}$ and $\tilde{\mathcal{P}}'$, as naturally calculating one gives the other. To do so, first compute the values $\hat{A}_i$ from M_i by noting that

$$\hat{A}_i = \tilde{T}^{-1} M_i, \text{ for } 1 \leq i \leq m.$$

Let K_ℓ be the kernel for $\hat{A}_i$, for $(\ell-1)r < i \leq \ell r$, and $1 \leq \ell \leq L$. Similar to before, let

$$\tilde{\mathcal{K}}_\ell := K_{\ell-1} \cap \bar{K}_\ell, \text{ for } 0 \leq \ell \leq L.$$

Again, each $\tilde{\mathcal{K}}_\ell$ has rank r, so find a basis C_ℓ for each $\tilde{\mathcal{K}}_\ell$. Define $\tilde{S}^{-1}$ row by row by setting rows 1 through r as the basis vectors of B_1, and in general rows $(\ell-1)r + 1$ through ℓr as the basis vectors of B_ℓ.

Finally, invert $\tilde{S}^{-1}$ and return $\tilde{S}$, and return $\tilde{\mathcal{P}}'$ in the form

$$\tilde{A}'_i := \hat{A}_i \tilde{S}^{-1}, \text{ for } 1 \leq i \leq m.$$

The most costly step is computing the kernels K_ℓ. There are m polynomials to evaluate, and they have $O(n^2)$ quadratic terms each. Each quadratic term costs $O(n^2)$ to evaluate, as it has 2 indeterminates. Thus, the total cost is $O(mn^4)$. This dominates the other costs here.

5.2.5 Using $(\tilde{T}, \tilde{\mathcal{P}}', \tilde{S})$

The attack gives almost everything necessary to operate as the legitimate user. The attacker has access to a system $(\tilde{T}, \tilde{\mathcal{P}}', \tilde{S})$ that faithfully replicates the behavior of the quadratic part of the system $(T, \mathcal{P}', S$. The linear and constant terms of the polynomial system, however, are not yet known. This is easily remedied. Note that the system

$$\mathcal{P} - \tilde{T} \circ \tilde{\mathcal{P}}' \circ \tilde{S}$$

is linear. Therefore, the system

$$\tilde{T}^{-1} \circ (\mathcal{P} - \tilde{T} \circ \tilde{\mathcal{P}}' \circ \tilde{S}) \circ \tilde{S}^{-1}$$

is also linear, and we have:

$$\begin{aligned} \tilde{T}^{-1} \circ (\mathcal{P} - \tilde{T} \circ \tilde{\mathcal{P}}' \circ \tilde{S}) \circ \tilde{S}^{-1} &= \tilde{T}^{-1} \circ (\mathcal{P} \circ \tilde{S}^{-1} - \tilde{T} \circ \tilde{\mathcal{P}}') \\ &= \tilde{T}^{-1} \circ \mathcal{P} \circ \tilde{S}^{-1} - \tilde{\mathcal{P}}'. \end{aligned}$$

This final expression can be used to determine the linear and constant coefficients of the alternate secret key polynomials $\tilde{p}'_i$ very quickly. For $x = (0, \ldots, 0, 1, 0, \ldots) \in \mathbb{F}_q^n$, where the i-th entry is one, this expression returns the linear coefficients for x_i on the various $\tilde{p}'_i$. Similarly, $x = 0 \in \mathbb{F}_q^n$ returns the constant coefficients for the various $\tilde{p}'_i$.

5.2.6 Efficacy

Wolf, Braeken, and Preneel [WBP04] conclude that between this attack variant and the others presented in [WBP04], all practical existing implementations of STS cryptosystems are insecure, due to the presence of a chain of kernels. This particular variant of the attacks offered by Wolf, Braeken and Preneel has a total expected complexity of $O(mn^3Lq^r + mn^4)$. As for attacking the schemes of Kasahara and Sakai [KS04b, KS04a] themselves, Wolf, Braeken, and Preneel [WBP04] showed that Kasahara and Sakai's schemes are part of the STS family, and noted that their new attack therefore would work on those schemes as well.

5.3 Attacking HFE

The HFE cryptosystem was discussed in Section 4.3. Recall that an HFE cryptosystem uses as many indeterminates as there are equations, so $n = m$. The private key of an HFE system can be considered, as always, as a triple (T, $\mathcal{P}'$, S). T and S are formed in the usual way (see Section 2.3.1). The main idea of HFE is that the private key term $\mathcal{P}'$ is given by

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

where ϕ is a bijection going from n-tuples in $\mathbb{F}_q$ to $\mathbb{F}_{q^n}$ and $\mathcal{Q}$ is a univariate polynomial over $\mathbb{F}_{q^n}$, defined in the following way

$$\mathcal{Q}(z) = \sum_{\substack{a,b \in \mathbb{Z}_{\geq 0} \\ q^a + q^b \leq d}} \alpha_{a,b} z^{q^a + q^b} + \sum_{\substack{c \in \mathbb{Z}_{\geq 0} \\ q^c \leq d}} \beta_c z^{q^c} + \gamma.$$

For further details, see Section 4.3.1. For the purpose of this attack, we will consider ϕ to be defined through

$$\begin{array}{rccc} \phi : & \mathbb{F}_q^n & \longrightarrow & \mathbb{F}_{q^n}, \\ & (x_1, x_2, \ldots, x_n) & \longmapsto & \sum_{i=1}^n x_i w_i \end{array}$$

where $\{w_1, w_2, \ldots, w_n\}$ is a basis for $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$. Kipnis and Shamir [KS99] proposed an attack on HFE that is interesting because it determines an equivalent private key and solves a MinRank problem to do so (see Section 3.4 for details on the MinRank problem). In the following, we will refer to this attack as the Kipnis-Shamir attack.

The attack proceeds similarly to the MinRank attack against STS (see Section 5.2), in that it first determines T, and then determines S and $\mathcal{P}'$. As in the STS attack, only the quadratic portion of the private key is recovered in the attack proper, with the remainder determined exactly as in Section 5.2.5. Note that this implies that the function $\mathcal{Q}$ is assumed to have the form

$$\mathcal{Q}(z) = \sum_{\substack{a,b \in \mathbb{Z}_{\geq 0} \\ q^a + q^b \leq d}} \alpha_{a,b} z^{q^a + q^b}$$

in this case. This can be rewritten, however, for a value of r given by the known value d, which is a parameter of the HFE cryptosystem (see Section 4.3.1). In this case, we get

$$\sum_{\substack{a,b \in \mathbb{Z}_{\geq 0} \\ q^a + q^b \leq d}} \alpha_{a,b} z^{q^a + q^b} = \sum_{a=0}^{r-1} \sum_{b=0}^{r-1} \delta_{a,b} z^{q^a + q^b},$$

where $r = 1 + \lceil \log_q(d) \rceil$. Also, as noted in Section 5.1, it is safe to assume that T and S are linear transformations when performing cryptanalysis of HFE cryptosystems.

5.3.1 Setup

In order to carry out a MinRank attack, matrices are required. In order to get the matrices that this attack uses, some setup is necessary. We would like to push all of our work into the land of univariate polynomials, which will lead to the matrices we will be using. We require a few lemmas to do this. These lemmas and their proofs are due to Kipnis and Shamir [KS99].

The first lemma states that any linear mapping from $\mathbb{F}_q^n$ to itself can be represented uniquely as an equivalent linear mapping from $\mathbb{F}_{q^n}$ to itself.

Lemma 5.2 *Let $\{w_1, w_2, \ldots, w_n\}$ be a basis for $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$. The set of mappings $\{A : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^n \mid A \text{ is linear}\}$ is equivalent to the set of univariate polynomials $\{\sum_{i=1}^n a_i X^{q^{i-1}} \mid \{a_1, a_2, \ldots, a_n\} \subset \mathbb{F}_{q^n}\}$. That is, for any linear mapping A, there exists a univariate polynomial $\sum_{i=1}^n a_i X^{q^{i-1}}$ such that for any $(v_1, v_2, \ldots, v_n), (x_1, x_2, \ldots, x_n) \in \mathbb{F}_q^n$, and with $x = \sum_{i=1}^n x_i w_i$ and $v = \sum_{i=1}^n v_i w_i$, we have that $v = \sum_{i=1}^n a_i x^{q^{i-1}}$ if and only if $(v_1, v_2, \ldots, v_n) = A(x_1, x_2, \ldots, x_n)$.*

Proof: There are q^{n^2} matrices of size $n \times n$ over $\mathbb{F}_q$, which is the number of possible linear mappings A. There are $(q^n)^n$ linear combinations of the n monomials $x^{q^{i-1}}$ over $\mathbb{F}_{q^n}$, given by the q^n choices for each of the n unknowns a_i. Now, as the function f, given by $f(x) = x^{q^{i-1}}$ is a linear function (see Theorem 4.1), then a linear combination of such functions is also a linear function. So, the set of linear mappings and the set of monomial sums $\sum_{i=1}^n a_i x^{q^{i-1}}$ are of the same size, and a surjective map from one to the other exists. Therefore, there is an isomorphism between them, and every linear map $A : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^n$ is uniquely represented by a univariate polynomial $\sum_{i=1}^n a_i x^{q^{i-1}}$ over the field $\mathbb{F}_{q^n}$.$\diamond$

The second lemma states that, in fact, any set of n multivariate polynomials in n indeterminates over $\mathbb{F}_q$ is uniquely representable by a univariate polynomial in $\mathbb{F}_{q^n}$

Lemma 5.3 *Let $\{w_1, w_2, \ldots, w_n\}$ be a basis for $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$. The set of mappings $\{ \{P_1, \ldots, P_n\} \mid P_i : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q$ is a multivariate polynomial, $1 \leq i \leq n\}$ is equivalent to the set of univariate polynomials $\{\sum_{i=1}^{q^n} a_i X^{i-1} \mid a_1, a_2, \ldots a_{q^n} \in \mathbb{F}_{q^n}\}$. That is, for any set of such multivariate polynomials $P_1(x_1, x_2, \ldots, x_n), \ldots, P_n(x_1, x_2, \ldots, x_n)$, there is a univariate polynomial $\sum_{i=1}^{q^n} a_i X^{i-1}$ such that for any $(v_1, v_2, \ldots, v_n), (x_1, x_2, \ldots, x_n) \in \mathbb{F}_q^n$, and with $x = \sum_{i=1}^n x_i w_i$ and $v = \sum_{i=1}^n v_i w_i$, we have that $v_j = P_j(x_1, x_2, \ldots, x_n)$ for all $j, 1 \leq j \leq n$ if and only if $v = \sum_{i=1}^{q^n} a_i x^{i-1}$.*

Proof: First, assume without loss of generality that $w_1 = 1$. Now, note that the mapping π_i

$$\pi_i : (x_1, x_2, \ldots, x_n) \longrightarrow (x_i, 0, 0, \ldots, 0)$$

is a linear mapping. Thus, by Lemma 5.2 this function has a univariate polynomial representation over $\mathbb{F}_{q^n}$. Let's call this polynomial representation f_i. To

represent the mapping

$$\pi(x_1, x_2, \ldots, x_n) \longrightarrow (\prod_{i=1}^{n} x_i^{c_i}, 0, 0, \ldots, 0)$$

as a univariate polynomial over $\mathbb{F}_{q^n}$, we form the product

$$U(x) \quad = \quad \prod_{i=1}^{n} (f_i(x_1, x_2, \ldots, x_n))^{c_i} \,. \tag{5.1}$$

To represent an arbitrary multivariate polynomial $P(x_1, x_2, \ldots, x_n)$ as a univariate polynomial over $\mathbb{F}_{q^n}$, we simply take an appropriate linear combination of products of form (5.1) to construct the univariate representation of that multivariate polynomial if it were in the first coordinate of the n-tuple. To take care of the case where that multivariate polynomial is in a different coordinate, we can construct it in the first coordinate as usual, and then shift it to a different coordinate k in the n-tuple, where $1 \leq k \leq n$, by multiplying by the basis element w_k. This is possible because the univariate polynomials take their coefficients from $\mathbb{F}_{q^n}$, not $\mathbb{F}_q$ itself. Finally, to construct the univariate polynomial that represents a system of n multivariate polynomials, as is the case here, we construct a representation of each multivariate polynomial at the first coordinate, shift it to its proper coordinate, and sum all n resulting univariate polynomials. Clearly, this process is invertible, and thus a representation in one form leads to a unique representation in the other form.$\diamond$

A corollary to this result is that the univariate representations desired can be found in polynomial time.

