Philippe Gaborit (Ed.)

LNCS 7932

Post-Quantum Cryptography

5th International Workshop, PQCrypto 2013
Limoges, France, June 2013
Proceedings

Springer

Lecture Notes in Computer Science 7932

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Philippe Gaborit (Ed.)

Post-Quantum Cryptography

5th International Workshop, PQCrypto 2013
Limoges, France, June 4-7, 2013
Proceedings

Volume Editor

Philippe Gaborit
University of Limoges
XLIM Laboratory, Department DMI
123, Avenue Albert Thomas, 87000 Limoges, France
E-mail: gaborit@unilim.fr

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-38615-2 e-ISBN 978-3-642-38616-9
DOI 10.1007/978-3-642-38616-9
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013938951

CR Subject Classification (1998): E.3, K.6.5, D.4.6, F.2, G.2.1, E.4, C.2.0

LNCS Sublibrary: SL 4 – Security and Cryptology

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

PQCrypto 2013, the 5th International Workshop on Post-Quantum Cryptography was held in Limoges, France, during June 4–7, 2013.

The workshop attracted 24 submissions, of which the Program Commmittee selected 17 for publication in the workshop proceedings. The accepted papers dealt with the topics of code-based cryptography, lattice-based cryptography, multivariate-cryptography, and cryptanalysis or implementations. The Program Committee included 23 subject-matter experts from 11 countries.

The workshop included two invited talks by Frédéric Magniez and Michael Naehrig and a recent results session chaired by Carlos Aguilar Melchor.

I would like to thank all the Program Committee members, who made great effort contributing their time, knowledge, and expertise. I also thank the external reviewers who assisted in the process.

I wish to thank the generous sponsors of PQCRypto 2013: the Region Limousin, the Limoges University, the Mathematics and Computer Science Departement of the XLIM laboratory and the XLIM laboratory. Special thanks are also due to Thierry Berger for his organizational effort as General Chair and to Odile Duval and Jean-Christophe Deneuville for their everyday help.

April 2013 Philippe Gaborit

Organization

General Chair

Thierry Berger	Limoges University, France

Program Chair

Philippe Gaborit	Limoges University, France

Steering Committee

Daniel J. Bernstein	University of Illinois at Chicago, USA
Johannes Buchmann	Technische Universität Darmstadt, Germany
Claude Crépeau	McGill University, Canada
Jintai Ding	University of Cincinnati, USA
Philippe Gaborit	University of Limoges, France
Tanja Lange	Technische Universiteit Eindhoven, The Netherlands
Daniele Micciancio	University of California, San Diego, USA
Werner Schindler	BSI, Germany
Nicolas Sendrier	INRIA, France
Shigeo Tsujii	Chuo University, Japan
Bo-Yin Yang	Academia Sinica, Taiwan

Program Committee

Carlos Aguilar Melchor	University of Limoges, France
Paulo Barreto	University of Sao Paulo, Brazil
Daniel J. Bernstein	University of Illinois at Chicago, USA
Xavier Boyen	QUT, Australia
Johannes Buchmann	TU Darmstadt, Germany
Stanislav Bulygin	TU Darmstadt, Germany
Claude Crépeau	McGill University, Canada
Jintai Ding	University of Cincinnati, USA
Pierre-Alain Fouque	University of Rennes I, France
Tim Guneysu	Rhur University Bochum, Germany
Sean Hallgren	University of Pennsylvania, USA
Kazukuni Kobara	AIST, Japan
Tanja Lange	TU Eindhoven, The Netherlands
Gregor Leander	Danmarks TU, Denmark

Michele Mosca	University of Waterloo, Canada
Bart Preneel	KU Leuven, Belgium
Michael Schneider	TU Darmstadt, Germany
Nicolas Sendrier	INRIA, France
Damien Stehlé	ENS Lyon, France
Jean-Pierre Tillich	INRIA, France
Bo-Yin Yang	Academia Sinica, Taiwan

External Reviewers

François Arnault
Rafael Baiao
Daniel Cabarcas
Jie Chen
Ming-Shing Chen
Adama Diene
Vivien Dubois
Nicolas Gama
Valérie Gauthier Umaña
Stefan Heyse
Jeffrey Hoffstein
Lei Hu
Andreas Hülsing
Rafael Misoczki
Kirill Morozov
Khoa Nguyen (special thanks)
Ryo Nojima
Ayoub Otmani
Christiane Peters
Albrecht Petzold
Thomas Pöppelmann
Koichi Sakumoto
John Schanck
Dimitris E. Simos
Yasuda Takanori
Chendong Tao
Enrico Thomae (special thanks)
Joop van de Pol
Ingo von Maurich
Patrick Weiden
Christopher Wolf
C.-H. Yu

Table of Contents

Using LDGM Codes and Sparse Syndromes to Achieve Digital Signatures*

Marco Baldi[1], Marco Bianchi[1], Franco Chiaraluce[1], Joachim Rosenthal[2], and Davide Schipani[3]

[1] Università Politecnica delle Marche, Ancona, Italy
{m.baldi,m.bianchi,f.chiaraluce}@univpm.it
[2] University of Zurich, Zurich, Switzerland
rosenthal@math.uzh.ch
[3] Nottingham Trent University, Nottingham, UK
davide.schipani@ntu.ac.uk

Abstract. In this paper, we address the problem of achieving efficient code-based digital signatures with small public keys. The solution we propose exploits sparse syndromes and randomly designed low-density generator matrix codes. Based on our evaluations, the proposed scheme is able to outperform existing solutions, permitting to achieve considerable security levels with very small public keys.

Keywords: Code-based digital signatures, LDGM codes, sparse syndromes.

1 Introduction

The problem of replacing current cryptographic primitives which will be subject to quantum computer attacks with alternative post-quantum solutions is fostering research on code-based systems, which are among the most promising options for this replacement.

Concerning asymmetric cryptography, the McEliece cryptosystem [21] and its recent improvements [9] already represent efficient solutions to replace quantum vulnerable systems, like RSA. The main drawback of the McEliece cryptosystem compared to RSA is the large size of its public keys. However, great steps have been done towards the reduction of the McEliece public key size. A possible solution consists in replacing the Goppa codes used in the original system with other families of codes. Among these, low-density parity-check (LDPC) codes have been considered since several years [1–3, 24], and most recent proposals based on them have been able to achieve significant reductions in the key size [6, 7, 23].

For what concerns digital signatures, the widespread DSA and RSA signature schemes will be endangered by quantum computers as well, and only a few

* This work was supported in part by the MIUR project "ESCAPADE" (Grant RBFR105NLC) under the "FIRB - Futuro in Ricerca 2010" funding program, and in part by the Swiss National Science Foundation under grant No. 132256.

P. Gaborit (Ed.): PQCrypto 2013, LNCS 7932, pp. 1–15, 2013.

replacements are available up to now, like hash-based signatures. Code-based digital signature schemes represent another post-quantum alternative to DSA and RSA signature schemes, but the development of efficient code-based solutions is still challenging.

The two main proposals of code-based signature schemes currently available are the Courtois-Finiasz-Sendrier (CFS) scheme [13] and the Kabatianskii-Krouk-Smeets (KKS) scheme [18]. An up-to-date discussion about these two systems can be found in [15] and [26], respectively.

The KKS scheme uses two codes with different sizes to create the trapdoor, one selecting the subset support of the other. An important weakness of this system was recently pointed out in [26], even though the KKS scheme can still be considered secure for some choices of its parameters.

The CFS signature scheme instead uses a hash-and-sign paradigm based on the fact that only the authorized signer can exploit the error correction capability of a secret code. The main components of the CFS scheme are a private t-error correcting code C and a public hash algorithm $\mathcal{H}$. The private code is described through its parity-check matrix H, while $H' = S \cdot H$ is made public, where S is a private random matrix. There must be a public function $\mathcal{F}$ able to transform (in a reasonable time) any hash value computed through $\mathcal{H}$ into a correctable syndrome for the code C. Then, syndrome decoding through C is performed by the signer, and the resulting error vector e is the digital signature, together with the parameters to be used in the function $\mathcal{F}$ for achieving the target. Verification is easily obtained by computing $H' \cdot e$ and comparing the result with the output of $\mathcal{F}$.

The main drawback of the CFS scheme concerns the function $\mathcal{F}$. In fact, it is very hard to find a function that quickly transforms an arbitrary hash vector into a correctable syndrome. In the original CFS scheme, two ways are proposed to solve this problem [15]: i) appending a counter to the message, or ii) performing complete decoding. Both these methods require a very special choice of the code parameters to be able to find decodable syndromes within a reasonable time. For this purpose, codes with very high rate and very small error correction capability are commonly used, and this has exposed the cryptosystem to attacks based on the generalized birthday algorithm [14], in addition to common attacks against code-based cryptosystems. This flaw is mainly due to the need to ensure that many vectors produced by the hash function $\mathcal{H}$ are correctable. In addition, in such a setting, the decoding complexity can be rather high, especially in the versions exploiting complete decoding.

In this paper, we propose a new solution based on a modification of the CFS scheme. The first variation is to consider only a subset of the possible syndromes, selecting the ones having a certain density (of not null elements). In addition, we replace traditional Goppa codes with low-density generator-matrix (LDGM) codes, which allow for a random based design and a considerable reduction in the public key size. As it will be shown in the following, this allows to relax many constraints on the code parameters, and therefore to use more practical codes which also make classical attacks against the CFS scheme inapplicable. In addition, syndrome decoding through the private code is reduced to

a straightforward procedure, with an extremely low complexity. The rationale of the proposed system is in the following observations:

- Given a private parity-check matrix in systematic form (with an identity block in the rightmost part), the signer can obtain an error vector associated to a given syndrome by simply transposing the syndrome and prepending it with an all zero vector. By obtaining the public parity-check matrix from the private one through a left and right multiplication by two dense secret matrices, the systematic form is lost and the same procedure cannot be exploited by an attacker. Moreover, the two parity-check matrices no longer describe the same code.
- The private error vector obtained by the signer can be disguised by adding to it a randomly selected codeword of the secret code.
- If both the private error vector and the random codeword are of moderately low weight, and the same holds for their transposition into the public code, they are difficult to discover by an attacker.
- If the private code is an LDGM code, it is very easy for the signer to randomly select a low weight codeword, since it is obtained as the sum of a small number of rows of its generator matrix, chosen at random. Although the private code is an LDGM code, its parity-check matrix in systematic form can be dense.

In the following sections we show how these observations are translated into practice in the proposed system. The organization of the paper is as follows. In Section 2, we describe the LDGM codes we use in the system and their characteristics. In Section 3, we define the main steps of the system, that are: key generation, signing procedure and verification procedure. In Section 4, we provide a preliminary assessment of the possible vulnerabilities affecting the system. In Section 5, we give some possible choices of the system parameters and, finally, Section 6 concludes the paper.

2 Low-Density Generator-Matrix Codes

LDGM codes have been used since a long time for transmission applications [12], and are recognized to achieve very good error correcting performance when used in concatenated schemes [16], [17].

A simple way to obtain an LDGM code with length n, dimension k and redundancy $r = n - k$ is to define its generator matrix in the form

$$G = [I_k|D], \tag{1}$$

where I_k is a $k \times k$ identity matrix, and D is a sparse $k \times r$ matrix. We suppose that the rows of G have Hamming weight $w_g \ll n$. An LDGM code can also be defined with G in a more general form than (1), that is, by randomly selecting k linearly independent vectors with length n and Hamming weight $w_g \ll n$, and using them as the rows of G. This approach requires to check the linear independence of the

rows of G, but it increases the degrees of freedom for random-based designs. Hence, we consider this more general solution for the design of the private code in the proposed system.

Due to their sparse nature, it is very likely that, by summing two or more rows of the generator matrix of an LDGM code, a vector with Hamming weight $\geq w_g$ is obtained. In this case, the LDGM code has minimum distance w_g. This is even more likely if the rows of G are chosen in such a way as to be quasi-orthogonal, that is, with a minimum number of overlapping ones. However, in the scheme we propose, we do not actually need that the secret code has minimum distance w_g. Hence, G can be designed completely at random, without any constraint on the number of overlapping ones between each pair of its rows.

The code defined through G as in (1) is systematic and admits a sparse parity check matrix H in the form

$$H = [D^T | I_r], \tag{2}$$

where ^{T} denotes transposition and I_r is an $r \times r$ identity matrix. Hence, such a code is an LDPC code as well. On the contrary, if G is designed completely at random, without imposing the form (1), it is not systematic and the LDGM code is generally not an LDPC code. This is the case for the private LDGM code which is used in the proposed system.

A particularly interesting class of LDGM codes is that of quasi-cyclic (QC) LDGM codes [4]. In fact, the QC property allows to reduce the memory needed to store the code characteristic matrices, which is an important feature in cryptographic applications where such matrices are used as private and public keys.

A general form for the generator matrix of a QC-LDGM code is as follows:

$$G_{QC} = \begin{bmatrix} C_{0,0} & C_{0,1} & C_{0,2} & \dots & C_{0,n_0-1} \\ C_{1,0} & C_{1,1} & C_{1,2} & \dots & C_{1,n_0-1} \\ C_{2,0} & C_{2,1} & C_{2,2} & \dots & C_{2,n_0-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{k_0-1,0} & C_{k_0-1,1} & C_{k_0-1,2} & \dots & C_{k_0-1,n_0-1} \end{bmatrix}, \tag{3}$$

where $C_{i,j}$ represents a sparse circulant matrix or a null matrix with size $p \times p$. Hence, in this case the code length, dimension and redundancy are $n = n_0 p$, $k = (n_0 - r_0)p = k_0 p$ and $r = r_0 p$, respectively. Since a circulant matrix is defined by one of its rows (conventionally the first), storing a binary matrix G_{QC} as in (3) requires $k_0 n_0 p = kn/p$ bits, and the corresponding parity-check matrix H_{QC} requires $r_0 n_0 p = rn/p$ bits to be stored. The proposed system uses a parity-check matrix as the public key; so, when adopting QC-LDGM codes, its public key size is rn/p bits.

An important feature of LDGM codes which will be exploited in the proposed scheme is that it is easy to obtain a random codeword c belonging to an LDGM code and having weight approximately equal to a fixed, small value w_c. Let us suppose that w_c is an integer multiple of w_g. Since the rows of G are sparse, it is highly probable that, by summing a small number of rows, the Hamming weight of the resulting vector is about the sum of their Hamming weights.

Hence, by summing $\frac{w_c}{w_g}$ rows of G, chosen at random, we get a random codeword with Hamming weight about w_c. Actually, due to some overlapping ones, the resulting weight could result smaller than w_c. In this case, some other row can be added, or some row replaced, or another combination of rows can be tested, in order to approach w_c. Moreover, as we will see in the following, using a random codeword with Hamming weight slightly smaller than w_c is not a problem in the proposed system. Based on the above considerations, the number of codewords with weight close to w_c which can be easily selected at random from an LDGM code having G with rows of weight w_g, with $w_g | w_c$, can be roughly estimated as

$$A_{w_c} \approx \binom{k}{\frac{w_c}{w_g}}. \tag{4}$$

3 System Description

In this section we describe the main steps of the proposed digital signature system.

3.1 Key Generation

The first part of the private key for the proposed system is formed by the $r \times n$ parity-check matrix H of an LDGM code $C(n, k)$, having length n and dimension k ($r = n - k$). The matrix H is in systematic form, with an identity block in the rightmost part. The private key also includes two other non-singular matrices: an $r \times r$ transformation matrix Q and an $n \times n$ scrambling matrix S (both defined below). The public key is then obtained as $H' = Q^{-1} \cdot H \cdot S^{-1}$.

The matrix S is a sparse non-singular matrix, with average row and column weight $m_S \ll n$. The matrix Q, instead, is a *weight controlling* transformation matrix as defined in [5]. For this kind of matrices, when s is a suitably chosen sparse vector, the vector $s' = Q \cdot s$ has a small Hamming weight, which is only a few times greater than that of s. As shown in [5], a matrix Q with such a feature can be obtained as the sum of an $r \times r$ low-rank dense matrix R and a sparse matrix T, chosen in such a way that $Q = R + T$ is non singular. In order to design R, we start from two $z \times r$ matrices, a and b, with $z < r$ properly chosen (see below), and define R as:

$$R = a^T \cdot b. \tag{5}$$

This way, R has rank $\leq z$. The matrix T is then chosen as a sparse matrix with row and column weight m_T, such that $Q = R + T$ is full rank.

It can be easily verified that, if the vector s is selected in such a way that $b \cdot s = 0_{z \times 1}$, where $0_{z \times 1}$ is the $z \times 1$ all-zero vector, then $R \cdot s = 0_{r \times 1}$ and $s' = Q \cdot s = T \cdot s$. Hence, the Hamming weight of s' is, at most, equal to m_T times that of s, and Q actually has the weight controlling feature we desire.

As we will see in Section 4.3, although it is relatively simple for an attacker to obtain the space defined by the matrix b, and its dual space, this does not

help to mount a key recovery attack. Hence, the matrix b, which is only a small part of Q, can even be made public.

When a QC code is used as the private code, H is formed by $r_0 \times n_0$ circulant matrices of size $p \times p$, and it is desirable to preserve this QC structure also for H', in such a way as to exploit its benefits in terms of key size. For this purpose, both Q and S must be formed by circulant blocks with the same size as those forming H. Concerning the matrix S, it is obtained in QC form (S_{QC}) by simply choosing at random a block of $n_0 \times n_0$ sparse or null circulant matrices such that the overall row and column weight is m_S.

Concerning the matrix Q, instead, a solution to obtain it in QC form is to define R as follows:

$$R_{QC} = \left(a_{r_0}^T \cdot b_{r_0}\right) \otimes \mathbf{1}_{p \times p}, \tag{6}$$

where a_{r_0} and b_{r_0} are two $z \times r_0$ binary matrices, $\mathbf{1}_{p \times p}$ is the all-one $p \times p$ matrix and $\otimes$ denotes the Kronecker product. Then, T_{QC} is chosen in the form of $n_0 \times n_0$ sparse circulant blocks with overall row and column weight m_T and Q_{QC} is obtained as $R_{QC} + T_{QC}$. This way, if H is in QC form, $H' = Q_{QC}^{-1} \cdot H \cdot S_{QC}^{-1}$ is in QC form as well. In the QC case, the condition we impose on s, that is, $b \cdot s = 0_{z \times 1}$ becomes $(b_{r_0} \otimes \mathbf{1}_{1 \times p}) \cdot s = 0_{z \times 1}$.

Such a condition, both in the generic and in the QC case, is equivalent to a set of z parity-check constraints for a code with length r and redundancy z. Hence, if we fix b such that this code has minimum distance d, then a vector s with weight $w < d$ cannot satisfy such condition, and Q loses its weight controlling feature on s. This is useful to reinforce the system against some vulnerabilities, and justifies the form used for the matrix Q.

Apart from the private and public key pair, the system needs two functions which are made public as well: a hash function $\mathcal{H}$ and a function $\mathcal{F}_\Theta$ that converts the output vector of $\mathcal{H}$ into a sparse r-bit vector s of weight $w \ll r$. The output of $\mathcal{F}_\Theta$ depends on the parameter Θ, which is associated to the message to be signed and made public by the signer. An example of implementation of $\mathcal{F}_\Theta$ is provided in the next section.

3.2 Signature Generation

In order to get a unique digital signature from some document M, the signer computes the digest $h = \mathcal{H}(M)$ and then finds Θ_M such that $s = \mathcal{F}_{\Theta_M}(h)$ verifies $b \cdot s = 0_{z \times 1}$. Since s has weight w, $s' = Q \cdot s$ has weight $\leq m_T w$. Concerning the implementation of the function $\mathcal{F}_\Theta(h)$, an example is as follows. Given a message digest $h = \mathcal{H}(M)$ of length x bits, similarly to what is done in the CFS scheme, it is appended with the y-bit value l of a counter, thus obtaining $[h|l]$. The value of $[h|l]$ is then mapped uniquely into one of the $\binom{r}{w}$ r-bit vectors of weight w, hence it must be $\binom{r}{w} \geq 2^{x+y}$. The counter is initially set to zero by the signer, and then progressively increased. This way, a different r-bit vector is obtained each time, until one orthogonal to b is found, for $l = \bar{l}$. This step requires the signer to test 2^z different values of the counter, on average. With this implementation of $\mathcal{F}_\Theta(h)$, we have $\Theta_M = \bar{l}$, and different signatures correspond to different vectors s, unless a hash collision occurs.

After having obtained s, the signer has to find a vector e of weight $\leq m_T w$ which corresponds to the private syndrome $s' = Q \cdot s$ through C. Since H is in systematic form, it can be written as $H = [X|I_r]$, where X is an $r \times k$ matrix and I_r is the $r \times r$ identity matrix. Hence, the private syndrome s' can be obtained from the error vector $e = [0_{1\times k}|s'^T]$. So, in this case, finding e simply translates into a vector transposition and some zero padding.

The signer finally selects a random codeword $c \in C$ with small Hamming weight (w_c), and computes the public signature of M as $e' = (e + c) \cdot S^T$. If the choice of the codeword c is completely random and independent of the document to be signed, the signature obtained for a given document changes each time it is signed, and the system becomes vulnerable to attacks exploiting many signatures of the same document. This can be simply avoided by choosing the codeword c as a deterministic function of the document M and, hence, of the public syndrome s. For example, s or, equivalently, $[h|\bar{l}]$ can be used as the initial state of the pseudo-random integer generator through which the signer extracts the indexes of the rows of G that are summed to obtain c. This way, the same codeword is always obtained for the same public syndrome.

To explain the role of the codeword c, let us suppose for a moment that the system does not include any random codeword, that is equivalent to fix $c = 0_{1\times n}, \forall M$. In this case, we could write $e' = W(s)$, where W is a linear bijective map from the set of public syndromes to the set of valid signatures. This can be easily verified, since it is simple to check that $W(s_1+s_2) = W(s_1)+W(s_2)$. So, an attacker who wants to forge a signature for the public syndrome s_x could simply express s_x as a linear combination of previously intercepted public syndromes, $s_x = s_{i_1} + s_{i_2} + \ldots s_{i_N}$, and forge a valid signature by linearly combining their corresponding signatures: $e'_x = e'_{i_1} + e'_{i_2} + \ldots e'_{i_N}$.

As mentioned, to prevent this risk, the signer adds a random codeword c, with weight $w_c \ll n$, to the error vector e, before multiplication by S^T. This way, the map W becomes an affine map which depends on the random codeword c, and it no longer has the set of valid signatures as its image. In fact, we can denote this new map as $W_c(s)$, such that $e'_1 = W_{c_1}(s_1)$ and $e'_2 = W_{c_2}(s_2)$, where c_1 and c_2 are two randomly selected codewords of the private code with weight w_c. If we linearly combine the signatures, we obtain $e'_f = e'_1 + e'_2 = W_{c_1}(s_1) + W_{c_2}(s_2) = W_{c_1+c_2}(s_1 + s_2)$. The vector $c_1 + c_2$ is still a valid codeword of the secret code, but it has weight $> w_c$ with a very high probability.

3.3 Signature Verification

After receiving the message M, its signature e' and the associated parameter Θ_M, the verifier first checks that the weight of e' is $\leq (m_T w + w_c)m_S$. If this condition is not satisfied, the signature is discarded. Then the verifier computes $\widehat{s} = \mathcal{F}_{\Theta_M}(\mathcal{H}(M))$ and checks that $\widehat{s}$ has weight w, otherwise the signature is discarded. If the previous checks have been positive, the verifier then computes $H' \cdot e'^T = Q^{-1} \cdot H \cdot S^{-1} \cdot S \cdot (e^T + c^T) = Q^{-1} \cdot H \cdot (e^T + c^T) = Q^{-1} \cdot H \cdot e^T = Q^{-1} \cdot s' = s$. If $s = \widehat{s}$, the signature is accepted; otherwise, it is discarded.

3.4 Number of Different Signatures

An important parameter for any digital signature scheme is the total number of different signatures. In our case, a different signature corresponds to a different r-bit vector s, having weight w. Only vectors s satisfying the z constraints imposed by b are acceptable, so the maximum number of different signatures is:

$$N_s \approx \frac{\binom{r}{w}}{2^z}. \tag{7}$$

4 Possible Vulnerabilities

For a security assessment of the proposed system, it would be desirable to find possible security reductions to some well known hard problems, and then to evaluate the complexity of practical attacks aimed at solving such problems. This activity is still at the beginning, and work is in progress in this direction. Hence, in this paper we only provide a sketch of some possible vulnerabilities we have already devised, which permit to obtain a first rough estimate of the security level of the system. Completing the security assessment will allow to improve the security level estimation, and possibly to find more effective choices of the system parameters.

From the definition of the proposed system, it follows that the published signature e' associated to a document M is always a sparse vector, with Hamming weight $\leq (m_T w + w_c) m_S$. Since e' is an error vector corresponding to the public syndrome s through the public code parity-check matrix H', having a low Hamming weight ensures that e' is difficult to find, starting from s and H'. This is achieved by using the weight controlling matrix Q and the sparse matrix S. If this was not the case, and e' was a dense vector, it would be easy to forge signatures, since a dense vector corresponding to s through H' is easy to find.

Based on these considerations, one could think that choosing both Q and S as sparse as possible would be a good solution. Let us suppose that they are two permutation matrices, P_1 and P_2. In this case, the public matrix would be $H' = P_1^T \cdot H \cdot P_2^T$, and both s' and e' would be sparse, thus avoiding easy forgeries. Actually, a first reason for avoiding to use permutation matrices is that, when masked only through permutations, the security of H decreases. In fact, using a doubly permuted version of H may still allow to perform decoding through the public code. However, neglecting for a moment this fact, we find that, in this case, e' would have weight $\leq w + w_c$. If e and c have disjoint supports, which is very likely true, since we deal with sparse vectors, w non-zero bits in e' would correspond to a reordered version of the non-zero bits in s. So, apart from the effect of the random codeword, we would simply have a disposition of the non-zero bits in s within e', according to a fixed pattern. This pattern could be discovered by an attacker who observes a sufficiently large number of signatures, so that the effect of the random codeword could be eliminated. In fact, by computing the intersection of the supports of many vectors s and their corresponding vectors e', the support of e' could be decomposed and the reordering of each bit disclosed.

Based on these considerations, we can conclude that the density of e' must be carefully chosen between two opposite needs:

- being sufficiently low to avoid forgeries;
- being sufficiently high to avoid support decompositions.

4.1 Forgery Attacks

In order to forge signatures, an attacker could search for an $n \times r$ right-inverse matrix H'_r of H'. Then, he could compute $f = (H'_r \cdot s)^T$, which is a forged signature. It is easy to find a right-inverse matrix able to forge dense signatures. In fact, provided that $H' \cdot H'^T$ is invertible, $H'_r = H'^T \cdot \left(H' \cdot H'^T\right)^{-1}$ is a right-inverse matrix of H'. The matrix H' is dense and the same occurs, in general, for $\left(H' \cdot H'^T\right)^{-1}$; so, H'_r is dense as well. It follows that, when multiplied by s, H'_r produces a dense vector, thus allowing to forge dense signatures. By using sparse signatures, with weight $\leq (m_T w + w_c) m_S$, the proposed system is robust against this kind of forged signatures.

However, the right-inverse matrix is not unique. So, the attacker could search for an alternative, possibly sparse, right-inverse matrix. In fact, given an $n \times n$ matrix Z such that $H' \cdot Z \cdot H'^T$ is invertible, $H''_r = Z \cdot H'^T \cdot \left(H' \cdot Z \cdot H'^T\right)^{-1}$ is another valid right-inverse matrix of H'. We notice that $H''_r \neq Z \cdot H'_r$. When H' contains an invertible $r \times r$ square block, there is also another simple way to find a right-inverse. It is obtained by inverting such block, putting its inverse at the same position (in a transposed matrix) in which it is found within H', and padding the remaining rows with zeros.

In any case, there is no simple way to find a right-inverse matrix that is also sparse, which is the aim of an attacker. Actually, for the matrix sizes considered here, the number of possible choices of Z is always huge. Moreover, there is no guarantee that any of them produces a sparse right-inverse. Searching for an $r \times r$ invertible block within H' and inverting it would also produce unsatisfactory results, since the overall density of H'^{-1} is reduced, but the inverse of the square block is still dense. So, the attacker would be able to forge signatures with a number of symbols 1 on the order of $r/2$, that is still too large for the system considered here. In fact, in the system examples we provide in Section 5, we always consider public signatures with weight on the order of $r/3$ or less.

A further chance is to exploit Stern's algorithm [29] (or other approaches for searching low weight codewords) to find a sparse representation of the column space of H'_r. If this succeeds, it would result in a sparse matrix $H_S = H'_r \cdot B$, for some $r \times r$ transformation matrix B. However, in this case, H_S would not be a right-inverse of H'.

For these reasons, approaches based on right-inverses seem to be infeasible for an attacker. An alternative attack strategy could be based on decoding. In fact, an attacker could try syndrome decoding of s through H', hoping to find a sparse vector f. He would have the advantage of searching for one out of many possible vectors, since he is not looking for a correctable error vector. Several algorithms could be exploited for solving such problem [8, 10, 11, 20, 27]. These algorithms

are commonly used to search for low weight vectors with a null syndrome, but, with a small modification, they can also be used to find vectors corresponding to a given (non-zero) syndrome. In addition, their complexity decreases when an attacker has access to a high number of decoding instances, and wishes to solve only one of them [28], which is the case for the proposed system. We will discuss the complexity issue in Section 5.

4.2 Support Decomposition Attacks

Concerning support decomposition, let us suppose that e and c have disjoint supports. In this case, the overall effect of the proposed scheme on the public syndrome s can be seen as the expansion of an $r \times 1$ vector s of weight w into a subset of the support of the $1 \times n$ vector e', having weight $\leq m_T m_S w$, in which each symbol 1 in s corresponds, at most, to $m = m_T m_S$ symbols 1 in e'.

An attacker could try to find the w sets of m (or less) symbols 1 within the support of e' in order to compute valid signatures. In this case, he will work as if the random codeword was absent, that is, $c = 0_{1\times n}$. Thus, even after succeeding, he would be able to forge signatures that are sparser than the authentic ones. In any case, this seems a rather dangerous situation, so we should aim at designing the system in such a way as to avoid its occurrence.

To reach his target, the attacker must collect a sufficiently large number L of pairs (s, e'). Then, he can intersect the supports (that is, compute the bit-wise AND) of all the s vectors. This way, he obtains a vector s_L that may have a small weight $w_L \geq 1$. If this succeeds, the attacker analyzes the vectors e', and selects the mw_L set bit positions that appear more frequently. If these bit positions are actually those corresponding to the w_L bits set in s_L, then the attacker has discovered the relationship between them. An estimate of the probability of success of this attack can be obtained through combinatorial arguments.

An even more efficient strategy could be to exploit information set decoding to remove the effect of the random codeword. In fact, an attacker knows that $e' = (e + c) \cdot S^T = e'' + c''$, with c'' such that $H' c'' = 0$. Hence, e'' can be considered as an error vector with weight $\leq m_T m_S w$ affecting the codeword c'' of the public code. So the attacker could consider a random subset of k coordinates of the public signature e' and assume that no errors occurred on these coordinates. In this case, he can easily recover c'' and, hence, remove the effect of the random codeword c. The probability that there are no errors in the chosen k coordinates is $\binom{n-m_T m_S w}{k} / \binom{n}{k}$, and its inverse provides a rough estimate of the work factor of this attack.

4.3 Key Recovery Attacks

An attacker could aim to mount a key recovery attack, that is, to obtain the private code. A potential vulnerability in this sense comes from the use of LDGM codes. As we have seen in Section 2, LDGM codes offer the advantage of having a predictable (and sufficiently high) number of codewords with a moderately low weight w_c, and of making their random selection very easy for the signer.

On the other hand, when the private code is an LDGM code, the public code admits a generator matrix in the form $G'_I = G \cdot S^T$, which is still rather sparse. So, the public code contains low weight codewords, coinciding with the rows of G'_I, which have weight approximately equal to $w_g \cdot m_S$. Since G'_I has k rows, and summing any two of them gives higher weight codewords with a very high probability, we can consider that the multiplicity of these words in the public code is k. They could be searched by using again Stern's algorithm [29] and its improved versions [8, 10, 11, 20, 27], in such a way as to recover G'_I. After that, G'_I could be separated into G and S^T by exploiting their sparsity. In Section 5 we discuss how to estimate the work factor of this attack.

Another possible vulnerability comes from the fact that the matrix b is public. Even if b was not public, an attacker could obtain the vector space generated by b, as well as its dual space, by observing $O(r)$ public syndromes s, since $b \cdot s = 0_{z \times 1}$. Hence, we must suppose that an attacker knows an $r \times r$ matrix V such that $R \cdot V = 0 \Rightarrow Q \cdot V = T \cdot V$. The attacker also knows that $H' = Q^{-1} \cdot H \cdot S^{-1}$ and that the public code admits any non-singular generator matrix in the form $G'_X = X \cdot G \cdot S^T$, which becomes $G'_Q = Q \cdot G \cdot S^T$ for $X = Q$. Obviously, G'_I is the sparsest among them, and it can be attacked by searching for low weight codewords in the public code, as we have already observed. Instead, knowing V is useless to reduce the complexity of attacking either H' or one of the possible G'_X, hence it cannot be exploited by an attacker to perform a key recovery attack.

4.4 Other Attacks

As for any other hash-and-sign scheme, classical collision birthday attacks represent a threat for the proposed system. Since the system admits up to N_s different signatures, it is sufficient to collect $\approx \sqrt{N_s}$ different signatures to have a high probability of finding a collision [19]. Hence, the security level reached by the system cannot exceed $\sqrt{N_s}$.

However, N_s can be made sufficiently high by increasing the value of w. The definition of the proposed system allows to choose its parameters in such a way as to guarantee this fact, as we will see in Section 5. This is possible since the choice of w is not constrained by the row weight of the private generator matrix. In fact, in the proposed scheme we do not actually need a private code of minimum distance greater than $2w$, because we rely on a decoding procedure which uniquely associates to a syndrome of a given weight an error vector with the same weight, though such an error vector is not necessarily unique.

Finally, it is advisable to consider the most dangerous attacks against the CFS scheme. It was successfully attacked by exploiting syndrome decoding based on the generalized birthday algorithm [14], even if the proposed attacking algorithm was not the optimal generalization of the birthday algorithm [22]. If we do not take into account some further improvement due to the QC structure of the public key, these algorithms provide a huge work factor for the proposed system parameters, since they try to solve the decoding problem for a random code. Just to give an idea, we obtain a work factor of more than 2^{200} binary operations even for the smallest key sizes we consider. However, there are some strategies that can

be implemented to improve the efficiency of the attack on structured matrices, like those of dyadic codes [25]. This improvement could be extended to QC codes as well, but the attack work factor, for the cryptosystems analyzed in [25], is lowered by (at most) 2^{10} binary operations, starting from a maximum value of 2^{344}. Hence, it is very unlikely that this strategy can endanger the signature scheme we propose.

5 System Examples

By using the preliminary security assessment reported in Section 4, we can find some possible choices of the system parameters aimed at reaching fixed security levels. For this purpose, we have considered all the vulnerabilities described in Section 4, and we have estimated the work factor needed to mount a successful attack exploiting each of them.

We have used the implementation proposed in [27] for estimating the work factor of low weight codeword searches. Actually, [27] does not contain the most up-to-date and efficient implementation of information set decoding. In fact, some improvements have appeared in the literature concerning algorithms for decoding binary random codes (as [20], [8]). These papers, however, aim at finding algorithms which are asymptotically faster, by minimizing their asymptotic complexity exponents. Instead, for computing the work factor of attacks based on decoding, we need actual operation counts, which are not reflected in these recent works. Also "ball collision decoding" [10] achieves significant improvements asymptotically, but they become negligible for finite code lengths and not too high security levels. For these reasons, we prefer to resort to [27], which provides a detailed analysis of the algorithm, together with a precise operation count for given code parameters. On the other hand, attacks against the proposed system which exploit decoding, i.e., trying to recover the rows of G or to forge valid signatures through decoding algorithms, are far away from providing the smallest work factors, and, hence, to determine the security level. For the choices of the system parameters we suggest, the smallest work factor is always achieved by attacks aiming at decomposing the signature support, which hence coincides with the security level. For the instances proposed in this section, the work factor of attacks based on decoding is on the order of 2^{2SL}, where SL is the claimed security level. Hence, even considering some reduction in the work factor of decoding attacks would not change the security level of the considered instances of the system. This situation does not change even if we consider the improvement coming from the "decoding one out of many" approach [28]. In fact, as shown in [28], even if an attacker has access to an unlimited number of decoding instances, the attack complexity is raised by a power slightly larger than 2/3.

Concerning support decomposition attacks, a rough estimation of their complexity has instead been obtained through simple combinatorial arguments, which are not reported here for the sake of brevity.

A more detailed analysis of the attacks work factor is out of scope of this paper, and will be addressed in future works, together with a more complete

Table 1. System parameters for some security levels (SL), with $d = 2$ and $w_L = 2$

SL (bits)	n	k	p	w	w_g	w_c	z	m_T	m_S	A_{w_c}	N_s	S_k (KiB)
80	9800	4900	50	18	20	160	2	1	9	$2^{82.76}$	$2^{166.10}$	117
120	24960	10000	80	23	25	325	2	1	14	$2^{140.19}$	$2^{242.51}$	570
160	46000	16000	100	29	31	465	2	1	20	$2^{169.23}$	$2^{326.49}$	1685

security assessment. This will also permit to refine the choice of the system parameters, in such a way as to find the best trade-off between the security level and the key size.

Table 1 provides three choices of the system parameters which are aimed at achieving 80-bit, 120-bit and 160-bit security, respectively. All these instances of the system use QC-LDGM codes with different values of p, also reported in the table, and consider the case in which the matrix Q is such that $d = w_L = 2$. Actually, achieving minimum distance equal to 2 (or more) is very easy: it is sufficient to choose $z > 1$ and to guarantee that the matrix b does not contain all-zero columns. For each instance of the system, the value of the key size S_k is also shown, expressed in kibibytes (1 KiB $= 1024 \cdot 8$ bits).

In the original version of the CFS system, to achieve an attack time and memory complexity greater than 2^{80}, we need to use a Goppa code with length $n = 2^{21}$ and redundancy $r = 21 \cdot 10 = 210$ [14]. This gives a key size on the order of $4.4 \cdot 10^8$ bits = 52.5 MiB. By using the parallel CFS proposed in [15], the same work factor can be reached by using keys with size ranging between $1.05 \cdot 10^7$ and $1.7 \cdot 10^8$ bits, that is, between 1.25 MiB and 20 MiB. As we notice from Table 1, the proposed system requires a public key of only 117 KiB to achieve 80-bit security. Hence, it is able to achieve a dramatic reduction in the public key size compared to the CFS scheme, even when using a parallel implementation of the latter.

Another advantage of the proposed solution compared to the CFS scheme is that it exploits a straightforward decoding procedure for the secret code. On the other hand, 2^z attempts are needed, on average, to find an s vector such that $b \cdot s = 0_{z \times 1}$. However, such a check is very simple to perform, especially for very small values of z, like those considered here.

6 Conclusion

In this paper, we have addressed the problem of achieving efficient code-based digital signatures with small public keys.

We have proposed a solution that, starting from the CFS schemes, exploits LDGM codes and sparse syndromes to achieve good security levels with compact keys. The proposed system also has the advantage of using a straightforward decoding procedure, which reduces to a transposition and a concatenation with an all-zero vector. This is considerably faster than classical decoding algorithms used for common families of codes.

The proposed scheme allows to use a wide range of choices of the code parameters. In particular, the low code rates we adopt avoid some problems of the classical CFS scheme, due to the use of codes with high rate and small correction capability.

On the other hand, using sparse vectors may expose the system to new attacks. We have provided a sketch of possible vulnerabilities affecting this system, together with a preliminary evaluation of its security level. Work is in progress to achieve more precise work factor estimates for the most dangerous attacks.