Lemma 5.4 *Let C be any collection of n homogeneous multivariate polynomials of degree d in n indeterminates over $\mathbb{F}_q$. Let $G(x)$ be the univariate polynomial representation of C over $\mathbb{F}_{q^n}$. The only powers of x with nonzero coefficients in $G(x)$ are sums of exactly d (not necessarily distinct) powers of q; i.e., $q^{i_1} + q^{i_2} + \cdots + q^{i_d}$. If d is constant, then $G(x)$ is sparse, and its coefficients can be found in polynomial time.*

Proof: From the proof of Lemma 5.3, we can note that any given univariate term generated of form (5.1) will be a product of exactly d univariate polynomials corresponding to linear functions as in Lemma 5.2. In the remainder of the steps, the degree of the polynomials are not changed, so every nonzero term has the form $x^{q^{i_1}+q^{i_2}+\cdots+q^{i_d}}$, as desired.

No matter the degree of $G(x)$ as a univariate polynomial (it may be exponential in n), at most $O(n^d)$ of its coefficients can be non-zero. If d is fixed, this is a polynomial of n. Given that such a sparse univariate polynomial representation exists, we can find its coefficients in polynomial time using sufficiently many input/output pairs.$\diamond$

We may now begin talking about the matrices we will be using in our MinRank problem. In the HFE cryptosystem, the private key $(T, \mathcal{P}', S)$ consists of two linear mappings T and S that may be converted to univariate polynomials

by Lemma 5.2. So, we may represent S as $S(X) = \sum_{i=1}^{n} s_i X^{q^{i-1}}$. As T is linear, T^{-1} is linear, so we may represent T^{-1} as $T^{-1}(X) = \sum_{i=1}^{n} t_i X^{q^{i-1}}$.

This leaves the public key $\mathcal{P}$ and the private key term $\mathcal{P}'$. By the assumptions of Section 5.3, the private key term $\mathcal{P}'$ has only quadratic terms, and therefore Lemma 5.4 applies. This fact allows us to compute the following representation of the private key term $\mathcal{P}'$:

$$\mathcal{P}'(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} x^{q^i+q^j} = \bar{x}A\bar{x}^t, \text{ where } A = [a_{ij}], \text{ and } \bar{x} = (x^{q^0}, x^{q^1}, \ldots, x^{q^{n-1}}).$$

This matrix A is unknown, as it is part of the secret key. However, a similar setup exists for $\mathcal{P}$, as Lemma 5.4 applies to the quadratic terms of $\mathcal{P}$. This leads to

$$\mathcal{P}(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij} x^{q^i+q^j} = \bar{x}M\bar{x}^t, \text{ where } M = [m_{ij}], \text{ and } \bar{x} = (x^{q^0}, x^{q^1}, \ldots, x^{q^{n-1}}).$$

The matrix M is computable from the public key. Note that for the same reasons the matrices of Section 5.2.1 were considered unique representations, we can consider A and M to be unique representations.

By our work, we can write $\mathcal{P}(x) = T \circ \mathcal{P}' \circ S(x)$ as a composition of univariate functions. Alternatively, we may write

$$T^{-1} \circ \mathcal{P}(x) = \mathcal{P}' \circ S(x),$$

where $T^{-1}(x) = \sum_{i=1}^{n} t_i x^{q^{i-1}}$ and $S(x) = \sum_{i=1}^{n} s_i x^{q^{i-1}}$. This equation leads to our MinRank problem.

Let $G = [g_{ij}]$ be an $n \times n$ matrix. Then we define the matrix G^{*k} from G entry-wise by:

$$G^{*k}_{ij} = g^{q^k}_{i-k,j-k}, \text{ where } i-k, j-k \text{ are computed modulo } n.$$

Note that here the range of modulo arithmetic is $[1, n]$. This leads to the following theorem, which is presented and proved as by Kipnis and Shamir [KS99].

Theorem 5.5 *Let $\bar{x} = (x^{q^0}, x^{q^1}, \ldots, x^{q^{n-1}})$. Let $u_1, u_2, \ldots, u_n$ be a basis for $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$.. Let M and A be as just defined, with matrices M^{*k} also as just defined computed for all k, $1 \leq k \leq n$. Let $T^{-1}(x) = \sum_{i=1}^{n} t_i x^{q^{i-1}}$ and $S(x) = \sum_{i=1}^{n} s_i x^{q^{i-1}}$. Then the matrix representation of $T^{-1} \circ \mathcal{P}$ is $\sum_{k=1}^{n} t_k M^{*(k-1)}$. Also, the matrix representation of $\mathcal{P}' \circ S(x)$ is WAW^t, where $W = [w_{ij}]$ is an $n \times n$ matrix defined by $w_{ij} = s^{q^i}_{j-i}$, and $j-i$ is computed modulo n, noting that here the range of modulo arithmetic is $[1, n]$.*

Proof: First, take the composition of T^{-1} and $\mathcal{P}$, given by:

$$T^{-1} \circ \mathcal{P}(x) = \sum_{k=1}^{n} t_k \left(\sum_{i=1}^{n}\sum_{j=1}^{n} M_{ij} x^{q^i+q^j} \right)^{q^{k-1}}.$$

There are several steps to take to change this into a form we prefer. We have:

$$\begin{aligned}
\sum_{k=1}^{n} t_k \left(\sum_{i=1}^{n}\sum_{j=1}^{n} M_{ij} x^{q^i+q^j} \right)^{q^{k-1}} &= \sum_{k=1}^{n} t_k \sum_{i=1}^{n}\sum_{j=1}^{n} M_{ij}^{q^{k-1}} x^{q^{i+k-1}+q^{j+k-1}} \\
&= \sum_{k=1}^{n} t_k \sum_{i=1}^{n}\sum_{j=1}^{n} M_{i-k+1,j-k+1}^{q^{k-1}} x^{q^i+q^j} \\
&= \sum_{k=1}^{n} t_k \sum_{i=1}^{n}\sum_{j=1}^{n} M_{ij}^{*(k-1)} x^{q^i+q^j} \\
&= \sum_{k=1}^{n} t_k \left(\bar{x} M^{*(k-1)} \bar{x}^t \right) \\
&= \bar{x} \left(\sum_{k=1}^{n} t_k M^{*(k-1)} \right) \bar{x}^t.
\end{aligned}$$

The first step of this simplification is to note that taking the q^k-th power is a linear function. The next step is to rewrite the indices by noting they can be cyclically rotated modulo n and that the exponents of q can be reduced modulo n as $x^{q^n} = x^{q^0} = x$. Finally, note that $M_{ij}^{*k} = M_{i-k,j-k}^{q^k}$. This finishes the simplification to the first matrix representation.

The second matrix representation is done similarly:

$$\begin{aligned}
\mathcal{P}' \circ S(x) &= \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij} \left(\sum_{k=1}^{n} s_k x^{q^{k-1}} \right)^{q^i+q^j} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij} \left(\sum_{k=1}^{n} s_k x^{q^{k-1}} \right)^{q^i} \left(\sum_{\ell=1}^{n} s_\ell x^{q^{\ell-1}} \right)^{q^j} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij} \left(\sum_{k=1}^{n} s_k^{q^i} x^{q^{i+k-1}} \right) \left(\sum_{\ell=1}^{n} s_\ell^{q^j} x^{q^{j+\ell-1}} \right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij} \left(\sum_{k=1}^{n} s_k^{q^i} x^{q^{i+k-1}} \right) \left(\sum_{\ell=1}^{n} s_\ell^{q^j} x^{q^{j+\ell-1}} \right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij} \left(\sum_{k=1}^{n} s_{k-i}^{q^i} x^{q^{k-1}} \right) \left(\sum_{\ell=1}^{n} s_{\ell-j}^{q^j} x^{q^{\ell-1}} \right).
\end{aligned}$$

As before, we used the linearity of exponentiation by q^i, cyclic index shifting and modulo arithmetic to simplify. Now we change the order of summation and

rewrite in terms of $W = [w_{ij}]$. This will net the second matrix representation.

$$\begin{aligned}
\mathcal{P}' \circ S(x) &= \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}\left(\sum_{k=1}^{n} s_{k-i}^{q^i} x^{q^{k-1}}\right)\left(\sum_{\ell=1}^{n} s_{\ell-j}^{q^j} x^{q^{\ell-1}}\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\sum_{k=1}^{n} s_{k-i}^{q^i} x^{q^{k-1}}\right) A_{ij}\left(\sum_{\ell=1}^{n} s_{\ell-j}^{q^j} x^{q^{\ell-1}}\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\sum_{k=1}^{n} w_{ik} x^{q^{k-1}}\right) A_{ij}\left(\sum_{\ell=1}^{n} w_{j\ell} x^{q^{\ell-1}}\right) \\
&= \sum_{k=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{\ell=1}^{n} w_{ik} x^{q^{k-1}} A_{ij} w_{j\ell} x^{q^{\ell-1}} \\
&= \bar{x} W A W^t \bar{x}^t.
\end{aligned}$$

And this concludes the proof.$_\diamondsuit$

We are now ready to construct our MinRank problem. Theorem 5.5 states that $T^{-1}\circ\mathcal{P}$, when represented as a matrix, can be written as a sum of matrices. Recall from Section 3.4 that the MinRank(r) problem is, given a finite set of matrices with entries in $\mathbb{F}_q$, to find a linear combination of those matrices of rank $\leq r$. We can compute M, and therefore the matrices M^{*k}, for $1 \leq k \leq n$. These are the matrices we will be dealing with. We still, however, require an r value. This is easy to determine in the following way. Recall that the private key univariate function has the form

$$\sum_{\substack{a,b\in\mathbb{Z}_{\geq 0}\\ q^a+q^b\leq d}} \alpha_{a,b} z^{q^a+q^b} = \sum_{a=0}^{r-1}\sum_{b=0}^{r-1} \delta_{a,b} z^{q^a+q^b},$$

where $r = 1 + \lceil \log_q(d) \rceil$. In particular, this implies that the matrix $\delta_{a,b}$ completely determines the private key function $\mathcal{Q}$. As $\delta_{a,b}$ is an $r \times r$ matrix, this implies that the secret matrix A cannot have a rank higher than r. So, as S can be represented by a full rank matrix, we have that the rank of $\mathcal{P}' \circ S(x)$ is at most r. Recall that $T^{-1} \circ \mathcal{P}(x) = \mathcal{P}' \circ S(x)$ to note that the rank of $T^{-1} \circ \mathcal{P}(x)$ is also at most r. Now we have the following MinRank(r) problem: Given the matrices $M^{*(k-1)}$, for $1 \leq k \leq n$, and $r = 1 + \lceil \log_q(d) \rceil$, find a set of coefficients $t_1^*, t_2^*, \ldots, t_n^*$ such that

$$\operatorname{rank}\left(\sum_{k=1}^{n} t_k^* M^{*(k-1)}\right) \leq r.$$

5.3.2 Calculating $\mathcal{T}$

Calculation of T occurs by solving the MinRank(r) problem introduced in the previous section. We assume that any set of coefficients $t_1^*, t_2^*, \ldots, t_n^*$ that solve

the MinRank(r) problem corresponds to the function T^{-1}. Courtois [Cou01] presents a succinct way of doing this.