References

1. Baldi, M., Chiaraluce, F.: Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes. In: Proc. IEEE International Symposium on Information Theory (ISIT 2007), Nice, France, pp. 2591–2595 (June 2007)
2. Baldi, M., Chiaraluce, F., Garello, R., Mininni, F.: Quasi-cyclic low-density parity-check codes in the McEliece cryptosystem. In: Proc. IEEE International Conference on Communications (ICC 2007), Glasgow, Scotland, pp. 951–956 (June 2007)
3. Baldi, M., Bodrato, M., Chiaraluce, F.: A new analysis of the McEliece cryptosystem based on QC-LDPC codes. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 246–262. Springer, Heidelberg (2008)
4. Baldi, M., Bambozzi, F., Chiaraluce, F.: On a Family of Circulant Matrices for Quasi-Cyclic Low-Density Generator Matrix Codes. IEEE Trans. on Information Theory 57(9), 6052–6067 (2011)
5. Baldi, M., Bianchi, M., Chiaraluce, F., Rosenthal, J., Schipani, D.: Enhanced public key security for the McEliece cryptosystem (2011), `http://arxiv.org/abs/1108.2462`
6. M. Baldi, M. Bianchi, and F. Chiaraluce. "Security and complexity of the McEliece cryptosystem based on QC-LDPC codes. IET Information Security (in press), `http://arxiv.org/abs/1109.5827`
7. Baldi, M., Bianchi, M., Chiaraluce, F.: Optimization of the parity-check matrix density in QC-LDPC code-based McEliece cryptosystems. To be presented at the IEEE International Conference on Communications (ICC 2013) - Workshop on Information Security over Noisy and Lossy Communication Systems, Budapest, Hungary (June 2013)
8. Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 520–536. Springer, Heidelberg (2012)
9. Bernstein, D.J., Lange, T., Peters, C.: Attacking and defending the mcEliece cryptosystem. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 31–46. Springer, Heidelberg (2008)
10. Bernstein, D.J., Lange, T., Peters, C.: Smaller decoding exponents: ball-collision decoding. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 743–760. Springer, Heidelberg (2011)
11. Chabaud, F., Stern, J.: The cryptographic security of the syndrome decoding problem for rank distance codes. In: Kim, K.-c., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 368–381. Springer, Heidelberg (1996)

12. Cheng, J.F., McEliece, R.J.: Some high-rate near capacity codecs for the Gaussian channel. In: Proc. 34th Allerton Conference on Communications, Control and Computing, Allerton, IL (October 1996)
13. Courtois, N.T., Finiasz, M., Sendrier, N.: How to achieve a McEliece-based digital signature scheme. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 157–174. Springer, Heidelberg (2001)
14. Finiasz, M., Sendrier, N.: Security bounds for the design of code-based cryptosystems. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 88–105. Springer, Heidelberg (2009)
15. Finiasz, M.: Parallel-CFS strengthening the CFS McEliece-based signature scheme. In: Proc. PQCrypto, Darmstadt, Germany, pp. 61–72, May 25-28 (2010)
16. Garcia-Frias, J., Zhong, W.: Approaching Shannon performance by iterative decoding of linear codes with low-density generator matrix. IEEE Commun. Lett. 7(6), 266–268 (2003)
17. González-López, M., Vázquez-Araújo, F.J., Castedo, L., Garcia-Frias, J.: Serially-concatenated low-density generator matrix (SCLDGM) codes for transmission over AWGN and Rayleigh fading channels. IEEE Trans. Wireless Commun. 6(8), 2753–2758 (2007)
18. Kabatianskii, G., Krouk, E., Smeets, B.: A digital signature scheme based on random error correcting codes. In: Darnell, M.J. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 161–167. Springer, Heidelberg (1997)
19. Lim, C.H., Lee, P.J.: On the length of hash-values for digital signature schemes. In: Proc. CISC 1995, Seoul, Korea, November 1995, pp. 29–31 (1995)
20. May, A., Meurer, A., Thomae, E.: Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 107–124. Springer, Heidelberg (2011)
21. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. DSN Progress Report, pp. 114–116 (1978)
22. Minder, L., Sinclair, A.: The extended k-tree algorithm. Journal of Cryptology 25(2), 349–382 (2012)
23. Misoczki, R., Tillich, J.-P., Sendrier, N., Barreto, P.S.L.M.: MDPC-McEliece: New McEliece variants from moderate density parity-check codes (2012), `http://eprint.iacr.org/2012/409`
24. Monico, C., Rosenthal, J., Shokrollahi, A.: Using low density parity check codes in the McEliece cryptosystem. In: Proc. IEEE International Symposium on Information Theory (ISIT 2000), Sorrento, Italy, p. 215 (June 2000)
25. Niebuhr, R., Cayrel, P.-L., Buchmann, J.: Improving the efficiency of Generalized Birthday Attacks against certain structured cryptosystems. In: Proc. WCC 2011, Paris, France, April 11-15 (2011)
26. Otmani, A., Tillich, J.-P.: An efficient attack on all concrete KKS proposals. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 98–116. Springer, Heidelberg (2011)
27. Peters, C.: Information-set decoding for linear codes over F_q. In: Sendrier, N. (ed.) PQCrypto 2010. LNCS, vol. 6061, pp. 81–94. Springer, Heidelberg (2010)
28. Sendrier, N.: Decoding one out of many. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 51–67. Springer, Heidelberg (2011)
29. Stern, J.: A method for finding codewords of small weight. In: Wolfmann, J., Cohen, G. (eds.) Coding Theory and Applications 1988. LNCS, vol. 388, pp. 106–113. Springer, Heidelberg (1989)

Quantum Algorithms for the Subset-Sum Problem

Daniel J. Bernstein[1,2], Stacey Jeffery[3], Tanja Lange[2], and Alexander Meurer[4]

[1] Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607–7045, USA
djb@cr.yp.to
[2] Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
tanja@hyperelliptic.org
[3] Institute for Quantum Computing, University of Waterloo, Canada
sjeffery@uwaterloo.ca
[4] Ruhr-University Bochum, Horst Görtz Institute for IT-Security
alexander.meurer@rub.de

Abstract. This paper introduces a subset-sum algorithm with heuristic asymptotic cost exponent below 0.25. The new algorithm combines the 2010 Howgrave-Graham–Joux subset-sum algorithm with a new streamlined data structure for quantum walks on Johnson graphs.

Keywords: subset sum, quantum search, quantum walks, radix trees, decoding, SVP, CVP.

1 Introduction

The subset-sum problem is the problem of deciding, given integers $x_1, x_2, \ldots, x_n$ and s, whether there exists a subset I of $\{1, 2, \ldots, n\}$ such that $\sum_{i \in I} x_i = s$; i.e., whether there exists a subsequence of $x_1, x_2, \ldots, x_n$ with sum s. Being able to solve this decision problem implies being able to find such a subset if one exists: for $n > 1$ one recursively tries $x_1, x_2, \ldots, x_{n-1}$ with sum s or, if that fails, sum $s - x_n$.

The reader should imagine, as a typical "hard" case, that $x_1, x_2, \ldots, x_n$ are independent uniform random integers in $\{0, 1, \ldots, 2^n\}$, and that s is chosen as a uniform random integer between $(n/2 - \sqrt{n})2^{n-1}$ and $(n/2 + \sqrt{n})2^{n-1}$. The number of subsets $I \subseteq \{1, 2, \ldots, n\}$ with $\sum_{i \in I} x_i = s$ then has a noticeable chance of being larger than 0 but is quite unlikely to be much larger, say larger than n.

This work was supported by the National Science Foundation under grant 1018836, by the Netherlands Organisation for Scientific Research (NWO) under grant 639.073.005, by NSERC Strategic Project Grant FREQUENCY, by the US ARO, and by the European Commission under Contract ICT-2007-216676 ECRYPT II. Permanent ID of this document: 797dcbe9c3389410ae4d5ae6dba33ab7. Date: 2013.04.06.

P. Gaborit (Ed.): PQCrypto 2013, LNCS 7932, pp. 16–33, 2013.

The subset-sum problem is, historically, one of the first problems to be proven NP-complete. A polynomial-time non-quantum algorithm for the subset-sum problem would violate the standard P $\neq$ NP conjecture; a polynomial-time quantum algorithm for the subset-sum problem would violate the standard NP $\not\subseteq$ BQP conjecture. There is, however, a very large gap between polynomial time and the time needed for a naive search through all 2^n subsets of $\{1, 2, \ldots, n\}$. The standard NP $\not\subseteq$ BQP conjecture does not rule out faster exponential-time algorithms, or even subexponential-time algorithms, or even algorithms that take polynomial time for *most* inputs. This paper studies faster exponential-time algorithms.

Variations. Often one is interested only in sums of fixed weight, or of limited weight. We are now given integers $x_1, x_2, \ldots, x_n$, s, and w; the problem is to decide whether there is a subset I of $\{1, 2, \ldots, n\}$ such that $\sum_{i \in I} x_i = s$ and $\#I = w$. In the special case $s = 0$ with $w \neq 0$, such a subset I immediately produces a short nonzero vector in the lattice L of vectors $v \in \mathbf{Z}^n$ satisfying $\sum_i x_i v_i = 0$: specifically, the characteristic function of I is a vector of length $\sqrt{w}$ in L. In many applications this is the shortest nonzero vector in L; in some applications this vector can be found by standard SVP algorithms.

For $s \neq 0$ one can instead compute a vector $r \in \mathbf{R}^n$ satisfying $\sum_i x_i r_i = s$, and then observe that subtracting the characteristic function of I from r produces an element of L. In many applications this is the vector in L closest to r; in some applications this vector can be found by standard CVP algorithms.

A variant of the same problem is the central algorithmic problem in coding theory. The input now consists of vectors $x_1, x_2, \ldots, x_n$, a vector s, and an integer w; these vectors all have the same length and have entries in the field $\mathbf{F}_2$ of integers modulo 2. The problem, as above, is to decide whether there is a subset I of $\{1, 2, \ldots, n\}$ such that $\sum_{i \in I} x_i = s$ and $\#I = w$.

We do not mean to suggest that these problems are identical. However, the algorithmic techniques used to attack subset-sum problems are among the central algorithmic techniques used to attack lattice problems and decoding problems. For example, the best attack known against code-based cryptography, at least asymptotically, is a very recent decoding algorithm by Becker, Joux, May, and Meurer [4], improving upon a decoding algorithm by May, Meurer, and Thomae [24]; the algorithm of [4] is an adaptation of a subset-sum algorithm by Becker, Coron, and Joux [3], improving analogously upon a subset-sum algorithm by Howgrave-Graham and Joux [17].

There is also a line of work on building cryptographic systems whose security is more directly tied to the subset-sum problem. For example, Lyubashevsky, Palacio, and Segev in [22] propose a public-key encryption system and prove that being able to break it implies being able to solve modular subset-sum problems of the following type: find a random subset $I \subseteq \{1, 2, \ldots, n\}$ given random $x_1, x_2, \ldots, x_n$ modulo M and given $\sum_{i \in I} x_i$ modulo M, where M is roughly $(10n \log n)^n$. They claim in [22, Section 1] that "there are currently no known quantum algorithms that perform better than classical ones on the subset sum problem".

Table 1.1. Heuristic asymptotic performance of various subset-sum algorithms. An algorithm using $2^{(e+o(1))n}$ operations is listed as "exponent" e.

Exponent	Quantum	Split	Algorithm
1	No	1	Brute force
0.5	Yes	1	Quantum search; §2
0.5	No	1/2	Left-right split; §2
0.5	No	1/4	Left-right split with a modulus; §4
0.375	Yes	1/4	Quantum search with a modulus; §4
0.337...	No	1/16	Moduli + representations; §5
0.333...	Yes	1/3	Quantum left-right split; §2
0.333...	Yes	1/2	Quantum walk; §3
0.3	Yes	1/4	Quantum walk with a modulus; §4
0.291...	No	1/16	Moduli + representations + overlap; [3]
0.241...	Yes	1/16	**New**; quantum walk + moduli + representations; §5

Contents of This Paper. We introduce the first subset-sum algorithm that beats $2^{n/4}$. Specifically, we introduce a quantum algorithm that, under reasonable assumptions, uses at most $2^{(0.241\ldots+o(1))n}$ qubit operations to solve a subset-sum problem. This algorithm combines quantum walks with the central "representations" idea of [17]. Table 1.1 compares this exponent $0.241\ldots$ to the exponents of other algorithms.

One can reasonably speculate that analogous quantum speedups can also be applied to the algorithms of [24] and [4]. However, establishing this will require considerable extra work, similar to the extra work of [24] and [4] compared to [17] and [3] respectively.

Cost Metric and Other Conventions. This paper follows the tradition of measuring algorithm cost as the number of bit operations or, more generally, qubit operations. In particular, random access to an array of size $2^{O(n)}$ is assumed to cost only $n^{O(1)}$, even if the array index is a quantum superposition.

We systematically suppress cost factors polynomial in n; our concern is with asymptotic exponents such as the $0.241\ldots$ in $2^{(0.241\ldots+o(1))n}$. We also assume that the inputs $x_1, x_2, \ldots, x_n, s$ have $n^{O(1)}$ bits. These conventions mean, for example, that reading the entire input $x_1, x_2, \ldots, x_n, s$ costs only 1.

Almost all of the algorithms discussed here split size-n sets into parts, either 2 or 3 or 4 or 16 parts, as indicated by the "Split" column in Table 1.1. Any reasonably balanced split is adequate, but to simplify the algorithm statements we assume that n is a multiple of 2 or 3 or 4 or 16 respectively.

The algorithms in this paper are designed to work well for random inputs, particularly in the "hard" case that $x_1, x_2, \ldots, x_n, s$ each have about n bits. Our analyses — like the analyses of state-of-the-art algorithms for integer factorization, discrete logarithms, and many other problems of cryptographic interest — are heuristic. We do not claim that the algorithms work for *all* inputs, and we do not claim that what we call the "hard" case is the worst case. Even for random inputs we do not claim that our analyses are proven, but we do speculate

that they are provable by an adaptation of the proof ideas stated in [17, eprint version].

Acknowledgments. This work was initiated during the Post-Quantum Cryptography and Quantum Algorithms workshop at the Lorentz Center in November 2012. We acknowledge helpful discussions with Andris Ambainis, Frédéric Magniez, Nicolas Sendrier, and Jean-Pierre Tillich.

2 Searches

Define Σ as the function that maps $I \subseteq \{1, 2, \ldots, n\}$ to $\sum_{i \in I} x_i$. Recall that we assume that $x_1, x_2, \ldots, x_n, s$ have $n^{O(1)}$ bits, and that we suppress polynomial cost factors; evaluating Σ therefore has cost 1.

The subset-sum problem is the problem of deciding whether there exists I with $\Sigma(I) = s$, i.e., whether the function $\Sigma - s$ has a root. A classical search for a root of $\Sigma - s$ uses 2^n evaluations of $\Sigma - s$, for a total cost of 2^n. Of course, the search can finish much sooner if it finds a root (one expects only 2^{n-1} evaluations on average if there is 1 root, and fewer if there are more roots); but as discussed in Section 1 we focus on "hard" cases where there are not many roots, and then the cost is 2^n (again, suppressing polynomial factors) with overwhelming probability.

This section reviews two standard ways to speed up this brute-force search. The first way is Grover's quantum search algorithm. The second way is decomposing $\Sigma(I)$ as $\Sigma(I_1) + \Sigma(I_2)$, where $I_1 = I \cap \{1, 2, \ldots, n/2\}$ and $I_2 = I \cap \{n/2 + 1, \ldots, n\}$; this split was introduced by Horowitz and Sahni in [16].

Review: The Performance of Quantum Search. Consider any computable function f with a b-bit input and a unique root. Grover's algorithm [15] finds the root (with negligible failure chance) using approximately $2^{b/2}$ quantum evaluations of f and a small amount of overhead.

More generally, consider any computable function f with a b-bit input and $r > 0$ roots. Boyer, Brassard, Høyer, and Tapp in [6] introduced a generalization of Grover's algorithm (almost exactly the same as Grover's original algorithm but stopping after a particular r-dependent number of iterations) that finds a root (again with negligible failure chance) using approximately $(2^b/r)^{1/2}$ quantum evaluations of f and a small amount of overhead. One can easily achieve the same result by using Grover's original algorithm sensibly (as mentioned in [15]): choose a fast but sufficiently random map from $b - \lceil \lg r \rceil$ bits to b bits; the composition of this map with f has a good chance of having a unique root; apply Grover's algorithm to this composition; repeat several times so that the failure chance becomes negligible.

Even more generally, consider any computable function f with a b-bit input. A more general algorithm in [6] finds a root using approximately $(2^b/r)^{1/2}$ quantum evaluations of f and a small amount of overhead, where r is the number of roots. If no root exists then the algorithm says so after approximately $2^{b/2}$ quantum evaluations of f. As above, the algorithm can fail (or take longer than expected),

but only with negligible probability; and, as above, the same result can also be obtained from Grover's algorithm.

As a trivial application, take $b = n$ and $f = \Sigma - s$: finding a root of $\Sigma - s$ costs $2^{n/2}$. Some implementation details of this quantum subset-sum algorithm appeared in [9] in 2009. We emphasize, however, that the same operation count is achieved by well-known non-quantum algorithms, and is solidly beaten by recent non-quantum algorithms.

Review: Left-Right Split. Define $L_1 = \{(\Sigma(I_1), I_1) : I_1 \subseteq \{1, 2, \ldots, n/2\}\}$ and $L_2 = \{(s - \Sigma(I_2), I_2) : I_2 \subseteq \{n/2 + 1, n/2 + 2, \ldots, n\}\}$. Note that each of these sets has size just $2^{n/2}$.

Compute L_1 by enumerating sets I_1. Store the elements of L_1 in a table, and sort the table by its first coordinate. Compute L_2 by enumerating sets I_2. For each $(s-\Sigma(I_2), I_2) \in L_2$, look for $s-\Sigma(I_2)$ by binary search in the sorted table. If there is a collision $\Sigma(I_1) = s - \Sigma(I_2)$, print out $I_1 \cup I_2$ as a root of $\Sigma - s$ and stop. If there are no collisions, print "there is no subset-sum solution" and stop.

This algorithm costs $2^{n/2}$. It uses $2^{n/2}$ memory, and one can object that random access to memory is expensive, but we emphasize that this paper follows the tradition of simply counting operations. There are several standard variants of this algorithm: for example, one can sort L_1 and L_2 together, or one can store the elements of L_1 in a hash table.

Quantum Left-Right Split. Redefine L_1 as $\{(\Sigma(I_1), I_1) : I_1 \subseteq \{1, 2, \ldots, n/3\}\}$; note that $n/2$ has changed to $n/3$. Compute and sort L_1 as above; this costs $2^{n/3}$.

Consider the function f that maps a subset $I_2 \subseteq \{n/3 + 1, n/3 + 2, \ldots, n\}$ to 0 if $s - \Sigma(I_2)$ is a first coordinate in L_1, otherwise to 1. Binary search in the sorted L_1 table costs only 1, so computing f costs only 1.

Now use quantum search to find a root of f, i.e., a subset $I_2 \subseteq \{n/3 + 1, \ldots, n\}$ such that $s - \Sigma(I_2)$ is a first coordinate in L_1. There are $2n/3$ bits of input to f, so the quantum search costs $2^{n/3}$.

Finally, find an I_1 such that $s - \Sigma(I_2) = \Sigma(I_1)$, and print $I_1 \cup I_2$. Like the previous algorithm, this algorithm finds a root of $\Sigma - s$ if one exists; any root I of $\Sigma - s$ can be expressed as $I_1 \cup I_2$.

Note that, with the original split of $\{1, \ldots, n\}$ into left and right halves, quantum search would not have reduced cost (modulo polynomial factors). Generalizing the original algorithm to allow an unbalanced split, and in particular a split into $n/3$ and $2n/3$, is pointless without quantum computers but essential for the quantum optimization. The split into $n/3$ and $2n/3$ imitates the approach used by Brassard, Høyer, and Tapp in [8] to search for hash-function collisions.

This algorithm uses $2^{n/3}$ memory, and as before one can object that random access to memory is expensive, especially when memory locations are quantum superpositions. See [5] for an extended discussion of the analogous objection to [8]. We again emphasize that this paper follows the tradition of simply counting operations; we do not claim that improved operation counts imply improvements in other cost models.

3 Walks

This section summarizes Ambainis's unique-collision-finding algorithm [2] (from the edge-walk perspective of [23]); introduces a new way to streamline Ambainis's algorithm; and applies the streamlined algorithm to the subset-sum context, obtaining cost $2^{n/3}$ in a different way from Section 2. This section's subset-sum algorithm uses collision finding as a black box, but the faster algorithms in Section 5 do not.

Review: Quantum Walks for Finding Unique Collisions. Consider any computable function f with b-bit inputs such that there is a unique pair of colliding inputs, i.e., exactly one pair (x, y) of b-bit strings such that $f(x) = f(y)$. The problem tackled in [2] is to find this pair (x, y).

The algorithm has a positive integer parameter $r < 2^b$, normally chosen on the scale of $2^{2b/3}$. At each step the algorithm is in a superposition of states of the form $(S, f(S), T, f(T))$. Here S and T are sets of b-bit strings such that $\#S = r$, $\#T = r$, and $\#(S \cap T) = r-1$; i.e., S and T are adjacent vertices in the "Johnson graph" of r-subsets of the set of b-bit strings, where edges are between sets that differ in exactly one element. The notation $f(S)$ means $\{f(x) : x \in S\}$.

The algorithm begins in a uniform superposition of states; setting up this superposition uses $O(r)$ quantum evaluations of f. The algorithm then performs a "quantum walk" that alternates two types of steps: diffusing each state to a new choice of T while keeping S fixed, and diffusing each state to a new choice of S while keeping T fixed. Only one element of T changes when S is fixed (and vice versa), so each step uses only $O(1)$ quantum evaluations of f.

Periodically (e.g., after every $2\lceil\sqrt{r}\rceil$ steps) the algorithm negates the amplitude of every state in which S contains a colliding pair, i.e., in which $\#f(S) < r$. Because $f(S)$ has already been computed, checking whether $\#f(S) < r$ does not involve any evaluations of f. One can object that this check is nevertheless extremely expensive; this objection is discussed in the "data structures" subsection below.

Ambainis's analysis shows that after roughly $2^b/\sqrt{r}$ steps the algorithm has high probability of being in a state in which S contains a colliding pair. Observing this state and then sorting the pairs $(f(x), x)$ for $x \in S$ reveals the colliding pair. Overall the algorithm uses only $O(2^{2b/3})$ evaluations of f.

As in the case of Grover's algorithm, this algorithm is easily generalized to the case that there are p pairs of colliding inputs, and to the case that p is not known in advance. The algorithm is also easily generalized to functions of S more complicated than "contains a colliding pair".

Data Structures. The most obvious way to represent a set of b-bit strings is as a sorted array. The large overlap between S and T suggests storing the union $S \cup T$, together with a pointer to the element not in S and a pointer to the element not in T; similar comments apply to the multisets $f(S)$ and $f(T)$. Keeping a running tally of $\#f(S)$ allows easily checking whether $\#f(S) < r$.

To decide whether a b-bit string x is suitable as a new element of T, one must check whether $x \in S$. Actually, what the diffusion steps need is not merely

knowing whether $x \in S$, but also knowing the number of elements of S smaller than x. ("Smaller" need not be defined lexicographically; the real objective is to compute a bijective map from b-bit strings to $\{1, 2, 3, \ldots, 2^b\}$ that maps S to $\{1, 2, 3, \ldots, r\}$.) The obvious sorted-array data structure allows these questions to be efficiently answered by binary search.

The big problem with this data structure is that inserting the new string into T requires, in the worst case, moving the other r elements of the array. This cost-r operation is performed at every step of the quantum walk, and dominates the total cost of the algorithm (unless evaluating f is very slow).

There is an extensive literature on classical data structures that support these operations much more efficiently. However, adapting a data structure to the quantum context raises three questions:

- Is the data-structure performance a random variable? Many data structures in the literature are randomized and provide good *average-case* performance but not good *worst-case* performance. The standard conversion of an algorithm to a quantum circuit requires first expressing the algorithm as a classical combinatorial circuit; the size of this circuit reflects the *worst-case* performance of the original algorithm.
- Does the performance of the data structure depend on S? For example, a standard hash table provides good performance for *most* sets S but not for *all* sets S.
- Is the data structure history-dependent? For most data structures, the representation of a set S depends on the order in which elements were added to and removed from the set. This spoils the analysis of the quantum walk through sets S, and presumably spoils the actual performance of the walk.

The first problem is usually minor: one can simply stop each algorithm after a constant time, where the constant is chosen so that the chance of an incorrect answer is negligible. The second problem can usually be converted into the first problem by some extra randomization: for example, one can choose a random hash function from a suitable family (as suggested by Wegman and Carter in [33]), or encrypt the b-bit strings before storing them. But the third problem is much more serious: it rules out balanced trees, red-black trees, most types of hash tables, etc.

Ambainis handles these issues in [2, Section 6] with an ad-hoc "combination of a hash table and a skip list", requiring several pages of analysis. We point out a much simpler solution: storing S etc. in a *radix tree*. Presumably this also saves time, although the speedup is not visible at the level of detail of our analysis.

The simplest type of radix tree is a binary tree in which the left subtree stores $\{x : (0, x) \in S\}$ and the right subtree stores $\{x : (1, x) \in S\}$; subtrees storing empty sets are omitted. To check whether $x \in S$ one starts from the root of the tree and follows pointers according to the bits of x in turn; the worst-case number of operations is proportional to the number of bits in x. Counting the number of elements of S smaller than x is just as easy if each tree node is augmented by a count of the number of elements below that node. A tree storing $f(S)$, with each leaf node accompanied by its multiplicity, allows an efficient running tally of the

number $\#f(S)$ of distinct elements of $f(S)$, and in particular quickly checking whether $\#f(S) < r$.

Randomizing the memory layout of the nodes for the radix tree for S (inductively, by placing each new node at a uniform random free position) provides history-independence for the classical data structure: each possible representation of S has equal probability to appear. Similarly, creating a uniform superposition over all possible memory layouts of the nodes produces a unique quantum data structure representing S.

Subset-Sum Solutions via Collisions. It is straightforward to recast the subset-sum problem as a collision-finding problem.

Consider the function f that maps $(1, I_1)$ to $\Sigma(I_1)$ for $I_1 \subseteq \{1, 2, \ldots, n/2\}$, and maps $(2, I_2)$ to $s - \Sigma(I_2)$ for $I_2 \subseteq \{n/2 + 1, n/2 + 2, \ldots, n\}$. Use the algorithm described above to find a collision in f. There are only $n/2 + 1$ bits of input to f, so the cost of this algorithm is only $2^{n/3}$.

In the "hard" cases of interest in this paper, there are not likely to be many collisions among inputs $(1, I_1)$, and there are not likely to be many collisions among inputs $(2, I_2)$, so the collision found has a good chance of having the form $\Sigma(I_1) = s - \Sigma(I_2)$, i.e., $\Sigma(I_1 \cup I_2) = s$. One can, alternatively, tweak the algorithm to ignore collisions among $(1, I_1)$ and collisions among $(2, I_2)$ even if such collisions exist.

4 Moduli

This section discusses the use of a "modulus" to partition the spaces being searched in Section 2. The traditional view is that this is merely a method to reduce memory consumption (which we do not measure in this paper); but moduli are also an essential building block for the faster algorithms of Section 5. This section reviews the traditional left-right split with a modulus, and then states a quantum algorithm with a modulus, as a warmup for the faster quantum algorithm of Section 5.

Schroeppel and Shamir in [29] introduced an algorithm with essentially the same reduction of memory consumption, but that algorithm does not use moduli and does not seem helpful in Section 5. Three decades later, Howgrave-Graham and Joux in [17, eprint version, Section 3.1] described the left-right split with a modulus as a "useful practical variant" of the Schroeppel–Shamir algorithm; Becker, Coron, and Joux stated in [3] that this was a "simpler but heuristic variant of Schroeppel–Shamir". A more general algorithm (with one minor restriction, namely a prime choice of modulus) had already been stated a few years earlier by Elsenhans and Jahnel; see [10, Section 4.2.1] and [11, page 2]. There are many earlier papers that used moduli to partition input spaces without stating the idea in enough generality to cover subset sums; we have not attempted to comprehensively trace the history of the idea.

Review: Left-Right Split with a Modulus. Choose a positive integer $M \approx 2^{n/4}$, and choose $t \in \{0, 1, 2, \ldots, M-1\}$. Compute

$$L_1 = \{(\Sigma(I_1), I_1) : I_1 \subseteq \{1, 2, \ldots, n/2\}, \quad \Sigma(I_1) \equiv t \pmod{M}\}.$$

The problem of finding all I_1 here is a size-$n/2$ subset-sum problem modulo M. This problem can, in turn, be solved as a small number of size-$n/2$ subset-sum problems without moduli, namely searching for subsets of $x_1 \bmod M, x_2 \bmod M, \ldots, x_{n/2} \bmod M$ having sum t or $t+M$ or $\ldots$ or $t+(n/2-1)M$. Note, however, that it is important for this size-$n/2$ subroutine to find *all* solutions rather than just *one* solution.

A reasonable choice of subroutine here is the original left-right-split algorithm (without a modulus). This subroutine costs $2^{n/4}$ for a problem of size $n/2$, and is trivially adapted to find all solutions. For this adaptation one must add the number of solutions to the cost, but in this context one expects only about $2^{n/2}/M \approx 2^{n/4}$ subsets I_1 to satisfy $\Sigma(I_1) \equiv t \pmod{M}$, for a total cost of $2^{n/4}$. One can also tweak this subroutine to work directly with sums modulo M, rather than separately handling t, $t+M$, etc.

Similarly compute

$$L_2 = \{(s - \Sigma(I_2), I_2) : I_2 \subseteq \{n/2+1, \ldots, n\}, \quad \Sigma(I_2) \equiv s - t \pmod{M}\}.$$

Store L_1 in a sorted table, and for each $(s - \Sigma(I_2), I_2) \in L_2$ check whether $s - \Sigma(I_2)$ appears in the table. If there is a collision $\Sigma(I_1) = s - \Sigma(I_2)$, print $I_1 \cup I_2$ as a root of $\Sigma - s$ and stop. Otherwise try another value of t, repeating until all choices of $t \in \{0, 1, 2, \ldots, M-1\}$ are exhausted.

One expects each choice of t to cost $2^{n/4}$, as discussed above. There are $M \approx 2^{n/4}$ choices of t, for a total cost of $2^{n/2}$. If there is a subset-sum solution then it will be found for some choice of t.

Quantum Search with a Modulus. The algorithm above is a classical search for a root of the function that maps t to 0 if there is a collision $\Sigma(I_1) = s - \Sigma(I_2)$ satisfying $\Sigma(I_1) \equiv t \pmod{M}$ (and therefore also satisfying $\Sigma(I_2) \equiv s - t \pmod{M}$). One way to take advantage of quantum computers here is to instead search for t by Grover's algorithm, which finds the root with only $\sqrt{M} \approx 2^{n/8}$ quantum evaluations of the same function, for a total cost of $2^{n/8}2^{n/4} = 2^{3n/8}$.

Quantum Walks with a Modulus. A different way to take advantage of quantum computers is as follows.

Recall that the collision-finding algorithm of Section 3 walks through adjacent pairs of size-r sets S, searching for sets that contain collisions under a function f. Each set S is stored in a radix tree, as is the multiset $f(S)$. Each radix tree is augmented to record at each node the number of distinct elements below that node, allowing fast evaluation of the number of elements of S smaller than a specified input and fast evaluation of whether S contains a collision.

One can design this collision-finding algorithm in four steps:

- Start with a simple classical collision-finding algorithm that computes $f(S)$ where S is the set of *all* 2^b b-bit strings.

- Generalize to a lower-probability algorithm that computes $f(S)$ where S is a set of only r strings and that checks whether S contains the collision.
- Build a data structure that expresses the entire computation of the lower-probability algorithm. Observe that this data structure allows efficiently moving from S to an adjacent set: a single element of S has only a small impact on the computation.
- Apply a quantum walk, walking through adjacent pairs of size-r sets S while maintaining this data structure for each S. This takes $O(\sqrt{r}/\sqrt{p})$ steps where p is the success probability of the previous algorithm, plus cost r to set up the data structure in the first place.

We now imitate the same four-step approach, starting from the classical left-right split with a modulus and ending with a new quantum subset-sum algorithm. The following description assumes that the correct value of t is already known; the overhead of searching for t is discussed after the algorithm.

First step: Classical algorithm. Recall that the subroutine to find all I_1 with $\Sigma(I_1) \equiv t$ computes $\Sigma(I_{11}) \bmod M$ for all $I_{11} \subseteq \{1, 2, \ldots, n/4\}$; computes $t - \Sigma(I_{12}) \bmod M$ for all $I_{12} \subseteq \{n/4+1, n/4+2, \ldots, n/2\}$; and finds collisions between $\Sigma(I_{11}) \bmod M$ and $t - \Sigma(I_{12}) \bmod M$. Similarly, the subroutine to find all I_2 finds collisions between $\Sigma(I_{21}) \bmod M$ for $I_{21} \subseteq \{n/2+1, n/2+2, \ldots, 3n/4\}$ and $s - t - \Sigma(I_{22}) \bmod M$ for $I_{22} \subseteq \{3n/4+1, 3n/4+2, \ldots, n\}$. The high-level algorithm finishes by finding collisions between $\Sigma(I_1)$ and $s - \Sigma(I_2)$.

Second step: Generalize to a lower-probability computation by restricting the sets that contain collisions. Specifically, instead of enumerating *all* subsets I_{11}, take a random collection S_{11} containing exactly r such subsets; here $r \leq 2^{n/4}$ is an algorithm parameter optimized below. Similarly take a random collection S_{12} of exactly r subsets I_{12}. Find collisions between $\Sigma(I_{11}) \bmod M$ and $t - \Sigma(I_{12}) \bmod M$; one expects about r^2/M collisions, producing r^2/M sets $I_1 = I_{11} \cup I_{12}$ satisfying $\Sigma(I_1) \equiv t \pmod M$. Similarly take random size-r sets S_{21} and S_{22} consisting of, respectively, subsets I_{21} and I_{22}; find collisions between $\Sigma(I_{21}) \bmod M$ and $s-t-\Sigma(I_{22}) \bmod M$, obtaining about r^2/M sets I_2 satisfying $\Sigma(I_2) \equiv s-t \pmod M$. Finally check for collisions between $\Sigma(I_1)$ and $s-\Sigma(I_2)$. One can visualize the construction of $I = I_1 \cup I_2$ as a three-level binary tree with I as the root, I_1 and I_2 as its left and right children, and $I_{11}, I_{12}, I_{21}, I_{22}$ as the leaves.

Recall that we are assuming that the correct value of t is known, i.e., that the desired subset-sum solution is expressible as $I_1 \cup I_2$ with $\Sigma(I_1) \equiv t \pmod M$ and $\Sigma(I_2) \equiv s - t \pmod M$. Then S_{11} has probability $r/2^{n/4}$ of containing the set $I_{11} = I_1 \cap \{1, 2, \ldots, n/4\}$. Similar comments apply to S_{12}, S_{21}, and S_{22}, for an overall success probability of $(r/2^{n/4})^4$.

The optimal choice of r is discussed later, and is far below the classical extreme $2^{n/4}$. This drop is analogous to the drop in list sizes from a simple left-right split (list size $2^{n/2}$) to the quantum-walk subset-sum algorithm of Section 3 (list size $2^{n/3}$). This drop has an increasingly large impact on subsequent levels of the tree: the number of sets $I_{11}, I_{12}, I_{21}, I_{22}$ is reduced by a factor of $2^{n/4}/r$, and the number of sets I_1, I_2 is reduced by a factor of $(2^{n/4}/r)^2$.

An interesting consequence of this drop is that one can reduce M without creating bottlenecks at subsequent levels of the tree. Specifically, taking $M \approx r$ means that one expects about r sets I_1 and about r sets I_2.

Third step: Data structure. This lower-probability computation is captured by a data structure that contains the following sets in augmented radix trees:

- The size-r set S_{11} of subsets $I_{11} \subseteq \{1, 2, \ldots, n/4\}$.
- The set $\{(\Sigma(I_{11}) \bmod M, I_{11}) : I_{11} \in S_{11}\}$.
- The size-r set S_{12} of subsets $I_{12} \subseteq \{n/4+1, n/4+2, \ldots, n/2\}$.
- The set $\{(t - \Sigma(I_{12}) \bmod M, I_{12}) : I_{12} \in S_{12}\}$.
- The set S_1 of $I_{11} \cup I_{12}$ for all pairs $(I_{11}, I_{12}) \in S_{11} \times S_{12}$ such that $\Sigma(I_{11}) \equiv t - \Sigma(I_{12}) \pmod M$, subject to the limit discussed below.
- The size-r set S_{21} of subsets $I_{21} \subseteq \{n/2+1, n/2+2, \ldots, 3n/4\}$.
- The set $\{(\Sigma(I_{21}) \bmod M, I_{21}) : I_{21} \in S_{21}\}$.
- The size-r set S_{22} of subsets $I_{22} \subseteq \{3n/4+1, 3n/4+2, \ldots, n\}$.
- The set $\{(s - t - \Sigma(I_{22}) \bmod M, I_{22}) : I_{22} \in S_{22}\}$.
- The set S_2 of $I_{21} \cup I_{22}$ for all pairs $(I_{21}, I_{22}) \in S_{21} \times S_{22}$ such that $\Sigma(I_{21}) \equiv s - t - \Sigma(I_{22}) \pmod M$.
- The set $\{(\Sigma(I_1), I_1) : I_1 \in S_1\}$.
- The set $\{(s - \Sigma(I_2), I_2) : I_2 \in S_2\}$.
- The set S of $I_1 \cup I_2$ for all pairs $(I_1, I_2) \in S_1 \times S_2$ such that $\Sigma(I_1) = s - \Sigma(I_2)$, subject to the limit discussed below.

Note that this data structure supports, e.g., fast removal of an element I_{11} from S_{11} followed by fast insertion of a replacement element I'_{11}. Checking for $\Sigma(I_{11}) \bmod M$ in the stored set $\{(t - \Sigma(I_{12}) \bmod M, I_{12})\}$ efficiently shows which elements have to be removed from S_1, and then a similar check shows which elements have to be removed from S.

Each element I_{11} of S_{11} affects very few elements of S_1; on average one expects "very few" to be $r/M \approx 1$. To control the time taken by each step of the algorithm we put a polynomial limit on the number of elements of S_1 involving any particular I_{11}. If this limit is reached then (to ensure history-independence) we use a random selection of elements, but this limit has negligible chance of affecting the algorithm output. Similar comments apply to I_{12}, I_{21}, and I_{22}.

Fourth step: Walk through adjacent pairs of 4-tuples $(S_{11}, S_{12}, S_{21}, S_{22})$ of size-r sets, maintaining the data structure above and searching for tuples for which the final set S is nonempty. Amplifying the $(r/2^{n/4})^4$ success probability mentioned above to a high probability requires a quantum walk consisting of $O(\sqrt{r}(2^{n/4}/r)^2)$ steps. Setting up the data structure in the first place costs $O(r)$.

For r on the scale of $2^{0.2n}$ these costs are balanced at $2^{0.2n}$; but recall that this assumes that t is already known. A classical search for t means repeating this algorithm $M \approx r \approx 2^{0.2n}$ times, for a total cost of $2^{0.4n}$. We do better by using amplitude amplification [7], repeating the quantum walk only $2^{0.1n}$ times, for a total cost of $2^{0.3n}$. We do not describe amplitude amplification in detail; this subset-sum algorithm is superseded by the approach of Section 5.

5 Representations

"Representations" are a technique to improve the "left-right split with a modulus" algorithm of Section 4. Howgrave-Graham and Joux introduced this technique in [17] and obtained a subset-sum algorithm that costs just $2^{(0.337\ldots+o(1))n}$. Beware that [17] incorrectly claimed a cost of $2^{(0.311\ldots+o(1))n}$; the underlying flaw in the analysis was corrected in [3] with credit to May and Meurer.

This section reviews the Howgrave-Graham–Joux algorithm, and then presents a new quantum subset-sum algorithm with cost only $2^{(0.241\ldots+o(1))n}$. The new quantum algorithm requires the quantum-walk machinery discussed in Section 3.

We simplify the algorithm statements in this section by considering only half-weight sets I; i.e., we search only for sets I with $\#I = n/2$ and $\Sigma(I) = s$. We comment, however, that straightforward generalizations of these algorithms, still within the same cost bound, handle smaller known weights (adjusting the set sizes shown below, at the expense of some complications in notation); also handle larger known weights (replacing I and s with their complements); and handle arbitrary unknown weights (trying all $n+1$ possible weights).

Review: The Basic Idea of Representations. Recall that the original left-right split partitions I as $I_1 \cup I_2$ where $I_1 \subseteq \{1, \ldots, n/2\}$ and $I_2 \subseteq \{n/2+1, \ldots, n\}$. The main idea of representations is to partition I in a different, ambiguous way as $I_1 \cup I_2$ with $I_1, I_2 \subseteq \{1, 2, \ldots, n\}$ and $\#I_1 = \#I_2 = n/4$. Note that there are $\binom{n/2}{n/4} \approx 2^{n/2}$ such partitions. The key observation is that finding only one out of these exponentially many *representations* (I_1, I_2) of I is sufficient to solve the subset-sum problem.

Recall also the idea of moduli: pick $t \in \{0, 1, 2, \ldots, M-1\}$ and hope that $\Sigma(I_1) \equiv t \pmod{M}$. In Section 4, there was only one choice of I_1, so one expects each choice of t to work with probability only about $1/M$, forcing a search through choices of t. In this section, there are $\approx 2^{n/2}$ choices of I_1, so one expects a single choice of t to work with high probability for M as large as $2^{n/2}$.

These observations motivate the following strategy. Pick a modulus $M \approx 2^{n/2}$ and choose a random target value $t \in \{0, 1, \ldots, M-1\}$. Compute

$$L_1 = \{(\Sigma(I_1), I_1) : I_1 \subseteq \{1, \ldots, n\},\ \#I_1 = n/4,\ \Sigma(I_1) \equiv t \pmod{M}\}$$

and

$$L_2 = \{(s - \Sigma(I_2), I_2) : I_2 \subseteq \{1, \ldots, n\},\ \#I_2 = n/4,\ \Sigma(I_2) \equiv s - t \pmod{M}\}.$$

If there is a collision $\Sigma(I_1) = s - \Sigma(I_2)$ satisfying $I_1 \cap I_2 = \{\}$, print $I_1 \cup I_2$ and stop. If there are no such collisions, repeat with another choice of t. One expects a negligible failure probability after a polynomial number of repetitions.