By construction, we know that the MinRank(r) problem we wish to solve has a solution. So, we know that there are $t_1^*, t_2^*, \ldots, t_n^*$ such that there exists an $\mathcal{M}$ such that

$$\mathcal{M} = \sum_{k=1}^{n} t_k^* M^{*(k-1)}, \text{ and } \operatorname{rank}(\mathcal{M}) \leq r.$$

Then any $(r+1) \times (r+1)$ submatrix of $\mathcal{M}$ has determinant 0, and this determinant is a degree $(r+1)$ equation in $t_1^*, t_2^*, \ldots, t_n^*$. There are $\binom{n}{r+1}$ ways to choose $r+1$ rows (and respectively columns) of $\mathcal{M}$, so there are up to $\binom{n}{r+1}^2$ different submatrices, and as many equations. Each of these equations will contain up to $\binom{n+r}{r+1}$ monomial terms. This is simply the number of possible degree $r+1$ monomials in n indeterminates. If we can accumulate $\binom{n+r}{r+1}$ linearly independent equations, then we can simply linearize these degree $r+1$ monomials and solve the resulting linear system using Gaussian reduction for a unique solution.

This gives a set of $\binom{n+r}{r+1}$ monomial equations in $t_1^*, t_2^*, \ldots, t_n^*$, which can be solved by brute force fairly easily, giving T^{-1}, and in turn giving T itself. So, the overall cost of calculating T is given by solving a system of $\binom{n+r}{r+1}$ equations and as many indeterminates. We can use simple properties of binomial coefficients to approximate this by

$$\binom{n+r}{r+1} \leq \left(\frac{(n+r)e}{r+1}\right)^{r+1} \leq \left(\frac{ne}{r+1} + e\right)^{r+1} \leq n^{r+1}.$$

The previous statement holds for $r \geq 2$ and $n \geq 32$, or $r \geq 3$ and $n \geq 9$, so for any reasonable choices of r, n this statement holds. Therefore the Gaussian reduction takes $O(n^{\omega(r+1)})$, where ω is, as usual, the exponent of Gaussian elimination. In terms of d, rather than r, we have a total cost of $O(n^{\omega \log_q d})$.

5.3.3 Calculating $\mathcal{S}$

Now that we know T, we may recover S fairly easily. This method was introduced by Kipnis and Shamir [KS99]. By recovering T, we also recovered a matrix $\mathcal{M}$ such that

$$\mathcal{M} = WAW^t.$$

At this point, W and A are still unknown. However, we know that $\operatorname{rank}(A) \leq r$, and of course $\operatorname{rank}(W) \leq n$. We can assume without loss of generality that $\operatorname{rank}(A) = r$ and that $\operatorname{rank}(W) = n$. The trick is to notice that the left kernel of $\mathcal{M}$ corresponds to the left kernel of WAW^t, and that therefore it corresponds to the left kernel of WA, since W^t is invertible. But W is also invertible, so the left kernel of P, which is exactly the vectors that are 0 in their first r entries, defines the left kernel of WAW^t. That is, if $v_1, v_2, \ldots, v_{n-r}$ is a basis of the left kernel of $\mathcal{M}$, then W maps the v_i values to a set of vectors that are 0 in their first r entries.

So, we compute the left kernel of $\mathcal{M}$, as $\mathcal{M}$ is known, and get a basis $v_1, v_2, \ldots, v_{n-r}$. Each basis vector leads to r equations in the n^2 indeterminates of W. This leads to a system of $r(n-r)$ equations in n^2 indeterminates. This does not appear to be enough equations. But recall that we may replace each w_{ij} by $s_{j-i}^{q^i}$, dropping the number of indeterminates from n^2 to n. Unfortunately, the equations are no longer linear over $\mathbb{F}_{q^n}$. Fortunately, if we replace each s_{j-i} by $\sum_{k=1}^{n} s_{j-i,k}u_k$, where the $s_{j-i,k}$ form a new set of n^2 indeterminates over $\mathbb{F}_q$, we get a linear combination of n indeterminates taken to the q^i-th power. As $f(z) = z^{q^i}$ is a linear function over $\mathbb{F}_q$, we still have a linear combination of the $s_{j-i,k}$ indeterminates. So, each $s_{j-i}^{q^i}$ is now a linear combination of the $s_{j-i,k}$ indeterminates, and each equation over $\mathbb{F}_{q^n}$ becomes a set of n linear equations over $\mathbb{F}_q$. So, now we have $r(n-r)n$ equations and n^2 indeterminates. As long as $r \geq 2$, we have an overdefined system, and we may solve this system of homogeneous equations up to multiplication by a constant. This gives S, from W.

5.3.4 Last steps and conclusions

Now that S and T are both known, we can use the quadratic portion of the public key to compute the quadratic portion of the private key. We use the technique of Section 5.2.5 to compute the lower degree terms of the private key, completing the attack. The hardest portion of the attack is solving the large linear system to determine T, giving an overall complexity of $O(n^{\omega \log_q d})$. Kipnis and Shamir note that their attack has polynomial complexity for a fixed d, and is subexponential in general. Thus, we see that d must be chosen large enough to render this attack ineffective. Kipnis and Shamir do not provide experimental evidence to indicate exactly how large d should be. As for the security of basic HFE, see Section 6.2.1.

5.4 Attacking MIC*

Patarin's attack [Pat95] on the MIC* cryptosystem family does not cleanly fit into an organized system of attacks. Recall that the MIC* family is described in Section 4.4. This attack relies on using the public key to generate a large set of equations in the plaintext indeterminates $x_1, x_2, \ldots, x_n$ and the ciphertext indeterminates $y_1, y_2, \ldots, y_n$. The equations to be generated in this attack all have the form

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \gamma_{ij}x_i y_j + \sum_{i=1}^{n} \alpha_i x_i + \sum_{i=1}^{n} \beta_i y_i + \delta_0 = 0. \tag{5.2}$$

Note that, given a ciphertext $Y = (y_1, y_2, \ldots, y_n)$, the system becomes degree one in the corresponding unknown plaintext $X = (x_1, x_2, \ldots, x_n)$, and solvable using standard Gaussian elimination. The idea behind the attack is that once these equations are found, each such independent equation fixes one of the x_i

in terms of the other x_k, where $i \neq k$. The remaining x_k are then determined via brute force searching. Naturally, then, this attack breaks down into two parts. First, we must show that equations of this form can be generated from the public key. Second, we must show that sufficient independent equations exist to make this attack worthwhile.

5.4.1 Equation generating

The goal of the Patarin attack [Pat95] is to find equations of the same form as described in (5.2). Recall that the public key $\mathcal{P}$ of the MIC* cryptosystem has the form

$$\mathcal{P} = T \circ \phi^{-1} \circ \mathcal{Q} \circ \phi \circ S,$$

and consists of n quadratic equations using n indeterminates. Also recall that the function $\mathcal{Q}$ is a univariate polynomial of the form

$$Q(z) = z^{q^\ell+1},$$

for some ℓ such that $\gcd(q^\ell + 1, q^n - 1) = 1$.

Lemma 5.6 *Let ϕ, S, T, and Q be MIC* parameters, where $Q(z) = z^{q^\ell+1}$. There exist equations of the form (5.2) in the plaintext x and the ciphertext y.*

Proof: First, consider only the equation $Q(z) = z^{q^\ell+1}$, where we let $z = a$ and $Q(z) = b$, giving

$$b = a^{q^\ell+1}.$$

Take each side of this equation to the power $q^\ell - 1$, and we have

$$b^{q^\ell-1} = a^{q^{2\ell}-1}.$$

Now multiply both sides by ab, and we have

$$ab^{q^\ell} = ba^{q^{2\ell}}. \tag{5.3}$$

Now, as the function $z \longrightarrow z^q$ is a linear function, as noted on page 43, and since the composition of linear functions is also linear, $ab^{q^\ell} - ba^{q^{2\ell}}$ is linear in a and b. However, one can determine a and b using only affine or linear mappings on the plaintext x and the ciphertext y. Given x, and letting $a = \phi \circ S(x)$, we may write $y = T \circ \phi^{-1}(b)$. Moreover, $b = \phi \circ T^{-1}(y)$, and thus by substitution in $ab^{q^\ell} - ba^{q^{2\ell}} = 0$, we get an equation of the form (5.2), as a is affine in x and b is affine in y.$\diamond$

Now the difficulty lies in generating equations of the form (5.2). This task is simple. Select any value of plaintext $x = (x_1, x_2, \ldots, x_n)$, and apply the public key $\mathcal{P}$ to calculate its associated ciphertext $y = (y_1, y_2, \ldots, y_n)$. Recall that this is unambiguous, as $\mathcal{P}$ is a bijection by construction. Each time we do so, we generate an input-output pair for an equation of form (5.2), which is known to exist by Lemma 5.6. By generating many such pairs (that is, choosing many

plaintexts x and calculating their associated ciphertexts y), we can attempt to solve the equation of form (5.2) for its coefficients γ_{ij}, α_i, β_i and δ_0. We do this by setting up some equations of the form

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\gamma_{ij}x_iy_j + \sum_{i=1}^{n}\alpha_i x_i + \sum_{i=1}^{n}\beta_i y_i + \delta_0 = 0. \tag{5.4}$$

At first glance, this may seem to be identical to an equation of form (5.2). In fact, they are quite different. In form (5.2), the indeterminates are the x_i and y_i, for $1 \leq i \leq n$. However, in this form, the indeterminates are γ_{ij}, α_i, β_i and δ_0. Thus, this form is solvable for its coefficients using standard techniques for a linear system, provided enough equations are found.

The equations themselves are found in practice by forming a basis for a particular vector space. Let V be the vector space of equations of form (5.4) over the base field $\mathbb{F}_q$. Finding a basis B of this vector space V gives precisely a largest set of independent equations of form (5.4), which are then interpretted as independent equations of the form (5.2). Patarin [Pat05] notes that V itself can be calculated using $O(n^2)$ equations of form (5.4) with at least a 99% chance of success. Once V is calculated, finding a basis B for V is a matter of Gaussian reductions. Patarin [Pat05] displays a technique to ensure that this step can be accomplished at cost $O(n^4 \log n \log^2 q)$, a cost dominated by a clever Gaussian reduction. This is a significant improvement over a naive Gaussian reduction on $O(n^2)$ indeterminates with multiplications that cost $O(\log q)$, which has cost $O(n^6 \log^2 q)$.