(We point out that if $t \equiv s - t \pmod{M}$ then computing L_1 immediately produces L_2. One can arrange for this by choosing a random odd M and taking t to be half of s modulo M; one can also choose a random even M if s is even. If other speed constraints prevent M from being chosen randomly then one can still try these special values of t first. Similar comments apply to the next level of

the Howgrave-Graham–Joux algorithm described below. Of course, the resulting speedup is not visible at the level of detail of our analysis.)

One expects $\#L_1 \approx \binom{n}{n/4}/2^{n/2} \approx 2^{0.311\ldots n}$: there are $\binom{n}{n/4}$ sets $I_1 \subseteq \{1,\ldots,n\}$ with $\#I_1 = n/4$, and one expects each I_1 to satisfy $\Sigma(I_1) \equiv t \pmod M$ with probability $1/M \approx 1/2^{n/2}$. (The calculation of $0.311\ldots$ relies on the standard approximation $\binom{n}{\alpha n} \approx 2^{H(\alpha)n}$, where $H(\alpha) = -\alpha\log_2\alpha - (1-\alpha)\log_2(1-\alpha)$.) The same comment applies to L_2. One also expects the number of collisions between L_1 and L_2 to be exponentially large, about $\#L_1\#L_2/2^{n/2} \approx 2^{0.122\ldots n}$, since each $\Sigma(I_1)$ is already known to match each $s - \Sigma(I_2)$ modulo M; but the only collisions satisfying $I_1 \cap I_2 = \{\}$ are collisions arising from subset-sum solutions.

The remaining task is to compute L_1 and L_2 in the first place. Howgrave-Graham and Joux solve these two weight-$n/4$ modular subset-sum problems by first applying another level of representations (using a smaller modulus that divides M), obtaining four weight-$n/8$ modular subset-sum problems; they solve each of those problems with a weight-$n/16$ left-right split. The details appear below.

Review: The Complete Howgrave-Graham–Joux Algorithm. Choose a positive integer $M_1 \approx 2^{n/4}$. Choose a positive integer $M \approx 2^{n/2}$ divisible by M_1. Choose randomly $s_1 \in \{0,1,\ldots,M-1\}$ and define $s_2 = s - s_1$. Choose randomly $s_{11} \in \{0,1,\ldots,M_1-1\}$ and define $s_{12} = s_1 - s_{11}$. Choose randomly $s_{21} \in \{0,1,\ldots,M_1-1\}$ and define $s_{22} = s_2 - s_{21}$. Also choose random subsets $R_{111}, R_{121}, R_{211}, R_{221}$ of $\{1,2,\ldots,n\}$, each of size $n/2$, and define R_{ij2} as the complement of R_{ij1}.

The following algorithm searches for a weight-$n/2$ subset-sum solution I decomposed as follows: $I = I_1 \cup I_2$ with $\#I_i = n/4$ and $\Sigma(I_i) \equiv s_i \pmod M$; furthermore $I_i = I_{i1} \cup I_{i2}$ with $\#I_{ij} = n/8$ and $\Sigma(I_{ij}) \equiv s_{ij} \pmod{M_1}$; furthermore $I_{ij} = I_{ij1} \cup I_{ij2}$ with $\#I_{ijk} = n/16$ and $I_{ijk} \subseteq R_{ijk}$. These constraints are shown as a tree in Figure 5.1. One expects a weight-$n/2$ subset-sum solution to decompose in this way with high probability (inverse polynomial in n), as discussed later, and if it does decompose in this way then it is in fact found by this algorithm.

Start with, for each $(i,j,k) \in \{1,2\}\times\{1,2\}\times\{1,2\}$, the set S_{ijk} of all subsets $I_{ijk} \subseteq R_{ijk}$ with $\#I_{ijk} = n/16$. Compute the sets

$$L_{ij1} = \{(\Sigma(I_{ij1}) \bmod M_1, I_{ij1}) : I_{ij1} \in S_{ij1}\},$$
$$L_{ij2} = \{(s_{ij} - \Sigma(I_{ij2}) \bmod M_1, I_{ij2}) : I_{ij2} \in S_{ij2}\}.$$

Merge L_{ij1} and L_{ij2} to obtain the set S_{ij} of $I_{ij1} \cup I_{ij2}$ for all pairs $(I_{ij1}, I_{ij2}) \in S_{ij1} \times S_{ij2}$ such that $\Sigma(I_{ij1}) \equiv s_{ij} - \Sigma(I_{ij2}) \pmod{M_1}$. Note that each $I_{ij} \in S_{ij}$ has $\Sigma(I_{ij}) \equiv s_{ij} \pmod{M_1}$ and $\#I_{ij} = n/8$. Next compute the sets

$$L_{i1} = \{(\Sigma(I_{i1}) \bmod M, I_{i1}) : I_{i1} \in S_{i1}\},$$
$$L_{i2} = \{(s_i - \Sigma(I_{i2}) \bmod M, I_{i2}) : I_{i2} \in S_{i2}\}.$$

Merge L_{i1} and L_{i2} to obtain the set S_i of $I_{i1} \cup I_{i2}$ for all pairs $(I_{i1}, I_{i2}) \in S_{i1} \times S_{i2}$ such that $\Sigma(I_{i1}) \equiv s_i - \Sigma(I_{i2}) \pmod M$ and $I_{i1} \cap I_{i2} = \{\}$. Note that each $I_i \in S_i$

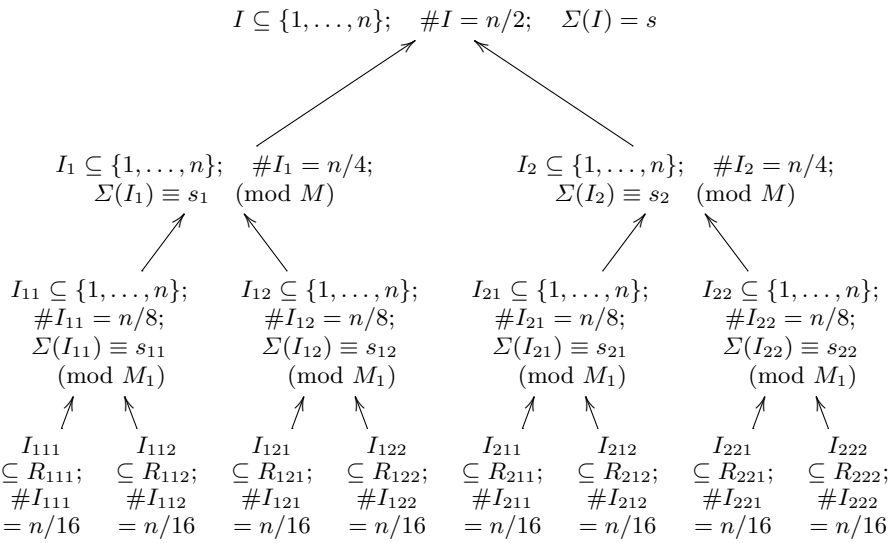


Fig. 5.1. Decomposition of a weight-$n/2$ subset-sum solution $I \subseteq \{1, 2, \ldots, n\}$

has $\Sigma(I_i) \equiv s_i \pmod M$ and $\#I_i = n/4$. Next compute the sets

$$L_1 = \{(\Sigma(I_1), I_1) : I_1 \in S_1\},$$
$$L_2 = \{(s - \Sigma(I_2), I_2) : I_2 \in S_2\}.$$

Merge L_1 and L_2 to obtain the set S of $I_1 \cup I_2$ for all pairs $(I_1, I_2) \in S_1 \times S_2$ such that $\Sigma(I_1) = s - \Sigma(I_2)$ and $I_1 \cap I_2 = \{\}$. Note that each $I \in S$ has $\Sigma(I) = s$ and $\#I = n/2$. If S is nonempty, print its elements and stop.

Review: Success Probability of the Algorithm. Consider any weight-$n/2$ subset $I \subseteq \{1, \ldots, n\}$ with $\Sigma(I) = s$. There are $\binom{n/2}{n/4} \approx 2^{n/2} \approx M$ ways to partition I as $I_1 \cup I_2$ with $\#I_1 = n/4$ and $\#I_2 = n/4$, and as discussed earlier one expects that with high probability at least one of these ways will satisfy $\Sigma(I_1) \equiv s_1 \pmod M$, implying $\Sigma(I_2) \equiv s_2 \pmod M$.

Similarly, there are $\binom{n/4}{n/8} \approx 2^{n/4} \approx M_1$ ways to partition I_1 as $I_{11} \cup I_{12}$ with $\#I_{11} = n/8$ and $\#I_{12} = n/8$. One expects that with high probability at least one of these ways will satisfy $\Sigma(I_{11}) \equiv s_{11} \pmod{M_1}$, implying $\Sigma(I_{12}) \equiv s_{12} \pmod{M_1}$ (since M_1 divides M and $s_{11} + s_{12} = s_1$). Analogous comments apply to I_2, I_{21}, I_{22}.

A uniform random subset of a set of size $n/8$ has size exactly $n/16$ with probability $\Theta(1/\sqrt{n})$, so with probability $\Theta(1/n^2)$ each of the four sets $I_{ij1} = I_{ij} \cap R_{ij1}$ has size exactly $n/16$, implying that each of the four complementary sets $I_{ij2} = I_{ij} \cap R_{ij2}$ also has size exactly $n/16$. (Experiments indicate that the probability is somewhat worse, although still inverse polynomial in n, if all R_{ij1}

are chosen to be identical, even if this set is randomized as discussed in [17]. The idea of choosing independent sets appeared in [4] with credit to Bernstein.)

Overall the probability of I being decomposed in this way, i.e., of I being found by this algorithm, is inverse polynomial in n. As above, one expects a negligible failure probability after a polynomial number of repetitions of the algorithm.

Review: Cost of the Algorithm. Each set S_{ijk} has size $\binom{n/2}{n/16} \approx 2^{0.271\ldots n}$. One expects $\#S_{ij1}\#S_{ij2}/M_1$ collisions $\Sigma(I_{ij1}) \equiv s_{ij} - \Sigma(I_{ij2}) \pmod{M_1}$, and therefore $\#S_{ij1}\#S_{ij2}/M_1 \approx 2^{0.293\ldots n}$ elements in S_{ij}.

Each $\Sigma(I_{i1})$ is already known by construction to be the same as each $s_i - \Sigma(I_{i2})$ modulo M_1. One expects it to be the same modulo M with probability M_1/M, for a total of $\#S_{i1}\#S_{i2}M_1/M \approx 2^{0.337\ldots n}$ collisions modulo M. This also dominates the algorithm's overall running time.

Relatively few of these collisions modulo M have $I_{i1} \cap I_{i2} = \{\}$. The only possible elements of S_1 are sets I_1 with $\Sigma(I_1) \equiv s_1 \pmod{M}$ and $\#I_1 = n/4$; one expects the number of such sets to be $\binom{n}{n/4}/2^{n/2} \approx 2^{0.311\ldots n}$. Furthermore, as discussed earlier, each $\Sigma(I_1)$ is already known by construction to be the same as each $s - \Sigma(I_2)$ modulo M, so one expects it to be the same integer with probability about $M/2^n$, for a total of $\#S_1\#S_2M/2^n \approx 2^{0.122\ldots n}$ collisions.

New: Quantum Walks with Moduli and Representations. We now combine quantum walks with the idea of representations. The reader is assumed to be familiar with the simpler quantum algorithm of Section 4.

We introduce a parameter $r \leq \binom{n/2}{n/16}$ into the Howgrave-Graham–Joux algorithm, and take each S_{ijk} as a random collection of exactly r weight-$n/16$ subsets I_{ijk} of R_{ijk}. The extreme case $r = \binom{n/2}{n/16}$ is the same as the original Howgrave-Graham–Joux algorithm: in this case S_{ijk} is the set of all weight-$n/16$ subsets of R_{ijk}. For smaller r this generalized algorithm has lower success probability, as discussed below, but is also faster. The resulting computation is captured by the following 29 sets, which we store in augmented radix trees:

- For each i, j, k, a set S_{ijk} consisting of exactly r weight-$n/16$ subsets of R_{ijk}.
- For each i, j, the set $L_{ij1} = \{(\Sigma(I_{ij1}) \bmod M_1, I_{ij1}) : I_{ij1} \in S_{ij1}\}$.
- For each i, j, the set $L_{ij2} = \{(s_{ij} - \Sigma(I_{ij2}) \bmod M_1, I_{ij2}) : I_{ij2} \in S_{ij2}\}$.
- For each i, j, the set S_{ij} of $I_{ij1} \cup I_{ij2}$ for all pairs $(I_{ij1}, I_{ij2}) \in S_{ij1} \times S_{ij2}$ such that $\Sigma(I_{ij1}) \equiv s_{ij} - \Sigma(I_{ij2}) \pmod{M_1}$, subject to the limit discussed below.
- For each i, the set $L_{i1} = \{(\Sigma(I_{i1}) \bmod M, I_{i1}) : I_{i1} \in S_{i1}\}$.
- For each i, the set $L_{i2} = \{(s_i - \Sigma(I_{i2}) \bmod M, I_{i2}) : I_{i2} \in S_{i2}\}$.
- For each i, the set S_i of $I_{i1} \cup I_{i2}$ for all pairs $(I_{i1}, I_{i2}) \in S_{i1} \times S_{i2}$ such that $\Sigma(I_{i1}) \equiv s_i - \Sigma(I_{i2}) \pmod{M}$ and $I_{i1} \cap I_{i2} = \{\}$, subject to the limit discussed below.
- The set $L_1 = \{(\Sigma(I_1), I_1) : I_1 \in S_1\}$.
- The set $L_2 = \{(s - \Sigma(I_2), I_2) : I_2 \in S_2\}$.
- The set S of $I_1 \cup I_2$ for all pairs $(I_1, I_2) \in S_1 \times S_2$ such that $\Sigma(I_1) = s - \Sigma(I_2)$ and $I_1 \cap I_2 = \{\}$, subject to the limit discussed below.

Like the data structure in the quantum walk of Section 4, this data structure supports, e.g., fast removal of an element I_{111} from S_{111} followed by fast insertion of a replacement element I'_{111}.

The optimal choice of r is discussed later; it is far below the starting list size $\binom{n/2}{n/16} \approx 2^{0.272n}$ used by Howgrave-Graham and Joux, and is even below $2^{n/4}$. One expects the number of collisions modulo M_1 to be $r^2/2^{n/4}$, which is smaller than r, and the list sizes on subsequent levels to be even smaller. Consequently this quantum algorithm ends up being bottlenecked at the bottom level of the tree, while the original algorithm is bottlenecked at a higher level.

Furthermore, one expects each element I_{ijk} of S_{ijk} to affect, on average, approximately $r/2^{n/4}$ elements of the set of collisions modulo M, and therefore to affect 0 elements of S_{ij} in almost all cases, 1 element of S_{ij} with exponentially small probability, 2 elements of S_{ij} with far smaller probability, etc. As in Section 4, to control the time taken by each step of the algorithm we put a polynomial limit on the number of elements of S_{ij} involving any particular I_{ijk}, a polynomial limit on the number of elements of S_i involving any particular I_{ij}, and a polynomial limit on the number of elements of S involving any particular I_i. A constant limit of 100 seems ample, and there is no obvious problem with a limit of 1.

Compared to the original Howgrave-Graham–Joux algorithm, the success probability of the generalized algorithm drops by a factor of $(r/\binom{n/2}{n/16})^8$, since each target I_{ijk} has chance only $r/\binom{n/2}{n/16}$ of being in S_{ijk}. Amplifying this to a high probability requires a quantum walk consisting of $O(\sqrt{r}(\binom{n/2}{n/16}/r)^4)$ steps. Setting up the data structure in the first place costs $O(r)$. For r on the scale of $\binom{n/2}{n/16}^{4/4.5}$ these costs are balanced at $\binom{n/2}{n/16}^{4/4.5}$, i.e., at $2^{(0.241\ldots+o(1))n}$.

References

[1] — (no editor): 20th annual symposium on foundations of computer science. IEEE Computer Society, New York (1979). MR 82a:68004. See [32]

[2] Ambainis, A.: Quantum walk algorithm for element distinctness. SIAM Journal on Computing 37, 210–239 (2007). http://arxiv.org/abs/quant-ph/0311001. Citations in this document: §3, §3, §3

[3] Becker, A., Coron, J.-S., Joux, A.: Improved generic algorithms for hard knapsacks. In: Eurocrypt 2011 [**27**] (2011). http://eprint.iacr.org/2011/474. Citations in this document: §1, §1, §1, §4, §5

[4] Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes in $2^{n/20}$: how $1 + 1 = 0$ improves information set decoding. In: Eurocrypt 2012 [**28**] (2012). http://eprint.iacr.org/2012/026. Citations in this document: §1, §1, §1, §1, §5

[5] Bernstein, D.J.: Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete? In: Workshop Record of SHARCS'09: Special-purpose Hardware for Attacking Cryptographic Systems (2009). http://cr.yp.to/papers.html#collisioncost. Citations in this document: §2

[6] Boyer, M., Brassard, G., Høyer, P., Tapp, A.: Tight bounds on quantum searching. Fortschritte Der Physik 46, 493–505 (1998). http://arxiv.org/abs/quant-ph/9605034v1. Citations in this document: §2, §2

[7] Brassard, G., Høyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. In: [**20**], pp. 53–74 (2002). http://arxiv.org/abs/quant-ph/0005055. Citations in this document: §4

[8] Brassard, G., Høyer, G., Tapp, A.: Quantum cryptanalysis of hash and claw-free functions. In: LATIN'98 [**21**], pp. 163–169 (1998). MR 99g:94013. Citations in this document: §2, §2

[9] Chang, W.-L., Ren, T.-T., Feng, M., Lu, L.C., Lin, K.W., Guo, M.: Quantum algorithms of the subset-sum problem on a quantum computer. International Conference on Information Engineering 2, 54–57 (2009). Citations in this document: §2

[10] Elsenhans, A.-S., Jahnel, J.: The diophantine equation $x^4 + 2y^4 = z^4 + 4w^4$. Mathematics of Computation 75, 935–940 (2006). http://www.uni-math.gwdg.de/jahnel/linkstopaperse.html. Citations in this document: §4

[11] Elsenhans, A.-S., Jahnel, J.: The Diophantine equation $x^4 + 2y^4 = z^4 + 4w^4$— a number of improvements (2006). http://www.uni-math.gwdg.de/jahnel/linkstopaperse.html. Citations in this document: §4

[12] Gilbert, H. (ed.): Advances in cryptology — EUROCRYPT 2010, 29th annual international conference on the theory and applications of cryptographic techniques, French Riviera, May 30–June 3, 2010, proceedings. LNCS, vol. 6110. Springer (2010). See [17]

[13] Goldwasser, S. (ed.): 35th annual IEEE symposium on the foundations of computer science. Proceedings of the IEEE symposium held in Santa Fe, NM, November 20–22, 1994. IEEE (1994). ISBN 0-8186-6580-7. MR 98h:68008. See [30]

[14] Grover, L.K.: A fast quantum mechanical algorithm for database search. In: [**26**], pp. 212–219 (1996). MR 1427516. http://arxiv.org/abs/quant-ph/9605043

[15] Grover, L.K.: Quantum mechanics helps in searching for a needle in a haystack. Physical Review Letters 79, 325–328 (1997). http://arxiv.org/abs/quant-ph/9706033. Citations in this document: §2, §2

[16] Horowitz, E., Sahni, S.: Computing partitions with applications to the knapsack problem. Journal of the ACM 21, 277–292 (1974). Citations in this document: §2

[17] Howgrave-Graham, N., Joux, A.: New generic algorithms for hard knapsacks. In: Eurocrypt 2010 [**12**] (2010). http://eprint.iacr.org/2010/189. Citations in this document: §1, §1, §1, §1, §4, §5, §5, §5

[18] Johnson, D.S., Feige, U. (eds.): Proceedings of the 39th annual ACM symposium on the theory of computing, San Diego, California, USA, June 11–13, 2007. Association for Computing Machinery (2007). ISBN 978-1-59593-631-8. See [23]

[19] Lee, D.H., Wang, X. (eds.): Advances in cryptology — ASIACRYPT 2011, 17th international conference on the theory and application of cryptology and information security, Seoul, South Korea, December 4–8, 2011, proceedings. LNCS, vol. 7073. Springer (2011). ISBN 978-3-642-25384-3. See [24]

[20] Lomonaco Jr., S.J., Brandt, H.E. (eds.): Quantum computation and information. Papers from the AMS Special Session held in Washington, DC, January 19–21, 2000. Contemporary Mathematics, vol. 305. American Mathematical Society (2002). ISBN 0-8218-2140-7. MR 2003g:81006. See [7]

[21] Lucchesi, C.L., Moura, A.V. (eds.): LATIN'98: theoretical informatics. Proceedings of the 3rd Latin American symposium held in Campinas, April 20–24, 1998. LNCS, vol. 1380. Springer (1998). ISBN 3-540-64275-7. MR 99d:68007. See [8]

[22] Lyubashevsky, V., Palacio, A., Segev, G.: Public-key cryptographic primitives provably as secure as subset sum. In: TCC 2010 [**25**], pp. 382–400 (2010). http://eprint.iacr.org/2009/576. Citations in this document: §1, §1

[23] Magniez, F., Nayak, A., Roland, J., Santha, M.: Search via quantum walk. In: STOC 2007 [**18**], pp. 575–584 (2007). http://arxiv.org/abs/quant-ph/0608026. Citations in this document: §3

[24] May, A., Meurer, A., Thomae, E.: Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In: Asiacrypt 2011 [**19**] (2011). http://www.cits.rub.de/imperia/md/content/may/paper/decoding.pdf. Citations in this document: §1, §1, §1

[25] Micciancio, D. (ed.): Theory of cryptography, 7th theory of cryptography conference, TCC 2010, Zurich, Switzerland, February 9–11, 2010, proceedings. LNCS, vol. 5978. Springer (2010). ISBN 978-3-642-11798-5. See [22]

[26] Miller, G.L. (ed.): Proceedings of the twenty-eighth annual ACM symposium on the theory of computing, Philadelphia, PA, May 22–24, 1996. Association for Computing Machinery (1996). ISBN 0-89791-785-5. MR 97g:68005. See [14]

[27] Paterson, K.G. (ed.): Advances in cryptology — EUROCRYPT 2011, 30th annual international conference on the theory and applications of cryptographic techniques, Tallinn, Estonia, May 15–19, 2011, proceedings. LNCS, vol. 6632. Springer (2011). ISBN 978-3-642-20464-7. See [3]

[28] Pointcheval, D., Johansson, T. (eds.): Advances in cryptology — EUROCRYPT 2012 — 31st annual international conference on the theory and applications of cryptographic techniques, Cambridge, UK, April 15–19, 2012, proceedings. LNCS, vol. 7237. Springer (2012). ISBN 978-3-642-29010-7. See [4]

[29] Schroeppel, R., Shamir, A.: A $T = O(2^{n/2})$, $S = O(2^{n/4})$ algorithm for certain NP-complete problems. SIAM Journal on Computing 10, 456–464 (1981). Citations in this document: §4

[30] Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: [**13**], pp. 124–134 (1994); see also newer version [**31**]. MR 1489242

[31] Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing 26, 1484–1509 (1997); see also older version [**30**]. MR 98i:11108. http://arxiv.org/abs/quant-ph/9508027

[32] Wegman, M.N., Lawrence Carter, J.: New classes and applications of hash functions. In: [**1**], pp. 175–182 (1979); see also newer version [**33**]

[33] Wegman, M.N., Lawrence Carter, J.: New hash functions and their use in authentication and set equality. Journal of Computer and System Sciences 22, 265–279 (1981); see also older version [**32**]. ISSN 0022-0000. MR 82i:68017. Citations in this document: §3

Improved Lattice-Based Threshold Ring Signature Scheme

Slim Bettaieb and Julien Schrek

XLIM-DMI, Université de Limoges,
123, av. Albert Thomas
87060 Limoges Cedex, France
{slim.bettaieb,julien.schrek}@xlim.fr

Abstract. We present in this paper an improvement of the lattice-based threshold ring signature proposed by Cayrel, Lindner, Rückert and Silva (CLRS) [LATINCRYPT '10]. We generalize the same identification scheme CLRS to obtain a more efficient threshold ring signature. The security of our scheme relies on standard lattice problems. The improvement is a significant reduction of the size of the signature. Our result is a t-out-of-N threshold ring signature which can be seen as t different ring signatures instead of N for the other schemes. We describe the ring signature induced by the particular case of only one signer. To the best of our knowledge, the resulted signatures are the most efficient lattice-based ring signature and threshold signature.

Keywords: Threshold ring signatures, lattices.

1 Introduction

Lattices were first used in cryptography with the LLL algorithm [1] in order to cryptanalyse some number theory primitives [2]. In 1996, the NTRU cryptosystem proposed an original idea to base cryptography on lattices assumptions.

In 1996, Ajtai [3] showed how to use the standard lattice problem GapSVP in order to build cryptographic schemes. More specifically he proved that the worst-case hardness of GapSVP is reduced to the average-case hardness of SIS problem. Lattice-based cryptography became a good alternative to number theory cryptography in regard to the quantum computer assumption. Nowadays several lattice-based cryptosystems show good results with strong security proofs.

The notion of group signature was first formalised by Chaum and van Heyst in 1991 [4]. A group signature scheme allows a user in a group to sign anonymously a message on behalf of the group. The group is administered by a group manager, who can accept to add users to the group and has the ability to determine the real identity of the signer when needed. Motivated by the following situation: a high-ranking official in the government wants to leak anonymously an important information to a journalist. Rivest, Shamir, and Tauman [5] introduced the concept of ring signature and proposed a scheme based on RSA.

P. Gaborit (Ed.): PQCrypto 2013, LNCS 7932, pp. 34–51, 2013.

As opposed to group signatures, ring signatures have no group manager and no anonymity revocation system.

The concept of ring signature was extended by Bresson, Stern and Szydlo to threshold ring signatures [6]. In this setting, t-out-of-N users interact together in order to produce a signature without giving any information of the set of signers which produced the signature. Since their introduction, several threshold ring signature schemes were proposed [7–9]. Those constructions have a particularity in common: a complexity in $\mathcal{O}(Nt)$.

Aguilar, Cayrel and Gaborit [10] proposed a new threshold ring signature in 2008 of size in $\mathcal{O}(N)$. It is a code-based signature considered as a very efficient scheme due to its complexity.

In 2010, Cayrel, Lindner, Rückert and Silva [11] presented a lattice based version of the scheme in [10]. This version generalised an identification scheme given in [12], which is more efficient than Stern's identification protocol used by Aguilar et al. [10]. The security of the new scheme is based on the shortest independent vectors problem SIVP.

Our Contributions. In this work, we present an improvement of the lattice-based threshold ring signature scheme given in [11]. The improvement is due to a different way to achieve anonymity. Our scheme looks like t different digital signature. The difference is an extra challenge-answer part to deal with anonymity. The first part has the same size as t digital signatures (we use seeds to represent random permutations and other masks). The other part which deals with anonymity is in complexity $O(Nt)$ which is more important than $O(N)$ in [11] but implies smaller signatures when Nt is smaller than the size of t digital signatures. We also detail a ring signature in the particular case $t = 1$, which has the smallest size among others lattice-based ring signatures [13, 14, 11]. A table of comparison can be found in section 6.

Organization. In Section 2, we give some definitions and notions related to lattices and cryptography as well as a description of the identification scheme from [12]. In section 3, we present and detail our ring signature scheme. In section 4, we construct the lattice-based threshold ring signature scheme. The proofs of security for the scheme are detailed in section 5. In section 6, we give some concrete instantiations of our schemes and a comparison with the schemes from [11].

2 Preliminaries

This section is split into three parts. In the first one we give some basic definitions about lattices and cryptography. The second part details formal security definitions about ring signatures. In the third part we describe the identification scheme CLRS presented in [12].

Notations. We use bold upper-case letters to denote matrices and bold lower-case letters to denote vectors. We denote the Euclidean norm of the vector $\mathbf{v}$

by $\|\mathbf{v}\|$. For q an integer, $\mathbb{Z}_q$ denotes the group of integers modulo q. For a set E, we use the notation $w \xleftarrow{\$} E$ to mean that w is chosen randomly at uniform from E. Let $wh(\mathbf{v})$ denote the Hamming weight of $\mathbf{v}$ (the number of non-zero elements in $\mathbf{v}$) and for any integer m we denote by S_m the set of permutation of $\{1, \ldots, m\}$. Let $N \in \mathbb{N}$, we denote by δ_j the vector in $\{0,1\}^N$ with 1 in the j-th position and 0 elsewhere.

2.1 Lattices in Cryptography

We recall the definition of lattices, the small integer solution problem SIS, the inhomogeneous small integer solution problem ISIS, the standard hard lattice problem SIVP and a lattice-based commitment scheme.

Definition 1. *Let $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ be set of n linearly independent vectors in $\mathbb{R}^m$. The lattice generated by B is the set*

$$\mathcal{L}(B) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \right\}$$

of all integer combination of vectors in B. We denote by $\lambda_1(\mathcal{L}(B))$ the shortest vector of the lattice $\mathcal{L}(B)$. For $i \in \{1, \ldots, n\}$, we denote by $\lambda_i(\mathcal{L})$ the successive minima which is the smallest values $\lambda_i(\mathcal{L})$ such that the sphere of radius $\lambda_i(\mathcal{L})$ of center the origin contains at least i linearly independent lattice vectors.

Definition 2 ($\mathbf{SIS}_{q,n,m,\alpha}$ problem). *Given a random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ find a non zero vector $\mathbf{v} \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{v}^T = 0$ and $\|\mathbf{v}\| \leq \alpha$.*

Definition 3 ($\mathbf{ISIS}_{q,n,m,\alpha}$ problem). *Given a random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ a vector $\mathbf{v} \in \mathbb{Z}_q^n$ and a real α find a vector $\mathbf{s} \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{s}^T = \mathbf{v} \bmod q$ and $\|\mathbf{s}\| \leq \alpha$.*

Definition 4 ($\mathbf{SIVP}_\gamma$ problem). *Given an n dimensional lattice $\mathcal{L}$, find n linearly independent lattice vectors of length at most $\gamma \cdot \lambda_n(\mathcal{L})$.*

In [15], Gentry *et al.* proved that the worst-case hardness of SIVP is reduced to the average-case hardness of SIS or ISIS problem. Their result is in the following theorem.

Theorem 1 ([15]). *Let $m, \alpha = poly(n)$ and for any prime $q \geq \alpha \cdot \omega(\sqrt{n \log n})$. There is a probabilistic polynomial time reduction from solving $SIVP_\gamma$ for $\gamma = \alpha \cdot \tilde{O}(\sqrt{n})$ in the worst case to solving $SIS_{q,n,m,\alpha}$ and $ISIS_{q,n,m,\alpha}$ on average with non negligible probability.*

In [16], Kawachi, Tanaka and Xagawa introduced a lattice-based commitment scheme COM. The proposed scheme satisfies the essential security properties, namely, the statistically-hiding and computationally-binding properties. Let $\mathsf{COM}(\mu, \rho)$ be a commitment scheme.

- COM is said to be a statistically hiding scheme if for any messages μ_1, μ_2, any attacker cannot distinguish between $\mathsf{COM}(\mu_1, \rho_1)$ and $\mathsf{COM}(\mu_2, \rho_2)$.
- The computational binding property ensure that no polynomial time attacker can change the committed message to another one.

2.2 Ring Signature

In this subsection we review some definitions and properties about ring signatures that will be used in the following sections.

Ring Signature. We use the same definition as in [17]. A ring signature is a digital signature where the signer is not known, however his membership of a particular set can be verified.

Definition 5 (Ring Signature Scheme). *A ring signature scheme consists of the three following algorithms:*

- $\mathsf{R.KeyGen}$: *A probabilistic polynomial time algorithm that takes as input a security parameter and outputs a key pair formed by a public key PK and a secret key SK.*
- $\mathsf{R.Sign}$: *A probabilistic polynomial time algorithm that takes as input a set of public keys $PK_1, \ldots, PK_N$, a message μ, and a signing key. The output is a ring signature of the message μ.*
- $\mathsf{R.Verify}$: *A deterministic algorithm that takes as input a ring signature, a message μ and the list of public keys of the users of the ring. The output of this algorithm is* accept *if the ring signature is valid or* reject *otherwise.*

Threshold Ring Signature. In 2002, Bresson, Stern and Szydlo introduced the concept of threshold ring signatures [6]. In such a scheme, a set of t users wants to collaborate to produce a signature while preserving the anonymity of the participating signers. We give a formal definition of t-out-of-N threshold ring signature scheme. We denote the set of users by $U' = \{1, \ldots, N\}$ and by S' the set of signers with $S' \subset U'$. Each user i in U' has a public/secret key pair (PK_i, SK_i).

Definition 6 (Threshold Ring Signature Scheme). *A threshold ring signature scheme consists of the three following algorithms:*

- $\mathsf{T.KeyGen}$: *outputs a public key PK and a secret key SK.*
- $\mathsf{T.Sign}(\mu, U', S')$: *an interactive protocol between t users that take on input a set of public keys correponding to users in U' a set of t secret keys corresponding to users in S' and a message μ. The output is a t-out-of-N-threshold ring signature σ on μ.*
- $\mathsf{T.Verify}(\mu, \sigma, t, U')$: *deterministic algorithm that takes as input a value t, a set of public keys corresponding to users in U', a message-signature pair (μ, σ), and outputs* accept *or* reject.

A t-out-of-N threshold ring signature scheme is said to be secure if it is source hiding and unforgeable.

Anonymity. The source hiding definition described in [17] and [11] does not fit with threshold ring signature schemes obtained using the Fiat-Shamir transform. Indeed, the one-way functions used during the commitments do not allow to build two identical signatures for different set of signers. Moreover, this remains almost impossible even if two sets of signers have the same secret keys. We introduce a new definition generalised from the anonymity property for ring signatures see Definition 4 in [18]. With this definition we introduce the notion of indistinguishability.

Definition 7 (Indistinguishable source hiding). *Given a t-out-of-N threshold ring signature and a probabilistic polynomial time adversary $\mathcal{A}$, consider the following game*

1. *For $i = 1$ to N, generate (PK_i, SK_i). Give to $\mathcal{A}$ the set of public keys $P = \{PK_1, \ldots, PK_N\}$. The adversary $\mathcal{A}$ is also given access to a signing oracle $\mathsf{OT.Sign}(\cdot,\cdot,\cdot)$, that returns $\sigma = \mathsf{T.Sign}(\mu, P, S)$ on input (μ, P, S) with S a set signers.*
2. *$\mathcal{A}$ outputs a message μ, distinct sets $\{i_{1,0}, \ldots, i_{t,0}\}$, $\{i_{1,1}, \ldots, i_{t,1}\}$ and a ring P for which $PK_{i_{l,j}} \in P$ for $l \in \{0,1\}$ and $j \in \{1, \ldots, t\}$. Adversary $\mathcal{A}$ is given $\{SK_1, \ldots, SK_N\} \setminus \{SK_{i_{1,0}}, \ldots, SK_{i_{t,0}}\}$.*
   Furthermore, a random bit b is chosen and $\mathcal{A}$ is given $\sigma \leftarrow \mathsf{T.Sign}(\mu, P, \{SK_{i_{1,b}}, \ldots, SK_{i_{t,b}}\})$.
3. *The adversary outputs a bit b', and succeeds if $b' = b$.*

Unforgeability. We give a formal definition of existential unforgeability under chosen message attacks in the setting of t-out-of-N threshold ring signature. The definition is described using a game between an existential forger $\mathcal{F}$ and a challenger $\mathcal{C}$, and it is similar to one used in [17] and [11].

Definition 8 (Existensial Unforgeability). *A threshold ring signature scheme is existentially unforgeable under a chosen message attack if for any probabilistic polynomial time $\mathcal{F}$, the probability that $\mathcal{F}$ succeeds in the following game is negligible:*

1. *The challenger $\mathcal{C}$ generates key pairs $\{PK_i, SK_i\}_{i=1}^N$, and gives the set of public keys $P = \{PK_i\}_{i=1}^N$ to $\mathcal{F}$.*
2. *$\mathcal{F}$ is given access to a signing oracle as in Definition 7.*
3. *$\mathcal{F}$ is also given access to a key exposure oracle $\mathsf{OExp}(\cdot)$, that returns a secret key SK_i on input i.*
4. *$\mathcal{F}$ outputs a t-out-of-N threshold ring signature $\sigma^\star$ for a new message $\mu^\star$.*

The adversary $\mathcal{F}$ succeeds if the verification $\mathsf{T.Verify}(\mu^\star, \sigma^\star, t, P) = 1$, $\mu^\star$ has not been asked by $\mathcal{F}$ in a signing query in step 2 of the game and the number of key exposure queries is strictly less than t.

2.3 CLRS Identification Scheme

CLRS is an identification scheme proposed by Cayrel, Lindner, Rückert and Silva [12]. It is a lattice-based version of a generalisation of the Stern code-based identification scheme presented in [19].

Input: n, m, q
$\mathbf{x} \xleftarrow{\$} \{0,1\}^m$, with $wh(\mathbf{x}) = m/2$
$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$
$\mathbf{y} \leftarrow \mathbf{A}\mathbf{x}^T \bmod q$
$\mathsf{COM} \xleftarrow{\$} \mathcal{F}$, a suitable family of commitment functions.
Output: $(SK, PK) = (\mathbf{x}, (\mathbf{y}, \mathbf{A}, \mathsf{COM}))$

Fig. 1. Key generation algorithm

P chooses $\sigma \xleftarrow{\$} S_m$, $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^m$, $\mathbf{r}_0 \xleftarrow{\$} \{0,1\}^n$ and $\mathbf{r}_1 \xleftarrow{\$} \{0,1\}^n$.
P computes $c_0 \leftarrow \mathsf{COM}(\sigma||\mathbf{A}\mathbf{u}||\mathbf{r}_0)$ and $c_1 \leftarrow \mathsf{COM}(\sigma(\mathbf{u})||\sigma(\mathbf{x})||\mathbf{r}_1)$.

1. [**first commitment**] P sends c_0 and c_1 to V.
2. [**first challenge**] V sends $\alpha \xleftarrow{\$} \mathbb{Z}_q$ to P.
3. [**second commitment**] P sets $\beta = \sigma(\mathbf{u} + \alpha\mathbf{x})$ and sends β to V.
4. [**second challenge**] V sends $b \xleftarrow{\$} \{0,1\}$, to P.
5. [**final answer**]
   If $b = 0$ then
   P sends σ and $\mathbf{r}_0$ to V.
   If $b = 1$ then
   P sends $\sigma(\mathbf{x})$ and r_1 to V.

Verification:
If $b = 0$ then V checks if

$$c_0 \stackrel{?}{=} \mathsf{COM}(\sigma \| \mathbf{A}\sigma^{-1}(\beta)^T - \alpha\mathbf{y} \| \mathbf{r}_0)$$

If $b = 1$ then V checks, $wh(\sigma(\mathbf{x})) \stackrel{?}{=} m/2$ and

$$c_1 \stackrel{?}{=} \mathsf{COM}(\beta - \alpha\sigma(\mathbf{x}) \| \sigma(\mathbf{x}) \| \mathbf{r}_1)$$

Fig. 2. CLRS Identification protocol

The Stern protocol was introduced in 1993, it was the first efficient code-based identification scheme. It has the specific property to be zero-knowledge, indeed nobody can give any information about the involved secret from the transcripts of the interactions between a prover and a verifier. The zero-knowledge property allows to obtain a digital signature scheme by applying the Fiat-Shamir transform. Stern's scheme uses challenges and answers to establish the authentication of the user. There is only one challenge by round, we call it a three-pass protocol.

A generalisation of that scheme was presented in [20] by using two challenges for each round. Hence, the digital signatures obtained from this five-pass protocol have a smaller size. The CLRS identification scheme is a lattice-based version of this protocol.

We use a variation of the CLRS identification scheme in our construction, as well as it was done in [11].

The Scheme. In Figure 2, we have an identification between a prover P and a verifier V. The prover's secret key is a vector $\mathbf{x}$ of Hamming weight $m/2$. The public key $\mathbf{y}$ is related to $\mathbf{x}$ by the equation $\mathbf{y} = \mathbf{A}\mathbf{x}^T \bmod q$.

High Level Description. The protocol is made with challenges and answers. The verifier asks some challenges to the prover who needs to answer correctly. The challenges focus on the secret value $\mathbf{x}$. The vector $\mathbf{x}$ has an Hamming weight of $m/2$ and verify $\mathbf{A}\mathbf{x}^T = \mathbf{y}$. We can see that two different properties are required to identify the secret. It is a fundamental observation to understand the scheme. Indeed, the verifer will either ask to verify the Hamming weight of $\mathbf{x}$ or verify the relation $\mathbf{A}\mathbf{x}^T = \mathbf{y}$.

The masks used to protect the secret properties are from two different kinds. A permutation σ allows to mask which of the vectors of some given Hamming weight is used. The random vector $\mathbf{u}$ allows to mask which one of the vector verifying $\mathbf{A}\mathbf{x} = \mathbf{y}$ is used. After masking the secret twice (with each mask), the prover can unmask the masked secret with the permutation σ or the additional random vector to prove either that the secret is a vector of given Hamming weight or that it verify $\mathbf{A}\mathbf{x} = \mathbf{y}$.

We can identify 5 steps in the scheme: first commitment step, first challenge, second commitment, second challenge and final answer. The first commitment aims to set the random permutation and the random vector. The first challenge α is here to prevent replay attacks. The second commitment β is the secret masked two times. The second challenge b consists on asking to reveal either the weight of the secret ($b = 1$) or a way to compute $\mathbf{A}\mathbf{x}^T$ ($b = 0$).

This protocol is a probabilistic protocol in the fact that it is possible to anticipate the challenges and respond correctly without knowing the secret key. The soundness is close to $1/2$. Therefore, to reduce the cheating probability, the protocol has to be repeated many times.

2.4 Fiat-Shamir Transform

To build signature schemes from canonical identification schemes, one can use the Fiat-Shamir transform.

The challenges of the scheme are built with a random oracle H instead of an honest verifier. They are generated randomly at uniform from the message and commitments. In this way, a malicious prover cannot anticipate challenges one at a time and needs to anticipate them at once. The cheating probability remains the same as in the authentication protocol. Pointcheval and Stern [21] proved that an unforgeable signature scheme can be obtained when applying the Fiat-Shamir transform.

Recently Cayrel et al [22] proved that the Fiat-Shamir transform can be extended to fit with the $(2n+1)$-pass identification schemes (n is an integer in our case $n = 2$). We briefly review this transformation. In order to transform a five-pass identification scheme to a signature scheme, the signer proceeds as follows. He uses two random oracle $\mathcal{O}_1$ and $\mathcal{O}_2$ to generate the challenges C_1 and C_2 given by the verifier in the identification scheme. Let μ be a message, the signature σ of μ is formed by R_1, C_1, R_2, C_2 and RSP, where R_1, R_2 and RSP are the values calculated by the prover in the identification scheme.

3 Ring Signature

A ring signature is a digital signature where the signer is not known, however his membership of a particular set can be verified. In the following we explain the idea described in [10] and [11], then we show the differences with our scheme. In this section we consider U as a set of N users.

3.1 Description of CLRS Ring Signature Scheme

The ring signature scheme in [11] is obtained by applying the Fiat-Shamir transform to a ring identification scheme. The idea is to construct, for each user i, a secret key $\mathbf{x}_i$ with the same property $\mathbf{A}\mathbf{x}_i^T = 0$. Hence, we cannot distinguish two users from their public keys. The problem is to construct several such secret keys. In fact, statistically, there is only one vector $\mathbf{v}$ of small enough weight w such that $\mathbf{A}\mathbf{v}^T = 0$. The construction consists on changing the matrix $\mathbf{A}$ for each user. Now they are identified by their public matrix $\mathbf{A}_i$ instead of their public value $\mathbf{A}_i\mathbf{x}_i^T$. To generate a ring identification, N identifications are done, one for each user. The real signer uses his secret key $\mathbf{x}_i$ for only one identification and the null vector for the others. The anonymity is obtained with a permutation of the users' commitments.

3.2 Our Ring Signature Scheme

This scheme uses the same key generation algorithm as the CLRS scheme presented in section 2.3. On the other hand, a matrix $\mathbf{M}$ is added to the public values. The matrix $\mathbf{M}$ is composed of all public elements $\mathbf{y}_1, \ldots, \mathbf{y}_N$ with $\mathbf{A}\mathbf{x_i}^T = \mathbf{y}_i$ for $i \in \{1, \ldots, N\}$.

The ring signature scheme can be obtained from the ring identification scheme in Figure 3 by applying the Fiat-Shamir transform. This scheme is very similar to the CLRS identification scheme described in section 2.3. As it is detailed in Figure 3, a new commitment β' and two masks $\mathbf{u}'$ and Σ were added as well as a secret value δ_j. Those new elements were designed to guarantee the anonymity of the real signer. The idea is to use δ_j as the secret identifying the real signer and to mask it in the same way as the secret key $\mathbf{x}_i$.

In this scheme, the $\mathbf{y}_j$ which defines the signer and which is used to compute the first commitment is masked in β'. Indeed, we have $\mathbf{y}_j = \mathbf{M}\delta_j^T$ with δ_j masked by Σ and $\mathbf{u}'$. The first commitment can be computed in the same way for each signer as far as $\mathbf{A}(\mathbf{x}_i + \mathbf{u})^T - \mathbf{M}(\delta_i + \mathbf{u}')^T$ is equal for each signer i. We use the same construction to mask $\mathbf{x}_i$ and δ_i because both values are identified by their weight and their image by a particular matrix ($\mathbf{A}$ or $\mathbf{M}$).

The fact that $wh(\Sigma(\delta_j)) = 1$ with Σ a permutation guarantees that one user in the ring U' is the real signer.

3.3 Features of the Scheme

This scheme, without β', u' and Σ, is the same as the identification scheme CLRS. We do not need to send N identifications to obtain anonymity like it was done in previous ring signature schemes. Therefore, a significant reduction of signature size is obtained for reasonable values of N and t.
An other notable property is the use of a unique random matrix $\mathbf{A}$ instead of N public matrices $\mathbf{A}_i$ in [11]. A consequence is a reduction of the size of the public keys.

4 Threshold Ring Signature

The threshold ring signature is a generalisation of the ring signature. The threshold ring signature is made by many signers instead of only one for the ring signature.

The authors of [6] and [10] claim that a threshold ring signature is not a repetition of several ring signatures. The reason is that we have to prove that at least t signer has been involved in the process. For example, a signer alone should not be able to sign twice in the same signature and produce a valid signature.
In our case, the threshold ring signature is more like t different ring signatures. That is why the signature size is close to t signatures instead of N for other schemes [10, 11].

4.1 From Ring Signature to Threshold Ring Signature

In this subsection we explain how to prove that t different users can make a threshold ring signature with our scheme. The threshold ring signature scheme is obtained by applying the Fiat-Shamir transform on the threshold ring identification scheme given in Figure 4. The idea is very simple, each δ_i represents a different user, which can be identified with the matrix $\mathbf{M}$. We only have to show that the δ_i, used in the identification, are different. We prove that point by verifying that the $\Sigma(\delta_i)$ are different because if $\Sigma(\delta_i) \neq \Sigma(\delta_j)$ then $\delta_i \neq \delta_j$, with Σ a permutation. From security perspective, Σ is known by all the signers, and this does not affect the unforgeability because Σ only masks δ_i which defines the identity of the signer. Moreover, no information about $\mathbf{x}_i$ is revealed. The

Let $U' = \{\text{users}\}$. Let $\mathbf{M}$ the matrix of all public elements $\mathbf{y}_1, \ldots, \mathbf{y}_N$ of users in U'.
$\mathbf{M} = \begin{pmatrix} | & | & \cdots & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_N \\ | & | & \cdots & | \end{pmatrix}$, with $\mathbf{y}_i = \mathbf{A}\mathbf{x}_i^T$ for all $i \in \{1, \ldots, N\}$.

The user S with index j in $\{1, \ldots, N\}$, do the following:
He sets $\delta_j \in \{0,1\}^N$ with 1 in the j-th position and 0 elsewhere. He chooses a random permutation Σ of $\{1, \ldots, N\}$, $\mathbf{u}' \xleftarrow{\$} \mathbb{Z}_q^N, \sigma \xleftarrow{\$} S_m$, $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^m$, $\mathbf{r}_0 \xleftarrow{\$} \{0,1\}^n$ and $\mathbf{r}_1 \xleftarrow{\$} \{0,1\}^n$.
He computes

$$c_0 \leftarrow \mathsf{COM}(\sigma||\Sigma||\mathbf{A}\mathbf{u}^T - \mathbf{M}\mathbf{u}'^T||\mathbf{r}_0) \text{ and } c_1 \leftarrow \mathsf{COM}(\sigma(\mathbf{u})||\Sigma(\mathbf{u}')||\sigma(\mathbf{x}_j)||\Sigma(\delta_j)\|\mathbf{r}_1).$$

1. [**first commitment**] S sends c_0 and c_1 to the verifier V.
2. [**first challenge**] V sends $\alpha \xleftarrow{\$} \mathbb{Z}_q$ to S.
3. [**second commitment**] S sends β and β' to V, with
$$\beta = \sigma(\mathbf{u} + \alpha\mathbf{x}_j) \text{ and } \beta' = \Sigma(\mathbf{u}' + \alpha\delta_j).$$
4. [**second challenge**] V sends $b \xleftarrow{\$} \{0,1\}$ to S.
5. [**final answer**]
   If $b = 0$ then
   S sends $\phi = \Sigma, \psi = \sigma$, and $\mathbf{a} = \mathbf{r}_0$ to V.
   If $b = 1$ then
   S sends $\chi = \Sigma(\delta_j), \mathbf{d} = \sigma(\mathbf{x}_j)$ and $\mathbf{e} = \mathbf{r}_1$ to V.

Verification :

If $b = 0$ then V checks if
$$c_0 \stackrel{?}{=} \mathsf{COM}(\psi\|\phi\|\mathbf{A}\psi^{-1}(\beta)^T - \mathbf{M}\phi^{-1}(\beta')^T\|\mathbf{a})$$
If $b = 1$ then V checks if
$$c_1 \stackrel{?}{=} \mathsf{COM}((\beta - \alpha\mathbf{d})\|(\beta' - \alpha\chi)\|\mathbf{d}\|\chi\|\mathbf{e})$$
$$\mathbf{d} \stackrel{?}{\in} \{0,1\}^m \text{ , } wh(\mathbf{d}) \stackrel{?}{=} m/2 \text{ and } wh(\chi) \stackrel{?}{=} 1.$$

Fig. 3. Ring identification scheme

anonymity property remains unchanged because signers are supposed to know each other when they produce the signature.

During the verification, when $b = 0$, we have just to verify that each $\Sigma(\delta_i)$ is different from the others. Thus, this proves that t different signers have cooperated to generate the threshold ring signature.

Let $\mathbf{M}$ the matrix of all public elements $\mathbf{y}_1, \ldots, \mathbf{y}_N$ of users in U'.
$\mathbf{M} = \begin{pmatrix} | & | & \cdots & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_N \\ | & | & \cdots & | \end{pmatrix}$, with $\mathbf{y}_i = \mathbf{A}\mathbf{x}_i^T$ for all $i \in \{1, \ldots, N\}$.
$\delta_i \in \{0,1\}^N$ the vector with 1 on the i-th position and 0 elsewhere else.
$U' = \{\text{users}\}$ and $S' = \{\text{signers}\}$, with $S' \subset U'$, $|S'| = t$ and $|U'| = N$.
L the leader, with $L \in S'$ and V the verifier.

1. [**first commitment**]

   L construct at random Σ a permutation of $\{1, \ldots, N\}$.

   for i from 1 to t do
   L choose the i-th signer $\mathcal{S}_j$
   S_j receives Σ from L.
   S_j constructs $\sigma_i \xleftarrow{\$} S_m$, $\mathbf{u}_i \xleftarrow{\$} \mathbb{Z}_q^m$, $\mathbf{u}'_i \xleftarrow{\$} \mathbb{Z}_q^N$, $\mathbf{r}_{0,i} \xleftarrow{\$} \{0,1\}^n$ and $\mathbf{r}_{1,i} \xleftarrow{\$} \{0,1\}^n$
   S_j computes : $c_{0,i} \leftarrow \mathsf{COM}(\sigma_i || \Sigma \| \mathbf{A}\mathbf{u}_i^T - \mathbf{M}\mathbf{u}_i'^T || \mathbf{r}_{0,i})$
   $c_{1,i} \leftarrow \mathsf{COM}(\sigma_i(\mathbf{u}_i) || \Sigma(\mathbf{u}'_i) \| \sigma_i(\mathbf{x}_j) || \Sigma(\delta_j) \| \mathbf{r}_{1,i})$
   S_j sends $c_{0,i}$ and $c_{1,i}$ to L.
   end for

   L computes $\mathbf{r}_0 \xleftarrow{\$} \{0,1\}^n$ and $\mathbf{r}_1 \xleftarrow{\$} \{0,1\}^n$.
   L computes the master commitments :
   $C_0 = \mathsf{COM}(c_{0,1} || \ldots || c_{0,t} || \mathbf{r}_0)$ and $C_1 = \mathsf{COM}(c_{1,1} || \ldots || c_{1,t} \| \mathbf{r}_1)$

   L sends C_0 and C_1 to V.

2. [**first challenge**] V sends α, such as $\alpha \xleftarrow{\$} \mathbb{Z}_q$.

3. [**second commitment**]

   We denote $\bar{\mathbf{x}}_i$ (respectively $\bar{\delta}_i$) the $\mathbf{x}_j$ (respectively δ_j) corresponding to the i-th signer $\mathcal{S}_j$.
   for i from 1 to t do
   L choose the i-th signer $\mathcal{S}_j$
   S_j computes $\beta_i \leftarrow \sigma_i(\mathbf{u}_i + \alpha \bar{\mathbf{x}}_i)$
   S_j computes $\beta'_i \leftarrow \Sigma(\mathbf{u}'_i + \alpha \bar{\delta}_i)$
   S_j sends β_i, β'_i to L.
   end for

   L sends $\mathbf{v}_1 = \beta_1, \ldots, \mathbf{v}_t = \beta_t$ and $\mathbf{w}_1 = \beta'_1, \ldots, \mathbf{w}_t = \beta'_t$ to V.

4. [**second challenge**] V sends $b = 0$ or $b = 1$ to L.

5. [**final answer**]

   if $b = 0$ then
   for i from 1 to t do
   L choose the i-th signer $\mathcal{S}_j$
   S_j sends $\sigma_i, \mathbf{r}_{0,i}$ to L
   end for
   if $b = 1$ then
   for i from 1 to t do
   L choose the i-th signer $\mathcal{S}_j$
   S_j sends $\mathbf{u}_i, \mathbf{u}'_i$ and $\mathbf{r}_{1,i}$ to L
   end for

   if $b = 0$ then
   L sends $\phi = \Sigma$, $\psi_1 = \sigma_1, \ldots, \psi_t = \sigma_t$, $\mathbf{a}_1 = \mathbf{r}_{0,1}, \ldots, \mathbf{a}_t = \mathbf{r}_{0,t}$ and $\rho = \mathbf{r}_0$ to V.
   if $b = 1$ then
   L sends $\chi_1 = \Sigma(\bar{\delta}_1), \ldots \chi_t = \Sigma(\bar{\delta}_t)$, $\mathbf{d}_1 = \sigma_1(\bar{\mathbf{x}}_1), \ldots, \mathbf{d}_t = \sigma_t(\bar{\mathbf{x}}_t)$, $\mathbf{e}_1 = \mathbf{r}_{1,1}, \ldots, \mathbf{e}_t = \mathbf{r}_{1,t}$ and $\varrho = \mathbf{r}_1$ to V.

Fig. 4. Threshold ring identification scheme

if $b = 0$ then
  for i from 1 to t
    Set $c_i = \mathsf{COM}(\psi_i \| \phi \| \mathbf{A}\psi_i^{-1}(\mathbf{v}_i)^T - \mathbf{M}\phi^{-1}(\mathbf{w}_i)^T \| \mathbf{a}_i)$
  end for
  V checks that $C_0 \stackrel{?}{=} \mathsf{COM}(c_1 \| \dots \| c_t \| \rho)$
if $b = 1$ then
  for i from 1 to t
    Set $c_i = \mathsf{COM}(\mathbf{v}_i - \alpha\mathbf{d}_i \| \mathbf{w}_i - \alpha\chi_i \| \mathbf{d}_i \| \chi_i \| \mathbf{e}_i)$

    V checks that $wh(\mathbf{d}_i) = m/2$ and $\mathbf{d}_i \in \{0,1\}^m$

    V checks that $wh(\chi_i) = 1$
  end for
  V checks that $\sum_{i=1}^{t} wh(\chi_i) = t$ and $\chi_i \in \{0,1\}^N$
  V checks that $C_1 \stackrel{?}{=} \mathsf{COM}(c_1 \| \dots \| c_t \| \varrho)$

Fig. 5. Verification algorithm

4.2 Lattice-Based Threshold Identification Scheme

In this subsection we present our threshold identification scheme. The scheme is in Figure 4, the verification algorithm is given in Figure 5.

5 Security Analysis

In this section we prove the security of our scheme. A threshold ring signature scheme must satisfy two properties. The source hiding, which ensures the anonymity of the users and the existential unforgeability, which is common to every digital signature.

5.1 Source Hiding

Lemma 1. *For any transcript τ of the threshold identification scheme in Figure 4 performed by a set S of t signers we have that:*

For any set S' of t signers there exists a transcript σ' performed by S' such that the differences between τ and τ' are in the commitment values.

Proof. We will prove the lemma for one round of the threshold identification scheme. Let τ be the transcript of the threshold identification, it is easy to see that τ consists of $((C_0, C_1), \alpha, (\beta_i, \beta_i'; i \in \{1, \dots, t\}), b, RSP)$ where RSP is the response sent by the leader L to the verifier V.

Let $(\mathbf{x}_i, \delta_i; i \in \{1, \dots, t\})$, be the secrets used by the signers in S and $(\mathbf{x}_i', \delta_i'; i \in \{1, \dots, t\})$, be the secrets used by the signers in S'. We will show how to choose $(\mathbf{u}_i, \mathbf{u}_i', \sigma_i, \Sigma)$ such that the transcript does not change except C_0 and C_1 when the secret keys $(\mathbf{x}_i, \delta_i; i \in \{1, \dots, t\})$ are replaced by $(\mathbf{x}_i', \delta_i'; i \in \{1, \dots, t\})$.

If $b = 0$, we set $U_i = \mathbf{u}_i + \alpha\mathbf{x}_i - \alpha\mathbf{x}'_i$ and $U'_i = \mathbf{u}'_i + \alpha\delta_i - \alpha\delta'_i$. Therefore, when we replace $(\mathbf{x}_i, \mathbf{u}_i, \mathbf{u}'_i, \delta_i; i \in \{1, \ldots, t\})$ by $(\mathbf{x}'_i, U_i, U'_i, \delta'_i; i \in \{1, \ldots, t\})$, we obtain that β_i and β'_i keep the same values. Since $\mathbf{u}_i$ and $\mathbf{u}'_i$ do not appears in RSP, then the only thing that change in the transcript are the values C_0 and C_1.

If $b = 1$, we choose π_i and Π such that $\pi_i(\mathbf{x}'_i) = \mathbf{x}_i$ and $\Pi(\delta'_i) = \delta_i$ and we set $U_i = \pi^{-1}(\mathbf{u}_i)$ and $U'_i = \Pi^{-1}(\mathbf{u}'_i)$. Therefore, when we replace $(\sigma_i, \mathbf{u}_i, \Sigma, \mathbf{u}'_i)$ by $(\sigma_i \circ \pi_i, U_i, \Sigma \circ \Pi, U'_i)$, we obtain that β_i and β'_i keep the same values as well as in RSP. Then, the only thing that change in the transcript are the values C_0 and C_1.

Applying this construction for all the rounds finishes the proof.

Lemma 2. *Let n, q be the parameters of the commitment scheme* COM *and rd the number of rounds of the threshold identification scheme. For any message μ and any threshold signature σ generated by a set S of t signers, we have : For any set S' of t signers there exists a signature σ' with probability*

$$p = 1 - \left(1 - \frac{1}{(2q)^{rd}}\right)^{2^{n \times rd}}$$

performed by S' such that the difference between σ and σ' are in the commitment values.

Proof. Using the Fiat-Shamir transform we can see the threshold signature as $(R_1, CH_1, R_2, CH_2, RSP)$ with R_1 is the concatenation of C_0 and C_1 of all the rounds, R_2 is the concatenation of all β_i and β'_i of all the rounds and RSP is the concatenation of final answers of all the rounds. The values of CH_1 and CH_2 are computed by two random oracle $\mathcal{O}_1$ and $\mathcal{O}_2$ such that $CH_1 = \mathcal{O}_1(\mu, R_1)$ with μ the message and $CH_2 = \mathcal{O}_2(\mu, CH_1, R_2)$. The Lemma 1 states the existence of another transcript equal to the signature except the values of C_0 and C_1. Those commitments are computed using the commitment scheme COM with random values $\mathbf{r}_0$ and $\mathbf{r}_1$. For each round of the transcript we can take an other $\mathbf{r}_0$ or $\mathbf{r}_1$ and we get another transcript equal to the signature except the values of C_0 and C_1. To finish the proof we compute the probability that one of those transcripts corresponds to a signature of μ. Most of the time the transcript is not a signature because the challenges are different than $\mathcal{O}_1(\mu, R_1)$ and $\mathcal{O}_2(\mu, CH_1, R_2)$. The probability that the transcript corresponds to a signature of μ can be computed by a Bernoulli distribution with parameters $p_B = \frac{1}{(2q)^{rd}}$ and $n_B = 2^{n \times rd}$. The parameter p_B corresponds to the probability to obtain a particular challenge and n_B correspond to the number of transcript. Then, the probability to obtain a signature is

$$1 - \left(1 - \frac{1}{(2q)^{rd}}\right)^{2^{n \times rd}}.$$

Theorem 2. *If there is a probabilistic polynomial time attacker $\mathcal{A}$ that can win the game of source hiding, then $\mathcal{A}$ can break the statistically hiding property of the commitment scheme* COM.

Proof. Let S_0 and S_1 be two sets of t signers. In the game described in Definition 7, the attacker is given as a challenge a threshold ring signature generated by S_b (with $b \in \{0,1\}$) and he has to guess with non negligible probability b' such that $b' = b$.

We consider that the attacker has access to the set of all the secret keys and chooses two sets of t signers S_0 and S_1. Then, he chooses a message μ and requests a challenge. After receiving the threshold ring signature σ of μ signed under S_b, the attacker has to guess the set which generated it.

By lemma 2, we obtain that there exists a signature σ' generated by $S_{\bar{b}}$ with probability p such that the difference between σ and σ' are in the commitments calculated by COM. In the parameters of the commitment scheme COM, q is polynomial on n, therefore the probability p is not negligible. If the attacker $\mathcal{A}$ wins the source hiding game, then $\mathcal{A}$ can distinguish between two outputs of the commitment scheme COM with non negligible probability.

5.2 Unforgeability

In this subsection we prove that if a forger can win the game in Definition 8, then a polynomial time algorithm can be built to solve a standard lattice problem. The proof of unforgeability is given in Theorem 3. We start by giving the following lemma which is used in the proof of unforgeability.

Lemma 3. *If there exist a probabilistic polynomial time algorithm $\mathcal{A}$ that is able to produce a t-out-of-N threshold signature scheme with probability greater than $\left(\frac{q+2}{2q}\right)^{rd}$, than either he can produce t values which can be used as secret keys or he can find a collision in the commitment scheme COM.*

Proof. If $\mathcal{A}$ produces a t-out-of-N threshold signature with probability greater than $\left(\frac{q+2}{2q}\right)^{rd}$, then he succeeds, in the corresponding threshold identification scheme, in at least one round with probability greater than $\frac{q+2}{2q}$. In each round there are $2q$ possible challenges, q possibilities for the first challenge step and 2 possibilities for the second challenge step. By the pigeon-hole principle we deduce that $\mathcal{A}$ can answer correctly with a particular commitment for two different challenges α, β in the first challenge step and any possible challenge in the second challenge step. Therefore, he can build the following transcript:

$$((C_0, C_1), \alpha, (\mathbf{v}_i, \mathbf{w}_i), 0, (\phi, \psi_i, \mathbf{a}_i, \rho)), \quad i \in \{1, \ldots, t\}$$

$$((C_0, C_1), \alpha, (\mathbf{v}_i, \mathbf{w}_i), 1, (\chi_i, \mathbf{d}_i, \mathbf{e}_i, \varrho)), \quad i \in \{1, \ldots, t\}$$

$$((C_0, C_1), \beta, (\mathbf{v}'_i, \mathbf{w}'_i), 0, (\phi', \psi'_i, \mathbf{a}'_i, \rho')), \quad i \in \{1, \ldots, t\}$$

$$((C_0, C_1), \beta, (\mathbf{v}'_i, \mathbf{w}'_i), 1, (\chi_i, \mathbf{d}_i, \mathbf{e}_i, \varrho)), \quad i \in \{1, \ldots, t\}$$

such that all those transcript succeeds in the verification protocol for that round. We have that either $\mathcal{A}$ can find a collision in COM or we obtain the following equations from the verification protocol.

1. $\psi_i = \psi'_i$, $\phi = \phi'$, $\mathbf{A}\psi_i^{-1}(\mathbf{v}_i)^T - \mathbf{M}\phi^{-1}(\mathbf{w}_i)^T = \mathbf{A}\psi_i'^{-1}(\mathbf{v}'_i)^T - \mathbf{M}\phi'^{-1}(\mathbf{w}'_i)^T$, $\mathbf{a}_i = \mathbf{a}'_i$ for $i \in \{1, \ldots, t\}$.
2. $\mathbf{v}_i - \alpha\mathbf{d}_i = \mathbf{v}'_i - \beta\mathbf{d}'_i$, $\mathbf{w}_i - \alpha\chi_i = \mathbf{w}'_i - \beta\chi'_i$, $\mathbf{d}_i = \mathbf{d}'_i$, $\chi_i = \chi'_i$, $\mathbf{e}_i = \mathbf{e}'_i$ for $i \in \{1, \ldots, t\}$.
3. $wh(\mathbf{d}_i) = wh(\mathbf{d}'_i) = m/2$, $\mathbf{d}_i \in \{0,1\}^m$, $\mathbf{d}'_i \in \{0,1\}^m$ for $i \in \{1, \ldots, t\}$.
4. $\sum_{i=1}^{t} wh(\chi_i) = t$, $\sum_{i=1}^{t} wh(\chi'_i) = t$, $\chi_i \in \{0,1\}^N$, $\chi'_i \in \{0,1\}^N$ for $i \in \{1, \ldots, t\}$.

From the equations in 2, we have that $\mathbf{v}'_i = \mathbf{v}_i - \alpha\mathbf{d}_i + \beta\mathbf{d}'_i$ and $\mathbf{w}'_i = \mathbf{w}_i - \alpha\chi_i + \beta\chi'_i$. When we replace $\mathbf{v}'_i$ and $\mathbf{w}'_i$ in the third equation in 1, we obtain:

$$\mathbf{A}\psi_i^{-1}(\mathbf{v}_i)^T - \mathbf{M}\phi^{-1}(\mathbf{w}_i)^T = \mathbf{A}\psi_i^{-1}(\mathbf{v}_i - \alpha\mathbf{d}_i + \beta\mathbf{d}_i)^T - \mathbf{M}\phi^{-1}(\mathbf{w}_i - \alpha\chi_i + \beta\chi_i)^T$$

$$0 = (\beta - \alpha)(\mathbf{A}\psi_i^{-1}(\mathbf{d}_i)^T - \mathbf{M}\phi^{-1}(\chi_i)^T)$$

Since $\alpha \neq \beta$ we have that $\mathbf{A}\psi_i^{-1}(\mathbf{d}_i)^T = \mathbf{M}\phi^{-1}(\chi_i)^T$. From the equation in 4, we have that $\mathbf{M}\phi^{-1}(\chi_i)^T$ correspond to t different public keys and $\psi_i^{-1}(\mathbf{d}_i)$ can be used to simulate t different secret keys. □

In the following theorem we prove the unforgeability of the threshold signature scheme.

Theorem 3 (Unforgeability). *If a forger wins the game in Definition 8 with probability p' in polynomial time, then the forger can solve an $ISIS_{q,n,m,\sqrt{m}}$ instance in polynomial time with probability $p'p\frac{1}{N^2}$ or find a collision in the commitment scheme* COM.

Proof. The challenger $\mathcal{C}$ is given an ISIS instance $(\mathbf{A}, \mathbf{y})$. Then, $\mathcal{C}$ chooses $k \in \{1 \ldots N\}$ and sets $\mathbf{x}_k := 0$, $\mathbf{y}_k := \mathbf{y}$. $\mathcal{C}$ generates $N - 1$ keys $(\mathbf{x}_i, \mathbf{y}_i)$ with $i \in \{1 \ldots N\}$ and $i \neq k$.

We start the game in Definition 8 with the forger $\mathcal{F}$ and the pairs $(\mathbf{x}_i, \mathbf{y}_i)$ with $i \in \{1 \ldots N\}$, as key pairs. We notice that only the key pair $(\mathbf{x}_k, \mathbf{y}_k)$ is not generated as a valid key pair and since $\mathbf{x}_k$ is not revealed, the set of key pairs $(\mathbf{x}_i, \mathbf{y}_i)$ is indistinguishable from a valid set of key pairs.

The challenger $\mathcal{C}$ simulates the signing oracle OT.Sign. When $\mathcal{F}$ requests the signing oracle, he send a query to the challenger $\mathcal{C}$ involving the public keys $\mathbf{y}_i$, $i \in I$ and $I \subseteq N$. If k is not in the set I, the challenger $\mathcal{C}$ is able to produce the corresponding signature. Otherwise the challenger produces a signature σ by replacing $\mathbf{x}_k$ by $\mathbf{x}_j$ with $j \notin I$. By Lemma 2, with probability p there exist a signature σ' such that σ' could be produced using the secret key corresponding to $\mathbf{y}_i$.

The challenger $\mathcal{C}$ also simulates the oracle OExp. If the forger $\mathcal{F}$ asks for the value of $\mathbf{x}_k$, the game ends without a forged signature. Since the forger do not ask for all the secret keys, the probability that he ask for $\mathbf{x}_k$ is less than $\frac{N-1}{N}$. Otherwise, the forger $\mathcal{F}$ wins the game and outputs a valid signature in polynomial time with probability p'.

According to Lemma 3, $\mathcal{F}$ can either simulate t different secret keys or find a collision on the commitment scheme COM. If he succeeds to simulate t secret

keys, at least one of them, $\mathbf{x}_l$, was not a query for the key exposure oracle. The vector $\mathbf{x}_l$ is such that $wh(\mathbf{x}_l) = m/2$ and $\mathbf{A}\mathbf{x}_l^T = \mathbf{y}_l$. Since k was chosen randomly at uniform in $\{1, \ldots, N\}$, then the probability that k is equal to l is $1/N$. If l is equal to k, then $\mathbf{x}_l$ is a solution of the $\text{ISIS}_{q,n,m,\sqrt{m}}$ instance $(\mathbf{A}, \mathbf{y})$.

With the interaction of the challenger and the forger, we can build a probabilistic polynomial time algorithm that can solve an $\text{ISIS}_{q,n,m,\sqrt{m}}$ instance with probability greater than $pp' \frac{1}{N^2}$. □

It was shown in [16] that the security of the commitment scheme COM is based on the average hardness of $\text{ISIS}_{q,n,m,\sqrt{m}}$. By Theorem 1 we have that there exist average-case/worst-case reduction from SIVP_γ to $\text{ISIS}_{q,n,m,\alpha}$ with an approximation factor $\gamma = \alpha \cdot \tilde{O}(\sqrt{n})$. Therefore, we have that for a prime $q = \tilde{O}(n)$, $m = O(n \log q)$ and $\alpha = \sqrt{m}$, the unforageability of our threshold ring signature scheme is based on $\text{SIVP}_{\tilde{O}(n)}$.

6 Parameters

In this section we compare our threshold ring signature and our ring signature to the CLRS ring and threshold ring signature in [11]. The comparison is only made with this scheme because others schemes in [13, 14] don't give instantiation parameters.

6.1 Parameters Assumption

We use the same parameters used in [11] (subsection 5.1) to compare the performance of the schemes. The parameters are $n = 64, m = 2048, q = 257$ and the length of the commitment of COM is 224 bits for bit-security equal to 111. To compute the size of the signature, we use a seed to represent each random permutation σ_i and Σ. In fact, the signer only has to send a seed from which the verifier can obtain the desired element and thus reduce the communication cost. The vector $\sigma_i(\mathbf{x}_i)$ is a vector in $\{0, 1\}^m$, its length is so m-bits instead of $\log_2 q \times m$ bits.

6.2 Ring Signature

In the following table we can see the significant reduction obtained with our ring signature scheme. Our scheme stay reasonable even for huge size of ring. The number of member of the ring is given by N.

Table 1. Comparison of lattice-based ring signature schemes in Mbytes

N	100	1000	5000	10000	100000
CLRS ring	24.43	244.24	1221.21	2442.42	24424.20
Scheme in Figure 3	0.26	0.37	0.84	1.43	12.05

6.3 Threshold Ring Signature

In the following table we can see different signature sizes for some values of t and N.

Table 2. Comparison of lattice-based threshold ring signature schemes in Mbytes

N	100	100	100	100	100	100	200	200	200	1000	1000	1000
t	2	10	30	50	70	100	2	10	50	2	10	50
CLRS threshold ring	24.43	24.43	24.43	24.43	24.43	24.43	48.85	48.85	48.85	244.24	244.24	244.24
Scheme in Figure 4	0.52	2.56	7.68	12.80	17.92	25.60	0.54	2.68	13.39	0.73	3.63	18.11

We see that our threshold scheme has a size close to t ring signatures and do not depend so much on the parameter N for the given parameters.

Acknowledgement. We deeply thank one of the anonymous reviewer of the conference which comments and suggestions were very helpful.

References

1. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261(4), 515–534 (1982)
2. Coppersmith, D.: Finding small solutions to small degree polynomials. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 20–31. Springer, Heidelberg (2001)
3. Ajtai, M.: Generating hard instances of lattice problems. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing., pp. 99–108. ACM (1996)
4. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
5. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001)
6. Bresson, E., Stern, J., Szydlo, M.: Threshold ring signatures and applications to ad-hoc groups. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 465–480. Springer, Heidelberg (2002)
7. Liu, J.K., Wei, V.K., Wong, D.S.: A separable threshold ring signature scheme. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 12–26. Springer, Heidelberg (2004)
8. Dallot, L., Vergnaud, D.: Provably secure code-based threshold ring signatures. In: Parker, M.G. (ed.) Cryptography and Coding 2009. LNCS, vol. 5921, pp. 222–235. Springer, Heidelberg (2009)
9. Zheng, D., Li, X., Chen, K.: Code-based ring signature scheme. IJ Network Security 5(2), 154–157 (2007)
10. Aguilar Melchor, C., Cayrel, P.-L., Gaborit, P.: A new efficient threshold ring signature scheme based on coding theory. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 1–16. Springer, Heidelberg (2008)

11. Cayrel, P.-L., Lindner, R., Rückert, M., Silva, R.: A lattice-based threshold ring signature scheme. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 255–272. Springer, Heidelberg (2010)
12. Cayrel, P.-L., Lindner, R., Rückert, M., Silva, R.: Improved zero-knowledge identification with lattices. In: Heng, S.-H., Kurosawa, K. (eds.) ProvSec 2010. LNCS, vol. 6402, pp. 1–17. Springer, Heidelberg (2010)
13. Wang, J., Sun, B.: Ring signature schemes from lattice basis delegation. In: Qing, S., Susilo, W., Wang, G., Liu, D. (eds.) ICICS 2011. LNCS, vol. 7043, pp. 15–28. Springer, Heidelberg (2011)
14. Brakerski, Z., Kalai, Y.: A framework for efficient signatures, ring signatures and identity based encryption in the standard model. Technical report, Cryptology ePrint Archive, Report 2010/086 (2010)
15. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, pp. 197–206. ACM (2008)
16. Kawachi, A., Tanaka, K., Xagawa, K.: Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 372–389. Springer, Heidelberg (2008)
17. Aguilar Melchor, C., Cayrel, P., Gaborit, P., Laguillaumie, F.: A new efficient threshold ring signature scheme based on coding theory. IEEE Transactions on Information Theory 57(7), 4833–4842 (2011)
18. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 60–79. Springer, Heidelberg (2006)
19. Stern, J.: A new identification scheme based on syndrome decoding. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 13–21. Springer, Heidelberg (1994)
20. Cayrel, P.L., Veron, P.: Improved code-based identification scheme. arXiv preprint arXiv:1001.3017 (2010)
21. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996)
22. El Yousfi Alaoui, S.M., Dagdelen, Ö., Véron, P., Galindo, D., Cayrel, P.-L.: Extended security arguments for signature schemes. In: Mitrokotsa, A., Vaudenay, S. (eds.) AFRICACRYPT 2012. LNCS, vol. 7374, pp. 19–34. Springer, Heidelberg (2012)

Degree of Regularity for HFEv and HFEv-

Jintai Ding[1,2,⋆] and Bo-Yin Yang[3,⋆⋆]

[1] University of Cincinnati, Cincinnati OH, USA
[2] Chongqing University, China,
jintai.ding@gmail.com
[3] Academia Sinica, Taipei, Taiwan,
by@crypto.tw

Abstract. In this paper, we first prove an explicit formula which bounds the degree of regularity of the family of HFEv ("HFE with vinegar") and HFEv- ("HFE with vinegar and minus") multivariate public key cryptosystems over a finite field of size q. The degree of regularity of the polynomial system derived from an HFEv- system is less than or equal to

$$\begin{array}{c} \frac{(q-1)(r+v+a-1)}{2}+2 \text{ if } q \text{ is even and } r+a \text{ is odd,} \\ \frac{(q-1)(r+v+a)}{2}+2 \text{ otherwise,} \end{array}$$

where the parameters v, D, q, and a are parameters of the cryptosystem denoting respectively the number of vinegar variables, the degree of the HFE polynomial, the base field size, and the number of removed equations, and r is the "rank" paramter which in the general case is determined by D and q as $r = \lfloor \log_q(D-1) \rfloor + 1$. In particular, setting $a = 0$ gives us the case of HFEv where the degree of regularity is bound by

$$\begin{array}{c} \frac{(q-1)(r+v-1)}{2}+2 \text{ if } q \text{ is even and } r \text{ is odd,} \\ \frac{(q-1)(r+v)}{2}+2 \text{ otherwise.} \end{array}$$

This formula provides the first solid theoretical estimate of the complexity of algebraic cryptanalysis of the HFEv- signature scheme, and as a corollary bounds on the complexity of a direct attack against the QUARTZ digital signature scheme. Based on some experimental evidence, we evaluate the complexity of solving QUARTZ directly using F_4/F_5 or similar Gröbner methods to be around 2^{92}.

Keywords: Degree of regularity, HFE, HFEv, HFEv-.

⋆ Partially sponsored by National Science Foundation of China, Grant #60973131 and U1135004, and the Charles Phelps Taft Research Center.

⋆⋆ Partially sponsored by Taiwan's National Science Council project #100-2628-E-001-004-MY3 and 101-2915-I-001-019.

P. Gaborit (Ed.): PQCrypto 2013, LNCS 7932, pp. 52–66, 2013.

1 Introduction

1.1 Questions

HFE (Hidden Field Equations) and its derivatives form one of the best known families of multivariate quadratic public-key cryptosystems. It was invented by Patarin as a modification of the Matsumoto-Imai cryptosystem C^* in 1997.

Shor's algorithm from 1994 and its extensions [30,34] will break RSA and ECC when large quantum computers became available. In this context, multivariate PKCs and in particular HFE [28] had been viewed as a possible candidate to replace RSA. Although it was shown by Faugère and Joux [19] that the basic form can be cryptanalyzed by a direct algebraic attack, simple HFE variations had already been designed to guard against known attacks. The best known of these is probably QUARTZ, a very conservatively designed HFE variant over $\mathbb{F}_2$ using both the "Vinegar" and "Minus" modifications [29]. QUARTZ (and all HFEv variants) have never been credibly cryptanalyzed.

We want to give a solid theoretical bound on the degree of regularity of HFEv and associated systems, such as QUARTZ, and thereby obtain a good estimate on the complexity of attacking HFEv, HFEv-, and ipHFE cryptosystems using Gröbner Bases.

1.2 Answers

One usually solves $p_1(x_1, \ldots, x_n) = p_2(x_1, \ldots, x_n) = \cdots = p_m(x_1, \ldots, x_n) = 0$ over $\mathbb{F}_q$ using Gröbner basis algorithms such as F_4/F_5. The critical parameter which determines the complexity is known as "the degree of regularity", which is the maximum degree of monomials that appear in the computation. If we denote by $(p_i)^h$ the homogeneous leading part of p_i, the degree of regularity of the system is the first degree at which we find non-trivial relations among the $(p_i)^h$'s, or if we set as the graded ring $B := \mathbb{F}_q[x_1, \ldots, x_n]/\langle x_1^q, \ldots, x_n^q \rangle$ and B_d its degree-d slice, we may state a definition as follows for the case of degree-2 equations (generalizable to higher/mixed degrees):

Definition 1.1. *For homogeneous quadratic polynomials $(\lambda_1, \ldots, \lambda_m) \in B_2^m$, let $\psi_d : B_d^m \to B_{d+2}$ be the map defined as $\psi(b_1, \ldots, b_m) = \sum_{i=1}^m b_i \lambda_i$. Then $R_d(\lambda_1, \ldots, \lambda_m) := \ker \psi_d$ defines the subspace of relations $\sum_{i=1}^m b_i \lambda_i = 0$. Further let $T_d(\lambda_1, \ldots, \lambda_n)$ be the subspace of trivial relations generated by elements*

$$\{b(\lambda_i e_j - \lambda_j e_i) | 1 \leq i < j \leq m,\, b \in B_{d-2}\},\ \textit{and}$$

$$\{b(\lambda_i^{q-1} - 1)e_i | 1 \leq i \leq m,\, b \in B_{d-2(q-1)}\}.$$

Here e_i means the i-th unit vector consisting of all zeros except one 1 at the i-th position. The degree of regularity of a homogeneous quadratic set is then

$$D_{reg}(\lambda_1, \ldots, \lambda_m) := \min\{d | R_{d-2}(\lambda_1, \ldots, \lambda_m)/T_{d-2}(\lambda_1, \ldots, \lambda_m) \neq \{0\}\},$$

and $D_{reg}(p_1, \ldots, p_m) := D_{reg}((p_1)^h, \ldots, (p_m)^h)$ for polynomials in general.