5.4.2 Efficiency

The efficiency of this attack relies entirely on the number of independent equations of form (5.2) generated. If roughly n such independent equations are generated, then the attack is very efficient. If much fewer than n independent equations are generated, then there is little to be gained by using it. Let λ be the number of independent equations that can be generated in the equation generation phase. Patarin [Pat95] notes that there is a lower bound on λ. Before it is possible to prove anything about this lower bound, a few lemmas will be necessary. The first two lemmas are proven as by Fouque, Granboulan and Stern [FGS05]. The third is proven as by Patarin [Pat95].

Lemma 5.7 *For any integers q, i, and n,* $\gcd(q^n - 1, q^i - 1) = q^{\gcd(n,i)} - 1$.

Proof: Let $(r_k)_{k\geq 0}$ be the sequence of integers obtained by the Euclidean algorithm applied to compute $\gcd(n, i)$. Therefore, $r_0 = n$ and $r_1 = i$. Let k_0 be the largest integer such that $r_{k_0} \neq 0$, so $r_{k_0} = \gcd(i, n)$.

In the same way, let $(R_k)_{k\geq 0}$ be the sequence of polynomials obtained from the Euclidean algorithm when applied to compute $\gcd(X^i - 1, X^n - 1)$. Therefore, $R_0 = X^i - 1$ and $R_1 = X^n - 1$. Then let t_0 be the largest integer such that $R_{t_0} \neq 0$, so that $R_{t_0} = \gcd(X^i - 1, X^n - 1)$. We will show by induction on k that for $0 \leq k \leq k_0 + 1$, $R_k = X^{r_k} - 1$. This is clear by definition for $0 \leq k \leq 1$.

So, assume $2 \leq k \leq k_0 + 1$. Recall that the Euclidean algorithm relates the successive values in the sequence $(r_k)_{k\geq 0}$ by $r_{k-2} = \alpha r_{k-1} + r_k$, where α is the integer part of the fraction r_{k-2}/r_{k-1}. Then we can write,

$$X^{r_{k-2}} - 1 = (X^{r_{k-1}} - 1)(X^{r_{k-2}-r_{k-1}} + X^{r_{k-2}-2r_{k-1}} + \cdots + X^{r_{k-2}-(\alpha+1)r_{k-1}}) + X^{r_k} - 1.$$

So, $X^{r_k} - 1$ is the remainder of the division of $R_{k-2} = X^{r_{k-2}} - 1$ by $R_{k-1} = X^{r_{k-1}} - 1$ as the sequence $(r_k)_{k\geq 0}$ is strictly decreasing; that is, $R_k = X^{r_k} - 1$. So, $R_{k_0+1} = X^0 - 1 = 0$, and $R_{k_0} \neq 0$. So, $R_{k_0} = R_{t_0} = X^{r_{k_0}} - 1 = X^{\gcd(n,i)} - 1$. Replace X by q to get the lemma.$\diamond$

Lemma 5.8 *In a finite field $\mathbb{F}_{q^n}$ with q^n elements, the equation $X^j = A$ has either 0 solutions or $\gcd(j, q^n - 1)$ solutions.*

Proof: If $\gcd(j, q^n - 1) = 1$, then the function X^j is invertible in $\mathbb{F}_{q^n}$, and has the function X^h as its inverse, where $h \equiv j^{-1} \pmod{q^n - 1}$. So, in this case, there is exactly one solution, $X = X^{jh} = A^h$.

If $\gcd(j, q^n - 1) = d > 1$, then there are slightly more difficulties. Let $j' = j/d$. Therefore $\gcd(j', q^n - 1) = 1$. So, let $h' = j'^{-1} \pmod{q^n - 1}$. Now we have $X^d = X^{j'dh'} = X^{jh'} = A^{h'}$. This merely removes the invertible portion of the function X^j. Let $A' = A^{h'}$. Now, it is possible that this equation has no solutions—this occurs when A' is not a d-th power of some value of $\mathbb{F}_{q^n}$. On the other hand, if a solution does exist, then in fact d solutions exist. The other solutions can be found by multiplying the original solution by the d-th roots of unity. The d d-th roots of unity are contained in $\mathbb{F}_{q^n}$ because $d \mid q^n - 1$.$\diamond$

Lemma 5.9 *Let $b \in \mathbb{F}_{q^n}^*$. Then the equation $ab^{q^\ell} = ba^{q^{2\ell}}$ has at most $q^{\gcd(\ell,n)}$ solutions $a \in \mathbb{F}_{q^n}$.*

Proof: Note first that the equation of this lemma is (5.3) of the proof of Lemma 5.6. It is clear that $a = 0$ is always a solution. Then suppose that $a \neq 0$. Then it is possible to cancel ab from both sides, giving

$$b^{q^\ell - 1} = a^{q^{2\ell} - 1} = (a^{q^\ell - 1})^{q^\ell + 1}.$$

Then recall that the function $Q(z) = z^{q^\ell + 1}$ is a bijection (see Section 4.4.3 for details). So, here, we have a solution a if and only if

$$a^{q^\ell - 1} = Q^{-1}(b^{q^\ell - 1}).$$

Recall now that b is given. So we are in the case of Lemma 5.8. Thus, there are either 0 solutions or $\gcd(q^\ell - 1, q^n - 1)$ solutions. Using Lemma 5.7, we learn that if any solutions exist, there are $q^{\gcd(\ell,n)} - 1$ of them, excluding $a = 0$. Add this solution to achieve the desired maximum number of solutions.$\diamond$

Finally, we can show that λ has a lower bound. The proof is as Patarin's [Pat95].

Theorem 5.10 *Let $b \in \mathbb{F}_{q^n}^*$. If the equation $ab^{q^\ell} = ba^{q^{2\ell}}$ is considered as a set of n equations over $\mathbb{F}_q$, then at least $n - \gcd(\ell, n)$ independent equations of degree one in the components of $a \in \mathbb{F}_{q^n}$ are obtained.*

Proof: First note that the equations obtained are all of degree 1 in a, so they are all of degree 1 in $a_1, a_2, \ldots, a_n$, where $a = (a_1, a_2, \ldots, a_n)$ as it is represented as an n-tuple of elements of $\mathbb{F}_q$. The equations also have at least one solution, corresponding to $a = 0$. Thus, if λ is the number of independent equations, there are exactly $q^{n-\lambda}$ solutions. However, there are at most $q^{\gcd(\ell,n)}$ solutions, by Lemma 5.9, so $n - \lambda \leq \gcd(\ell, n)$, and thus $\lambda \geq n - \gcd(\ell, n)$.$_\diamond$

Thus, for a given ciphertext y, Patarin's attack reduces the problem from a search of q^n possible plaintexts x to a search of $q^{n-\lambda}$ possible plaintexts. We have that $\lambda \geq n - \gcd(\ell, n)$. Patarin [Pat05] notes that due to certain technical aspects of the possible choices of ℓ and n, it is true that $\gcd(\ell, n) \leq n/3$. Therefore, no more than $q^{n/3}$ possible plaintexts need be searched. However, it is a fact that many times, $\gcd(\ell, n) = 1$, which results in a simple search of q plaintexts.

In this best case and average case scenario, the dominant cost of the attack is the initial phase of equation generation, which is, as mentioned, $O(n^4 \log n \log^2 q)$. The surprising efficiency of this attack makes the MIC* system unsafe for cryptographic use.

An efficiency sidenote

Again recall that the efficiency of this attack hinges on finding a number of equations of form (5.2). The lower bound on λ, the number of such equations found, seems to be a loose one. There is not yet an explanation as to why this bound is loose, or, more usefully, there is not yet a tighter bound on the number of equations that can be found in this way.

5.5 Attacking PMI

The attack on the PMI cryptosystem outlined by Fouque, Granboulan and Stern [FGS05] reduces to an attack on the MIC* cryptosystem. The trick of this PMI attack is to find a clever way of reducing the PMI system to the MIC* system. Doing so relies on differential cryptanalysis, first discussed in Section 3.5. The PMI cryptosystem was introduced in Section 4.5. Recall that it has a public key function of the form

$$\mathcal{P} = T \circ (\phi^{-1} \circ \mathcal{Q} \circ \phi + \mathcal{A} \circ \pi) \circ S.$$

This form may be rewritten as

$$\mathcal{P} = T \circ \phi^{-1} \circ \mathcal{Q} \circ \phi \circ S + T \circ \mathcal{A} \circ \pi \circ S,$$

which reveals that the public key of a PMI cryptosystem is exactly a public key $T \circ \phi^{-1} \circ \mathcal{Q} \circ \phi \circ S$ of an MIC* cryptosystem, plus some noise in the form of

$T \circ \mathcal{A} \circ \pi \circ S$. The attack will remove that noise, reducing to the case of the MIC* cryptosystem, for which effective attacks have been shown to exist, as noted in Section 5.4.

Removing the noise $T \circ \mathcal{A} \circ \pi \circ S$ is as easy as finding a way to replace it with a constant. This can be done by computing the linear space $\mathcal{K}$, the kernel of the linear portion of $\pi \circ S$. As noted in Section 3.5.2, this attack uses differential equations. These differential equations are used to create a test for membership in $\mathcal{K}$.

5.5.1 Membership in $\mathcal{K}$

The test for membership in $\mathcal{K}$ is based on several lemmas. First, let $k_1, k_2 \in \mathbb{F}_q^n$ and let $\mathcal{P}_1$ and $\mathcal{P}_2$ be systems of quadratic equations, and let T and S be affine bijections as usual. Then recall some facts about $L_{\mathcal{P},k}(x) = d\mathcal{P}_k(x) - d\mathcal{P}_k(0)$:

$$L_{\mathcal{P}_1,k_1+k_2} = L_{\mathcal{P}_1,k_1} + L_{\mathcal{P}_1,k_2}. \tag{5.5}$$

$$L_{\mathcal{P}_1+\mathcal{P}_2,k_1} = L_{\mathcal{P}_1,k_1} + L_{\mathcal{P}_2,k_1}. \tag{5.6}$$

$$L_{T\circ\mathcal{P}_1\circ S,k} = T \circ L_{\mathcal{P}_1,S(k_1)} \circ S + T \circ \mathcal{P}_1 \circ S(0) - T \circ \mathcal{P}_1(0). \tag{5.7}$$

$$L_{\mathcal{P}_1,0} = 0. \tag{5.8}$$

Each of these lemmas is proven as by Fouque, Granboulan and Stern [FGS05]. The first two lemmas are Lemma 5.7 and Lemma 5.8, as found on page 66 and the following. They provide the support needed to deal with the lemmas and remarks in the remainder of this section. They are simple statements from algebra. The remaining lemmas concern $L_{\mathcal{P},k}$, as defined above.

Lemma 5.11 *If $\mathcal{U} = T \circ \phi^{-1} \circ \mathcal{Q} \circ \phi \circ S$ is the public key of a **MIC*** system over $\mathbb{F}_q$ of characteristic 2, dimension n and exponent $q^\ell+1$, then* $\dim(\ker L_{T\circ\phi^{-1}\circ\mathcal{Q}\circ\phi\circ S,k}) = \gcd(\ell, n)$.

Proof: The function $T \circ \phi^{-1} \circ \mathcal{Q} \circ \phi \circ S$ is a composition of the bijective functions T and S, the linear bijection ϕ and the function $\mathcal{Q}$. Therefore,

$$\dim(\ker L_{T\circ\phi^{-1}\circ\mathcal{Q}\circ\phi\circ S,k}) = \dim(\ker L_{\mathcal{Q},k}).$$

To show this, first note that by Fact 5.5, we have that for any bijection Φ,

$$\begin{aligned} L_{\mathcal{Q}\circ\Phi,k}(x) &= L_{\mathcal{Q},\Phi(k)} \circ \Phi(x) + \mathcal{Q} \circ \Phi(0) - \mathcal{Q}(0) \\ &= \mathcal{Q} \circ \Phi(x+k) - \mathcal{Q} \circ \Phi(x) - \mathcal{Q} \circ \Phi(k) + \mathcal{Q} \circ \Phi(0) + \mathcal{Q} \circ \Phi(0) - \mathcal{Q}(0) \\ &= \mathcal{Q} \circ \Phi(x+k) - \mathcal{Q} \circ \Phi(x) - \mathcal{Q} \circ \Phi(k) + \mathcal{Q} \circ \Phi(0). \end{aligned}$$

So, if there is a bijection composed with $\mathcal{Q}$ on the right, it clearly does not change the dimension of the kernel. Thus,

$$\dim(\ker L_{\mathcal{Q}\circ\phi\circ S,k}) = \dim(\ker L_{\mathcal{Q},k}).$$

Similarly, by Fact 5.5, we have that for any bijection Φ,

$$\begin{aligned} L_{\Phi\circ\mathcal{Q},k}(x) &= \Phi\circ L_{\mathcal{Q},k}(x) + \Phi\circ\mathcal{Q}(0) - \Phi\circ\mathcal{Q}(0) \\ L_{\Phi\circ\mathcal{Q},k}(x) &= \Phi\circ L_{\mathcal{Q},k}(x). \end{aligned}$$

So, if there is a bijection composed with $\mathcal{Q}$ on the left, it clearly does not change the dimension of the kernel. Therefore it is true that

$$\dim(\ker L_{T\circ\phi^{-1}\circ\mathcal{Q}\circ\phi\circ S,k}) = \dim(\ker L_{\mathcal{Q},k}).$$

This implies that only the dimension of the kernel of the linear portion of the equation $\mathcal{Q}(z) = z^{q^\ell+1}$ matters. So, for this proof let $z = \phi(x)$ and $j = \phi(k)$. Then $L_{\mathcal{Q},j} = (z^{q^\ell}\cdot j + z\cdot j^{q^\ell})$. Then clearly $z = 0$ is in the kernel of $L_{\mathcal{Q},j}$. For the same reason, $0 \neq z \in \ker(L_{\mathcal{Q},j})$ if and only if $z^{q^\ell}\cdot j + z\cdot j^{q^\ell} = 0$. This equation can be rewritten as

$$z^{q^\ell+1}\cdot\left(\frac{j}{z} + \left(\frac{j}{z}\right)^{q^\ell}\right) = 0.$$

As $z \neq 0$, let $X = \frac{j}{z}$, and note that it remains to solve $X + X^{q^\ell} = 0$ in the finite field $\mathbb{F}_{q^n}$. When $X \neq 0$ (equivalently, when $j \neq 0$), this equation becomes $X^{q^\ell-1} = 1$ in a finite field of characteristic 2. So, because $X = 1$ is a solution, by Lemma 5.8, there are $\gcd(q^\ell-1, q^n-1)$ solutions. By Lemma 5.7, $\gcd(q^\ell-1, q^n-1) = q^{\gcd(\ell,n)}-1$. Add the solution $X = 0$ and there are a total of $q^{\gcd(\ell,n)}$ solutions. So, $\dim(\ker L_{\mathcal{U},k}) = \gcd(\ell, n)$. $\diamond$

Note that $X = 1$ is always a solution, which means that $x = k$, and thus k is always in the kernel. If $\gcd(\ell, n) = 1$, then k is the only solution.

Lemma 5.12 *If $\mathcal{P}$ is the public key of the **PMI** cryptosystem and $k \in \mathcal{K}$, then* $\dim(\ker(L_{\mathcal{P},k})) = \gcd(\ell, n)$.

Proof: Assume that $k \in \mathcal{K}$. Recall that $\mathcal{P}$ can be written as

$$\mathcal{P} = T\circ\phi^{-1}\circ\mathcal{Q}\circ\phi\circ S + T\circ\mathcal{A}\circ\pi\circ S,$$

where the left term in the addition is the public key $\mathcal{U}$ of an instance of the MIC* cryptosystem. We write

$$\mathcal{U} = T\circ\phi^{-1}\circ\mathcal{Q}\circ\phi\circ S.$$

Now we will show that for this MIC* cryptosystem $\mathcal{U}$ associated with $\mathcal{P}$, we have

$$L_{\mathcal{P},k} = L_{\mathcal{U},k}.$$

Step one is to notice that $k \in \mathcal{K}$ is equivalent to $\pi\circ S(k) = \pi\circ S(0)$. Then we compute

$$\begin{aligned} L_{\mathcal{P},k}(x) - L_{\mathcal{U},k}(x) &= L_{T\circ A\circ\pi\circ S,k}(x) \\ &= T\circ A\circ\pi\circ S(x+k) - T\circ A\circ\pi\circ S(x) \\ &\quad -T\circ A\circ\pi\circ S(k) + T\circ A\circ\pi\circ S(0). \end{aligned}$$

Therefore, since $\pi \circ S(k) = \pi \circ S(0)$, we have

$$\begin{aligned} T^{-1}(L_{\mathcal{P},k}(x) - L_{\mathcal{U},k}(x)) &= A \circ \pi \circ S(x+k) - A \circ \pi \circ S(x) \\ &\quad -A \circ \pi \circ S(k) + A \circ \pi \circ S(0) \\ &= A \circ \pi \circ S(x+k) - A \circ \pi \circ S(x) \\ &= 0. \end{aligned}$$

Therefore, $T^{-1} \circ L_{\mathcal{P},k}(x) = T^{-1} \circ L_{\mathcal{U},k}(x)$. So $\dim(\ker(L_{\mathcal{P},k}) = \dim(\ker(T^{-1} \circ L_{\mathcal{P},k}) = \dim(\ker(T^{-1} \circ L_{\mathcal{U},k}) = \dim(\ker(L_{\mathcal{U},k})$. $\diamond$

These lemmas lead to a corollary result.

Theorem 5.13 *If $\mathcal{P}$ is the public key of the **PMI** cryptosystem and if* $\dim(\ker(L_{\mathcal{P},k})) \neq \gcd(\ell, n)$*, then $k \notin \mathcal{K}$.*

Proof: This statement is the contrapositive of Lemma 5.12. $\diamond$

Now it is possible to define a test to decide whether or not a vector k is in $\mathcal{K}$. Let H be a boolean function such that $H(k) = 1$ if $\dim(\ker(L_{\mathcal{P},k})) \neq \gcd(\ell, n)$, and $H(k) = 0$ otherwise. Thus, if $H(k) = 1$, then with probability 1, $k \notin \mathcal{K}$. If $H(k) = 0$, then it is not known whether or not k is in $\mathcal{K}$. The computation of the dimension of the kernel is the most expensive part of this test, taking $O(n^3)$ field operations to determine the rank and nullity of the linear transformation $L_{\mathcal{P},k}$.

In order to see that this test is reasonable, Fouque, Granboulan and Stern [FGS05] make the following remark.

Remark 5.14 *If $\mathcal{P}$ is the public key of the **PMI** cryptosystem and $k \notin \mathcal{K}$, then often* $\dim(\ker(L_{\mathcal{P},k})) \neq \gcd(\ell, n)$.

It is noted by Fouque, Granboulan and Stern [FGS05] that this remark can be verified experimentally. The paper also gives a rationale for this experimental success. The argument is that the term $L_{T \circ A \circ \pi \circ S,k}(x)$ is no longer null, as $k \notin \mathcal{K}$, but that since it is a random-looking linear function, the dimension of the kernel of the sum $L_{\mathcal{P},k}$ follows the distribution of the dimension of the kernel of random linear maps.

5.5.2 Determining $\mathcal{K}$

The test H defined at the end of Section 5.5.1 can be used to create algorithms that will determine a basis for $\mathcal{K}$. Let $\alpha = \Pr[H(k) = 0]$ and let $\beta = \Pr[k \in \mathcal{K}] = q^{-r}$. The algorithms that will use the test H will use the linearity of $\mathcal{K}$ as well. Each algorithm uses the assumption that for any fixed value k and random value k', the probability that $H(k + k') = 0$ is independent of the probability that $H(k') = 0$.

The two suggested algorithms of [FGS05] are rather different, and it is declared that a real-world attack will use a mix of the two algorithms. Thus, they are both included here.

If it is true that $\alpha - \beta \ll \beta$, then there are almost no false positives in the test H. However, this is rarely the case. Normally, $\gcd(\ell, n) = 1$, and therefore $\beta \ll \alpha$.

Technique 1

This technique operates under the idea that if for many different $k' \in \mathcal{K}$, $k+k' \in \mathcal{K}$, then k is in $\mathcal{K}$ as well. So, if there are many different k' such that $H(k') = 0$ and $H(k+k') = 0$, then $k \in \mathcal{K}$ with high probability. Let $p(k) = \Pr[H(k+k') = 0 | H(k') = 0]$. Note that for a random k, $k + k'$ is uniformly distributed, so $p(k) = \alpha$. However, if $k \in \mathcal{K}$, then

$$\begin{aligned} p(k) &= \Pr[k' \in \mathcal{K} | H(k') = 0] + \Pr[k' \notin \mathcal{K} | H(k') = 0] \cdot \Pr[H(k + k') = 0 | k + k' \notin \mathcal{K}] \\ &= \frac{\beta}{\alpha} + \frac{\alpha - \beta}{\alpha} \cdot \frac{\alpha - \beta}{1 - \beta}. \end{aligned}$$

Thus, if in addition we assume that $\beta \ll \alpha$, in the case $k \in \mathcal{K}$, we have $p(k)/\alpha = \frac{1-\beta/\alpha}{1-\beta} + \frac{\beta}{\alpha^2} \simeq 1 + \beta(\alpha^{-1} - 1)^2$. So, the difference in the expected value of $p(k)$, depending on whether or not $k \in \mathcal{K}$, is on the order of $\alpha\beta$. So, take $N = \frac{1}{(\alpha\beta)^2}$ elements k' such that $H(k') = 0$ and compute the average of $H(k + k')$. This computed average will determine whether or not k is an element of $\mathcal{K}$, as if it is nearly α, then it is random, while if this average is on the order of $\alpha\beta$ removed from α, then $k \in \mathcal{K}$. The complexity of this test is about β^{-2}, according to [FGS05].

Thus, to determine $\mathcal{K}$ totally, this test must run on the order of $n\beta^{-1}$ times, as approximately n distinct elements of $\mathcal{K}$ are necessary to form a basis, and it takes approximately β^{-1} randomly chosen elements to choose an element of $(\mathbb{F}_q)^n$ that is an element of $\mathcal{K}$. This implies a total complexity on the order of $n\beta^{-3} = nq^{3r}$ to determine $\mathcal{K}$ using this technique.