We find an upper bound to D_{reg} for HFEv and HFEv-, which like in earlier studies depends on the size of the base field q, the rank of the HFE polynomial r, the number of removed equations a (if "minus" is used), and additionally the number of vinegar variables v, but in general on not the number of variables n:

$$D_{\text{reg}} \leq \frac{(q-1)(r+v+a-1)}{2} + 2, \quad \text{if } q \text{ is even and } r+a \text{ is odd,}$$
$$D_{\text{reg}} \leq \frac{(q-1)(r+v+a)}{2} + 2, \quad \text{otherwise.}$$

For small numbers we evaluated D_{reg} of random tests for HFEv and HFEv- using MAGMA and in each case the bound is relatively tight (see Section 4.1) which lends credence to predictions using our bound above for the Gröbner bases complexity.

As an example, substituting the actual parameters of QUARTZ we get $D_{\text{reg}} \leq 9$. Assuming that it is indeed 9, we can compute the number of bit-operations required to break it as $\approx 2^{92}$ (see Section 4.1), so QUARTZ should be reasonably secure for now.

This also shows that the break of an instance of internally perturbed HFE in [18], which is very much related to HFEv, is likely a case of overly aggressive parameters rather than of systematic problems.

1.3 Related Work

The C^* cryptosystem can be seen as a simple case of an HFE cryptosystem, and [14] noted that Patarin's linearization attack [27] was equivalent to the degree of regularity of C^* being three (in line with the formula in that paper).

The Square cryptosystem [7] is a C^* system with rank 1 and an odd base field. [14] proves a *lower* bound on its degree of regularity, showing a direct algebraic attack with Gröbner basis to be infeasible. However, such a result does not mean that the system is secure, because Square is actually broken by a different attack.

[9] was the first to claim to "break" HFE (cryptanalyze in significantly under design security), and [10] the earliest to mention HFEv- and HFE- specifically. But neither was followed up with a concrete implementation, and all interest was attracted to the news of Faugère's actually breaking HFE Challenge 1 [19].

[21] started to investigate algebraically the degree of regularity of HFE, but [17] seems to be the first rigorous study of the subject, which is continued by [14, 15].

2 Background

In the standard formulation of a multivariate public-key cryptosystem over a finite field $\mathbb{F}$, the public-key $P : \mathbb{F}^n \mapsto \mathbb{F}^m = T \circ Q \circ S$ is a composition of two invertible affine maps $S : \mathbb{F}^n \mapsto \mathbb{F}^n$ and $T : \mathbb{F}^m \mapsto \mathbb{F}^m$, and a quadratic map (possibly with some parameters) $Q : \mathbb{F}^n \mapsto \mathbb{F}^m$ which is easily invertible

when all parameters are given. The maps S and T are part of the secret key, and properties of the central map Q determines most of the properties of the cryptosystem.

2.1 The HFEv, ipHFE and HFEv- Cryptosystems

Let $\mathbb{F} \cong \mathbb{F}_q$ be a finite field of order q and $\mathbb{K}$ a degree-n extension of $\mathbb{F}$, with a "canonical" isomorphism ϕ identifying $\mathbb{K}$ with the vector space $\mathbb{F}^n$. That is, $\mathbb{F}^n \xrightarrow{\phi} \mathbb{K}$, $\mathbb{K} \xrightarrow{\phi^{-1}} \mathbb{F}^n$. Any function or map F from $\mathbb{K}$ to $\mathbb{K}$ can be expressed *uniquely* as a polynomial function with coefficients in $\mathbb{K}$ and degree less than q^n, namely

$$F(X) = \sum_{i=0}^{q^n-1} a_i X^i, \quad a_i \in \mathbb{K}.$$

Denote by $\deg_{\mathbb{K}}(F)$ the degree of $F(X)$ for any map F. Using ϕ, we can build a new map $F' : \mathbb{F}^n \to \mathbb{F}^n$

$$P(x_1, .., x_n) = (p_1(x_1, .., x_n), \ldots, p_n(x_1, .., x_n)) = \phi^{-1} \circ F \circ \phi(x_1, .., x_n),$$

which is essentially F but viewed from the perspective of $\mathbb{F}^n$. We can identify F and F' unless there is a chance of confusion.

An $\mathbb{F}$-degree-2 or $\mathbb{F}$-quadratic function from $\mathbb{K}$ to $\mathbb{K}$ can in this framework be seen to be a polynomial all of whose monomials have exponent $q^i + q^j$ or q^i or 0 for some i and j. The general form of this $\mathbb{F}$-quadratic function is $Q(X) = \sum_{i,j=0}^{n-1} a_{ij} X^{q^i+q^j} + \sum_{i=0}^{n-1} b_i X^{q^i} + c.$, the *extended Dembowski-Ostrom polynomial map*. Such a $Q(X)$ with a fixed low $\mathbb{K}$-degree is used to build the HFE multivariate public key cryptosystems, as in the following

$$Q(X) = \sum_{i,j=0}^{q^i+q^j \le D, j \le i} a_{ij} X^{q^i+q^j} + \sum_{i=0}^{q^i \le D} b_i X^{q^i} + c;$$

Note that the coefficients are values in $\mathbb{K}$, and all coefficients $a_{ii} = 0$ if $q = 2$, since those are covered by the b-part of the coefficients.

For a recent overview of multivariate cryptosystems, including all the common modifiers such as "minus", "internal perturbation", and "vinegar" see [16]. It gives this formulation of HFEv, which uses the vinegar modification [23], built from this polynomial:

$$Q(X, \bar{X}) := \sum_{i,j} a_{ij} X^{q^i+q^j} + \sum_{i,j} b_{ij} X^{q^i} \bar{X}^{q^j} + \sum_{i,j} \alpha_{ij} \bar{X}^{q^i+q^j} + \sum_i b_i X^{q^i} + \sum_i \beta'_i \bar{X}^{q^i} + c \tag{1}$$

where the auxiliary variable $\bar{X}$ occupies only a subspace of small rank v in $\mathbb{K} \cong \mathbb{F}^n$. The function Q is quadratic in the components of X and $\bar{X}$, and so is $P = T \circ Q \circ S$ for affine bijections T and S in $\mathbb{F}^n$ and $\mathbb{F}^{n+v}$. We hope that P is hard to invert to the adversary, while the legitimate user, with the knowledge of

(S, T) can compute X by substituting a random $\bar{X}$, then solving for X via root-finding algorithms such as Berlekamp (or Cantor-Zassenhaus, if $q \neq 2$). To limit the effort of Berlekamp, we restrict the maximum degree D of the polynomial. QUARTZ has the parameter set $(q, n, D, v, a) = (2, 103, 129, 4, 3)$.

We note that to verify in QUARTZ, one invokes the public map multiple times, but the ability to defeat QUARTZ still principally rests on inverting an HFEv- public map.

In an HFEv- cryptosystem, the public key P becomes P^-, that is, it is released minus the last a equations. Again we hope that inverting P^- is intractible without the trapdoor. The legitimate user can invert P^- simply by appending a random numbers from $\mathbb{F}_q$ to to the ciphertext or signature before inverting P.

Another closely related scheme to HFEv is ipHFE (internally perturbed HFE). Suppose in Eq. 1, $\bar{X}$ is not a free variable, but is instead the image of ℓ, a map from $\mathbb{F}^n$ onto $\mathbb{F}^v$. So the central map is really $Q'(X) := Q(X, \ell(X))$. To invert Q', the legitimate user would *guess* the values at positions in $V = \ell(X)$, solve for X, and then check whether $V = \ell(X)$. So the inversion process becomes less efficient in the sense that it takes in the worst case q^v tries to get one answer. From this description, we can see that ipHFE is the same as HFEv with the prefix modification (i.e., one or more limbs of the plaintext in a multivariate scheme becomes pre-determined).

2.2 Conventional Wisdom about HFE Security

There is no "proof of security" for any variant of HFE or any of the usual multivariate PKC proposals that reduce to a difficult computational problem commonly used for cryptography. However, similarly the security of NTRU depends on the hardness of lattice problems, but does not reduce to them. There are lattice-based systems which reduce to hard lattice problems, but these are much less efficient than NTRU. Analogously, there are multivariate PKCs that are "provably secure" in the sense that a break of such a PKC would imply an advance in the solution to an MQ-related computational problem [22,32,33], which happen to be much less efficient. Hence we take the approach that only careful study of cryptanalytic techniques can determine the security of a cryptosystem.

It is unfortunate, then, that HFE Challenge 1 was proposed when we understood the algebra behind it much less. It is even more unfortunate that some of the proposed HFE variants were overly aggressive and were promptly broken [3, 4, 18] just like many other multivariate schemes, because the public perception became biased against the HFE family.

HFE variants also gained a further reputation for being flimsy, more specifically poly-time-solvable [17, 21] with further mathematical studies. In particular [21] sketched a way to bound the degree of regularity for HFE when $q = 2$, using an approach to lift the problem back to the extension $\mathbb{K}$, an idea first suggested by Kipnis-Shamir [24]. They managed to describe a connection of the degree of regularity of the HFE system to the degree of regularity of a lifted system over the big field. Heuristic asymptotic bounds were found when $q = 2$

leading to the conclusion that if $D = O(n)$ the complexity of Gröbner basis solvers for the corresponding HFE systems is quasi-polynomial.

In some ways, this reputation is actually somewhat unfair, since simple HFE variations such as QUARTZ have resisted known attacks for a long time, and it is actually known in various contexts how the degree of operations in an algebraic attack varies (cf. [14]). We hope to achieve a more realistic evaluation of the security of HFE-related schemes. In particular we hope that better understanding of the degree of regularity under algebraic attcks can establish some HFE variants as fundamentally sound cryptosystems which had previously been proposed with overly aggressive parameters, rather than fundamentally broken systems (like C^*-).

2.3 Algebraic Cryptanalysis

Aside from cases in which brute-force enumeration [5] seems to the best *practical* way to solve systems, almost all of today's algebraic algorithms to solve

$$p_1(x_1, \ldots, x_n) = p_2(x_1, \ldots, x_n) = \cdots = p_m(x_1, \ldots, x_n) = 0$$

over $\mathbb{F}_q$ go back to Buchberger's algorithm for computing Gröbner bases [6]. Lazard proposed the following critical simplification (later reinvented as the XL Method): multiply the equations with monomials to form a collection of relations up to a some degree d. Linearize (i.e., treat each individual monomial as a variable), and use well-studied matrix algorithms over $\mathbb{F}_q$ on the resulting matrix (the *extended Macaulay matrix*) [11,25].

The Degree of Regularity. The critical concept in the complexity analysis of algebraic polynomial solving algorithms is the concept of *degree of regularity*. As given in Definition 1.1, the degree of regularity of the polynomial system is the lowest degree where we find a *non-trivial* degree drop. Conventional wisdom has it that in general this is the degree at which $\mathrm{F}_4/\mathrm{F}_5$ and similar algorithms usually terminate. Therefore D_{reg} is used to characterize the complexity of the algorithms.

We first note that almost all modern Gröbner Bases methods improve on XL as follows: suppose we fix a degree d and multiply each p_i with all monomials of degree $d - \deg p_i$ to create a large collection of relations of degree d. Order the monomials and linearize these equations to obtain the Macaulay matrix $\mathrm{Mac}^{(d)}(p_1, \ldots, p_m)$. Try to eliminate the highest degree monomials from $\mathrm{Mac}^{(d)}(p_1, \ldots, p_m)$ to create relations of degree $d-1$ or lower.

After we find such polynomials with degree drop, we multiply them by individual variables, and we obtain equations of at most degree d, which are effectively elimination remnants from higher-degree relations. If necessary, we can repeat this process many times until we can solve for all the variables. This describes MutantXL or XL2 [13,36] which will terminate at the same degree as $\mathrm{F}_4/\mathrm{F}_5$ [36]. Any superiority of the latter comes from having fewer redundant equations being generated or going through the elimination.

In Definition 1.1, we can see that the subspace T_d of trivial syzygies represents a "known-to-be-useless" degree drop in the following sense: Let $p_i = c^{(i)} + \sum_k b_k^{(i)} x_k + \sum_{k \leq \ell} a_{k\ell}^{(i)} x_k x_\ell$. For a polynomial p, let $(p)^h$ represent the homogeneous highest degree part of the polynomial p, and (p) a corresponding row in a Macaulay-type matrix. Clearly $(p_j)^h (p_i)^h - (p_i)^h (p_j)^h = 0$ is a trivial syzygy, which is equivalent to the combination of degree-4 rows $\left(\sum_{k\ell} a_{k\ell}^{(i)} (x_k x_\ell p_j)\right) - \left(\sum_{k\ell} a_{k\ell}^{(j)} (x_k x_\ell p_i)\right)$ being of degree-3 (or fewer). Equally clearly this "degree-drop" will not give us anything useful since

$$\left(c^{(i)}(p_j) + \sum_k b_k^{(i)} (x_k p_j) + \sum_{k\ell} a_{k\ell}^{(i)} (x_k x_\ell p_j)\right)$$
$$= \left(c^{(j)}(p_i) + \sum_k b_k^{(j)} (x_k p_i) + \sum_{k\ell} a_{k\ell}^{(j)} (x_k x_\ell p_i)\right),$$

given that both give $(p_i p_j)$. Thus we just "found" a linear combination of polynomials we already have at degree 3. So a trivial or *principal* syzygy between the top-degree parts $(p_i)^h$ leads to a *trivial* degree drop useless for generating new equations. We must verify that a degree-drop is non-trivial before we can claim that we have reached the degree of regularity.

Issue of Terminology. There is some confusion about the term "the degree of regularity". The rank of Macaulay matrices at a given degree can be derived as the coefficients of certain generating functions, with the heuristic assumption that there are no non-trivial syzygies. A system where this holds for all degrees is called *regular*. However this can be the case only for underdetermined systems over characteristic zero fields. Otherwise at a sufficiently high degree the generating function eventually has a non-positive coefficient, and regularity becomes impossible. Systems for which the rank of the Macaulay matrices follows the heuristic for as long as possible are called "semi-regular" [12]. Definition 1.1 follows [17] in that the degree of regularity is defined as "the first appearance of non-trivial degree fall", i.e., where the system ceases to behave as semi-regular.

The heuristic formulas that have since long been known to hold for the degree of regularity of most random systems (including asymptotics) are given by Bardet et al [1,2,37]. However, this formula does not hold for most systems with structure.

Conventional wisdom also accepts that when $m/n = h + o(1)$ where h is a constant not far removed from 1, solving m "generic" or "random" equations in n variables is exponential in time and space in n. We can do a tiny bit better. That is, for sufficiently large h we may solve the system faster than just guessing variables first (cf. e.g. [8, 35]), but it is still exponential time and space in m (and/or n).

Invariance of Degree of Regularity. The degree of regularity is invariant under invertible linear transformation in both the domain and the codomain.

So if $P = T \circ Q \circ S$ is the public map of a multivariate PKC with the central map Q with both S and T invertible affine transformations, then the degree of regularity in solving X from $P(X) = Y$ depends only on Q, and can be written $D_{\text{reg}}(Q)$.

3 Main Results

To recap, suppose we wish to solve an HFEv system with $\mathbb{K} \cong \mathbb{F}^n$, where $\mathbb{F} = \mathbb{F}_q$, with degree D and v vinegar variables. We would have then $n + v$ variables and n equations. However, MutantXL or F_4/F_5 algorithms deal with determined or overdetermined equations. The standard way to get around this problem is to guess some v variables and bring it down to a system with n variables. As noted earlier, we have now an ipHFE instance. We try to analyze the direct attack as in [14,15,17]. First, let us present our main results.

Theorem 3.1. *Let r be the rank of the HFE polynomial and v the number of vinegar variables. We may bound the degree of regularity of HFEv as follows:*

$$D_{\text{reg}} \leq \frac{(q-1)(r+v-1)}{2} + 2, \quad \textit{if } q \textit{ is even and } r \textit{ is odd,} \tag{2}$$

$$D_{\text{reg}} \leq \frac{(q-1)(r+v)}{2} + 2, \quad \textit{otherwise.} \tag{3}$$

This result is sufficient to bound the complexity of a direct algebraic attack against HFEv. If we assume that the direct algebraic attack is the best attack on HFEv systems, this would be the most important bound required to evaluate the security of odd-field HFEv and derivatives.

However, QUARTZ is an instance of HFEv-, not just HFEv. We recall that HFEv-, of which QUARTZ is a special case is derived from HFEv by removing a few public key polynomials. We normally have $n+v$ variables and $n-a$ equations. To solve a HFEv- case, we again first guess v-values. Then we have n variables and $n-a$ equations. As we mentioned, this is essentially an ipHFE system. Now we need to bound the degree of regularity of a direct algebraic attack on HFEv on such a system.

Theorem 3.2. *Let r be the rank of the HFE polynomial, v the number of vinegar variables, and a the number of "minus" equations, then we may bound the degree of regularity as follows:*

$$D_{\text{reg}} \leq \frac{(q-1)(r+a+v-1)}{2} + 2, \quad \textit{if } q \textit{ is even and } r+a \textit{ is odd,}$$

$$D_{\text{reg}} \leq \frac{(q-1)(r+a+v)}{2} + 2, \quad \textit{otherwise.}$$

We will now show how our main results is proved.

To prove Equation (3) in Theorem 3.1, we must use a result that link the degree of regularity on a big-field multivariate to the rank of the central map.

Proposition 3.3. *[14, Theorem 4.1] For central maps Q that corresponds to quadratic maps, we have*

$$D_{reg}(Q) \leq \frac{(q-1)\,\mathrm{Rank}(Q)}{2} + 2.$$

We now need to show that the rank of an HFEv central polynomial with v vinegar variables is no higher than that of the original HFE polynomial plus v. First, we rewrite the HFEv polynomial so that it is more easily handled.

Proposition 3.4. *The associated polynomial when solving an HFEv or an ipHFE system over the big field $\mathbb{K}$ can be written as:*

$$\begin{aligned}\bar{P}(X) = &\sum_{i=0}^{q^i<D} \Big((\sum_{j=0}^{q^i+q^j \leq D, j\leq i} a_{ij} X^{q^i+q^j}) + (\sum_{l=0}^{v-1} a'_{il} X^{q^i} \bar{X}_l) \Big) \\ &+ \sum_{i=0}^{v-1} \sum_{j=i}^{v-1} a''_{ij} \bar{X}_i \bar{X}_j + \sum_{i=0}^{q^i \leq D} b_i X^{q^i} + \sum_{i=0}^{v-1} u_i \bar{X}_i + c, \qquad (4)\end{aligned}$$

where $\bar{X}_i := \mathrm{Tr}(\alpha_i X)$ for suitably chosen α_i. The map Tr *is the trace function, which is also given by* $\mathrm{Tr}(X) := \sum_{j=0}^{n-1} (X)^{q^j}$.,

Proof. We note that Tr is a nontrivial linear map of $\mathbb{F}^n \to \mathbb{F}$. For some representation of $\mathbb{F}^n \cong \mathbb{K}$, we can write it as a projection into the first component. With a suitably chosen α_i, we can make the first component of $\alpha_i X$ any nontrivial linear map of the components of X. So we can express each of the components of $\bar{X} = \ell(X)$ in Eq. 1 as $\mathrm{Tr}(\alpha_i X)$ for some α_i.

So Theorem 3.1 can be proved if we can show that:

Proposition 3.5. *The rank of the quadratic form associated with the polynomial $\bar{P}$ above, written $R(\bar{P})$, is bounded by:*

$$R(\bar{P}) \leq R(P) + v.$$

To obtain this we need this result about quadratic forms:

Proposition 3.6. *[26, Chapter 6] The rank of a quadratic form F is less than or equal to the minimum number of linear forms one needs to express F as a quadratic function in them. That is, if one can write F as a quadratic function of linear forms $\ell_1, \ldots, \ell_r$, then* $\mathrm{Rank}\, F \leq r$.

Definition 3.7. *Let F be a quadratic form over a field k, and $F(X,Y) := X^t F Y$ be the bilinear (symmetric) form associated with F over the field k^n. Let*

$$N_F = \{X \in k^n | F(X,Y) = 0, \quad \textit{for any} \quad Y \in k^n\}.$$

N_F as linear subspace is called the radical for the bilinear form F.

Note that for any F of rank r, we can write F in the linear forms $\ell_1, \ldots, \ell_r$, and any X with $\ell_1(X) = \cdots = \ell_r(X) = 0$ is in N_F. So by using the following observation, we see that the dimension of N_F is $n - r$.

Proposition 3.8. *Let z_ℓ, $\ell = 0, \ldots, v-1$, be linear functions from $\mathbb{F}^n$ to $\mathbb{F}$, i.e., $z_\ell : (x_1, \ldots, x_n) \mapsto \sum \beta_i^{(\ell)} x_i$. Then the dimension of the intersection of kernels $K(z_i) := \{X \in \mathbb{F}^n | z_i(X) = 0\}$ is bounded by*

$$\dim\left(\bigcap_i^{v-1} K(z_i)\right) \geq n - v.$$

Proposition 3.9. *Under the conditions and notation of Definition 3.7 and Proposition 3.8,*

$$\dim(N_F \bigcap K(Z)) \geq n - r - v.$$

The last proposition follows from 3.7 and 3.8, basically by inclusion-exclusion.

Proposition 3.10. *Let $F(x_0, ..x_{n-1})$ be a quadratic form (or polynomial) whose rank is r. Here each variable x_i can additionally be considered as a linear map or function from $\mathbb{F}^n$ to $\mathbb{F}$. In this manner it would be viewed the i-th component map $x_i(u_0, \ldots, u_{n-1}) = u_i$, for $(u_0, \ldots, u_{n-1}) \in \mathbb{F}^n$. Let $A : \mathbb{F}^n \to \mathbb{F}^n$ be an invertible linear transformation (with A^{-1} its inverse), such that*

$$F(A(x_0), \ldots, A(x_{n-1}))) = \sum_{i=0}^{r} \sum_{j=i}^{r} a_{ij} x_i x_j,$$

where $A(x_i)$ is the function from $\mathbb{F}^n$ to $\mathbb{F}$ derived from $x_i \circ A$. Let

$$\bar{F}(x) = F(x_0, ..x_{n-1}) + \sum_{i=0}^{r} (\sum_{\ell=0}^{v-1} a'_{i\ell} A^{-1}(x_i) z_\ell),$$

where each z_ℓ is a linear function from $\mathbb{F}^n$ to $\mathbb{F}$, i.e., $z_\ell : (x_0, \ldots, x_{n-1}) \mapsto \sum \beta_i^{(\ell)} x_i$. Then

$$\text{Rank}(\bar{F}) \leq \text{Rank}(F) + v.$$

This follows from Proposition 3.9.

Now we further note that the process of fixing v variables to get a determined system corresponds to introducing v linear relations of the form

$$\sum_i a_i X^{q^i} + \sum_{j=1}^{v} b_j \overline{X}_j = 0.$$

From this we can substitute for each of the $\overline{X}_j$, a linear combination of the X^{q^i} (which is itself linear in X), which shows that the quadratic form $\overline{P}$ of HFEv or ipHFE can be expressed using v extra linear forms than the Dembowski-Ostrom polynomial map P (that is, without the v forms $\overline{X}_i$). Then since the rank of a quadratic form is bounded by the number of linear forms used to express it, we have Proposition 3.5, and Equation (3) then follows.

We note that the above line of reasoning is good only for odd q because in binary fields the rank of the associated matrix to a symmetric form is always even, creating various off-by-one errors in the above process, we may go through steps akin to that in [14] to patch those off-by-one problems (to be included in a full journal version), and account for the binary field cases in Theorem 3.1.

A note on HFE over tower fields. An HFE-derivative cryptosystem built over $\mathbb{F}_{q^k}$ is also one over $\mathbb{F}_q$. So we can (for example) attack an HFE-type instance built over $\mathbb{F}_{16}$ by solve it as a system over $\mathbb{F}_2$. However, in this situation the rank parameter r would usually be $\lfloor \log_{16}(D-1) \rfloor + 1$, not $\lfloor \log_2(D-1) \rfloor + 1$. The reason is that the central Dembowski-Ostrom polynomial, and therefore the rank r, is an entity in the big field and does not vary according to our viewpoint.

Proving Theorem 3.2 Again let us examine only odd characteristic cases for now. From the definition of HFEv-, it may be viewed as (HFE-)v. I.e., just as a central map of HFEv is one of HFE *plus a quadratic function with the extra variables in the form of a vector in an unknown subspace of dimension v (the "vinegar subspace")*, in exactly the fashion, HFEv- is HFEv plus a quadratic function with extra unknowns in that same vinegar subspace.

Put another way, let $\hat{P}^-$ be the public key of an HFEv- instance which is derived from the corresponding public key of an HFEv instance:

$$\hat{P}^-(x_1, \ldots, x_n) = (\hat{p}_1(x_1, \ldots, x_n), \ldots, \hat{p}_{n-a}(x_1, \ldots, x_n), 0, ..0).$$

We can then depict $\hat{P}^-$ as the vinegar form of an HFE- instance with central map Q^-. Q^- is a quadratic map, and can hence expressible as an extended Dembowski-Ostrom map. In other words, Q^- is also the central map of an HFE instance.

Now, according to [15, Section 4, Proposition 1] we have $\text{Rank}(Q) \leq \text{Rank}(Q) + a$, where a is the number of "minus" equations. This holds because all the arguments there depend only on rank and not on exponents in the formulas.

Finally, we use Proposition 3.10 with $\hat{P}^-$ as the central map of an HFEv instance. We conclude that $\text{Rank}(P^-) < \text{Rank}(Q) + r + a$ which leads to the odd-q half of Theorem 3.2.

4 Testing, Implication and Discussion

Having given a bound for the degree of regularity for HFEv- (and ipHFE) systems, we give some experimental results and discuss what this means for QUARTZ.

4.1 Tests and Results

We ran MAGMA-2.7.12 on random systems for each parameter $n \leq 13$, $r \leq 4$, $a, v \leq 2$, and $q \leq 5$, on a workstation (with 2x Opteron 6212 and 32GB of RAM) to find D_{reg} on 4–20 randomly generated HFEv and HFEv- systems, and for $q = 2$ further for 1 random system each up to $n = 29$. We added $x_i^q - x_i$ for each i as part of the system of equations, so as to trigger field-specific optimizations that MAGMA might have for $q = 2$. In each case, D_{reg} proves to be the *smaller* of *either* the minimum of the bound in the formula above *or*, if we use $[u]S$ to mean the coefficient of the term u in a corresponding series expansion of S:

$$\min \left\{ d : [x^d] \left(\left(\frac{1-x^q}{1-x} \right)^n \left(\frac{1-x^2}{1-x^{2q}} \right)^m \right) < 0 \right\},$$

for m equations and n variables in $\mathbb{F}_q$. The cryptic expression above denotes the smallest d such that the coefficient of x^d in the Maclaurin expansion of $\left(\frac{1-x^q}{1-x}\right)^n \left(\frac{1-x^2}{1-x^{2q}}\right)^m$ becomes negative. It is actually the usual heuristic expression for D_{reg} for random systems, such as those found in [1] (for $q = 2$ only).

The numbers may seem too small to be conclusive, but for 13 variables and equations over $\mathbb{F}_7$ or 14 variables and equations over $\mathbb{F}_5$ MAGMA is already running out of memory, and these results lend credence to predictions using our bound for the Gröbner Bases complexity for HFEv and HFEv- systems. We can now try to justify the predictions for QUARTZ given in Section 1.

4.2 Implications for QUARTZ

We have obtained a bound on the degree of regularity of 9 for QUARTZ (which has $q = 2$, $n = 103$, $r = 7$, $a = 3$, $v = 4$), which represents a big drop already compared to degree 13 for a random system of that size (cf. formula above). However, if the bound is reasonably tight, the number of columns (monomials) involved in the elimination should be roughly the number of top-level monomials, which are $T := \binom{n}{D_{\text{reg}}} = \binom{100}{9} \gtrsim 2^{40}$ in total. A dense-matrix elimination would require 2^{80} bits of storage which is clearly not feasible.

Let us assume an extremely optimistic scenario for the attacker, such that a putative sparse-matrix-enabled $\mathrm{F}_4/\mathrm{F}_5$ attack is possible. Since each row has $\tau = \binom{100}{2} \geq 2^{12}$ terms, we will require about 2^{52} bits of memory. This is very large still, but not impossible in the mid-term future. We further use the number of bit-operations in the most time-consuming Wiedemann or Block Wiedemann type elimination methods as the estimate of the attack complexity, then we get the evaluation of the complexity given in Section 1:

$$C_{\mathrm{F}_4/\mathrm{F}_5} \geq 3\tau T^2 \gtrsim 3 \cdot 2^{12} \cdot (2^{40})^2 \geq 2^{92}.$$

Note: This evaluation above is highly optimistic in that it makes the implicit assumption that there is no penalty for accessing large memory. This may be very wrong in two ways:

- There is a very perceptible cost penalty in assembling a large amount of RAM which is either accessible on one machine or is networked using high speed interconnect to every other machine.
- Accessing a large amount of memory is slower; most server motherboards takes a speed penalty when using the maximum number of memory modules, and accessing memory on other machines of course incurs terrible latency.

What this might mean practically is that *it might be more advantageous to attack QUARTZ by brute-force [5]*, which imposes no communication requirements (i.e., networking and memory bandwidth and latencies) and is embarassingly parallelizable (hence perfectly scalable).

Final Remark: In some of the cases previously studied, we can *prove* tightness of the bounds. Clearly more of this type of work is needed, where theoretical bounds for attacks are given, just like the studies of theoretical bounds on differential probabilities in AES.

References

1. Bardet, M., Faugère, J.-C., Salvy, B.: On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In: Proceedings of the International Conference on Polynomial System Solving, pp. 71–74 (2004); Previously INRIA report RR-5049
2. Bardet, M., Faugère, J.-C., Salvy, B., Yang, B.-Y.: Asymptotic expansion of the degree of regularity for semi-regular systems of equations. In: Gianni, P. (ed.) MEGA 2005 Sardinia, Italy (2005)
3. Bettale, L., Faugère, J.-C., Perret, L.: Cryptanalysis of multivariate and odd-characteristic HFE variants. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 441–458. Springer, Heidelberg (2011)
4. Billet, O., Macario-Rat, G.: Cryptanalysis of the square cryptosystems. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 451–468. Springer, Heidelberg (2009)
5. Bouillaguet, C., Chen, H.-C., Cheng, C.-M., Chou, T., Niederhagen, R., Shamir, A., Yang, B.-Y.: Fast exhaustive search for polynomial systems in $\mathbb{F}_2$. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 203–218. Springer, Heidelberg (2010)
6. Buchberger, B.: Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. PhD thesis, Innsbruck (1965)
7. Clough, C., Baena, J., Ding, J., Yang, B.-Y., Chen, M.-S.: Square, a new multivariate encryption scheme. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 252–264. Springer, Heidelberg (2009)
8. Courtois, N., Goubin, L., Meier, W., Tacier, J.-D.: Solving underdefined systems of multivariate quadratic equations. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 211–227. Springer, Heidelberg (2002)
9. Courtois, N.T.: The security of hidden field equations (HFE). In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 266–281. Springer, Heidelberg (2001)
10. Courtois, N.T., Daum, M., Felke, P.: On the security of HFE, HFEv- and Quartz. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 337–350. Springer, Heidelberg (2002)
11. Courtois, N.T., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000), `http://www.minrank.org/xlfull.pdf`
12. Diem, C.: The XL-algorithm and a conjecture from commutative algebra. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 323–337. Springer, Heidelberg (2004)
13. Ding, J., Buchmann, J., Mohamed, M.S.E., Mohamed, W.S.A.E., Weinmann, R.-P.: Mutant XL. Talk at the First International Conference on Symbolic Computation and Cryptography (SCC 2008), Beijing (2008)
14. Ding, J., Hodges, T.J.: Inverting HFE systems is quasi-polynomial for all fields. In: Rogaway [31], pp. 724–742
15. Ding, J., Kleinjung, T.: Degree of regularity for HFE−. Cryptology ePrint Archive, Report 2011/570 (2011), `http://eprint.iacr.org/`

16. Ding, J., Yang, B.-Y.: Multivariate public key cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post Quantum Cryptography, 1st edn., pp. 193–241. Springer, Berlin (2008) ISBN 3-540-88701-6
17. Dubois, V., Gama, N.: The degree of regularity of HFE systems. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 557–576. Springer, Heidelberg (2010)
18. Dubois, V., Granboulan, L., Stern, J.: Cryptanalysis of HFE with internal perturbation. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 249–265. Springer, Heidelberg (2007)
19. Faugère, J.-C., Joux, A.: Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 44–60. Springer, Heidelberg (2003)
20. Fischlin, M., Buchmann, J., Manulis, M. (eds.): PKC 2012. LNCS, vol. 7293. Springer, Heidelberg (2012)
21. Granboulan, L., Joux, A., Stern, J.: Inverting HFE is quasipolynomial. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 345–356. Springer, Heidelberg (2006)
22. Huang, Y.-J., Liu, F.-H., Yang, B.-Y.: Public-key cryptography from new multivariate quadratic assumptions. In: Fischlin et al. [20], pp. 190–205
23. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999)
24. Kipnis, A., Shamir, A.: Cryptanalysis of the HFE public key cryptosystem by relinearization. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 19–30. Springer, Heidelberg (1999), `http://www.minrank.org/hfesubreg.ps`
25. Lazard, D.: Gröbner-bases, Gaussian elimination and resolution of systems of algebraic equations. In: van Hulzen, J.A. (ed.) ISSAC 1983 and EUROCAL 1983. LNCS, vol. 162, pp. 146–156. Springer, Heidelberg (1983)
26. Lidl, R., Niederreiter, H.: Finite Fields, 2nd edn. Encyclopedia of Mathematics and its Application, vol. 20. Cambridge University Press (2003)
27. Patarin, J.: Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt '88. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 248–261. Springer, Heidelberg (1995)
28. Patarin, J.: Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 33–48. Springer, Heidelberg (1996), `http://www.minrank.org/hfe.pdf`
29. Patarin, J., Courtois, N.T., Goubin, L.: QUARTZ, 128-bit long digital signatures. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 282–288. Springer, Heidelberg (2001)
30. Proos, J., Zalka, C.: Shor's discrete logarithm quantum algorithm for elliptic curves. Quantum Information & Computation 3(4), 317–344 (2003)
31. Rogaway, P. (ed.): Advances in Cryptology – CRYPTO 2011. LNCS, vol. 6841. Springer, Heidelberg (2011)
32. Sakumoto, K.: Public-key identification schemes based on multivariate cubic polynomials. In: Fischlin et al. [20], pp. 172–189
33. Sakumoto, K., Shirai, T., Hiwatari, H.: Public-key identification schemes based on multivariate quadratic polynomials. In: Rogaway [31], pp. 706–723

34. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing 26(5), 1484–1509 (1997)
35. Thomae, E., Wolf, C.: Solving underdetermined systems of multivariate quadratic equations revisited. In: Fischlin et al. [20], pp. 156–171.
36. Yang, B.-Y., Chen, J.-M.: All in the XL family: Theory and practice. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 67–86. Springer, Heidelberg (2005)
37. Yang, B.-Y., Chen, J.-M.: Theoretical analysis of XL over small fields. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 277–288. Springer, Heidelberg (2004)

Software Speed Records for Lattice-Based Signatures

Tim Güneysu[1], Tobias Oder[1], Thomas Pöppelmann[1], and Peter Schwabe[2,⋆]

[1] Horst Görtz Institute for IT-Security, Ruhr-University Bochum, Germany
[2] Digital Security Group, Radboud University Nijmegen, The Netherlands

Abstract. Novel public-key cryptosystems beyond RSA and ECC are urgently required to ensure long-term security in the era of quantum computing. The most critical issue on the construction of such cryptosystems is to achieve security *and* practicability at the same time. Recently, lattice-based constructions were proposed that combine both properties, such as the lattice-based digital signature scheme presented at CHES 2012. In this work, we present a first highly-optimized SIMD-based software implementation of that signature scheme targeting Intel's Sandy Bridge and Ivy Bridge microarchitectures. This software computes a signature in only 634988 cycles on average on an Intel Core i5-3210M (Ivy Bridge) processor. Signature verification takes only 45036 cycles. This performance is achieved with full protection against timing attacks.

Keywords: Post-quantum cryptography, lattice-based cryptography, cryptographic signatures, software implementation, AVX, SIMD.

1 Introduction

Besides breakthroughs in classical cryptanalysis the potential advent of quantum computers is a serious threat to the established discrete-logarithm problem (DLP) and factoring-based public-key encryption and signature schemes, such as RSA, DSA and elliptic-curve cryptography. Especially when long-term security is required, all DLP or factoring-based schemes are somewhat risky to use. The natural consequence is the need for more diversification and investigation of potential alternative cryptographic systems that resist attacks by quantum computers. Unfortunately, it is challenging to design secure *post-quantum* signature schemes that are efficient in terms of speed and key sizes. Those which are known to be very efficient, such as the lattice-based NTRU-sign [15] have been shown to be easily broken [19]. Multivariate quadratic (MQ) signatures, e.g., Unbalanced Oil and Vinegar (UOV), are fast and compact, but their public keys are huge with around 80 kB and thus less suitable on embedded systems – even with optimizations the keys are still too large (around 8 Kb) [20].

⋆ This work was supported by the National Institute of Standards and Technology under Grant 60NANB10D004. Permanent ID of this document: `ead67aa537a6de60813845a45505c313`. Date: March 28, 2013

P. Gaborit (Ed.): PQCrypto 2013, LNCS 7932, pp. 67–82, 2013.

The introduction of special ring-based (ideal) lattices and their theoretical analysis (see, e.g., [18]) provides a new class of signature and encryption schemes with a good balance between key size, signature size, and speed. The speed advantage of ideal lattices over standard lattice constructions usually stems from the applicability of the Number Theoretic Transform (NTT), which allows operations in quasi-linear runtime of $\mathcal{O}(n \log n)$ instead of quadratic complexity. In particular, two implementations of promising lattice-based constructions for encryption [12] and digital signatures [14] were recently presented and demonstrate that such constructions can be efficient in reconfigurable hardware. However, as the proof-of-concept implementation in [12] is based on the generic NTL library [22], it remains still somewhat unclear how these promising schemes perform on high-performance processors that include modern SIMD multimedia extensions such as SSE and AVX.

Contribution. The main contribution of this work is the first optimized software implementation of the lattice-based signature scheme proposed in [14]. It is an aggressively optimized variant of the scheme originally proposed by Lyubashevsky [17] without Gaussian sampling. We use security parameters $p = 8383489, n = 512, k = 2^{14}$ that are assumed to provide an equivalent of about 80 bits of security against attacks by quantum computers and 100 bits of security against classical computers. With these parameters, public keys need only 1536 bytes, private keys need 256 bytes and signatures need 1184 bytes. On one core of an Intel Core i5-3210M processor (Ivy Bridge microarchitecture) running at 2.5 GHz, our software can compute more than 3900 signatures per second or verify more than 55000 signatures per second. To maximize reusability of our results we put the software into the public domain[1]. We will additionally submit our software to the eBACS benchmarking project [4] for public benchmarking.

Outline. In Section 2 we first provide background information on the implemented signature scheme. Our implementation and optimization techniques are described in Section 3 and evaluated and compared to previous work in Section 4. We conclude with future work in Section 5.

2 Signature Scheme Background

In this section we briefly revisit the lattice-based signature scheme implemented in this work. For more detailed information as well as security proofs, please refer to [14,17].

2.1 Notation

In this section we briefly recall the notation from [14]. We use a similar notation and denote by $\mathcal{R}^{p^n}$ the polynomial ring $\mathbb{Z}[x]_p\langle x^n + 1\rangle$ with integer coefficients in the range $[-\frac{p-1}{2}, \frac{p-1}{2}]$ where n is a power of two. The prime p must satisfy the

[1] The software is available at `http://cryptojedi.org/crypto/#lattisigns`

congruence relation $p \equiv 1 \pmod{2n}$ to allow us to use the quasi-linear-runtime NTT-based multiplication. For any positive integer k, we denote by $\mathcal{R}_k^{p^n}$ the set of polynomials in $\mathcal{R}^{p^n}$ with coefficients in the range $[-k, k]$. The expression $a \xleftarrow{\$} D$ denotes the uniformly random sampling of a polynomial a from the set D.