Technique 2

This technique works with graphs. We define a graph whose vertices are the elements k such that $H(k) = 0$. These are precisely the elements that might be in the kernel. For each pair of vertices (k, k') we place an edge if and only if $H(k + k') = 0$. Note that in this graph, all the vertices of $\mathcal{K}$ are connected to each other, and are therefore in the same large clique. In practice, the entirety of this graph is too large to create. Instead, it is formed up to a restricted number of vertices N. If $N > n/\beta$, then according to Fouque, Granboulan and Stern [FGS05], it is likely that the graph has sufficient vertices to contain $n - r$ independent elements of $\mathcal{K}$. The goal is to seek a largest clique in the graph we construct, which will contain a basis of $\mathcal{K}$, given the same assumption of independence used by the first technique, and using general results on random graphs, again as noted by Fouque, Granboulan and Stern. Note that if the probability that $H(k + k') = 0$ is independent of the probability $H(k) = 0$, then for this graph restricted to N randomly chosen vertices, there are αN^2

edges. By our assumptions, these edges are randomly distributed wherever they do not correspond to elements of $\mathcal{K}$. Now Fouque, Granboulan and Stern use results from graph theory to estimate that the number of vertices in the clique of maximal order in a random graph of N vertices with a probability α between each edge is $\frac{2\log N}{log 1/\alpha} + O(\log\log N)$. Thus, if βN is much larger than $\frac{2\log N}{log 1/\alpha}$, there will be a unique large clique, and it will contain a basis of $\mathcal{K}$. Again, assuming $\beta \ll \alpha$, this condition is equivalent to $N \approx \beta^{-1}\log\beta^{-1}$. Thus, the whole complexity for finding $\mathcal{K}$ is $a^{2r}\log q^{2r}$.

Combining Techniques

In [FGS05] it is advised that in an implemented attack, the attacker would use Technique 1 to identify some good elements (ones likely to be in $\mathcal{K}$) and Technique 2 to extract a basis of $\mathcal{K}$ from a large clique in a graph of N vertices, some of which were chosen using technique 2.

5.5.3 Attacking the MIC* portion

Now that the desired kernel $\mathcal{K}$ has been found, it is possible to mount the last stage of the attack. We iterate q^r attacks on the various possible MIC* schemes formed by replacing $T \circ A \circ \pi \circ S$ by each of the q^r possible constants, in the q^r affine subspaces parallel to $\mathcal{K}$. Since for speedy decryption, q^r is not too large, this attack is feasible. The remaining question, is the attack on the MIC* scheme of Section 5.4 feasible when x is restricted to a subspace is yes, according to [FGS05]. In fact, it is noted that this can only increase the number of independent equations found in the attack, which favours the attacker.

The attack, therefore, ends by reducing to the case of the MIC* cryptosystem, and performing an attack on that system. The setup phase prior to running the attack on the MIC* portion takes, according to Fouque, Granboulan and Stern [FGS05] is a precomputation of order $O(nq^{3r} + n^6 q^r)$ and a precomputation of order $O(n^3 \times q^r \times q^{\gcd(\ell,n)})$ using Technique 1. This cost dwarfs the expected cost of an attack on the MIC* cryptosystem (see Section 5.4 for details). Fouque, Granboulan and Stern [FGS05] did not implement this attack themselves, so they did not have practical results. Still, as a consequence, Ding and Gower [DG05] modified PMI to attempt to defeat this attack. See Section 6.2.3 for details.

Chapter 6

Special Topics

6.1 Quantum computing is not a panacea

It would certainly be desirable, at least for a cryptanalyst, to be able to use a quantum computer (perhaps our cryptanalyst is from the future) to defeat an $\mathcal{MQ}$-based scheme in polynomial time. Already, RSA and ECC-based cryptography are known to be susceptible to such attacks. So far, however, there has not been a quantum algorithm found that does such a thing. In fact, one of the nicer aspects of $\mathcal{MQ}$-based cryptosystems is that no clear route to defeat them using the power of quantum computing is known. In general, it is not known whether a quantum computer can solve an instance of an $\mathcal{NP}$-complete problem in less than exponential time in the size of the problem. Since the $\mathcal{MQ}$ problem can be shown to be $\mathcal{NP}$-complete, as noted by Patarin and Goubin [PG97], the two major quantum algorithms are not useful.

The first major quantum algorithm is Shor's algorithm [Sho97], which is used to solve what is known as the Hidden Subgroup Problem for an Abelian group in polynomial time. The Discrete Logarithm Problem and the Factorization Problem can be seen as instances of a Hidden Subgroup Problem for an Abelian group, so Shor's algorithm can solve such problems in polynomial time as well. Thus, in a world with a sufficiently advanced quantum computer, RSA and elliptic curve cryptography are not secure. However, to the knowledge of this author, there is no known transformation of the problem of finding the plaintext x of an $\mathcal{MQ}$-based scheme from its associated ciphertext y, using only the public key $\mathcal{P}$ into an instance of the Hidden Subgroup Problem for an Abelian group. So, Shor's algorithm does not appear to be helpful.

This leaves us with Grover's algorithm [Gro96]. Grover's algorithm allows any space of N elements to be searched in $O(\sqrt{N})$ time. It has been shown by Bennet, Bernstein, Brassard and Vazirani [BBBV97] to be optimal for solving $\mathcal{NP}$-complete problems in this way. So, if one views the problem of decrypting a ciphertext y into its plaintext x as solving an instance of an $\mathcal{MQ}$ problem, then quantum computing is certainly not a panacea. Quantum computing provides

a quadratic speed-up using Grover's algorithm, and that is all. This is certainly nothing to ignore, but such an ability on the part of an attacker necessitates requiring larger keys, not scrapping a system entirely. Furthermore, it is the same attacking power that an attacker has to solve the ECDLP in a conventional computational model as-is.

It is, of course, possible that as no $\mathcal{MQ}$-based cryptosystem uses a truly random instance of the $\mathcal{MQ}$ problem (they must have trapdoors to be useful), there is a quantum algorithm (or several algorithms) that are capable of solving certain classes of $\mathcal{MQ}$-based cryptosystems in polynomial time. However, no such algorithms exist at this time, to the knowledge of the author.

6.2 Patching systems

Not every system introduced in this survey has been totally defeated, and there exist variants of those systems that resist all known attacks. We now seek to shed some light on a few such systems. The aim of this section is to note that not every $\mathcal{MQ}$-based cryptosystem has been defeated, and that a system that has a powerful attack can often be patched to prevent such an attack.

6.2.1 Basic HFE pros and cons

The HFE cryptosystem is described in Section 4.3. Recall that it relies on a system of n public multivariate equations in n indeterminates, and has a private univariate polynomial of degree no greater than d. However, not all agree as to the security of the system. For some, such as Wolf and Preneel [WP04], the basic HFE system is broken from a cryptanalytic point of view, due to the Kipnis-Shamir attack of Section 5.3. Recall that the Kipnis-Shamir attack of Section 5.3 is a MinRank-based attack that first recovers T, then recovers the pair $\mathcal{P}'$ and S simultaneously by solving a large linear system derived from the public key and the calculated T. Others, such as Faugère and Joux [FJ03], are a bit less inclined to declare the scheme insecure, calling for a "re-evaluation of the security of HFE based cryptosystems." On the other hand, Courtois [Cou05] maintains a website with security recommendations and estimates for HFE, despite his own work on HFE. Courtois's work [Cou01] indicates that attacking HFE has a complexity of $O(n^{\omega \log_q d})$, where ω is the exponent of Gaussian elimination. Faugère [Fau03] has shown through experiments that for HFE systems with $n \leq 160$ and $d \leq 512$, a Gröbner basis attack will run in $O(n^{10})$ time. This is a significant improvement over the bound of $O(n^{\omega \log_q d})$ given by Courtois. Some of Courtois' recommended security parameters, as mentioned in Section 4.3.4, become unusable in this light, and the final recommendation, $q = 2, d = 25, n = 251$ lies beyond the scope of Faugère's results only because Faugère's results did not cover $n \geq 161$. So, while this choice of parameters may formally lie outside the bounds of all known attacks, it is on less firm ground than one might expect. Still, even in the worst case, if a Gröbner basis attack will run in $O(n^{10})$ time even for $q = 2, d = 25, n = 251$, we have about

2^{80} operations before the Gröbner basis attack completes. That is, as noted in Section 2.3.4, approximately the security level of a RSA using a 1024-bit modulus. Note, though, that the estimate of Courtois [Cou01] for this case is $O(n^{4.64\omega})$. For $\omega = 3$, this becomes $O(n^{13.93})$. The estimate of Faugère [Fau03] is still $O(n^{10})$, by assumption. So, Faugère's estimate is significantly smaller than the estimate of Courtois, if it still holds for $n = 251$.

So, who is correct? It seems only a matter of time before further experiments increase the known range in which Gröbner bases attacks outperform the more specific attack of Kipnis and Shamir [KS99]. Between that and general computing power increases, the required size for n will keep increasing, resulting in larger keys and slower performance. Most likely, HFE is a dead-end—parameters sufficient to resist Gröbner basis attacks will quickly prove so large as to render decryption (which has a complexity of $O(n^4 d^2 \log(d))$) inhibitively slow.

6.2.2 HFE-

HFE- (read: "HFE minus") is a variant of basic HFE that resists all known attacks. Key generation is done in the following way. Let $(T,\ \mathcal{P}',\ S)$ be the private key of a basic HFE system. It remains unchanged as the private key of our HFE- system. The public key, however, is now given by:

$$\mathcal{P} = R \circ T \circ \mathcal{P}' \circ S,$$

where R is the function given by:

$$\begin{array}{rccc} R: & \mathbb{F}_q^n & \longrightarrow & \mathbb{F}_q^{n-\ell}, \\ & (y_1, y_2, \ldots, y_n) & \longmapsto & (y_1, y_2, \ldots, y_{n-\ell}). \end{array}$$

In HFE-, the value ℓ is a parameter. Note that the only difference between basic HFE and HFE- is that basic HFE retains the last ℓ indeterminates of the ciphertext, while they are never computed for HFE-. Encryption is essentially identical to basic HFE: simply input the plaintext into the public key, as usual. Decryption is accomplished by first guessing the values of the ℓ indeterminates deleted by R, and then attempting to decrypt as usual. So, where previously 1 decryption was necessary, now we expect to use $O(q^\ell)$ decryption attempts. Thus, ℓ cannot be too large, or the system becomes unusable.

Current attacks are ineffective against HFE-: the Kipnis-Shamir attack of Section 5.3 falls apart; the final ℓ public matrices of a basic HFE system simply do not exist in the HFE- equivalent system, and it is no longer possible to work with T directly, even in the abstract. The work of Faugère and Joux [FJ03], which is simply a set of estimates for the cost of a Gröbner basis attack, falls apart as well, as an HFE- cryptosystem does not follow the estimates Faugère [Fau03] calculated for basic HFE. Faugère and Joux [FJ03] note that every dropped ciphertext character y_i, for $n - \ell + 1 \leq i \leq n$, essentially doubles the maximum possible degree d of a system with respect to their original framework, as found in [Fau03]. So, for ℓ not too large, an HFE- system moves well beyond

the parameters $n \leq 160, d \leq 512$ of Faugère's [Fau03] Gröbner basis attack on basic HFE.

6.2.3 PMI+

The PMI cryptosystem is described in Section 4.5. Recall that it relies on a so-called "internal perturbation" of the MIC* cryptosystem for its security, and that there is an attack that reduces attacking the PMI cryptosystem to attacking the MIC* cryptosystem multiple times (see Section 5.5 for details). In order to circumvent this attack, Ding and Gower [DG05] have introduced a new variant of PMI, which they have dubbed PMI+ (read: "PMI plus"), as it functions by simply adding more quadratic equations to the original PMI cryptosystem.

Key generation in PMI+ is nearly identical. In the original PMI cryptosystem, the private key consists of $(T, \phi, \mathcal{Q}, \mathcal{A}, \pi, S)$, where ϕ and $\mathcal{Q}$ are as described in Section 4.4.1 and π is as described in Section 4.5.1. Recall that $\mathcal{Q}$ is a univariate function given by:

$$Q(z) = z^{q^\ell+1},$$

where $\ell \in [1, n-1]$ is uniformly randomly chosen until

$$\gcd(q^\ell + 1, q^n - 1) = 1.$$

As usual, the system uses n indeterminates with m multivariate quadratic equations in the public key

$$\mathcal{P} = T \circ \phi^{-1} \circ \mathcal{Q} \circ \phi + \mathcal{A} \circ \pi \circ S.$$

Recall that the permutation in the PMI cryptosystem is to add an internal pertubation to a standard MIC* cryptosystem in the form of n random quadratic polynomials (represented by $\mathcal{A}$) in r indeterminates created by an affine linear transformation π of the output of S. The PMI+ cryptosystem is similar. The private key is changed in two ways. First, a new parameter b and an associated map $\mathcal{B}$ are introduced, given by:

$$\begin{aligned} \mathcal{B}: \quad \mathbb{F}_q^n &\longrightarrow \mathbb{F}_q^{n+b}, \\ \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} &\longmapsto \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \\ B_1(v_1, v_2, \ldots v_n) \\ B_2(v_1, v_2, \ldots v_n) \\ \vdots \\ B_b(v_1, v_2, \ldots v_n) \end{pmatrix}, \end{aligned}$$

where each of the B_i, for $1 \leq i \leq b$ is a uniformly randomly determined multivariate quadratic equation in n indeterminates with coefficients from $\mathbb{F}_q$. In

the PMI+ cryptosystem, S is an affine invertible transformation on n indeterminates, as before, but T is now an affine invertible transformation on $n + b$ indeterminates. The PMI+ private key is given by $(T, \mathcal{B}, \phi, \mathcal{Q}, \mathcal{A}, \pi, S)$, and the public key is given by:

$$\mathcal{P} = T \circ \mathcal{B} \circ \phi^{-1} \circ \mathcal{Q} \circ \phi + \mathcal{A} \circ \pi \circ S.$$

In other words, the PMI+ cryptosystem simply adds b multivariate quadratic equations to the vector of multivariate quadratic equations determined by $\phi^{-1} \circ \mathcal{Q} \circ \phi + \mathcal{A} \circ \pi \circ S$.

Encryption is no different than in the original PMI cryptosystem, although it is of course done with a PMI+ public key. Decryption is accomplished in the same way as the original PMI cryptosystem, by simply ignoring the polynomials $B_1, B_2, \ldots, B_b$ when they are encountered, and proceeding as usual for decrypting a PMI ciphertext.

Ding *et al.* [DG05, DGS+05] show through experiments that for $\delta = \gcd(\ell, n)$, adding $b = \delta + 10$ equations is sufficient to secure a PMI+ system with $n = 136$, $q = 2$, $r = 6$ and $\gcd(\ell, n) = \delta$ against the attack on PMI of Section 5.5. Recall that this was a differential analysis-based attack that reduced the PMI system to the case of the MIC* cryptosystem from which it was derived. They note that doing so increases the expected attack time to at least 2^{80} operations. They also note that the following parameters are sufficient to provide a baseline security requirement of 2^{80} operations:

$$\begin{aligned} q &= 2 \\ n &> 84 \\ r &= 6 \\ \gcd(\ell, n) &= 1 \\ b &= 11. \end{aligned}$$

Ding and Gower [DG05] caution against using more than the suggested choice of $b = \delta + 10$ added equations, as sufficiently overdefined systems of equations can be attacked more efficiently, as noted by Courtois *et al.* [CKPS00].

This system's resistance to further attacks has not yet been tested, so it would be premature to consider this "fix" to PMI to be both necessary and sufficient for a secure $\mathcal{MQ}$-based scheme.

6.3 Multivariate schemes not appearing in this thesis

A number of multivariate schemes exist that do not appear in this thesis. A complete listing of all multivariate encryption schemes was certainly not the goal of this thesis, so schemes for which no novel attacks exist, such as the HFEv scheme introduced by Wolf and Preneel [WP04] and shown by Ding and Schmidt [DS05] to succumb to Gröbner basis-based attacks, were passed over. Some schemes, while based on the $\mathcal{MQ}$ problem, fail to be useful as encryption

schemes. These tend to be signature schemes rather than encryption schemes in any case. One example of such schemes is the so-called unbalanced oil and vinegar scheme, which uses a trapdoor not even mentioned in this thesis. According to Braeken, Wolf and Preneel [BWP04], the unbalanced oil and vinegar scheme has the advantage of remaining secure against forgery attempts provided the appropriate key choices are made. Braeken, Wolf and Preneel [BWP04] also note that these same key choice requirements render unbalanced oil and vinegar schemes unusable as encryption schemes.

Additionally, most of the schemes in this thesis have signature scheme variants, and all of these signature scheme variants have been ignored as well.

Chapter 7

Future work and conclusions

As this thesis draws to a close, we wrap up with some of the unfinished business in attacking $\mathcal{MQ}$-based encryption schemes. We close with some conclusions.

7.1 Future work

As always in cryptography, there is work to be done. Here, that remaining work will be divided into two types: future work in *attacking* $\mathcal{MQ}$-based schemes and future work in *creating* $\mathcal{MQ}$-based schemes. In keeping with the title of this thesis, the former will be discussed first and in greater detail.

7.1.1 Future work in attacking $\mathcal{MQ}$-based schemes

The first thought one might have when considering the "future work" in attacking $\mathcal{MQ}$-based schemes is to attempt to break the patched systems. Indeed, there is work to be done there. In general, the method of Gröbner basis-based attacks is used on a fairly regular basis. There is as of now no good system in place for determining how long a Gröbner basis-based attack will require to extract a plaintext from a general $\mathcal{MQ}$-based encrypted ciphertext. Estimates have been made for HFE by Faugère and Joux [FJ03], and other estimates have been made for PMI by Ding *et al.* [DGS+05], but in both cases, the estimates are simply the result of computing many Gröbner basis-based attacks on said schemes. While this is a valid technique, it does not generalize to other schemes, and may not generalize to the original systems themselves beyond certain parameter choices. We saw in Section 6.2.1 that the experiments of Faugère and Joux [FJ03] only extended to $n \leq 160$ and $d \leq 512$ (See Section 4.3.1 for details on the key structure of HFE). In any case, improvements to our knowledge of the speed of Gröbner basis-based attacks (especially with respect to basic HFE)

will probably put the last nails in the coffin for using basic HFE with a reasonable choice of parameters—ones that do not render decryption uselessly slow. Continuing on this "attack the patched systems" line of thought, we also have HFE- and PMI+ to consider. So far, both resist all known attacks. Perhaps the attacks on HFE and PMI can be extended to cover HFE- and PMI+. Or perhaps other attacks exist suited more to HFE- and PMI+ themselves.

So far, most attacks on $\mathcal{MQ}$-based schemes are either extremely specific, such as the MinRank-based attack on HFE, or extremely generic, such as a Gröbner basis-based attack. While it has been mentioned that the generic attack using Gröbner bases has a difficult to predict running time, not much has been said about the specific attacks. They all suffer from a common problem– once the specific weakness they exploit has been removed, these attacks often become useless, or at least much less effective, as noted by Ding and Gower *et al.* [DG05] for PMI+, and by Wolf and Preneel [WP04] for HFE-. Extending these specific attacks to a more general case may eventually become necessary to attack new cryptosystems. An example where this may be the case is the IPHFE cryptosystem introduced by Ding and Schmidt [DS05]. It uses the exact same internal perturbation of the PMI cryptosystem, but applies it to the HFE cryptosystem, rather than the MIC* cryptosystem. Ding and Schmidt [DS05] note that encryption and decryption hold no real surprises to those familiar with the HFE system and the PMI system (see Sections 4.3, 4.5, respectively), and go on to suggest without proof that the IPHFE cryptosystem resists all the attacks on HFE (both the one due to Kipnis and Shamir [KS99] and the much faster Gröbner basis-based attack, whose effectiveness was noted by Faugére and Joux [FJ03]). Since the debut of the IPHFE cryptosystem [DS05], to our knowledge no work has been reported on it. Ding and Gower [DG05] suggest that the attack of Fouque, Granboulan and Stern [FGS05] on PMI may not work on the IPHFE cryptosystem. However, they admit that they lack the experimental and theoretical evidence to confirm this.

So, what may have happened with the IPHFE system is that by combining various "base" ideas for $\mathcal{MQ}$-based encryption schemes, a new scheme has been created that is stronger than its constituent parts. Finding ways to extend the HFE and PMI attacks to cover IPHFE is one idea. The natural counter to this idea from the perspective of the cryptanalyst is to find ways to generalize the current crop of attacks on HFE and PMI and other systems. Hopefully, such generalizations will make it easier to understand why, say, IPHFE is (or is not) more secure than HFE or PMI.

One attack technique in this paper is particularly deserving of further study. The Dixon resultant-based technique of Section 3.3 is a polynomial time algorithm, according to Tang and Feng [TF05], when dealing with particular kinds of sparse systems of quadratic equations. Expanding the set of sparse systems that work, as well as possibly making such a technique work in general systems are obvious new avenues of research.

7.1.2 Future work in creating $\mathcal{MQ}$-based schemes

The IPHFE cryptosystem mentioned in the previous section is a good starting point for the future work in creating $\mathcal{MQ}$-based schemes. While further work must be done to confirm or deny this, the general idea of combining existing ideas seems to be a good one. In fact, Wolf and Preneel have an excellent paper, titled *Taxonomy of Public Key Schemes based on the problem of $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Equations* [WP05]. In [WP05, Section 7], Wolf and Preneel suggest several new schemes, each created by combining ideas from various older schemes in new ways.

Of course, it is always desirable to find new trapdoor conditions for the private key term $\mathcal{P}'$ not created by combining old trapdoor conditions, a sentiment also expressed by Wolf and Preneel [WP05, Section 8]. To generalize a bit, any new idea for constructing an $\mathcal{MQ}$-based encryption scheme will remain welcome, even if there is the trend is to reuse older ideas in new ways in future schemes, rather than to introduce new trapdoor conditions.

Finally, in another partial echo of Wolf and Preneel [WP05, 8], it should be noted that there are currently no $\mathcal{MQ}$-based encryption schemes that are both secure and efficient, compared to ECC and RSA. As is, $\mathcal{MQ}$-based encryption schemes must choose security (e.g., HFE or HFE-, which have very slow decryption speeds at a usable security level) or speed (e.g., MIC*, which is not at all secure). Finding a secure, efficient technique would be most desirable.

7.2 Conclusions

There are many attacks on $\mathcal{MQ}$-based encryption schemes. However, we have noted that the attacks on such schemes fall largely into 2 groups. First, there are system-agnostic attacks, such as the methods using Gröbner bases (and, historically, the XL method) and Dixon resultants. Second, there are attacks that remain highly specific; MinRank attacks and differential attacks require very specific knowledge of the system they are used against. Most other specific attacks fail to earn themselves the distinction of a category, and are simply considered specific attacks, such as Patarin's [Pat05] attack on MIC*.