Algorithm 1. KEY GENERATION ALGORITHM GEN(p, n)

Input: Parameters p, n
Output: $(t)_{pk}$, $(s_1, s_2)_{sk}$
1 $s_1, s_2 \xleftarrow{\$} \mathcal{R}_1^{p^n}$;
2 $t \leftarrow as_1 + s_2$;

2.2 Definition

According to the description in [14] we have chosen a to be a randomly generated global constant. For the key generation described in Algorithm 1 we therefore basically perform sampling of random values from the domains $\mathcal{R}_1^{p^n}$ followed by a polynomial multiplication with the global constant and an addition. The private key sk consists of the values s_1, s_2 while t is the public key pk. Algorithm 2 signs a message m specified by the user. In step 1 two polynomials y_1, y_2 are chosen uniformly at random with coefficients in the range $[-k, k]$. In step 2 a hash function is applied on the higher-order bits of ay_1+y_2 which outputs a polynomial c by interpreting the first 160-bit of the hash output as a sparse polynomial. In step 3 and 4, y_1 and y_2 are used to mask the private key by computing z_1 and z_2. The algorithm only continues if z_1 and z_2 are in the range $[-(k-32), k-32]$ and restarts otherwise. The polynomial z_2 is then compressed into $z_2^{'}$ in step 7 by Compress. This compression is part of the aggressive size reduction of the signature $\sigma = (z_1, z_2^{'}, c)$ since only some portions of z_2 are necessary to maintain the security of the scheme. For the implemented parameter set Compress has a chance of failure of less than two percent which results in the restart of the whole signing process.

The verification algorithm VER as described in Algorithm 3 first ensures that all coefficients of $z_1, z_2^{'}$ are in the range $[-(k-32), k-32]$ and rejects the input otherwise by returning $b = 0$ to indicate an invalid signature. In the next step, $az_1+z_2^{'}-tc$ is computed, transformed into the higher-order bits and then hashed. If the polynomial c from the signature and the output of the hash match, the signature is valid and the algorithm outputs $b = 1$ to indicate its success.

In Algorithm 4 the transformation of a polynomial into a higher-order representation is described. This algorithm exploits the fact that every polynomial $Y \in \mathcal{R}^{p^n}$ can be written as

$$Y = Y^{(1)}(2(k-32)+1) + Y^{(0)}$$

where $Y^{(0)} \in \mathcal{R}_{k-32}^{p^n}$ and thus every coefficient of $Y^{(0)}$ is in the range $[-(k-32), k-32]$. Due to this bijectional relationship, every polynomial Y can be also written as the tuple $(Y^{(1)}, Y^{(0)})$.

Algorithm 2. SIGNING ALGORITHM SIGN(s_1, s_2, m)

Input: $s_1, s_2 \in \mathcal{R}_1^{p^n}$, message $m \in \{0,1\}^*$
Output: $z_1, z_2^{'} \in \mathcal{R}_{k-32}^{p^n}$, $c \in \{0,1\}^{160}$

1 $y_1, y_2 \xleftarrow{\$} \mathcal{R}_k^{p^n}$;
2 $c \leftarrow \mathrm{H}(\mathsf{Transform}(ay_1 + y_2), m)$;
3 $z_1 \leftarrow s_1 c + y_1$;
4 $z_2 \leftarrow s_2 c + y_2$;
5 **if** z_1 *or* $z_2 \notin \mathcal{R}_{k-32}^{p^n}$ **then**
6 go to step 1;
7 $z_2^{'} \leftarrow \mathsf{Compress}(ay_1 + y_2 - z_2, z_2, p, k-32)$;
8 **if** $z_2^{'} = \perp$ **then**
9 go to step 1;

Algorithm 3. VERIFICATION ALGORITHM VER$(z_1, z_2^{'}, c, t, m)$

Input: $z_1, z_2^{'} \in \mathcal{R}_{k-32}^{p^n}$, $t \in \mathcal{R}^{p^n}$, $c \in \{0,1\}^{160}$, message $m \in \{0,1\}^*$
Output: b

1 **if** z_1 *or* $z_2^{'} \notin \mathcal{R}_{k-32}^{p^n}$ **then**
2 $b \leftarrow 0$;
3 **else**
4 **if** $c = H(\mathsf{Transform}(az_1 + z_2^{'} - tc), m)$ **then**
5 $b \leftarrow 1$;
6 **else**
7 $b \leftarrow 0$;

Algorithm 5 describes the compression algorithm Compress which takes a polynomial y, a polynomial z with small coefficients and the security parameter k as well as p as input. It is designed to return a polynomial $z^{'}$ that is compacted but still maintains the equality between the higher-order bits of $y + z$ and $y + z^{'}$ so that $(y + z)^{(1)} = (y + z^{'})^{(1)}$. In particular, the parameters of the scheme are chosen in a way that the if-condition specified in step 3 is true only for rare cases. This is important since only values assigned to $z^{'}[i]$ in step 6 to step 13 can be efficiently encoded.

The hash function H maps an arbitrary-length input $\{1,0\}^*$ to a 512-coefficient polynomial with 32 coefficients in $\{-1,1\}$ and all other coefficients zero. The whole process of generating this string and its transformation into a polynomial with the above described character is shown in Algorithm 6. In step 1 the message is concatenated with a binary representation of the polynomial x generated by the algorithm BinRep. It takes a polynomial $x \in \mathcal{R}^{p^n}$ as input and outputs a (somehow standardized) binary representation of this polynomial. The 160-bit hash value is processed by partitioning it into 32 blocks of 5 side-by-side bits (beginning with the lowest ones) that each correspond to a particular region in the

Algorithm 4. HIGHER-ORDER TRANSFORMATION ALGORITHM TRANSFORM(y, k)

Input: $y \in \mathcal{R}^{p^n}$, k
Output: $y^{(1)}$
for i=0 **to** $n-1$ **do**
  $y^{(0)}[i] \leftarrow y[i] \mod (2(k-32)+1)$;
  $y^{(1)}[i] \leftarrow \frac{y[i]-y^{(0)}[i]}{2(k-32)+1}$;
return $y^{(1)}$;

polynomial c. These bits are $r_4r_3r_2r_1r_0$ where $(r_3r_2r_1r_0)_2$ represents the position in the region interpreted as a 4-bit unsigned integer and the bit r_4 determines if the value of the coefficient is -1 or 1.

2.3 Parameters and Security

Parameters that offer a reasonable security margin of approximately 100 bits of comparable classical symmetric security are $n = 512$, $p = 8383489$, and $k = 2^{14}$This parameter set is the primary target of this work. For some intuition on how these parameters were selected, how the security level has been computed, for a second parameter set and a security proof in the random-oracle model we refer again to [14].

In general, the security of the signature scheme is based on the Decisional Compact Knapsack ($\text{DCK}_{p,n}$) problem and the hardness of finding a preimage in the hash function. For solving the DCK problem one has to distinguish between uniform samples from $\mathcal{R}^{p^n} \times \mathcal{R}^{p^n}$ and samples from the distribution $(a, as_1 + s_2)$ with a being chosen uniformly at random from $\mathcal{R}^{p^n}$ and s_1, s_2 being chosen uniformly at random from $\mathcal{R}_1^{p^n}$. In comparison to the Ring-LWE problem [18], where s_1, s_2 are chosen from a Gaussian distribution of a certain range, this just leads to s_1, s_2 with coefficients being either ± 1 or zero. Therefore, the DCK problem is an "aggressive" variant of the LWE problem but is not affected by the Arora-Ge algorithm as only one sample is given for the DCK problem and not the required polynomially-many [1]. Note also that extraction of the private key from the public key requires to solve the search variant of the DCK problem. In [14] the hardness of breaking the signature scheme for the implemented parameter set is computed based on the root Hermite factor of 1.0066 and stated to provide roughly 100 bits of security. Finding a preimage in the hash function has classical time complexity of 2^l but is lowered to $2^{l/2}$ by Grover's quantum algorithm [13]. As we use an output bit length of $l = 160$ from the hash function the implemented scheme achieves a security level of roughly 80 bits of security against attacks by a quantum computer.

Algorithm 5. Compression Algorithm Compress(y, z, p, k)

Input: $y \in \mathcal{R}_k^{p^n}$, $z \in \mathcal{R}_{k-32}^{p^n}$, p, k
Output: $z' \in \mathcal{R}_k^{p^n}$

$uncompressed \leftarrow 0$;
for i=0 **to** $n-1$ **do**
  if $|y[i]| > \frac{p-1}{2} - k$ **then**
    $z'[i] \leftarrow z[i]$;
    $uncompressed \leftarrow uncompressed + 1$;
  else
    write $y[i] = y[i]^{(1)}(2k+1) + y[i]^{(0)}$ where $-k \leq y[i]^{(0)} \leq k$
    if $y[i]^{(0)} + z[i] > k$ **then**
      $z[i]' \leftarrow k$;
    else if $y[i]^{(0)} + z[i] < -k$ **then**
      $z[i]' \leftarrow -k$;
    else
      $z[i]' \leftarrow 0$;
if $uncompressed \leq \frac{6kn}{p}$ **then**
  return z';
else
  return $\perp$;

3 Software Optimization

In this section we show our approach to high-level optimization of algorithms and low-level optimization to make best use of the target micro-architecture.

3.1 High-Level Optimization

In the following we present high-level ideas to speed-up the polynomial multiplication, runtime behavior as well as randomness generation.

Polynomial Multiplication. In order to achieve quasi-linear speed in $\mathcal{O}(n \log n)$ when performing the essential polynomial-multiplication operation we use the Fast Fourier Transform (FFT) or more specifically the Number Theoretic Transform (NTT) [21]. The advantages offered by the NTT have recently been shown by a hard- and software implementation of an ideal lattice-based public key cryptosystem [12]. The NTT is defined in a finite field or ring for a given primitive n-th root of unity ω. The generic forward $\mathrm{NTT}_\omega(a)$ of a sequence $\{a_0, .., a_{n-1}\}$ to $\{A_0, \ldots, A_{n-1}\}$ with elements in $\mathbb{Z}_p$ and length n is defined as $A_i = \sum_{j=0}^{n-1} a_j \omega^{ij} \bmod p,\ i = 0, 1, \ldots, n-1$ with the inverse $\mathrm{NTT}_\omega^{-1}(A)$ just using ω^{-1} instead of ω.

Algorithm 6. Hash Function Invocation $\mathrm{H}(x, m)$

Input: Polynomial $x \in \mathcal{R}^{p^n}$, message $m \in \{0,1\}^*$, hash function $\tilde{H}(\{0,1\}^*) \rightarrow \{0,1\}^{160}$
Output: $c \in \mathcal{R}_1^{p^n}$ with at most 32 coefficients being -1 or 1

```
r ← H̃(m||BinRep(x));
for i=0 to n − 1 do
    c[i] = 0;
for i=0 to 31 do
    pos ← 8 · r_{5i+3} + 4 · r_{5i+2} + 2 · r_{5i+1} + r_{5i};
    if r_{5i+4} = 0 then
        c[i · 16 + pos] ← −1;
    else
        c[i · 16 + pos] ← 1;
```

For lattice-based cryptography it is also convenient that most schemes are defined in $\mathbb{Z}_p[\mathbf{x}]/\langle x^n + 1\rangle$ and require reduction modulo $x^n + 1$. As a consequence, let ω be a primitive n-th root of unity in $\mathbb{Z}_p$ and $\psi^2 = \omega$. Then when $a = (a_0, \ldots a_{n-1})$ and $b = (b_0, \ldots b_{n-1})$ are vectors of length n with elements in $\mathbb{Z}_p$ let $d = (d_0, \ldots d_{n-1})$ be the negative wrapped convolution of a and b (thus $d = a * b \mod x^n + 1$). Let $\bar{a}, \bar{b}$ and $\bar{d}$ be defined as $(a_0, \psi a_1, \ldots, \psi^{n-1} a_{n-1})$, $(b_0, \psi b_1, \ldots, \psi^{n-1} b_{n-1})$, and $(d_0, \psi d_1, \ldots, \psi^{n-1} d_{n-1})$. It then holds that $\bar{d} = NTT_w^{-1}(NTT_w(\bar{a}) \circ NTT_w(\bar{b}))$ [24], where $\circ$ means componentwise multiplication. This avoids the doubling of the input length of the NTT and also gives us a modular reduction by $x^n + 1$ for free. If parameters are chosen such that n is a power of two and that $p \equiv 1 \bmod 2n$, the NTT exists and the negative wrapped convolution can be implemented efficiently.

In order to achieve high NTT performance, we precompute all constants $\omega^i, \omega^{-i}, \psi^i$ as well as $n^{-1} \cdot \psi^i$ for $i \in 0 \ldots n-1$. The multiplication by n^{-1}, which is necessary in the NTT^{-1} step, is directly performed as we just multiply by $n^{-1} \cdot \psi^{-i}$.

Storing Parameters in NTT Representation. The polynomial a is used as input to the key-generation algorithm and can be chosen as a global constant. By setting $\tilde{a} = \mathrm{NTT}(a)$ and storing $\tilde{a}$ we just need to perform $\mathrm{NTT}^{-1}(\tilde{a} \circ \mathrm{NTT}(y_1))$, which consists of one forward transform, one point multiplication and one backward transform. This is implemented in the `poly_mul_a` function and is superior to the general-purpose NTT multiplication, which requires three transforms.

Random Polynomials. During signature generation we need to generate two polynomials with random coefficients uniformly distributed in $[-k, k]$. To obtain these polynomials, we first generate $4 \cdot (n + 16) = 2112$ random bytes using the Salsa20 stream cipher [2] and a seed from the Linux kernel random-number generator `/dev/urandom`. We interprete these bytes as an array of $n+16$ unsigned 32-bit integers. To convert one such a 32-bit integer r to a polynomial coefficient c in $[-k, k]$ we first check whether $r \geq (2k+1) \cdot \lfloor 2^{32}/(2k+1) \rfloor$. If it is, we discard

this integer and move to the next integer in the array. Otherwise we compute $c = (r \mod (2k+1)) - k$.

The probability that an integer is discarded is $(2^{32} \mod (2k+1))/2^{32}$. For our parameters we have $(2^{32} \mod (2k+1)) = 4$. The probability to discard a randomly chosen 32-bit integer is thus $4/2^{32} = 2^{-30}$. The 16 additional elements in our array (corresponding to one block of Salsa20) make it extremely unlikely that we do not sample enough random elements to set all coefficients of the polynomial. In this highly unlikely case we simply sample another 2112 bytes of randomness.

During key generation we use the same approach to generate polynomials with coefficients in $\{-1, 0, 1\}$. The difference is that we sample bytes instead of 32-bit integers. We again sample one additional block of Salsa20 output, now corresponding to 64 additional elements. A byte is discarded only if its value is 255, the chance to discard a random byte is thus 2^{-8}.

3.2 Low-Level Optimization

The performance of the signature scheme is largely determined by a small set of operations on polynomials with $n = 512$ coefficients over $\mathbb{Z}_p$ where p is a 23-bit prime. This section first describes how we represent polynomials and what implementation techniques we use to accelerate operations on these polynomials.

Representation of Polynomials. We represent each 512-coefficient polynomial as an array of 512 double-precision floating-point values. Each such array is aligned on a 32-byte boundary, meaning that the address in memory is divisible by 32. This representation has the advantage that we can use the single-instruction multiple-data (SIMD) instructions of the AVX instruction-set extension in modern Intel and AMD CPUs. These instructions operate on vectors of 4 double-precision floats in 256-bit-wide, so called `ymm` vector registers. These registers and the corresponding AVX instructions can be found, for example, in the Intel Sandy Bridge, Intel Ivy Bridge, and AMD Bulldozer processors. The following performance analysis focuses on Ivy Bridge processors; Section 4 also reports benchmarks from a Sandy Bridge processor.

Both Sandy Bridge and Ivy Bridge processors can perform one AVX double-precision-vector multiplication and one addition every cycle. This corresponds to 4 multiplications (`vmulpd` instruction) and 4 additions (`vaddpd` instruction) of polynomial coefficients each cycle. However, arithmetic cost is not the main bottleneck in our software as loads and stores are often necessary because only 64 polynomial coefficients fit into the 16 available `ymm` registers. The performance of loads and stores is more complex to determine than arithmetic throughput. In principle, the processor can perform two loads and one store every two cycles. However, this maximal throughput can be reduced by bank conflicts. For details see [10, Section 8.13].

Modular Reduction of Coefficients. To perform a modular reduction of a coefficient x, we first compute $c = x \cdot \overline{p^{-1}}$, then round c, then multiply c by p and then subtract c from x. The first step uses a precomputed double-precision

approximation $\overline{p^{-1}}$ of the inverse of p. When reducing all coefficients of a polynomial, the multiplications and the subtraction are performed on four coefficients in parallel with the `vmulpd` and `vsubpd` AVX instructions, respectively. The rounding is also done on four coefficients in parallel using the `vroundpd` instruction. Note that depending on the rounding mode we can obtain the reduced value of x in different intervals. If we perform a truncation we obtain x in $[0, p-1]$, if we round to the nearest integer we obtain x in $[-((p-1)/2), (p-1)/2]$. We only need rounding to the nearest integer (`vroundpd` with rounding-mode constant `0x08`). Both representations are required at different stages of the computation; `vroundpd` supports choosing the rounding mode.

Lazy Reduction. The prime p has 23 bits. A double-precision floating-point value has a 53-bit mantissa and one sign bit. Even the product of two coefficients does not use the whole available precision, so we do not have to perform modular reduction after each addition, subtraction or even multiplication. We can thus make use of the technique known as *lazy reduction*, i.e., of performing reduction modulo p only when necessary.

Optimizing the NTT. The most speed-critical operation for signing is polynomial multiplication and we can thus use the NTT transformation as described above. We start from a standard fast iterative algorithm (see, e.g., [9]) for computing the FFT/NTT and adapt it to the target architecture. The transformation of a polynomial f with coefficients $f_0, \ldots, f_{511}$ to or from NTT representation consist of an initial permutation of the coefficients followed by $\log_2 n = 9$ levels of operations on coefficients. On level 0, pick up f_0 and f_1, multiply f_1 with a constant (a power of ω), add the result to f_0 to obtain the new value of f_0 and subtract the result from f_0 to obtain the new value of f_1. Then pick up f_2 and f_3 and perform the same operations to find the new values for f_2 and f_3 and so on. The following levels work in a similar way except that the distance of pairs of elements that are processed together is different: on level i process elements that are 2^i positions apart. For example, on level 2 pick up and transform f_0 and f_4, then f_1 and f_5 etc. On level 0 we can omit the multiplication by a constant, because the constant is 1.

The obvious bottleneck in this computation are additions (and subtractions): Each level performs 256 additions and 256 subtractions accounting for a total of $9 \cdot 512 = 4608$ additions requiring at least 1152 cycles. In fact the lower bound of cycles is much higher, because after each multiplication by a constant we need to reduce the coefficients modulo p. This takes one `vroundpd` instruction and one subtraction. The `vroundpd` instruction is processed in the same port as additions and subtractions, we thus get a lower bound of $(9 \cdot 512 + 8 \cdot 512)/4 = 2176$ cycles. To get close to this lower bound, we need to make sure that all the additions can be efficiently processed in AVX instructions by minimizing overhead from memory access, multiplications or vector-shuffle instructions.

Starting from level 2, the structure of the algorithm is very friendly for 4-way vector processing: For example, we can load (f_0, f_1, f_2, f_3) into one vector register, load (f_4, f_5, f_6, f_7) in another vector register, load the required constants (c_0, c_1, c_2, c_3) into a third vector register and then use one vector multiplication,

one vector addition and one vector subtraction to obtain $(f_0+c_0f_4, f_1+c_1f_5, f_2+c_2f_6, f_3+c_3f_7)$ and $(f_0-c_0f_4, f_1-c_1f_5, f_2-c_2f_6, f_3-c_3f_7)$. However, on levels 0 and 1 the transformations are not that straightforwardly done in vector registers. On level 0 we do the following: Load f_0, f_1, f_2, f_3 into one register; perform vector multiplication of this register with $(1, -1, 1, -1)$ and store the result in another register; perform a `vhaddpd` instruction of these two registers which results exactly in $(f_0 + v_1, f_0 - f_1, f_2 + f_3, f_2 - f_3)$. On level 1 we do the following: Load f_0, f_1, f_2, f_3; multiply with a vector of constants, reduce the result modulo p; use the `vperm2f128` instruction with constant argument `0x01` to obtain $c_2f_2, c_3f_3, c_0f_0, c_1f_1$ in another register and perform vector register multiplication of this register by $(1, 1, -1, -1)$; add the result to (f_0, f_1, f_2, f_3) to obtain the desired $(f_0 + c_2f_2, f_1 + c_1f_1, f_0 - c_2f_2, f_1 - c_3f_3)$.

A remaining bottleneck is memory access. To minimize loads and stores, we merge levels 0,1,2, levels 3,4,5 and levels 6,7,8. The idea is that on one level two pairs of coefficients are interacting; through two levels it is 4-tuples of coefficients that interact and through 3 levels it is 8-tuples of coefficients that interact. On levels 0,1 and 2 we load these 8 coefficients; perform all transformations through the 3 levels and store them again, then proceed to the next 8 coefficients. On higher levels we load 32 coefficients, perform all transformations through 3 levels on them, store them and then proceed to the next 32 coefficients.

In total, one NTT transformation takes 4484 cycles on the Ivy Bridge processor. This includes about 500 cycles for the initial coefficient permutation. We are continuing to investigate the difference between the lower bound on cycles dictated by vector additions and the cycles actually taken by our software.

Addition and Subtraction. Addition and subtraction of polynomials simply means loading coefficients, performing double-precision floating-point addition or subtraction, and storing the result coefficient. This is completely parallel, so we do this in 256 vector loads, 128 vector additions or subtractions, and 128 vector stores.

Higher-Order Transformation. The higher-order transformation described in Algorithm 4 is a nice example of the power of representing polynomial coefficients as double-precision floats: The only operation required is the multiplication by the precomputed value $\overline{(2(k-32)+1)^{-1}}$ (a double-precision approximation of $(2(k-32)+1)^{-1}$) and a subsequent rounding towards the nearest integer. As for the coefficient reduction we perform these computations using the `vmulpd` and `vroundpd` instructions.

4 Performance Analysis and Benchmarks

In this section we analyze the performance of our software and report benchmarks for key generation (`crypto_keypair`), as well as the signing (`crypto_sign`) and verification (`crypt_sign_open`) algorithm. Our software implements the eBATS API [4] for signature software, but we did *not* use SUPERCOP for benchmarking. The reason is that SUPERCOP reports the *median* of multiple runs

to filter out benchmarks that are polluted by, for example, an interrupt that occurred during some of the computations. Considering the median of timings when signing would be overly optimistic and cut off legitimate benchmarks of signature generations that took very long because they required many attempts. Therefore, for signing we report the *average* of 100000 signature generations; for key-pair generation, verification and lower-level functions we report the median of 1000 benchmarks. However, we will submit our software to eBACS for public benchmarking and discuss the issue with the editors of eBACS. Note that our software for signing is obviously not running in constant time but the timing variation is independent of secret data; our software is fully protected against timing attacks.

We performed benchmarks on two different machines:

- a machine called `h9ivy` at the University of Illinois at Chicago with an Intel Core i5-3210M CPU (Ivy Bridge) at 2500 MHz and 4 GB of RAM; and
- a machine called `h6sandy` at the University of Illinois at Chicago with an Intel Core i3-2310M CPU (Sandy Bridge) at 2100 MHz and 4 GB of RAM.

All software was compiled with gcc-4.7.2 and compiler flags `-O3 -msse2avx -march=corei7-avx -fomit-frame-pointer`. During the benchmarks Turbo-Boost and hyperthreading were switched off. The performance results for the most important operations are given in Table 1. The message length was 59 bytes for the benchmarking of `crypto_sign` and `crypto_sign_open`.

Table 1. Cycle counts of our software; $n = 512$ and $p = 8383489$

Operation	Sandy Bridge cycles	Ivy Bridge cycles
crypto_sign_keypair	33894	31140
crypto_sign	681500	634988
crypto_sign_open	47636	45036
ntt	4480	4484
poly_mul	16052	16096
poly_mul_a	11100	11044
poly_setrandom_maxk	12788	10824
poly_setrandom_max1	6072	5464

Polynomial-Multiplication Performance. The multiplication of two polynomials (`poly_mul`) takes 16096 cycles on the Ivy Bridge. Out of those, $3 \cdot 4484 = 13452$ cycles are for 3 NTT transformations (`ntt`).

Key-Generation Performance. Generating a key pair takes 31140 cycles on the Ivy Bridge. Out of those, $2 \cdot 5464 = 10928$ cycles are required to generate two random polynomials (`poly_setrandom_max1`); 11044 cycles are required for a multiplication by the constant system parameter a (`poly_mul_a`); the remaining

9168 cycles are required for one polynomial addition, compression of the two private-key polynomials and packing of the public-key polynomial into a byte array.

Signing Performance. Signing takes 634988 cycles on average on the Ivy Bridge. Each signing *attempt* takes 85384 cycles. We need 7 attempts on average, so those attempts account for about $7 \cdot 85384 = 597688$ cycles; the remaining cycles are required for constant overhead for extracting the private key from the byte array, copying the message to the signed message etc. *Some* of the remaining cycles may also be due to some measurements being polluted as explained above.

Out of the 85384 cycles for each signing attempt, $2 \cdot 10824 = 21648$ cycles are required to generate two random polynomials (`poly_setrandom_maxk`); $2 \cdot 16096 = 32192$ cycles are required for two polynomial multiplications; 11084 cycles are required for a multiplication with the system parameter a; the remaining 20460 cycles are required for hashing, the higher order transformation, four polynomial additions, one polynomial subtraction and testing whether the polynomial can be compressed.

Verification Performance. Verifying a signature takes 45036 cycles on the Ivy Bridge. Out of those, 16096 cycles are required for a polynomial multiplication; 11084 cycles are required for a multiplication with a; the remaining 17856 cycles are required for hashing, the high-order transformation, a polynomial addition and a polynomial subtraction, decompression of the signature, and unpacking of the public key from a byte array.

Comparison. As we provide the first software implementation of the signature scheme we cannot compare our result to other software implementations. In [14] only a hardware implementation is given which is naturally hard to compare to. For different types of FPGAs and parallelism, an implementation of sign/verify of 931/998 (Spartan-6 LX16) up to 12627/14580 (Virtex-6 LX130) messages/signatures per second is reported. However, the architecture is quite different; in particular it uses a configurable number of high-clock-frequency schoolbook multipliers instead of an NTT multiplier. The explanation for the low verification performance on the FPGA, compared with the software implementation, is that only one such multiplier is used in the verification engine.

Another target for comparison is a recently reported implementation of an ideal lattice-based encryption system in soft- and hardware [12]. In software, the necessary polynomial arithmetic relies on Shoup's NTL library [22]. Measurements confirmed that our basic arithmetic is faster than their prototype implementation (although their parameters are smaller) as we can rely on AVX, a hand-crafted NTT implementation and optimized modular reduction.

Various other implementations of post-quantum signature schemes have been described in the literature and many of them have been submitted to eBACS [4]. In Table 2 we compare our software in terms of security, speed, key sizes and signature size to the Rainbow, TTS, and C^* (pFLASH) software presented in [8], and the MQQ-Sig software presented in [11]. The cycle counts of these imple-

mentations are obtained from the eBACS website and have been measured on the same Intel Ivy Bridge machine that we used for benchmarking (`h9ivy`). We reference these implementations by their names in eBACS (in `typewriter` font) and their corresponding paper. For most of these multivariate schemes, the signing performance is much better, verification performance is somewhat better, but they suffer from excessive public-key sizes.

We furthermore compare to software described in the literature that has not been submitted to eBACS, specifically the implementation of the parallel-CFS code-based signature scheme presented in [16], the implementation of the treeless signature scheme TSS12 presented in [23], and the implementation of the hash-based signature scheme XMSS [6]. For those implementations we give the performance numbers from the respective paper and indicate the CPU used for benchmarking. Parallel-CFS not only has much larger keys, signing is also several orders of magnitude slower than with the lattice-based signature software presented in this paper. However, we expect that verification with parallel-CFS is very fast, but [16] does not give performance numbers for verification. The TSS software is using the scheme originally proposed in [17]. It makes an interesting target for comparison as it is similar to our scheme but relies on weaker assumptions. However, the software is much slower for both signing and verification. Hash-based signature schemes are also an interesting post-quantum signature alternative due to their well understood security properties and relatively small keys. However, the `XMSS` software presented in [6] is still an order of magnitude slower than our implementation and produces considerably larger signatures.

Finally we include two non-post-quantum signature schemes in the comparison in Table 2. First, the Ed25519 elliptic-curve signature scheme [3] and second, RSA-2048 signatures based on the OpenSSL implementation (`ronald2048`). Comparing to those schemes shows that our implementation and also most of the multivariate-signature software can even be faster or at least quite comparable to established schemes in terms of performance. However, the key and signature sizes of those two non-post-quantum signature are not beaten by any post-quantum proposal, yet.

Other lattice-based signature schemes that have a security reduction in the standard model are given in [7] and [5]. However, those papers do not give concrete parameters, security estimates or describe an implementation.

5 Future Work

As the initial implementation work has been carried out it is now necessary in future work to evaluate the security claims of the scheme by careful cryptanalysis and development of potential attacks. Especially, as the implemented scheme relaxes some assumptions that are required for connection to worst-case lattice problems more confidence is needed for real world usage. Other future work is the investigation of efficiency on more constrained devices like ARM (which, in some versions, also feature a SIMD unit) or even low-cost 8-bit processors.

Table 2. Comparison of different post-quantum signature software; **pk** stands for public key; **sk** stands for private key. The sizes are given in bytes. All software was benchmarked on `h9ivy` if not indicated otherwise.

Software	Security	Cycles		Sizes	
This work	100 bits	**sign:**	634988	**pk:**	1536
		verify:	45036	**sk:**	256
				sig:	1184
`mqqsig160` [12]	80 bits	**sign:**	1996	**pk:**	206112
		verify:	33220	**sk:**	401
				sig:	20
`mqqsig192` [12]	96 bits	**sign:**	3596	**pk:**	333540
		verify:	63488	**sk:**	465
				sig:	24
`mqqsig224` [12]	112 bits	**sign:**	3836	**pk:**	529242
		verify:	65988	**sk:**	529
				sig:	28
`mqqsig256` [12]	128 bits	**sign:**	4560	**pk:**	789552
		verify:	87904	**sk:**	593
				sig:	32
`rainbow5640` [9]	80 bits	**sign:**	53872	**pk:**	44160
		verify:	34808	**sk:**	86240
				sig:	37
`rainbowbinary16242020` [9]	80 bits	**sign:**	29364	**pk:**	102912
		verify:	17900	**sk:**	94384
				sig:	40
`rainbowbinary256181212` [9]	80 bits	**sign:**	33396	**pk:**	30240
		verify:	27456	**sk:**	23408
				sig:	42
`pflash1` [9]	80 bits	**sign:**	1473364	**pk:**	72124
		verify:	286168	**sk:**	5550
				sig:	37
`tts6440` [9]	80 bits	**sign:**	33728	**pk:**	57600
		verify:	49248	**sk:**	16608
				sig:	43
Parallel-CFS [17]	80 bits	**sign:**	4200000000[a]	**pk:**	20968300
$(20, 8, 10, 3)$		**verify:**	-	**sk:**	4194300
				sig:	75
TSS12 [24]	80 bits	**sign:**	93633000[b]	**pk:**	13087
$(n = 512)$		**verify:**	13064000[b]	**sk:**	13240
				sig:	8294
XMSS [7]	82 bits	**sign:**	7261100[c]	**pk:**	912
$(H = 20, w = 4,$ AES-128)		**verify:**	556600[c]	**sk:**	19
				sig:	2451
`ed25519` [4]	128 bits	**sign:**	67564	**pk:**	32
		verify:	209328	**sk:**	64
				sig:	64
`ronald2048`	112 bits	**sign:**	5768360	**pk:**	256
(RSA-2048 based on		**verify:**	77032	**sk:**	2048
OpenSSL)				**sig:**	256

[a] Benchmarked on an Intel Xeon W3670 (3.20 GHz)
[b] Benchmarked on an AMD Opteron 8356 (2.3 GHz)
[c] Benchmarked on an Intel i5-M540 (2.53 GHz)

Acknowledgments. We would like to thank Michael Schneider, Vadim Lyubashevsky, and the anonymous reviewers for their helpful comments.

References

1. Arora, S., Ge, R.: New Algorithms for Learning in Presence of Errors. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 403–415. Springer, Heidelberg (2011)
2. Bernstein, D.J.: The Salsa20 Family of Stream Ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 84–97. Springer, Heidelberg (2008)
3. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y.: High-speed high-security signatures. J. Cryptographic Engineering 2(2), 77–89 (2012)
4. Bernstein, D.J., Lange, T.: eBACS: ECRYPT benchmarking of cryptographic systems, `http://bench.cr.yp.to` (accessed January 25, 2013)
5. Boyen, X.: Lattice Mixing and Vanishing Trapdoors: A Framework for Fully Secure Short Signatures and More. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 499–517. Springer, Heidelberg (2010)
6. Buchmann, J., Dahmen, E., Hülsing, A.: XMSS - A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 117–129. Springer, Heidelberg (2011)
7. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai Trees, or How to Delegate a Lattice Basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)
8. Chen, A.I.-T., Chen, M.-S., Chen, T.-R., Cheng, C.-M., Ding, J., Kuo, E.L.-H., Lee, F.Y.-S., Yang, B.-Y.: SSE Implementation of Multivariate PKCs on Modern x86 CPUs. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 33–48. Springer, Heidelberg (2009)
9. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press (2009)
10. Fog, A.: The microarchitecture of Intel, AMD and VIA CPUs: An optimization guide for assembly programmers and compiler makers (2010), `http://www.agner.org/optimize/microarchitecture.pdf` (version February 29, 2012)
11. Gligoroski, D., Ødegård, R.S., Jensen, R.E., Perret, L., Faugère, J.-C., Knapskog, S.J., Markovski, S.: MQQ-SIG – an ultra-fast and provably CMA resistant digital signature scheme. In: Chen, L., Yung, M., Zhu, L. (eds.) INTRUST 2011. LNCS, vol. 7222, pp. 184–203. Springer, Heidelberg (2012)
12. Göttert, N., Feller, T., Schneider, M., Buchmann, J., Huss, S.: On the Design of Hardware Building Blocks for Modern Lattice-Based Encryption Schemes. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 512–529. Springer, Heidelberg (2012)
13. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Miller, G.L. (ed.) Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, pp. 212–219. ACM (1996)
14. Güneysu, T., Lyubashevsky, V., Pöppelmann, T.: Practical Lattice-Based Cryptography: A Signature Scheme for Embedded Systems. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 530–547. Springer, Heidelberg (2012)

15. Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J.H., Whyte, W.: NTRUSIGN: Digital Signatures Using the NTRU Lattice. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 122–140. Springer, Heidelberg (2003)
16. Landais, G., Sendrier, N.: Implementing CFS. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 474–488. Springer, Heidelberg (2012)
17. Lyubashevsky, V.: Lattice Signatures without Trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012)
18. Lyubashevsky, V., Peikert, C., Regev, O.: On Ideal Lattices and Learning with Errors over Rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010)
19. Nguyên, P.Q., Regev, O.: Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 271–288. Springer, Heidelberg (2006)
20. Petzoldt, A., Thomae, E., Bulygin, S., Wolf, C.: Small Public Keys and Fast Verification for $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Public Key Systems. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 475–490. Springer, Heidelberg (2011)
21. John, M.: Pollard. The Fast Fourier Transform in a finite field. Mathematics of Computation 25(114), 365–374 (1971)
22. Shoup, V.: NTL: A library for doing number theory, `http://www.shoup.net/ntl/` (accessed March 18, 2013)
23. Weiden, P., Hülsing, A., Cabarcas, D., Buchmann, J.: Instantiating treeless signature schemes. IACR Crptology ePrint archive report 2013/065 (2013), `http://eprint.iacr.org/2013/065`
24. Winkler, F.: Polynomial Algorithms in Computer Algebra (Texts and Monographs in Symbolic Computation). Springer, 1st edn. (1996)

Solving the Shortest Vector Problem in Lattices Faster Using Quantum Search

Thijs Laarhoven[1], Michele Mosca[2], and Joop van de Pol[3]

[1] Dept. of Mathematics and Computer Science, Eindhoven University of Technology
t.m.m.laarhoven@tue.nl
[2] Institute for Quantum Computing and Dept. of C&O, University of Waterloo
and
Perimeter Institute for Theoretical Physics
michele.mosca@uwaterloo.ca
[3] Dept. of Computer Science, University of Bristol
joop.vandepol@bristol.ac.uk

Abstract. By applying Grover's quantum search algorithm to the lattice algorithms of Micciancio and Voulgaris, Nguyen and Vidick, Wang et al., and Pujol and Stehlé, we obtain improved asymptotic quantum results for solving the shortest vector problem. With quantum computers we can provably find a shortest vector in time $2^{1.799n+o(n)}$, improving upon the classical time complexity of $2^{2.465n+o(n)}$ of Pujol and Stehlé and the $2^{2n+o(n)}$ of Micciancio and Voulgaris, while heuristically we expect to find a shortest vector in time $2^{0.312n+o(n)}$, improving upon the classical time complexity of $2^{0.384n+o(n)}$ of Wang et al. These quantum complexities will be an important guide for the selection of parameters for post-quantum cryptosystems based on the hardness of the shortest vector problem.

Keywords: lattices, shortest vector problem, sieving, quantum algorithms, quantum search.

1 Introduction

Large-scale quantum computers will redefine the landscape of computationally secure cryptography, including breaking public-key cryptography based on integer factorization or the discrete logarithm problem [57] or the Principle Ideal Problem in real quadratic number fields [25], providing sub-exponential attacks for some systems based on elliptic curve isogenies [16], speeding up exhaustive searching [9,23], counting [12] and (with appropriate assumptions about the computing architecture) finding collisions and claws [4, 11, 13], among many other quantum algorithmic speed-ups [15, 42, 58].

Currently, a small set of systems [8] are being studied intensely as possible systems to replace those broken by large-scale quantum computers. These systems can be implemented with conventional technologies and to date seem resistant to substantial quantum attacks. It is critical that these systems receive intense

P. Gaborit (Ed.): PQCrypto 2013, LNCS 7932, pp. 83–101, 2013.

scrutiny for possible quantum or classical attacks. This will boost confidence in the resistance of these systems to (quantum) attacks, and allow us to fine-tune secure choices of parameters in practical implementations of these systems.

One such set of systems bases its security on the computational hardness of certain lattice problems. Since the late 1990s, there has been a lot of research into the area of lattice-based cryptography, resulting in encryption schemes [27, 50], digital signature schemes [20, 39] and even fully homomorphic encryption schemes [10,21]. Each of the lattice problems that underpin the security of these systems can be reduced to the shortest vector problem. Conversely, the decisional variant of the shortest vector problem can be reduced to the average case of such lattice problems. For a more detailed summary on the security of lattice-based cryptography, see [35, 45].

In this paper, we closely study the best-known algorithms for solving the shortest vector problem on a lattice, and how quantum algorithms may speed up these algorithms. By challenging and improving the best asymptotic complexities of these algorithms, we increase the confidence in the security of lattice-based schemes. Understanding these algorithms is critical when selecting key-sizes and other security parameters.

1.1 Lattices

Lattices are discrete subgroups of $\mathbb{R}^n$. Given a set of n linearly independent vectors $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ in $\mathbb{R}^n$, we define the lattice generated by these vectors as $L = \{\sum_{i=1}^n \lambda_i \mathbf{b}_i : \lambda_i \in \mathbb{Z}\}$. We call the set B a basis of the lattice L. This basis is not unique; applying a unimodular matrix transformation to the vectors of B leads to a new basis B' of the same lattice L.

In lattices, we generally work with the Euclidean or ℓ_2-norm, which we will denote by $\|\cdot\|$. For bases B, we write $\|B\| = \max_i \|\mathbf{b}_i\|$. We refer to a vector $\mathbf{s} \in L \setminus \{\mathbf{0}\}$ such that $\|\mathbf{s}\| \leq \|\mathbf{v}\|$ for any $\mathbf{v} \in L \setminus \{\mathbf{0}\}$ as a shortest (non-zero) vector of the lattice. Its length is denoted by $\lambda_1(L)$. Given a basis B, we write $\mathcal{P}(B) = \{\sum_{i=1}^n \lambda_i \mathbf{b}_i : 0 \leq \lambda_i < 1\}$ for the fundamental domain of B.

One of the most important hard problems in the theory of lattices is the Shortest Vector Problem (SVP). Given a basis of a lattice, the Shortest Vector Problem consists of finding a shortest vector in this lattice. In many applications, finding a short vector instead of a shortest vector is also sufficient. The Approximate Shortest Vector Problem with approximation factor γ (SVP_γ) asks to find a non-zero lattice vector $\mathbf{v} \in L$ with length bounded from above by $\|\mathbf{v}\| \leq \gamma\lambda_1(L)$.

1.2 Related Work

The Approximate Shortest Vector problem is integral in the cryptanalysis of lattice-based cryptography [18]. For small values of γ, this problem is known to be NP-hard [2, 31], while for certain exponentially large γ, polynomial time algorithms exist, such as the LLL algorithm of Lenstra, Lenstra and Lovász [37]. Other algorithms trade extra running time for a better γ, such as LLL with deep insertions [55] and the BKZ algorithm of Schnorr and Euchner [55].