Furthermore, we have seen that often a system is created, as PMI was by Ding [Din04] and then quickly broken, as Fouque, Granboulan and Stern [FGS05] did with their PMI attack. Then, after a round of fixes, the cycle starts anew. For MIC*, there were HFE and PMI as follow-up systems. More recently, the ideas of PMI and HFE have been combined to create IPHFE. And by looking at the dates of the relevant papers, the time lag between a new system, a new attack, and a new variant that resists that attack is quickly decreasing. The MIC* to HFE cycle took nearly a decade [MI88, Pat95, Pat96b] while the PMI to PMI+ cycle has happened over the past two years or so [Din04, FGS05, DG05]. Arguably, the change between PMI and PMI+ was less complex than the change between MIC* and HFE, but it is perhaps unfair to consider one security improvement more difficult to conceive than another sim-

ply because it is harder to implement efficiently. Judging by the widening pool of names associated with multivariate cryptography papers, it seems as if more individuals are working in the area as well.

In any case, $\mathcal{MQ}$-based cryptosystems seem to be gaining momentum and recognition. Hopefully, they will gain more attention as time goes by, and more people will work in this area. They are an alternative to the world of public key cryptography based on the integer factorization and discrete logarithm problems. In the same vein, the lack of effective quantum-based attacks on $\mathcal{MQ}$-based cryptosystems makes them an interesting "back-up" set of cryptosystems in the off-chance some organization is hiding a useful quantum computer somewhere, or if such a machine is ever built.

Bibliography

[AFI+04] Gwénolé Ars, Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, and Makoto Sugita, *Comparison between XL and Gröbner basis algorithms*, ASIACRYPT 2004 (P.J. Lee, ed.), Lecture Notes in Computer Science, vol. 3329, Springer, 2004, pp. 338–353.

[BBBV97] Charles H. Bennet, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani, *The strengths and weaknesses of quantum computation*, SIAM Journal on Computing **26** (1997), 1510–1523.

[Buc65] Bruno Buchberger, *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomial*, Ph.D. thesis, Leopold-Franzens-Universität, 1965.

[BWP04] An Braeken, Christopher Wolf, and Bart Preneel, *A study of the security of unbalanced oil and vinegar signature schemes*, Cryptology ePrint Archive, Report 2004/222, 2004, `http://eprint.iacr.org/`.

[CKPS00] Nicolas T. Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir, *Efficient algorithms for solving overdefined systems of multivariate polynomial equations*, 2000, Available at `http://www.minrank.org/xlfull.pdf`.

[CLO92] David Cox, John Little, and Donal O'Shea, *Ideals, varieties and algorithms*, Springer-Verlag, 1992.

[CLO98] David Cox, John Little, and Donal O'Shea, *Using algebraic geometry*, Springer-Verlag, 1998.

[Cou01] Nicolas T. Courtois, *The security of hidden field equations (HFE)*, The Cryptographer's Track at RSA Conference 2001 (D. Naccache, ed.), Lecture Notes in Computer Science, vol. 2020, Springer, 2001, `http://www.minrank.org/hfesec.\{ps|dvi|pdf\}`, pp. 266–281.

[Cou05] Nicolas T. Courtois, *Hidden field equations public key cryptosystem home page (HFE)*, 2005, `http://www.minrank.org/hfe/`, last modified 07/31/05.

[DG05] Jintai Ding and Jason E. Gower, *Inoculating multivariate schemes against differential attacks*, Cryptology ePrint Archive, Report 2005/255, 2005, `http://eprint.iacr.org/`.

[DGS+05] Jintai Ding, Jason E. Gower, Dieter Schmidt, Christopher Wolf, and Zhijun Yin, *Complexity estimates for the F4 attack on the Perturbed Matsumoto-Imai cryptosystem*, Manuscript, 15 pages, 2005, Available at `http://math.uc.edu/~aac/pub/pmi-groebner.pdf`.

[Die04] Claus Diem, *The XL-algorithm and a conjecture from commutative algebra*, ASIACRPYT 2004 (P.J. Lee, ed.), vol. 3329, Springer, 2004, pp. 323–337.

[Din04] Jintai Ding, *A new variant of the Matsumoto-Imai cryptosystem through perturbation.*, Public Key Cryptography—PKC 2004 (Feng Bao, Robert H. Deng, and Jiangying Zhou, eds.), vol. 2947, Springer, 2004, pp. 305–318.

[DS05] Jintai Ding and Dieter Schmidt, *Cryptanalysis of HFEv and internal perturbation of HFE*, Public Key Cryptography—PKC 2005 (Serge Vaudenay, ed.), Lecture Notes in Computer Science, vol. 3386, Springer, 2005, pp. 288–301.

[Fau99] Jean-Charles Faugère, *A new efficient algorithm for computing Gröbner bases (F4)*, Journal of Pure and Applied Algebra (1999), 61–88.

[Fau02] Jean-Charles Faugère, *A new efficient algorithm for computing Gröbner bases without reduction to zero F5.*, Proceedings of ISSAC (T. Mora, ed.), ACM Press, July 2002, pp. 75–83.

[Fau03] Jean-Charles Faugère, *Algebraic cryptanalysis of HFE using Gröbner bases*, Tech. report, INRIA, 2003.

[FGS05] Pierre-Alain Fouque, Louis Granboulan, and Jacques Stern, *Differential cryptanalysis for multivariate schemes*, Advances in Cryptology—EUROCRYPT 2005 (Ronald Cramer, ed.), Lecture Notes in Computer Science, vol. 3494, Springer, 2005, pp. 341–353.

[FJ03] Jean-Charles Faugère and Antoine Joux, *Algebraic cryptanalysis of hidden field equations (HFE) using Gröbner bases*, Advances in Cryptology—CRYPTO 2003 (Dan Boneh, ed.), Lecture Notes in Computer Science, vol. 2729, Springer, 2003, pp. 44–60.

[GC00] Louis Goubin and Nicolas T. Courtois, *Cryptanalysis of the TTM cryptosystem*, Advances in Cryptology—ASIACRYPT 2000 (Tatsuaki Okamoto, ed.), Lecture Notes in Computer Science, vol. 1976, Springer, 2000, pp. 44–57.

[Gro96] Lov Grover, *A fast quantum mechanical algorithm for database search*, Proceedings of the 28th Annual ACM Symposium on the Theory of Computing, 1996, p. 212.

[JKJMR05] Antoine Joux, Sèbastien Kunz-Jacques, Frédéric Muller, and Pierre-Michel Ricordel., *Cryptanalysis of the tractable rational map cryptosystem*, Public Key Cryptography—PKC 2005 (Serge Vaudenay, ed.), Lecture Notes in Computer Science, vol. 3386, Springer, 2005, pp. 258–274.

[Kal03] Burt Kaliski, *TWIRL and RSA key size*, Available at `http://www.rsasecurity.com/rsalabs/node.asp?id=2004`, 2003, May 6, 2003 revision.

[KS99] Aviad Kipnis and Adi Shamir, *Cryptanalysis of the HFE public key cryptosystem*, Advances in Cryptology—CRYPTO 1999 (Michael Wiener, ed.), Lecture Notes in Computer Science, vol. 1666, Springer, 1999, pp. 19–30.

[KS04a] Masao Kasahara and Ryuichi Sakai, *A construction of public key cryptosystem for realizing ciphertext of size 100 bit and digital signature scheme*, Institute of Electronics Information and Communication Engineers Transactions Fundamentals **E87-A(1)** (2004), 102–109, Electronic version: `http://search.ieice.org/2004/files/e000a01.htm#e87-a,1,102`.

[KS04b] Masao Kasahara and Ryuichi Sakai, *A construction of public-key cryptosystem based on singular simultaneous equations*, Symposium on Cryptography and Information Security—SCIS 2004, The Institute of Electronics, Information and Communication Engineers, January 27–30 2004, 6 pages.

[KSY94] Deepak Kapur, Tushar Saxena, and Lu Yang, *Algebraic and geometric reasoning using Dixon resultants*, ACM ISSAC 94 (1994), 99–107.

[MI88] Tsutomo Matsumoto and Hideki Imai, *Public quadratic polynomial-tuples for efficient signature verification and message-encryption*, Advances in Cryptology—EUROCRYPT 1988 (Christoph G. Günther, ed.), Lecture Notes in Computer Science, vol. 330, Springer, 1988, pp. 419–453.

[Pat95] Jacques Patarin, *Cryptanalysis of the Matsumoto and Imai public key scheme of EUROCRYPT '88*, Advances in Cryptology—CRYPTO 1995 (Don Coppersmith, ed.), Lecture Notes in Computer Science, vol. 963, Springer, 1995, pp. 248–261.

[Pat96a] Jacques Patarin, *HFE first challenge*, 1996, `http://www.minrank.org/challenge1.txt`.

[Pat96b] Jacques Patarin, *Hidden field equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms*, 1996, Extended Version.

[Pat05] Jacques Patarin, *Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88*, Personal correspondence, 34 pages, 2005, This is an extended version of Pat95.

[PG97] Jacques Patarin and Louis Goubin, *Trapdoor one-way permutations and multivariate polynomials*, International Conference on Information Security and Cryptology 1997, Lecture Notes in Computer Science, vol. 1334, Springer, 1997, Extended Version: `http://citeseer.nj.nec.com/patarin97trapdoor.html`, pp. 356–368.

[PGC98] Jacques Patarin, Louis Goubin, and Nicolas T. Courtois, *Improved algorithms for isomoprhisms of polynomials*, Advances in Cryptology—EUROCRYPT 1998 (Kaisa Nyberg, ed.), Lecture Notes in Computer Science, vol. 1403, Springer, 1998, Extended Version: `http://www.minrank.org/ip6long.ps`, pp. 184–200.

[SFB96] Jeffrey Outlaw Shallit, Gudmund Skovbjerg Frandsen, and Jonathan F. Buss, *The computational complexity of some problems of linear algebra*, Tech. report, Basic Research in Computer Science, 1996, Available at `http://www.brics.dk/RS/96/33/`.

[Sho97] Peter Shor, *Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM Journal on Scientific Computing **26** (1997), 1484.

[TF05] Xijin Tang and Yong Feng, *A new efficient algorithm for solving systems of multivariate polynomial equations*, Cryptology ePrint Archive, Report 2005/312, 2005, `http://eprint.iacr.org/`.

[vzGG03] Joachim von zur Gathen and Jürgen Gerhard, *Modern computer algebra*, Cambridge, 2003.

[WBP04] Christopher Wolf, An Braeken, and Bart Preneel, *Efficient cryptanalysis of RSE(2)PKC and RSSE(2)PKC*, Cryptology ePrint Archive, Report 2004/237, 2004, `http://eprint.iacr.org/`.

[WP04] Christopher Wolf and Bart Preneel, *Asymmetric cryptography: Hidden field equations*, Cryptology ePrint Archive, Report 2004/072, 2004, `http://eprint.iacr.org/`.

[WP05] Christopher Wolf and Bart Preneel, *Taxonomy of public key schemes based on the problem of multivariate quadratic equations*, Cryptology ePrint Archive, Report 2005/077, 2005, `http://eprint.iacr.org/`.