The current state-of-the-art for classically finding short vectors is BKZ 2.0 [14], which is essentially the original BKZ algorithm with the improved SVP subroutine of Gama et al. [19]. Implementations of this algorithm, due to Chen and Nguyen [14], and Aono and Naganuma [5], currently dominate the Lattice Challenge Hall of Fame [36]. The BKZ algorithm and its variants require a low-dimensional exact SVP solver as a subroutine. In theory, any of the known methods for finding a shortest vector could be used. For SVP solvers there is a similar online challenge [59], where the record is currently held by Kuo et al. [32].

In 2003, Ludwig [38] used quantum algorithms to speed up one such basis reduction algorithm, Random Sampling Reduction (RSR), which is due to Schnorr [56]. By replacing a random sampling from a big list by a quantum search, Ludwig achieves a quantum algorithm that is asymptotically faster than previous results. Ludwig also details the effect that this faster quantum algorithm would have had on the practical security of the lattice-based encryption scheme NTRU [27], had there been a quantum computer in 2005.

Enumeration. The classical method for finding shortest vectors is enumeration, dating back to work by Pohst [44], Kannan [30] and Fincke and Pohst [17] in the first half of the 1980s. In order to find a shortest vector, one enumerates all lattice vectors inside a giant ball around the origin. If the input basis is only LLL-reduced, enumeration runs in $2^{O(n^2)}$ time, where n is the lattice dimension. The algorithm by Kannan uses a stronger preprocessing of the input basis, and runs in $2^{O(n \log n)}$ time. Both approaches use only polynomial space in n.

Sieving/Saturation. In 2001, Ajtai et al. [3] introduced a technique called sieving, leading to the first probabilistic algorithm to solve SVP in time $2^{O(n)}$. Starting with a huge list of short vectors, the algorithm repeatedly applies a sieve to this list to end up with a smaller list of shorter lattice vectors. Eventually, we hope to be left with a list of lattice vectors of length $O(\lambda_1(L))$. Due to the size of the list, the space requirement of sieving is $2^{O(n)}$. Later work [26, 41, 43, 48] investigated the constants in both exponents and ways to reduce these.

Recently, in 2009, Micciancio and Voulgaris [41] started a new branch of sieving algorithms, which may be more appropriately called saturation algorithms. While sieving starts out with a long list and repeatedly applies a sieve to reduce its length, saturation algorithms iteratively add vectors to an initially empty list, hoping that at some point the space of short lattice vectors is "saturated", and two of the vectors in the list are at most $\lambda_1(L)$ apart. The time and space requirements of these algorithms are also $2^{O(n)}$. In 2009, Pujol and Stehlé [46] showed that with this method, SVP can provably be solved in time $2^{2.465n+o(n)}$.

Voronoi. In 2010, Micciancio and Voulgaris presented a deterministic algorithm for solving SVP based on constructing the Voronoi cell of the lattice [40]. In time $2^{2n+o(n)}$ and space $2^{n+o(n)}$, this algorithm is able to find a shortest vector in any lattice. Currently this is the best provable asymptotic result for classical SVP solvers.

Practice. While many methods have surpassed the enumeration algorithms in terms of classical provable asymptotic time complexities, in practice the enumeration methods still dominate the field. The version of enumeration that is currently used in practice is due to Schnorr and Euchner [55] with improvements by Gama et al. [19]. It does not incorporate the stronger version of preprocessing of Kannan [30] and hence has an asymptotic time complexity of $2^{O(n^2)}$. However, due to the small hidden constants in the exponents and the exponential space complexity of the other algorithms, enumeration is actually faster than other methods for common values of n. That said, the other methods are still quite new, so a further study of these other methods may tip the balance.

1.3 Quantum Search

In this paper we will study how quantum algorithms can be used to speed up the SVP algorithms outlined above. For this, we will make use of Grover's quantum search algorithm [23], which considers the following problem:

Given a list L of length N and a function $f : L \to \{0, 1\}$, such that the number of elements $e \in L$ with $f(e) = 1$ is small. Construct an algorithm "search" that, given L and f as input, returns an $e \in L$ with $f(e) = 1$, or determines that (with high probability) no such e exists. We assume for simplicity that f can be evaluated in unit time.

Classical Algorithm. With classical computers, the natural way to find such an element is to go through the whole list, until one of these elements is found. This takes on average $O(N)$ time. This is also optimal up to a constant factor; no classical algorithm can find such an element in less than $\Omega(N)$ time.

Quantum Algorithm. Using quantum search [9, 12, 23], we can find such an element in time $O(\sqrt{N})$. This is optimal up to a constant factor, as any quantum algorithm needs at least $\Omega(\sqrt{N})$ evaluations of f [6].

Throughout the paper, we will write $x \leftarrow \text{search}_{e \in L}(f(e) = 1)$ to highlight subroutines that perform a search in a long list. This assignment returns true if an element $e \in L$ with $f(e) = 1$ exists (and assigns such an element to x), and returns false if no such e exists. This allows us to give one description for both the classical and quantum versions of each algorithm, as the only difference between the two versions is which version of the subroutine is used.

For both of these classical and quantum algorithms, we assume a RAM model of computation where the jth entry of the list L can be looked up in constant time (or polylogarithmic time). In the case that L is a virtual list where the jth element can be computed in time polynomial in the length of j (thus polylogarithmic in the length of the list L), then look-up time is not an issue. When L is indeed an unstructured list of values, for classical computation, the assumption of a RAM-like model has usually been valid in practice. However, there are fundamental reasons for questioning it [7], and there are practical computing

architectures where the assumption does not apply. In the case of quantum computation, a practical RAM-like quantum memory (e.g. [22]) looks particularly challenging, especially for first generation quantum computers. Some authors have studied the limitations of quantum algorithms in this context [7, 24, 28].

Some algorithms (e.g. [4]) must store a large database of information in regular quantum memory (that is, memory capable of storing quantum superpositions of states). In contrast, quantum searching an actual list of N (classical) strings requires the N values to be stored in quantumly addressable classical memory (e.g. as Kuperberg discusses in [34]) and $O(\log N)$ regular qubits. Quantumly addressable classical memory in principle could be much easier to realize in practice than regular qubits. Furthermore, quantum searching for a value $x \in \{0,1\}^n$ satisfying $f(x) = 1$ for a function $f : \{0,1\}^n \to \{0,1\}$ which can be implemented by a circuit on $O(n)$ qubits only requires $O(n)$ regular qubits, and there is no actual list to be stored in memory. In this paper, the quantum search algorithms used require the lists of size N to be stored in quantumly addressable classical memory and use $O(\log N)$ regular qubits and $O(\sqrt{N})$ queries into the list of numbers.

In this work, we consider (conventional) classical RAM memories for the classical algorithms, and RAM-like quantumly addressable classical memories for the quantum search algorithms. This is both a first step for future studies in assessing the impact of more practical quantum architectures, and also represents a more conservative approach in determining parameter choices for lattice-based cryptography that should be resistant against the potential power of quantum algorithmic attacks. Future work may also find ways to take advantage of advanced quantum search techniques, such as those surveyed in [51].

1.4 Contributions and Outline

In this paper, we show that quantum algorithms can significantly speed up sieving and saturation algorithms. The constant in the exponent decreases by approximately 25% in all cases, leading to an improvement upon both provable and heuristic asymptotic results for solving the Shortest Vector Problem:

- Provably, we can find a shortest vector in any lattice in time $2^{1.799n+o(n)}$.
- Heuristically, we can find a shortest vector in any lattice in time $2^{0.312n+o(n)}$.
- Extrapolating from classical experiments, with quantum computers we expect to be able to find a shortest vector in any lattice in time about $2^{0.39n}$.

Table 1 contains a comparison between our contributions and previous results, in both the classical and quantum setting. While the Voronoi Cell algorithm is asymptotically the best algorithm in the provable classical setting, our quantum saturation algorithm has better asymptotics in the provable quantum setting.

Why do we only consider sieving and saturation algorithms, and not the more practical enumeration or the theoretically faster Voronoi cell algorithms? It turns out that it is not as simple to significantly speed up these algorithms using similar techniques. For some intuition why this is the case, see Appendix C.

Table 1. A comparison of the results as expressed in logarithmic leading order terms, with provable results above and heuristic results below

	Classical		Quantum		
Algorithm	Time	Space	Time	Space	
(Enumeration)	$O(n \log n)$	$O(\log n)$	-	-	(Appendix C)
Pujol and Stehlé [46]	$2.47n$	$1.24n$	$1.80n$	$1.29n$	(Section 3.1)
(Voronoi)	$2.00n$	$1.00n$	-	-	(Appendix C)
Micciancio and Voulgaris [41]	$0.52n$	$0.21n$	$0.39n$	$0.21n$	(Section 3.2)
Nguyen and Vidick [43]	$0.42n$	$0.21n$	$0.32n$	$0.21n$	(Section 2.1)
Wang et al. [60]	$0.39n$	$0.26n$	$0.32n$	$0.21n$	(Section 2.2)

The outline of this paper is as follows. In Section 2 we look at sieving algorithms, and how quantum algorithms lead to speed-ups. In Section 3, we look at saturation algorithms, and their estimated time and space complexities on a quantum computer. Technical details regarding some of these results can be found in Appendices A and B.

2 Sieving Algorithms

Sieving was first introduced by Ajtai et al. [3] and later improved theoretically [26, 41, 43, 48] and practically [43, 60] in various papers. In these algorithms, first an exponentially long list of lattice vectors is generated. Then, by iteratively applying a sieve to this list, the size of the list, as well as the lengths of the vectors in the list are reduced. After a polynomial number of applications of the sieve, we hope to be left with a short but non-empty list of very short vectors, from which we can then obtain a shortest vector of the lattice with high probability.

2.1 The Heuristic Algorithm of Nguyen and Vidick

Nguyen and Vidick [43] considered a heuristic, practical variant of the sieve algorithm of Ajtai et al. [3], which provably returns a shortest vector under a certain natural, heuristic assumption. A slightly modified but equivalent version of this algorithm is given in Algorithm 1.

Description of the Algorithm. The algorithm starts by generating a big list S of random lattice vectors with length at most $n\|B\|$. Then, by repeatedly applying a sieve to this list, shorter lists of shorter vectors are obtained, until the list is completely depleted. In that case, we go back one step, and look for the closest pair of lattice vectors in the last non-empty list.

The sieving step consists of splitting the previous list S_{prev} in a set of 'centers' C and a new list of vectors S that will be used for the next sieve. For each vector

Algorithm 1 The Heuristic Sieve Algorithm of Nguyen and Vidick

Input: An LLL-reduced basis B of L, and constants $\gamma \in (\frac{2}{3}, 1)$ and $N = 2^{O(n)}$
Output: A short non-zero lattice vector $\mathbf{s}$

1: $S \leftarrow \emptyset$
2: **for** $i \leftarrow 1$ **to** N **do**
3: $\quad \mathbf{v} \in_R B_n(\mathbf{0}, \|B\|) \cap L$
4: $\quad S \leftarrow S \cup \{\mathbf{v}\}$
5: **while** $S \setminus \{\mathbf{0}\} \neq \emptyset$ **do**
6: $\quad S_{\text{prev}} \leftarrow S \setminus \{\mathbf{0}\}$
7: $\quad R \leftarrow \max_{\mathbf{v} \in S_{\text{prev}}} \|\mathbf{v}\|$
8: $\quad C \leftarrow \{\mathbf{0}\}$
9: $\quad S \leftarrow \emptyset$
10: $\quad$ **for all** $\mathbf{v} \in S_{\text{prev}}$ **do**
11: $\quad\quad$ **if** $\mathbf{c} \leftarrow \text{search}_{\mathbf{c} \in C}(\|\mathbf{v} - \mathbf{c}\| \leq \gamma R)$ **then**
12: $\quad\quad\quad S \leftarrow S \cup \{\mathbf{v} - \mathbf{c}\}$
13: $\quad\quad$ **else**
14: $\quad\quad\quad C \leftarrow C \cup \{\mathbf{v}\}$
15: $\mathbf{s} \leftarrow \text{argmin}_{\mathbf{v} \in S_{\text{prev}}} \|\mathbf{v}\|$
16: **return** $\mathbf{s}$

$\mathbf{v}$ in S_{prev}, the algorithm first checks if a vector $\mathbf{c}$ in C exists that is close to $\mathbf{v}$. If this is the case, then we add the difference $\mathbf{v} - \mathbf{c}$ to S_{prev}. If this is not the case, then $\mathbf{v}$ is added to C. Since the set C consists of vectors with a bounded norm and a specified minimum distance between any two points, one can bound the size of C from above using a result of Kabatiansky and Levenshtein [29] regarding sphere packings. In other words, C will be sufficiently small, so that the list S will be sufficiently large. After applying the sieve, we discard all vectors in C and apply the sieve again to the vectors in $S_{\text{prev}} = S$.

At each iteration of the sieve, the maximum norm of the vectors in the list decreases from some constant R to at most γR, where γ is some geometric factor smaller than 1. Nguyen and Vidick conjecture that throughout the algorithm, the longest vectors in S are uniformly distributed over the space of all n-dimensional vectors with norms between γR and R.

Heuristic 1. *[43] At any stage of Algorithm 1, the vectors in $S \cap C_n(\gamma R, R)$ are uniformly distributed in $C_n(\gamma R, R)$, where $C_n(r_1, r_2) = \{\mathbf{x} \in \mathbb{R}^n : r_1 \leq \|\mathbf{x}\| \leq r_2\}$.*

Classical Complexities. In Line 11 of Algorithm 1, we have highlighted an application of a search subroutine that could be replaced by a quantum search. Using a standard classical search algorithm for this subroutine, under this heuristic assumption Nguyen and Vidick give the following estimate for the time and space complexity of their algorithm.

Lemma 1. *[43] On a classical computer, assuming that Heuristic 1 holds, Algorithm 1 will return a shortest vector of a lattice in time at most $2^{0.415n+o(n)}$ and space at most $2^{0.208n+o(n)}$.*

Quantum Complexities. If we use a quantum search subroutine in Line 11, the complexity of this subroutine decreases from $\tilde{O}(|C|)$ to $\tilde{O}(\sqrt{|C|})$. Since this search is part of the bottleneck for the time complexity, applying a quantum search here will decrease the running time significantly. Note that in Line 15, it also seems like a search of a list is performed. In reality, this final search of S_{prev} can be done in constant time by using appropriate data structures, e.g., by keeping the vectors in S and S_{prev} sorted from short to long, or by manually keeping track of the shortest vector in S.

Since replacing the classical search by a quantum search does not change the internal behaviour of the algorithm, the estimates and heuristics are as valid as they were in the classical setting. The time complexity does change, as the following theorem explains. For details, see Appendix A.

Theorem 1. *On a quantum computer, assuming that Heuristic 1 holds, Algorithm 1 will return a shortest vector of a lattice in time* $2^{0.312n+o(n)}$ *and space* $2^{0.208n+o(n)}$.

In other words, applying quantum search to Nguyen and Vidick's sieve algorithm leads to a 25% decrease in the exponent of the runtime.

2.2 The Heuristic Algorithm of Wang et al.

To improve upon the time complexity of the algorithm of Nguyen and Vidick, Wang et al. [60] introduced a further trade-off between the time complexity and the space complexity. Their algorithm uses two lists of centers C_1 and C_2 and two geometric factors γ_1 and γ_2, instead of the single list C and single geometric factor γ in the algorithm of Nguyen and Vidick. For details, see [60].

Classical Complexities. The classical time complexity of this algorithm is bounded from above by $\tilde{O}(|S| \cdot (|C_1| + |C_2|))$, while the space required is at most $O(|S| + |C_1| + |C_2|)$. Optimizing the constants γ_1 and γ_2 leads to $\gamma_1 = 1.0927$ and $\gamma_2 \to 1$, with an asymptotic time complexity of less than $2^{0.384n+o(n)}$ and a space complexity of about $2^{0.256n+o(n)}$.

Quantum Complexities. By using the quantum search algorithm for searching the lists C_1 and C_2, the time complexity is reduced to $\tilde{O}(|S| \cdot (\sqrt{|C_1|} + \sqrt{|C_2|}))$, while the space complexity remains $O(|S| + |C_1| + |C_2|)$. Re-optimizing the constants for a minimum time complexity leads to $\gamma_1 \to \sqrt{2}$ and $\gamma_2 \to 1$, leading to the same time and space complexities as the quantum-version of the algorithm of Nguyen and Vidick. Due to the simpler algorithm and smaller constants, a quantum version of the algorithm of Nguyen and Vidick will most likely be more efficient than a quantum version of the algorithm of Wang et al.

3 Saturation Algorithms

Saturation algorithms were only recently introduced by Micciancio and Voulgaris [41], and further studied by Pujol and Stehlé [46] and Schneider [52].

Algorithm 2 The Provable Saturation Algorithm of Pujol and Stehlé

Input: An LLL-reduced basis B of L, $\mu \simeq \lambda_1(L)$, $\xi > \frac{1}{2}$, $R > 2\xi$, $N_1^{\max}, N_2 = 2^{O(n)}$
Output: A non-zero lattice vector $\mathbf{s}$ of norm less than μ

$\gamma \leftarrow 1 - \frac{1}{n}$
$T \leftarrow \emptyset$
$N_1 \in_R [0, N_1^{\max} - 1]$
for $i \leftarrow 1$ **to** N_1 **do**
  $\mathbf{x} \in_R B_n(\mathbf{0}, \xi\mu)$
  $\mathbf{v}' \leftarrow \mathbf{x} \mod \mathcal{P}(B)$
  while $\mathbf{t} \leftarrow \text{search}_{\mathbf{t} \in T}(\|\mathbf{v}' - \mathbf{t}\| < \gamma\|\mathbf{v}'\|)$ **do**
    $\mathbf{v}' \leftarrow \mathbf{v}' - \mathbf{t}$
  $\mathbf{v} \leftarrow \mathbf{v}' - \mathbf{x}$
  if $\|\mathbf{v}\| \geq R\mu$ **then**
    $T \leftarrow T \cup \{\mathbf{v}\}$
$S \leftarrow \emptyset$
for $i \leftarrow 1$ **to** N_2 **do**
  $\mathbf{x} \in_R B_n(\mathbf{0}, \xi\mu)$
  $\mathbf{v}' \leftarrow \mathbf{x} \mod \mathcal{P}(B)$
  while $\mathbf{t} \leftarrow \text{search}_{\mathbf{t} \in T}(\|\mathbf{v}' - \mathbf{t}\| < \gamma\|\mathbf{v}'\|)$ **do**
    $\mathbf{v}' \leftarrow \mathbf{v}' - \mathbf{t}$
  $\mathbf{v} \leftarrow \mathbf{v}' - \mathbf{x}$
  $S \leftarrow S \cup \{\mathbf{v}\}$
$\{\mathbf{s}_1, \mathbf{s}_2\} \leftarrow \text{search}_{\{\mathbf{s}_1, \mathbf{s}_2\} \in S \times S}(0 < \|\mathbf{s}_1 - \mathbf{s}_2\| < \mu)$
return $\mathbf{s}_1 - \mathbf{s}_2$

Instead of starting with a huge list and making the list smaller and smaller, this method starts with a small or empty list, and keeps adding more and more vectors to the list. Building upon the same result of Kabatiansky and Levenshtein about sphere packings [29], we know that if the list reaches a certain size and all vectors have a norm bounded by a sufficiently small constant, two of the vectors in the list must be close to one another. Thus, if we can guarantee that new short lattice vectors keep getting added to the list, then at some point, with high probability, we can find a shortest vector as the difference between two of the list vectors.

3.1 The Provable Algorithm of Pujol and Stehlé

Using the Birthday paradox, Pujol and Stehlé [46] showed that the constant in the exponent of the time complexity of the original algorithm of Micciancio and Voulgaris [41, Section 3.1] can be reduced by almost 25%. The algorithm is presented in Algorithm 2.

Description of the Algorithm. The algorithm can roughly be divided in three stages, as follows.

First, the algorithm generates a long list T of lattice vectors with norms between $R\mu$ and $\|B\|$. This 'dummy' list is only used for technical reasons, and

in practice one does not seem to need such a list. Note that besides the actual lattice vectors $\mathbf{v}$, to generate this list we also consider slightly perturbed vectors $\mathbf{v}'$ which are not in the lattice, but are at most $r\mu$ away from $\mathbf{v}$. This is purely a technical modification to make the proofs work, as experiments show that without such perturbed vectors, saturation algorithms also work fine [40,46,52].

After generating T, we generate a fresh list of short lattice vectors S. The procedure for generating these vectors is similar to that of generating T, with two exceptions: (i) now all sampled lattice vectors are added to S (regardless of their norms), and (ii) the vectors are reduced with the dummy list T rather than with vectors in S. The latter guarantees that the vectors in S are i.i.d.

Finally, when S has been generated, we hope that it contains two distinct lattice vectors $\mathbf{s}_1$, $\mathbf{s}_2$ that are at most μ apart. So we search $S \times S$ for a pair $\{\mathbf{s}_1, \mathbf{s}_2\}$ of close, distinct lattice vectors, and return their difference.

Classical Complexities. With a classical search applied to the subroutines in Lines 7, 16, and 20, Pujol and Stehlé obtained the following results.

Lemma 2. *[46] Let $\xi \approx 0.9476$ and $R \approx 3.0169$. Then, using polynomially many queries to Algorithm 2, we can find a shortest vector in a lattice with probability exponentially close to 1, using time at most $2^{2.465n+o(n)}$ and space at most $2^{1.233n+o(n)}$.*

Quantum Complexities. Applying a quantum search algorithm to the search-subroutines in Lines 7, 16, and 20 leads to the following result. Details are given in Appendix B.

Theorem 2. *Let $\xi \approx 0.9086$ and $R \approx 3.1376$. Then, using polynomially many queries to the quantum version of Algorithm 2, we can find a shortest vector in a lattice with probability exponentially close to 1, using time at most $2^{1.799n+o(n)}$ and space at most $2^{1.286n+o(n)}$.*

So the constant in the exponent of the time complexity decreases by about 27% when using quantum search.

Remark. If we generate S in parallel, we can potentially achieve a time complexity of $2^{1.470n+o(n)}$, by setting $\xi \approx 1.0610$ and $R \approx 4.5166$. However, it would require exponentially many parallel quantum computers of size $O(n)$ to achieve a substantial theoretical speed-up over the $2^{1.799n+o(n)}$ of Theorem 2. (Recall that quantum searching a list of c^n elements (with $c > 1$) requires the list to be stored in quantumly addressable classical memory (versus regular quantum memory) and otherwise can be searched using only $O(n)$ qubits and $O(c^{n/2})$ queries to the list.)

3.2 The Heuristic Algorithm of Micciancio and Voulgaris

In practice, just like sieving algorithms, saturation algorithms are much faster than their worst-case running times and provable time complexities suggest.

Algorithm 3 The Heuristic Saturation Algorithm of Micciancio and Voulgaris

Input: An LLL-reduced basis B of L, and a constant C_0
Output: A short non-zero lattice vector $\mathbf{s}$

$S \leftarrow \{\mathbf{0}\}$
$Q \leftarrow \emptyset$
$c \leftarrow 0$
while $c < C_0$ **do**
  if $Q \neq \emptyset$ **then**
    $\mathbf{v} \in_R Q$
    $Q \leftarrow Q \setminus \{\mathbf{v}\}$
  else
    $\mathbf{v} \in_R B_n(\mathbf{0}, \|B\|) \cap L$
  while $\mathbf{s} \leftarrow \text{search}_{\mathbf{s} \in S}(\max\{\|\mathbf{s}\|, \|\mathbf{v} - \mathbf{s}\|\} \leq \|\mathbf{v}\|)$ **do**
    $\mathbf{v} \leftarrow \mathbf{v} - \mathbf{s}$
  while $\mathbf{s} \leftarrow \text{search}_{\mathbf{s} \in S}(\max\{\|\mathbf{v}\|, \|\mathbf{v} - \mathbf{s}\|\} \leq \|\mathbf{s}\|)$ **do**
    $S \leftarrow S \setminus \{\mathbf{s}\}$
    $Q \leftarrow Q \cup \{\mathbf{v} - \mathbf{s}\}$
  if $\mathbf{v} = \mathbf{0}$ **then**
    $c \leftarrow c + 1$
  else
    $S \leftarrow S \cup \{\mathbf{v}\}$
$\mathbf{s} \leftarrow \text{argmin}_{\mathbf{v} \in S \setminus \{\mathbf{0}\}} \|\mathbf{v}\|$
return $\mathbf{s}$

Micciancio and Voulgaris [41] gave a heuristic variant of their saturation algorithm, for which they could not give a (heuristic) bound on the time complexity, but with a better bound on the space complexity, and a better practical time complexity. The algorithm is given in Algorithm 3.

Description of the Algorithm. The algorithm is similar to Algorithm 2, with the following main differences: (i) we do not explicitly generate two lists S, T to apply the birthday paradox; (ii) we do not use the geometric factor $\gamma < 1$ but always reduce a vector if it can be reduced; (iii) we also reduce the existing list vectors with newly sampled vectors, so that each two vectors in the list are pairwise Gauss-reduced; and (iv) instead of specifying the number of iterations, we run the algorithm until we reach a predefined number of collisions C_0.

Classical Complexities. Micciancio and Voulgaris state that the algorithm above has an experimental time complexity of about $2^{0.52n}$ and a space complexity which is most likely bounded from above by $2^{0.208n}$ due to the kissing constant [41, Section 5]. This is much faster than the theoretical time complexity of $2^{1.799n}$ of the quantum-enhanced saturation algorithm discussed in Section 3.1.

Remark 1. In practice, the algorithm of Micciancio and Voulgaris is faster than the one of Nguyen and Vidick of Section 2.1, even though the leading term in

the exponent is larger. So asymptotically, this algorithm is dominated by the algorithm of Nguyen and Vidick, but in practice and for small dimensions, the algorithm of Micciancio and Voulgaris seems to perform better.

Remark 2. Schneider states [52] that the time complexity roughly scales like $2^{0.57n-23.5}$, instead of the $2^{0.52n}$ claimed by Micciancio and Voulgaris. Although asymptotically this time complexity is worse than the one of Micciancio and Voulgaris, the cross-over point of these rough approximations is around $n \approx 470$. So for most values of n that SVP solvers handle in practice, the term -23.5 is more significant than the small increase caused by n, and the conjectured time complexity of Schneider is better than that of Micciancio and Voulgaris.

Quantum Complexities. To this heuristic algorithm, the quantum speed-ups can also be applied. Generally, these saturation algorithms generate a list S of reasonably short lattice vectors by (i) first sampling a long, random lattice vector $\mathbf{v} \in L$; (ii) reducing the vector $\mathbf{v}$ with lattice vectors already in S; (iii) possibly reducing the vectors in S with this new vector $\mathbf{v}$; and (iv) finally adding $\mathbf{v}$ to S. The total classical time complexity of these algorithms is of the order $|S|^2$ due to (ii) and (iii), but by applying quantum speed-ups to these steps, this becomes $|S|^{3/2}$. This means that the exponent in the time complexity is generally reduced by about 25%, which is comparable to the improvement in Section 3.1. In practice, we therefore expect a time complexity of about $2^{0.39n}$ for the heuristic algorithm of Micciancio and Voulgaris with quantum search speed-ups, with constants that may make this algorithm faster than the sieving algorithm of Section 2.1.

Acknowledgments. This report is partly a result of fruitful discussions at the Lorentz Center Workshop on Post-Quantum Cryptography and Quantum Algorithms, Nov. 5–9, Leiden, The Netherlands. In particular, we would like to thank Felix Fontein, Nadia Heninger, Stacey Jeffery, Stephen Jordan, Michael Schneider, Damien Stehlé and Benne de Weger for the valuable discussions there. Finally, we thank the anonymous reviewers for their helpful comments and suggestions.

The first author is supported by DIAMANT and ECRYPT II (ICT-2007-216676). The second author is supported by Canada's NSERC (Discovery, SPG FREQUENCY, and CREATE CryptoWorks21), MPrime, CIFAR, ORF and CFI; IQC and Perimeter Institute are supported in part by the Government of Canada and the Province of Ontario. The third author is supported in part by EPSRC via grant EP/I03126X.

References

1. Aharonov, D., Regev, O.: A Lattice Problem in Quantum NP. In: 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 210–219. IEEE Press, New York (2003)

2. Ajtai, M.: The Shortest Vector Problem in L_2 is NP-hard for Randomized Reductions. In: 30th Annual ACM Symposium on Theory of Computing (STOC), pp. 10–19. ACM, New York (1998)
3. Ajtai, M., Kumar, R., Sivakumar, D.: A Sieve Algorithm for the Shortest Lattice Vector Problem. In: 33rd Annual ACM Symposium on Theory of Computing (STOC), pp. 601–610. ACM, New York (2001)
4. Ambainis, A.: Quantum Walk Algorithm for Element Distinctness. In: 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 22–31. IEEE Press, New York (2003)
5. Aono, Y., Naganuma, K.: Heuristic Improvements of BKZ 2.0. IEICE Tech. Rep. 112(211), 15–22 (2012)
6. Bennett, C.H., Bernstein, E., Brassard, G., Vazirani, V.: Strengths and Weaknesses of Quantum Computing. SIAM J. Comput. 26(5), 1510–1523 (1997)
7. Bernstein, D.J.: Cost analysis of hash collisions: Will quantum computers make SHARCs obsolete? In: SHARCS 2009: Special-purpose Hardware for Attacking Cryptographic Systems (2009)
8. Buchmann, J., Ding, J. (eds.): PQCrypto 2008. LNCS, vol. 5299. Springer, Heidelberg (2008)
9. Boyer, M., Brassard, G., Høyer, P., Tapp, A.: Tight Bounds on Quantum Searching. Fortschritte der Physik 46, 493–505 (1998)
10. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) Innovations in Theoretical Computer Science, ITCS 2012, pp. 309–325. ACM (2012)
11. Brassard, G., Høyer, P., Tapp, A.: Quantum cryptanalysis of hash and claw-free functions. In: Lucchesi, C.L., Moura, A.V. (eds.) LATIN 1998. LNCS, vol. 1380, pp. 163–169. Springer, Heidelberg (1998)
12. Brassard, G., Høyer, P., Mosca, M., Tapp, A.: Quantum Amplitude Amplification and Estimation. AMS Contemporary Mathematics Series Millennium Vol. entitled Quantum Computation & Information, vol. 305 (2002)
13. Buhrman, B., Dürr, C., Heiligman, M., Høyer, P., Magniez, F., Santha, M., de Wolf, R.: Quantum Algorithms for Element Distinctness. SIAM J. Comput. 34(6), 1324–1330 (2005)
14. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better Lattice Security Estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011)
15. Childs, A., Van Dam, W.: Quantum algorithms for algebraic problems. Rev. Mod. Phys. 82, 1–52 (2010)
16. Childs, A.M., Jao, D., Soukharev, V.: Constructing elliptic curve isogenies in quantum subexponential time. arXiv:1012.4019 (2010)
17. Fincke, U., Pohst, M.: Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. Math. Comp. 44, 463–471 (1985)
18. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008)
19. Gama, N., Nguyen, P.Q., Regev, O.: Lattice Enumeration Using Extreme Pruning. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 257–278. Springer, Heidelberg (2010)
20. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Dwork, C. (ed.) STOC 2008, pp. 197–206. ACM (2008)
21. Gentry, C.: A fully homomorphic encryption scheme (Doctoral dissertation, Stanford University) (2009)

22. Giovannetti, V., Lloyd, S., Maccone, L.: Quantum Random Access Memory. Phys. Rev. Lett. 100, 160501 (2008)
23. Grover, L.K.: A Fast Quantum Mechanical Algorithm for Database Search. In: 28th Annual ACM Symposium on Theory of Computing (STOC), pp. 212–219. ACM, New York (1996)
24. Grover, L., Rudolph, T.: How significant are the known collision and element distinctness quantum algorithms? Quantum Info. Comput. 4(3), 201–206 (2004)
25. Hallgren, S.: Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem. J. ACM. 54(1), 653–658 (2007)
26. Hanrot, G., Pujol, X., Stehlé, D.: Algorithms for the Shortest and Closest Lattice Vector Problems. In: Chee, Y.M., Guo, Z., Ling, S., Shao, F., Tang, Y., Wang, H., Xing, C. (eds.) IWCC 2011. LNCS, vol. 6639, pp. 159–190. Springer, Heidelberg (2011)
27. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998)
28. Jeffery, S.: Collision Finding with Many Classical or Quantum Processors. Master's thesis, University of Waterloo (2011)
29. Kabatiansky, G., Levenshtein, V.I.: On Bounds for Packings on a Sphere and in Space. Problemy Peredachi Informacii 14(1), 3–25 (1978)
30. Kannan, R.: Improved Algorithms for Integer Programming and Related Lattice Problems. In: 15th Annual ACM Symposium on Theory of Computing (STOC), pp. 193–206. ACM, New York (1983)
31. Khot, S.: Hardness of approximating the shortest vector problem in lattices. Journal of the ACM 52(5), 789–808 (2005)
32. Kuo, P.C., Schneider, M., Dagdelen, Ö., Reichelt, J., Buchmann, J., Cheng, C.M., Yang, B.Y.: Extreme Enumeration on GPU and in Clouds. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 176–191. Springer, Heidelberg (2011)
33. Kuperberg, G.: A Subexponential-Time Quantum Algorithm for the Dihedral Hidden Subgroup Problem. SIAM J. Comput. 35(1), 170–188 (2005)
34. Kuperberg, G.: Another Subexponential-Time Quantum Algorithm for the Dihedral Hidden Subgroup Problem. arXiv, Report 1112/3333, pp. 1–10 (2011)
35. Laarhoven, T., van de Pol, J., de Weger, B.: Solving Hard Lattice Problems and the Security of Lattice-Based Cryptosystems. Cryptology ePrint Archive, Report 2012/533, pp. 1–43 (2012)
36. TU Darmstadt Lattice Challenge, `http://www.latticechallenge.org/`
37. Lenstra, A.K., Lenstra, H., Lovász, L.: Factoring Polynomials with Rational Coefficients. Math. Ann. 261(4), 515–534 (1982)
38. Ludwig, C.: A Faster Lattice Reduction Method Using Quantum Search. In: Ibaraki, T., Katoh, N., Ono, H. (eds.) ISAAC 2003. LNCS, vol. 2906, pp. 199–208. Springer, Heidelberg (2003)
39. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012)
40. Micciancio, D., Voulgaris, P.: A Deterministic Single Exponential Time Algorithm for Most Lattice Problems based on Voronoi Cell Computations. In: 42nd Annual ACM Symposium on Theory of Computing (STOC), pp. 351–358. ACM, New York (2010)
41. Micciancio, D., Voulgaris, P.: Faster Exponential Time Algorithms for the Shortest Vector Problem. In: 21st Annual ACM Symposium on Discrete Algorithms (SODA), pp. 1468–1480. ACM, New York (2010)

42. Mosca, M.: Quantum Algorithms. In: Meyers, R. (ed.) Encyclopedia of Complexity and Systems Science (2009)
43. Nguyen, P.Q., Vidick, T.: Sieve Algorithms for the Shortest Vector Problem are Practical. J. Math. Crypt. 2(2), 181–207 (2008)
44. Pohst, M.: On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. ACM SIGSAM Bulletin 15(1), 37–44 (1981)
45. van de Pol, J.: Lattice-based cryptography. Master's thesis. Eindhoven University of Technology (2011)
46. Pujol, X., Stehlé, D.: Solving the Shortest Lattice Vector Problem in Time $2^{2.465n}$. Cryptology ePrint Archive, Report 2009/605, pp. 1–7 (2009)
47. Regev, O.: A Subexponential Time Algorithm for the Dihedral Hidden Subgroup Problem with Polynomial Space. arXiv, Report 0405/151, pp. 1–7 (2004)
48. Regev, O.: Lattices in Computer Science. Lecture Notes for a Course at the Tel Aviv University (2004)
49. Regev, O.: Quantum Computation and Lattice Problems. SIAM J. Comput. 33(3), 738–760 (2004)
50. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: 37th Annual ACM Symposium on Theory of Computing (STOC), pp. 84–93 (2005)
51. Santha, M.: Quantum Walk Based Search Algorithms. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 31–46. Springer, Heidelberg (2008)
52. Schneider, M.: Analysis of Gauss-Sieve for Solving the Shortest Vector Problem in Lattices. In: Katoh, N., Kumar, A. (eds.) WALCOM 2011. LNCS, vol. 6552, pp. 89–97. Springer, Heidelberg (2011)
53. Schneider, M.: Sieving for Short Vectors in Ideal Lattices. Cryptology ePrint Archive, Report 2011/458, pp. 1–19 (2011)
54. Schnorr, C.P.: A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms. Theoretical Computer Science 53(2-3), 201–224 (1987)
55. Schnorr, C.P., Euchner, M.: Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems. Mathematical Programming 66(2-3), 181–199 (1994)
56. Schnorr, C.P.: Lattice reduction by random sampling and birthday methods. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 145–156. Springer, Heidelberg (2003)
57. Shor, P.W.: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM J. Comput. 26(5), 1484–1509 (1997)
58. Smith, J.: Mosca. M.: Algorithms for Quantum Computers. In: Handbook of Natural Computing, pp. 1451–1492. Springer (2012)
59. SVP Challenge, `http://latticechallenge.org/svp-challenge/`
60. Wang, X., Liu, M., Tian, C., Bi, J.: Improved Nguyen-Vidick Heuristic Sieve Algorithm for Shortest Vector Problem. In: 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS), pp. 1–9. ACM, New York (2011)

A Analysis of the Sieve Algorithm of Nguyen and Vidick

Nguyen and Vidick showed that if their heuristic assumption holds, the time and space complexities of their algorithm can be bounded from above as follows.

Lemma 3. *[43] On a classical computer, assuming Heuristic 1 holds, Algorithm 1 will return a shortest vector of a lattice in time* $2^{2c_h n+o(n)}$ *and space* $2^{c_h n+o(n)}$*, where* $\frac{2}{3} < \gamma < 1$ *and*

$$c_h = -\log_2(\gamma) - \frac{1}{2}\log_2\left(1 - \frac{\gamma^2}{4}\right). \tag{1}$$

To obtain a minimum time complexity, γ should be chosen as close to 1 as possible. Letting $\gamma \to 1$ leads to an asymptotic time complexity of less than $2^{0.415n+o(n)}$ and an asymptotic space complexity of less than $2^{0.208n+o(n)}$.

To obtain these estimates, it is first noted that the sizes of S and C are bounded from above by $2^{c_h n+o(n)}$. The space complexity is therefore bounded from above by $O(|S| + |C|) = 2^{c_h n+o(n)}$, and since for every element in S the algorithm has to search the list C, the time complexity is bounded from above by $\tilde{O}(|S| \cdot |C|) = 2^{2c_h n+o(n)}$.

Using quantum search on the list C, the time complexity decreases to $\tilde{O}(|S| \cdot \sqrt{|C|}) = 2^{\frac{3}{2}c_h n+o(n)}$, while the space complexity remains the same. This leads to the following result.

Lemma 4. *On a quantum computer, assuming Heuristic 1 holds, Algorithm 1 will return a shortest vector of a lattice in time* $2^{\frac{3}{2}c_h n+o(n)}$ *and space* $2^{c_h n+o(n)}$*.*

Optimizing γ to obtain a minimum time complexity again corresponds to letting γ tend to 1 from below, leading to an asymptotic time complexity of $2^{0.312n+o(n)}$ and space complexity of $2^{0.208n+o(n)}$, as stated in Theorem 1.

B Analysis of the Saturation Algorithm of Pujol and Stehlé

In the classical setting, the time complexities of the different parts of the algorithm are as follows. The constants are explained in the lemma below.

- Cost of generating T: $\tilde{O}(N_1^{\max} \cdot |T|) = 2^{(c_g+2c_t)n+o(n)}$.
- Cost of generating S: $\tilde{O}(N_2 \cdot |T|) = 2^{(c_g+c_b/2+c_t)n+o(n)}$.
- Cost of searching S for a pair of close vectors: $\tilde{O}(|S|^2) = 2^{(2c_g+c_b)n+o(n)}$.

The space complexity is at most $O(|T| + |S|) = 2^{\max(c_t, c_g+c_b/2)n+o(n)}$. In [46], this lead to the following lemma.

Lemma 5. *[46] Let $\xi > \frac{1}{2}$ and $R > 2\xi$, and suppose $\mu > \lambda_1(L)$. Then, with c_b, c_t, c_g, N_B, N_V, N_G, $N_1^{\max}$, N_2 chosen according to:*

$$c_b = \log_2(R) + 0.401, \qquad N_B = 2^{c_b n + o(n)}, \tag{2}$$

$$c_t = \frac{1}{2}\log_2\left(1 + \frac{2\xi}{R - 2\xi}\right) + 0.401, \qquad N_T = 2^{c_t n + o(n)}, \tag{3}$$

$$c_g = \frac{1}{2}\log_2\left(\frac{4\xi^2}{4\xi^2 - 1}\right), \qquad N_G = 2^{c_g n + o(n)}, \tag{4}$$

$$N_1^{\max} = 2^{(c_g + c_t)n + o(n)}, \qquad N_2 = 2^{(c_g + c_b/2)n + o(n)}, \tag{5}$$

with probability at least $\frac{1}{16}$, Algorithm 2 returns a lattice vector $\mathbf{s} \in L \setminus \{\mathbf{0}\}$ with $\|\mathbf{s}\| < \mu$, in time at most $2^{tn+o(n)}$ and space at most $2^{sn+o(n)}$, where t and s are given by

$$t = \max\left(c_g + 2c_t, c_g + \frac{c_b}{2} + c_t, 2c_g + c_b\right), \quad s = \max\left(c_t, c_g + \frac{c_b}{2}\right). \tag{6}$$

In the quantum setting, the costs are as follows.

- Cost of generating T: $\tilde{O}(N_1^{\max} \cdot \sqrt{|T|}) = 2^{(c_g + 3c_t/2)n + o(n)}$.
- Cost of generating S: $\tilde{O}(N_2 \cdot \sqrt{|T|}) = 2^{(c_g + c_b/2 + c_t/2)n + o(n)}$.
- Cost of searching S for a pair of close vectors: $\tilde{O}(\sqrt{|S|^2}) = 2^{(c_g + c_b/2)n + o(n)}$.

The total space complexity is still the same as in the classical setting, i.e., at most $O(|T| + |S|) = 2^{\max(c_t, c_g + c_b/2)n + o(n)}$. This leads to the following lemma.

Lemma 6. *Let $\xi > \frac{1}{2}$ and $R > 2\xi$, and suppose $\mu > \lambda_1(L)$. Then, with c_b, c_t, c_g, N_B, N_V, N_G, $N_1^{\max}$, N_2 chosen according to Equations (2) to (5), with probability at least $\frac{1}{16}$, Algorithm 2 returns a lattice vector $\mathbf{s} \in L \setminus \{\mathbf{0}\}$ with $\|\mathbf{s}\| < \mu$ on a quantum computer in time at most $2^{\tilde{t}n+o(n)}$ and space at most $2^{\tilde{s}n+o(n)}$, where $\tilde{t}$ and $\tilde{s}$ are given by*

$$\tilde{t} = \max\left(c_g + \frac{3c_t}{2}, c_g + \frac{c_b}{2} + \frac{c_t}{2}, c_g + \frac{c_b}{2}\right), \quad \tilde{s} = \max\left(c_t, c_g + \frac{c_b}{2}\right). \tag{7}$$

Optimizing ξ and R for the minimum time complexity, we get $\xi \approx 0.9086$ and $R \approx 3.1376$ as in Theorem 2. Note that if S is generated in parallel with exponentially many quantum computers, the cost of the second part of the algorithm becomes negligible, and the exponent in the time complexity changes to

$$\tilde{t}' = \max\left(c_g + \frac{3c_t}{2}, c_g + \frac{c_b}{2}\right). \tag{8}$$

In that case, the optimal choice of ξ and R (with respect to minimizing the time complexity) would be $\xi \approx 1.0610$ and $R \approx 4.5166$, leading to a time complexity of less than $2^{1.470n+o(n)}$.

C Other SVP Algorithms

C.1 Enumeration

Recall that enumeration considers all lattice vectors inside a giant ball around the origin that is known to contain at least one lattice vector. Let L be a lattice with basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$. Consider each lattice vector $\mathbf{u} \in L$ as a linear combination of the basis vectors, i.e., $\mathbf{u} = \sum_i u_i \mathbf{b}_i$. Now, we can represent each lattice vector by its coefficient vector $(u_1, \ldots, u_n)$. We would like to have all combinations of values for $(u_1, \ldots, u_n)$ such that the corresponding vector $\mathbf{u}$ lies in the ball. We could try any combination and see if it lies within the ball by computing the norm of the corresponding vector, but there is a smarter way that ensures we only consider vectors that lie within the ball and none that lie outside.

To this end, enumeration algorithms search from right to left, by identifying all values for u_n such that there might exist $u'_1, \ldots, u'_{n-1}$ such that the vector corresponding to $(u'_1, \ldots, u'_{n-1}, u_n)$ lies in the ball. To identify these values $u'_1, \ldots, u'_{n-1}$, enumeration algorithms use the Gram-Schmidt orthogonalization of the lattice basis as well as the projection of lattice vectors. Then, for each of these possible values for u_n, the enumeration algorithm considers all possible values for u_{n-1} and repeats the process until it reaches possible values for u_1. This leads to a search which is serial in nature, as each value of u_n will lead to different possible values for u_{n-1} and so forth. Unfortunately, we can only really apply the quantum search algorithm to problems where the list of objects to be searched is known in advance.

One might suggest to forego the smart way to find short vectors and just search all combinations of $(u_1, \ldots, u_n)$ with appropriate upper and lower bounds on the different u_i's. Then it becomes possible to apply quantum search, since we now have a predetermined list of vectors and just need to compute the norm of each vector. However, it is doubtful that this will result in a faster algorithm, because the recent heuristic changes by Gama et al. [19] have reduced the running time of enumeration dramatically (roughly by a factor $2^{n/2}$) and these changes only complicate the search area further by changing the ball to an ellipsoid. There seems to be no simple way to apply quantum search to the enumeration algorithms that are currently used in practice, but perhaps the algorithms can be modified in some way.

C.2 Voronoi Cell

Consider a set of points in the Euclidean space. For any given point in this set, its Voronoi cell is the region that contains all vectors that lie closer to this point than to any of the other points in the set. Now, given a Voronoi cell, we define a relevant vector to be any vector in the set whose removal from the set will change this particular Voronoi cell. If we pick our lattice as the set and we consider the Voronoi cell around the zero vector, then any shortest vector is also a relevant vector. Furthermore, given the relevant vectors of the Voronoi cell we can solve the closest vector problem in $2^{2n+o(n)}$ time.

So how can we compute the relevant vectors of the Voronoi cell of a lattice L? Micciancio and Voulgaris [40] show that this can be done by solving $2^n - 1$ instances of CVP in the lattice $2L$. However, in order to solve CVP we would need the relevant vectors which means we are back to our original problem. However, Micciancio and Voulgaris show that these instances of CVP can also be solved by solving several related CVP instances in a lattice of lower rank. They give a basic and an optimized version of the algorithm. The basic version only uses LLL as preprocessing and solves all these related CVP instances in the lower rank lattice separately. As a consequence, the basic algorithm runs in time $2^{3.5n+o(n)}$ and in space $2^{n+o(n)}$. The optimized algorithm uses a stronger preprocessing for the lattice basis, which takes exponential time. But since the most expensive part is the computation of the Voronoi relevant vectors, this extra preprocessing time does not increase the asymptotic running time as it is executed only once. In fact, having the reduced basis decreases the asymptotic running time to $\tilde{O}(2^{3n})$. Furthermore, the optimized algorithm employs a trick that allows it to reduce 2^k CVP instances in a lattice of rank k to a single instance of an enumeration problem related to the same lattice. The optimized algorithm solves CVP in time $\tilde{O}(2^{2n})$ using $\tilde{O}(2^n)$ space.

Now, in the basic algorithm, it would be possible to speed up the routine that solves the CVP given the Voronoi relevant vectors using a quantum computer. It would also be possible to speed up the routine that removes non-relevant vectors from the list of relevant vectors using a quantum computer. Combining these two changes gives a quantum algorithm with an asymptotic running time $\tilde{O}(2^{2.5n})$, which is still slower than the optimized classical algorithm. It is not possible to apply these same speedups to the optimized algorithm due to the aforementioned trick with the enumeration problem. The algorithm to solve this enumeration problem makes use of a priority queue, which means the search is not trivially parallellized. Once again, there does not seem to be a simple way to apply quantum search to this special enumeration algorithm. However, it may be possible that the algorithm can be modified in such a way that quantum search can be applied.

An Efficient Attack of a McEliece Cryptosystem Variant Based on Convolutional Codes

Grégory Landais and Jean-Pierre Tillich

SECRET Project - INRIA Rocquencourt
Domaine de Voluceau, B.P. 105 78153 Le Chesnay Cedex, France
{gregory.landais,jean-pierre.tillich}@inria.fr

Abstract. Löndahl and Johansson proposed last year a variant of the McEliece cryptosystem which replaces Goppa codes by convolutional codes. This modification is supposed to make structural attacks more difficult since the public generator matrix of this scheme contains large parts that are generated completely at random. They proposed two schemes of this kind, one of them consists in taking a Goppa code and extending it by adding a generator matrix of a time varying convolutional code. We show here that this scheme can be successfully attacked by looking for low-weight codewords in the public code of this scheme and using it to unravel the convolutional part. It remains to break the Goppa part of this scheme which can be done in less than a day of computation in the case at hand.

Keywords: Code-based cryptography, McEliece cryptosystem, convolutional codes, cryptanalysis.

1 Introduction

In [Sho97], Peter Shor showed that all cryptosystems based on the hardness of factoring or taking a discrete logarithm can be attacked in polynomial time with a quantum computer (see [BBD09] for an extensive report). This threatens most if not all public-key cryptosystems deployed in practice, such as RSA [RSA78] or DSA [Kra91]. Cryptography based on the difficulty of decoding a linear code, on the other hand, is believed to resist quantum attacks and is therefore considered as a viable replacement for those schemes in future applications. Yet, independently of their so-called "post-quantum" nature, code-based cryptosystems offer other benefits even for present-day applications due to their excellent algorithmic efficiency, which is up to several orders of complexity better than traditional schemes.

The first code-based cryptosystem is the McEliece cryptosystem [McE78], originally proposed using Goppa codes. Afterwards several code families have been suggested to replace the Goppa codes in this scheme: generalized Reed–Solomon codes (GRS) [Nie86] or subcodes of them [BL05], Reed–Muller codes [Sid94], algebraic geometry codes [JM96], LDPC codes [BBC08], MDPC codes

P. Gaborit (Ed.): PQCrypto 2013, LNCS 7932, pp. 102–117, 2013.

[MTSB12] or more recently convolutional codes [LJ12]. Some of these schemes allow to reduce the public key size compared to the original McEliece cryptosystem while presumably keeping the same level of security against generic decoding algorithms.

However, for several of the aforementioned schemes it has been shown that a description of the underlying code suitable for decoding can be obtained- this breaks the corresponding scheme. This has been achieved for generalized Reed-Solomon codes in [SS92] and for subcodes of generalized Reed-Solomon codes in [Wie10]. In this case, the attack takes polynomial time and recovers the complete structure of the underlying generalized Reed–Solomon code from the public key G'. The Reed-Muller code scheme has also been attacked, but this time the algorithm recovering the secret description of the permuted Reed-Muller code has sub-exponential complexity [MS07] which is enough for attacking the scheme with the parameters proposed in [Sid94] but which is not sufficient to break the scheme completely. Algebraic geometry codes are broken in polynomial time but only for low genus hyperelliptic curves [FM08]. Finally, it should be mentioned that a first version of the scheme based on LDPC codes proposed in [BC07] has been successfully attacked in [OTD10] (but the new scheme proposed in [BBC08] seems to be immune to this kind of attack), that a variant [BBC$^+$11] of the generalized Reed-Solomon scheme which was supposed to resist to the attack of [SS92] has recently been broken in [GOT12], that another variant of the Generalized Reed-Solomon scheme [Wie06] has been broken in [CGG$^+$13], both by an approach that is related to the distinguisher of Goppa codes which is proposed in [FGO$^+$11] (see [FGO$^+$10] for the full version).

The original McEliece cryptosystem with Goppa codes is still unbroken. It was modified in [BCGO09, MB09] by considering quasi-cyclic or quasi-dyadic versions of Goppa codes (or more generally of alternant codes in [BCGO09]) in order to reduce significantly the key size. However, in this case it was shown in [FOPT10, UL09] that the added structure allows a drastic reduction of the number of unknowns in algebraic attacks and most of the schemes proposed in [BCGO09, MB09] were broken by this approach. This kind of attack has exponential complexity and it can be thwarted by choosing smaller cyclic or dyadic blocks in this approach in order to increase the number of unknowns of the algebraic system. When the rate of the Goppa code is close to 1 (as is the case in signature schemes for instance [CFS01]) then it has been shown in [FGO$^+$11] that the public key can be distinguished from a random public key. This invalidates all existing security proofs of the McEliece cryptosystem when the code rate is close to 1 since they all rely on the hardness of two problems: the hardness of decoding a generic linear code on one hand and the indistinguishability of the Goppa code family on the other hand.

These algebraic attacks motivate the research of alternatives to Goppa codes in the McEliece cryptosystem and it raises the issue of what kind of codes can be chosen in the McEliece cryptosystem. The proposal with convolutional codes made in [LJ12] falls into this thread of research. What makes this new scheme interesting is the fact that its secret generator matrix contains large

parts which are generated completely at random and has no algebraic structure as in other schemes such as generalized Reed-Solomon codes, algebraic geometry codes, Goppa codes or Reed-Muller codes.

In [LJ12] two schemes are given. The first one simply considers as the secret key the generator matrix of a time varying tail-biting convolutional code. A scheme for which it is supposed to resist to attacks of time complexity of about 2^{80} elementary operations is suggested and has reasonable decoding complexity. This construction presents however the drawback that the complexity of decoding scales exponentially with the security level measured in bits. The authors give a second scheme which is scalable and which is built upon a Goppa code and extends it by adding a generator matrix of a time varying convolutional code.

We study the security of this second scheme in this article. It was advocated that the convolutional structure of the code can not be recovered due to the fact that the dual code has large enough minimum distance. However, we show here that this extra defense can be successfully attacked by looking for low-weight codewords in the public code of this scheme. By a suitable filtering procedure of these low weight codewords we can unravel the convolutional part.

The main point that makes this attack feasible is the following phenomenon: the public code of this scheme contains subcodes of much smaller support but whose rate is not much smaller than the rate of the public code. The support of such codes can be easily found by low weight codewords algorithms. It is worthwhile to notice that the code-based KKS signature scheme [KKS97] was broken with exactly the same approach [OT11]. It turns out that the support of these subcodes reveals the convolutional structure. By suitably puncturing the public code, only the Goppa part remains. Deciphering an encrypted message can then be done because for the concrete parameters example provided in [LJ12], algorithms for decoding general linear codes can be used in this case to decode the Goppa code successfully. This attack works successfully on the parameters proposed in [LJ12] and needs only a few hours of computation. It should be possible to change the parameters of the scheme to avoid this kind of attack. In order to do so an improved attack is suggested in Subsection 5.1, its complexity is analyzed in Section 5. This suggests that it should be possible to repair the scheme by fixing the parameters in a more conservative way. Some indications about how to perform such a task are given in Subsection 5.3.

2 The McEliece Scheme Based on Convolutional Codes

The scheme can be summarized as follows.

Secret key.

- $\boldsymbol{G}_{\text{sec}}$ is a $k \times n$ generator matrix which has a block form specified in Figure 1;
- $\boldsymbol{P}$ is an $n \times n$ permutation matrix;
- $\boldsymbol{S}$ is a $k \times k$ random invertible matrix over $\mathbb{F}_2$.

Public key. $\boldsymbol{G}_{\text{pub}} \stackrel{\text{def}}{=} \boldsymbol{S}\boldsymbol{G}_{\text{sec}}\boldsymbol{P}$.

Encryption. The ciphertext $\boldsymbol{c} \in \mathbb{F}_2^n$ of a plaintext $\boldsymbol{m} \in \mathbb{F}_2^k$ is obtained by drawing at random $\boldsymbol{e}$ in $\mathbb{F}_2^n$ of weight equal to some quantity t and computing $\boldsymbol{c} \stackrel{\text{def}}{=} \boldsymbol{m}\boldsymbol{G}_{\text{pub}} + \boldsymbol{e}$.

Decryption. It consists in performing the following steps

1. Calculating $\boldsymbol{c}' \stackrel{\text{def}}{=} \boldsymbol{c}\boldsymbol{P}^{-1} = \boldsymbol{m}\boldsymbol{S}\boldsymbol{G}_{\text{sec}} + \boldsymbol{e}\boldsymbol{P}^{-1}$ and using the decoding algorithm of the code with generator matrix $\boldsymbol{G}_{\text{sec}}$ to recover $\boldsymbol{m}\boldsymbol{S}$ from the knowledge of $\boldsymbol{c}'$;
2. Multiplying the result of the decoding by $\boldsymbol{S}^{-1}$ to recover $\boldsymbol{m}$.

The point of the whole construction is that if t is well chosen, then with high probability the Goppa code part can be decoded, and this allows a sequential decoder of the time varying convolutional code to decode the remaining errors. From now on we will denote by $\mathscr{C}_{\text{pub}}$ the code with generator matrix $\boldsymbol{G}_{\text{pub}}$ and by $\mathscr{C}_{\text{sec}}$ the code with generator matrix $\boldsymbol{G}_{\text{sec}}$.

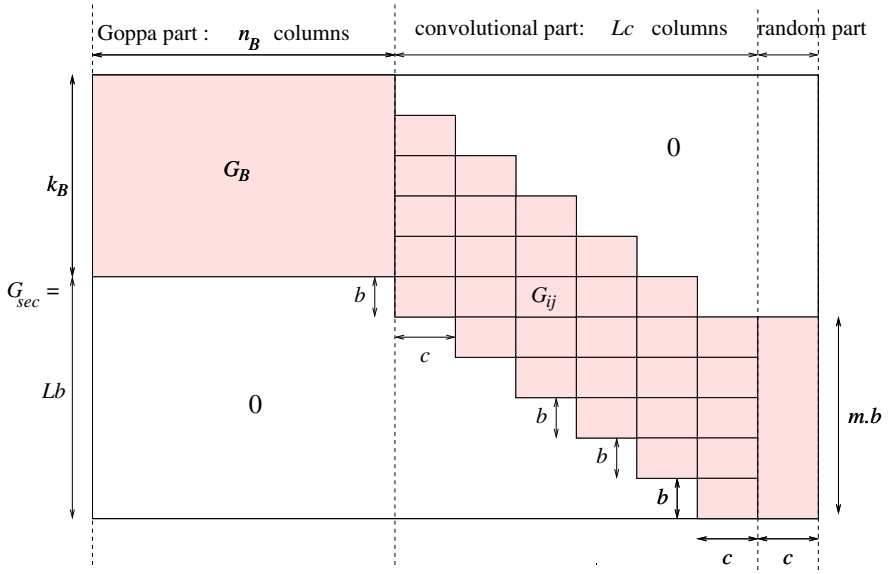


Fig. 1. The secret generator matrix. The areas in light pink indicate the only non zero parts of the matrix. G_B is a generator matrix of a binary Goppa code of length n_B and dimension k_B. This matrix is concatenated with a matrix of a time varying binary convolutional code where b bits of information are transformed into c bits of data (the corresponding G_{ij} blocks are therefore all of size $b \times c$) and terminated with c random columns at the end. The dimension of the corresponding code is $k \stackrel{\text{def}}{=} k_B + Lb$ and the length is $n \stackrel{\text{def}}{=} n_B + (L+1)c$ where L is the time duration of the convolutional code.

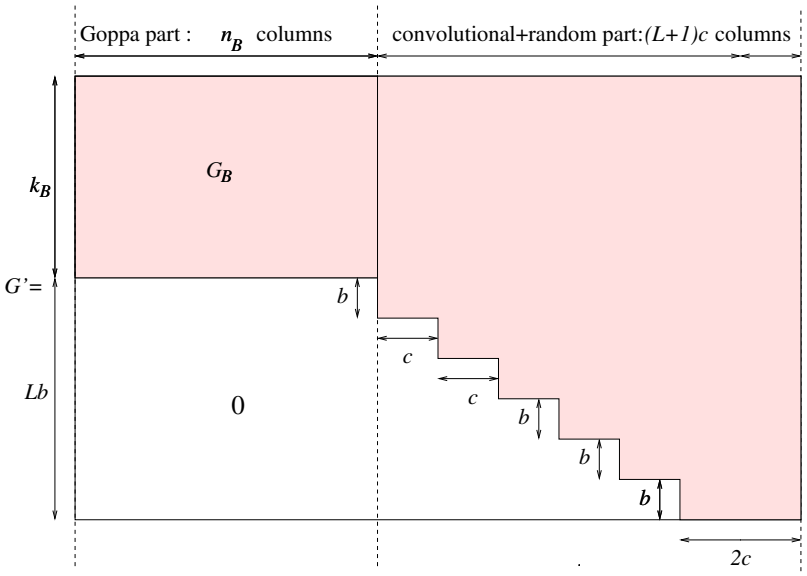


Fig. 2. The generator matrix of an equivalent code obtained by our approach.$\boldsymbol{G}'_B$ denotes the generator matrix of a Goppa code that is equivalent to the code with generator matrix $\boldsymbol{G}_B$

3 Description of the Attack

The purpose of this section is to explain the idea underlying our attack which is a message recovery attack taking advantage of a partial key recovery attack. The attack is divided into two main steps. The first step consists in a (partial) key recovery attack aiming at unraveling the convolutional structure. The second part consists in a message recovery attack taking advantage of the fact that if the convolutional part is recovered, then an attacker can decrypt a message with good probability if he is able to decode a linear code of dimension k_B and length n_B when there are less than $t_B \stackrel{\text{def}}{=} t\frac{n_B}{n}$ errors (this is the average number of errors that the Goppa code has to decode).

3.1 Unraveling the Convolutional Structure

The authors of [LJ12] have chosen the parameters of their scheme so that it remains hard to find low-weight codewords in the dual of the public code $\mathscr{C}_{\text{pub}}$. It is advocated in [LJ12] that in their case the only deviation from a random code is the convolutional structure in terms of low weight parity-checks. For instance, the following parameters are suggested $(n, k, c, b, t) = (1800, 1200, 30, 20, 45)$ and in the construction phase the authors propose to throw away any code that would have parity-checks of weight less than 125. However, the fact that the structure

of $\mathscr{C}_{\text{pub}}$ leads in a natural way to low weight codewords is not taken into account. Indeed, we expect many (i.e. about 2^{b-1}) codewords of weight less than or equal to c. This comes from the fact that the subcode of $\mathscr{C}_{\text{pub}}$ generated by the last b rows of $\boldsymbol{G}_{\text{sec}}$ (and permuted by $\boldsymbol{P}$) has support of size $2c$ and dimension b. Therefore any algorithm aiming at finding codewords of weight less than c say should output such codewords. Looking at the support of such codewords reveals the $2c$ last columns of $\boldsymbol{G}_{\text{sec}}$. By puncturing these columns and starting this process again but this time by looking for codewords of weight less than $c/2$ (since this time the punctured code contains a subcode of dimension b and support of size c arising from the penultimate block of rows of $\boldsymbol{G}_{\text{sec}}$) will reveal the following block of c columns of the matrix. In other words we expect to capture by these means a first subcode of dimension b and support the $2c$ last positions of $\mathscr{C}_{\text{sec}}$. Then we expect a second subcode of dimension b with support the $3c$ last positions of $\mathscr{C}_{\text{pub}}$ and so on and so forth. Finally we expect to be able after suitable column swapping to obtain the generator matrix G' of an equivalent code to $\mathscr{C}_{\text{pub}}$ that would have the form indicated in Figure 2.

More precisely the algorithm for finding a generator matrix of a code equivalent to $\mathscr{C}_{\text{pub}}$ is given by Algorithm 1 given below.

Algorithm 1. An algorithm for finding $\boldsymbol{G}'$.

input: $\boldsymbol{G}_{\text{pub}}$ the public generator matrix
output: a generator matrix $\boldsymbol{G}'$ of a code equivalent to $\mathscr{C}_{\text{pub}}$ which has the form indicated in Fig. 2

$\mathcal{L} \leftarrow []$
for $i = L, \ldots, 1$ **do**
  $\boldsymbol{G} \leftarrow \texttt{GeneratorMatrixPuncturedCode}(\mathscr{C}_{\text{pub}}, \mathcal{L})$
  $\boldsymbol{G} \leftarrow \texttt{LowWeight}(\boldsymbol{G}, w)$
  $w \leftarrow \texttt{Function}(i)$
  $\boldsymbol{G}_i \leftarrow \texttt{ExtendedGeneratorMatrix}(\boldsymbol{G}, \mathcal{L}, \mathscr{C}_{\text{pub}})$
  $\mathcal{L} \leftarrow \texttt{Support}(\boldsymbol{G}) || \mathcal{L}$
end for
$\boldsymbol{G} \leftarrow \texttt{GeneratorMatrixPuncturedCode}(\mathscr{C}_{\text{pub}}, \mathcal{L})$
$\boldsymbol{G}_0 \leftarrow \texttt{ExtendedGeneratorMatrix}(\boldsymbol{G}, \mathcal{L}, \mathscr{C}_{\text{pub}})$
$\boldsymbol{G}'$ is the concatenation of the rows of $\boldsymbol{G}_0, \boldsymbol{G}_1, \ldots, \boldsymbol{G}_L$.
return $\boldsymbol{G}'$

We assume here that:

- the function `GeneratorMatrixPuncturedCode` takes as input a code $\mathscr{C}$ of length n and an ordered set of positions $\mathcal{L}$ which is a sublist of $[1, 2, \ldots, n]$ and outputs a generator matrix of $\mathscr{C}$ punctured in the positions belonging to $\mathcal{L}$;
- `Function` will be a certain function which will be specified later on;
- `Support`($\mathscr{C}$) yields the (ordered) support of $\mathscr{C}$ and $||$ is the concatenation of lists;

- the function `LowWeight` takes as input a code $\mathscr{C}$ and a weight w. It outputs a generator matrix of a subcode of $\mathscr{C}$ obtained by looking for codewords of weight less than or equal to w. Basically a certain number of codewords of weight $\leq w$ are produced and the positions that are involved in at least t codewords are put in a list $\mathcal{L}$ (where t is some threshold depending on the weight w, the length n of the code, its dimension k and the number of codewords produced by the previous call of the function), which means that i is taken as soon as there are at least c elements in $\mathscr{C}$ for which $c_i = 1$. Then a generator matrix for the subcode of $\mathscr{C}$ formed by the codewords of $\mathscr{C}$ whose coordinates outside $\mathcal{L}$ are all equal to 0 is returned. See Algorithm 2 for further details.
- the function `ExtendedGeneratorMatrix` takes as input a generator matrix of some code $\mathscr{C}'$, an ordered set of positions $\mathcal{L}$ and a code $\mathscr{C}$ such that $\mathscr{C}'$ is the result of the puncturing of $\mathscr{C}$ in the positions belonging to $\mathcal{L}$. It outputs a generator matrix of the permuted subcode $\mathscr{C}''$ of $\mathscr{C}$ whose positions are reordered in such a way that the first positions correspond to the positions of $\mathscr{C}'$ and the remaining positions to the ordered list $\mathcal{L}$. This code $\mathscr{C}''$ corresponds to the codewords of $\mathscr{C}'$ that are extended as codewords of $\mathscr{C}$ over the positions belonging to $\mathcal{L}$ in an arbitrary linear way.

3.2 Finishing the Job: Decoding the Code with Generator Matrix G'_B

If we are able to decode the code with generator matrix $\boldsymbol{G}'_B$, then standard sequential decoding algorithms for convolutional codes will allow to decode the last $(L+1)c$ positions. Let $\boldsymbol{G}'_B$ be the generator matrix of a code equivalent to the secret Goppa code chosen for the scheme specified in Figure 2. Decoding such a code can be done by algorithms aiming at decoding generic linear codes such as Stern's algorithm [Ste88] and its subsequent improvements [Dum91, BLP11, MMT11, BJMM12]. This can be done for the parameters suggested in [LJ12].

4 Implementation of the Attack for the Parameters Suggested in [LJ12]

We have carried out the attack on the parameters suggested in [LJ12]. They are provided in Table 1.

Table 1. Parameters for the second scheme suggested in [LJ12]

n	n_B	k	k_B	b	c	L	m	t (number of errors)
1800	1020	1160	660	20	30	25	12	45

Setting the weight parameter w accurately when calling the function `LowWeight` is the key for finding the 60 last positions. If w is chosen to be too large, for

Algorithm 2. `LowWeight`$(\boldsymbol{G}, w)$

input:

- $\boldsymbol{G}$ a certain $k \times n$ generator matrix of a code $\mathscr{C}$;
- w a certain weight

output: a generator matrix $\boldsymbol{G}'$ of a subcode of $\mathscr{C}$ obtained from the supports of a certain subset of codewords of weight w in $\mathscr{C}$.

$\mathscr{C} \leftarrow$ `LowWeightCodewordSearch`$(\boldsymbol{G}, w)$ {Produces a set of linear combinations of rows of $\boldsymbol{G}$ of weight $\leq w$}
Initialize an array tab of length n to zero
$t \leftarrow$ `Threshold`$(w, n, k, |\mathscr{C}|)$
for all $c \in \mathscr{C}$ **do**
  for $i \in [1..n]$ **do**
    if $c_i = 1$ **then**
      tab[i] $\leftarrow$ tab[i] + 1
    end if
  end for
end for
$\mathcal{L} \leftarrow []$
for $i \in [1..n]$ **do**
  if tab[i] $\geq$ t **then**
    $\mathcal{L} \leftarrow \mathcal{L}||\{i\}$
  end if
end for
$\boldsymbol{G}' \leftarrow$ `ShortenedCode`$(\boldsymbol{G}, \mathcal{L})$ {Produces a generator matrix for the subcode of $\mathscr{C}$ formed by the codewords of $\mathscr{C}$ whose coordinates outside $\mathcal{L}$ are all equal to 0.}
return $\boldsymbol{G}'$.

instance when $w = 22$, running Dumer's low weight codeword search algorithm [Dum91] gave the result given in Figure 3 concerning the frequencies of the code positions involved in the codewords of weight less than 22 output by the algorithm and stored in table *tab* during the execution of the algorithm.

We see in Figure 3 that this discriminates the 90 last code positions and not as we want the 60 last code positions. However choosing w to be equal to 18 enables to discriminate the 60 last positions as shown in Figure 4.

Data used in Figure 4 come from 3900 codewords generated in one hour and a half on an Intel Xeon W3550 (3 GHz) CPU by a monothread implementation in C of Dumer's algorithm. The message recovery part of the attack involving the Goppa code consists in decoding 25.5 errors on average in a linear code of dimension 660 and length 1020. The time complexity is about 2^{42}. This second part of the attack could be achieved using the previous program on the same computer in about 6.5 hours on average.

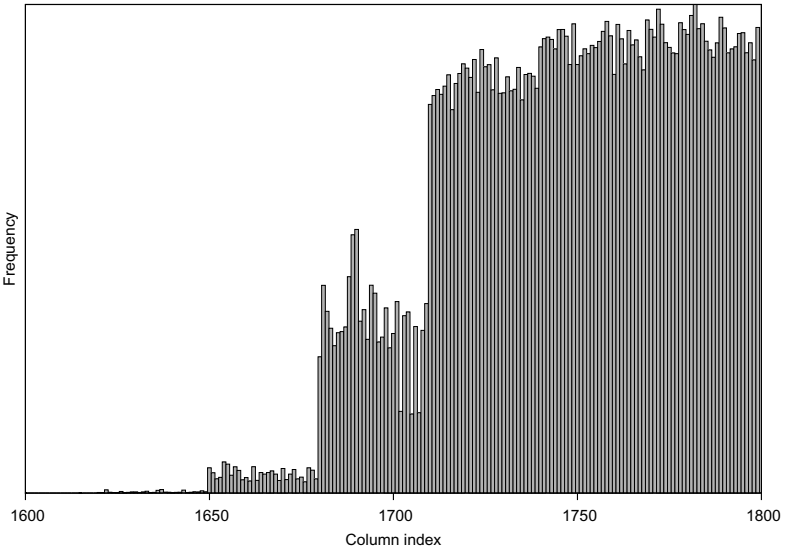


Fig. 3. The frequencies of the code positions involved in codewords of weight ≤ 22 output by Dumer's algorithm

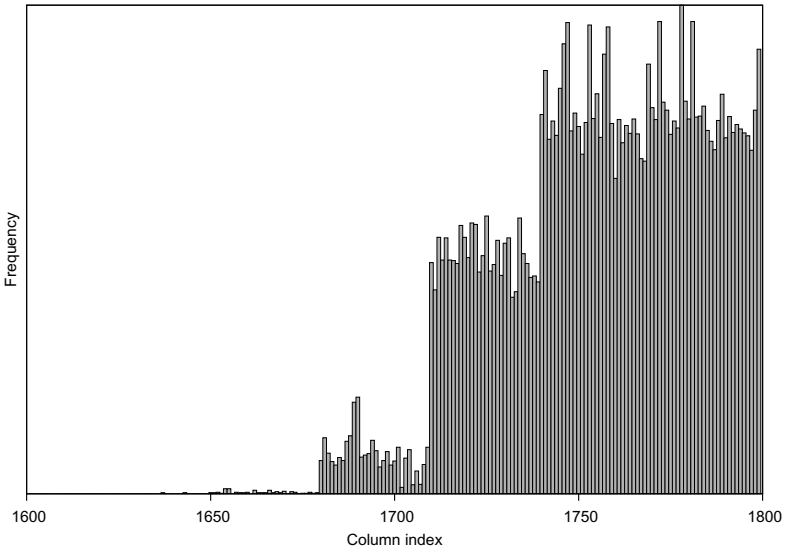


Fig. 4. The frequencies of the code positions involved in codewords of weight ≤ 18 output by Dumer's algorithm

5 Analysis of the Security of the Scheme

5.1 An Improved Attack

The purpose of this section is to provide a very crude analysis of the security of the scheme. We will not analyze our attack detailed in Section 3, since even though it was enough to break the second scheme suggested in [LJ12] it is not the most efficient one. We will give a sketch of a better attack and a rough analysis for it. Basically, the real threat on this scheme comes from the fact that there exists a subcode $\mathscr{C}$ of $\mathscr{C}_{\text{pub}}$ of very small support (of size $2c$ here), namely the code generated by the last b rows of $\boldsymbol{G}_{\text{sec}}$ permuted by the secret permutation matrix $\boldsymbol{P}$. For instance, there are about 2^{b-1} codewords of weight less than or equal to c that should be found by a low weight codeword searching algorithm and that should reveal the support of $\mathscr{C}$. This is basically the idea underlying our attack. However there are other subcodes of rather small support that yield low weight codewords, namely the codes $\mathscr{C}_s$ generated by the $s \times b$ last rows of $\boldsymbol{G}$ for s ranging between 2 and L. The support of $\mathscr{C}_s$ is of size $(s+1)c$. Notice that its rate gets closer and closer to the rate $\frac{2}{3}$ (which is more or less the rate of the final code) as s increases. This is a phenomenon that helps low weight codeword algorithms as will be explained later on.

An improvement of our attack would consist in using a low weight codeword algorithm in order to find one of the codewords of $\mathscr{C}_s$ and to use this codeword c to bootstrap from here to find the whole support of $\mathscr{C}_s$. This is very much in the spirit of the attack against the KKS scheme which is explained in Algorithm 2 which can be found in Subsection 4.4 of [OT11]. With this approach, by using the codeword that has already been found, it is much easier to find new ones belonging to the same subcode with small support by imposing that the information set used for finding low weight codewords is chosen outside the positions belonging to the support of c. The complexity of the whole attack is dominated in this case by the complexity of finding just one codeword in $\mathscr{C}$ when there is a good way to identify the candidates in $\mathscr{C}$ (which can be done by checking the weight of c). Notice that it is very likely that $\mathscr{C}$ is actually the sub code of $\mathscr{C}$ of dimension b that has the smallest support. Recall here that this is precisely the notion captured by the generalized Hamming weights of a code [WY93], w_i being defined as the smallest support of a subcode of dimension i. In other words w_1 is nothing but the minimum distance of the code and in our case it is likely that $w_b = 2c$ (and more generally $w_{sb} = (s+1)c$ for $s = 1..L$). In other words, the problem which should be difficult to solve is the following one

Problem 1. Find one of the subcodes of dimension $s \times b$ whose support size is the $s \times b$-th generalized Hamming weight of $\mathscr{C}_{\text{pub}}$.

We will focus now on the following approach to solve this problem. Consider a low weight codeword algorithm that aims at finding low weight codewords in a code of dimension k by picking up a random set of positions $\mathcal{I}$ of size slightly larger than k, say $k+l$ and that looks for all (or at least a non-negligible fraction) of codewords that have weight equal to some small quantity p over these

positions. These quantities are very good candidates for having low weight over the whole support. This is precisely the approach that is followed in the best low weight codeword search algorithms such as [Ste88, Dum91, MMT11, BJMM12]. We run such an algorithm for several different sets $\mathcal{I}$ and will be interested in the complexity of outputting at least one codeword that belongs to $\mathscr{C}$. This is basically the approach that has been very successful to break the KKS scheme [OT11] and that is the natural candidate to break the [LJ12] scheme.

To analyze such an algorithm we will make some simplifying assumptions

- The cost of checking one of those $\mathcal{I}$ is of order $O\left(\mathscr{L} + \frac{\mathscr{L}^2}{2^l}\right)$ where $\mathscr{L} = \sqrt{\binom{k+l}{p}}$. We neglect here the cost coming from writing the parity-check matrix in systematic form and this does not really cover the recent improvements in [MMT11, BJMM12]. We have made here such an approximation for sake of simplicity. We refer to [FS09] for an explanation of this cost.
- Denote by k' the dimension of the subcode $\mathscr{C}$, by n' the size of its support $\mathcal{J}$. We assume that the result of the puncturing of $\mathscr{C}$ by all positions that do no belong to $\mathcal{J}$ behaves like a random code of dimension k' and length n'.

Our main result to analyze such an algorithm consists in the following proposition.

Proposition 1. *Let*

- $f(x)$ *be the function defined over the positive reals by* $f(x) \stackrel{def}{=} \max\left(x(1-x/2), 1-\frac{1}{x}\right)$;
- $\pi(s) \stackrel{def}{=} \frac{\binom{n'}{s}\binom{n-n'}{k+l-s}}{\binom{n}{k+l}}$;
- $\lambda(s) \stackrel{def}{=} \binom{s}{p} 2^{k'-s}$.
- $C(k,l,p) \stackrel{def}{=} \mathscr{L} + \frac{\mathscr{L}^2}{2^l}$ *where* $\mathscr{L} \stackrel{def}{=} \sqrt{\binom{k+l}{p}}$;
- $\Pi \stackrel{def}{=} \sum_{s=1}^{n'} \pi(s) f(\lambda(s))$.

Then the complexity that the low weight codeword search algorithm outputs an element in $\mathscr{C}$ *is of order*

$$O\left(\frac{C(k,l,p)}{\Pi}\right).$$

5.2 Proof of Proposition 1

Our first ingredient is a lower bound on the probability that a given set $X \subseteq \mathbb{F}_2^n$ intersects a random linear code $\mathscr{C}_{\text{rand}}$ of dimension k and length n picked up uniformly at random. This lemma gives a sharp lower bound even when X is very large and when there is a big gap between the quantities $\mathbf{prob}(X \cap \mathscr{C}_{\text{rand}} \neq \emptyset) = \mathbf{prob}(\cup_{x \in X}\{x \in \mathscr{C}_{\text{rand}}\})$ and $\sum_{x \in X} \mathbf{prob}(x \in \mathscr{C}_{\text{rand}})$.

Lemma 1. *Let X be some subset of $\mathbb{F}_2^n$ of size m and let f be the function defined over the positive reals by $f(x) \stackrel{def}{=} \max\left(x(1-x/2), 1-\frac{1}{x}\right)$. We denote by x the quantity $\frac{m}{2^{n-k}}$, then*

$$\mathbf{prob}(X \cap \mathscr{C}_{rand} \neq \emptyset) \geq f(x).$$

This lemma can be found in [OT11] and it is proved there.

Let us finish now the proof of Proposition 1. Denote by $\mathcal{J}$ the support of $\mathscr{C}$:

$$\mathcal{J} \stackrel{\text{def}}{=} \mathsf{supp}(\mathscr{C}).$$

Let us first calculate the expected number of sets $\mathcal{I}$ we have to consider before finding an element of $\mathscr{C}$. Such an event happens precisely when there is a nonzero word in $\mathscr{C}$ whose restriction to $\mathcal{I} \cap \mathcal{J}$ is of weight equal to p. Let $\mathscr{C}_{\mathcal{I}\cap\mathcal{J}}$ be the restriction of the codewords of $\mathscr{C}$ to the positions that belong to $\mathcal{I} \cap \mathcal{J}$, that is

$$\mathscr{C}_{\mathcal{I}\cap\mathcal{J}} \stackrel{\text{def}}{=} \{(c_i)_{i\in\mathcal{I}\cap\mathcal{J}} : (c_i)_{1\leq i\leq n} \in \mathscr{C}\}.$$

Let X be the set of non-zero binary words of support $\mathcal{I} \cap \mathcal{J}$ that have weight equal to p. Denote by W the size of $\mathcal{I} \cap \mathcal{J}$. The probability that W is equal to s is precisely

$$\mathbf{prob}(W = s) = \frac{\binom{n'}{s}\binom{n-n'}{k+l-s}}{\binom{n}{k+l}} = \pi(s).$$

Then the probability Π that a certain choice of $\mathcal{I}$ gives among the codewords considered by the algorithm a codeword of $\mathscr{C}$ can be expressed as

$$\Pi = \sum_{s=1}^{n'} \mathbf{prob}(W = s)\mathbf{prob}(X \cap \mathscr{C}_{\mathcal{I}\cap\mathcal{J}} \neq \emptyset) \tag{1}$$

$$\geq \sum_{s=1}^{n'} \pi(s) f(\lambda(s)) \tag{2}$$

by using Lemma 1 with $\mathscr{C}_{\mathcal{I}\cap\mathcal{J}}$ and the aforementioned X. Therefore the average number of iterations that have to be performed before finding an element in $\mathscr{C}$ is equal to $\frac{1}{\Pi}$ and this yields immediately Proposition 1.

5.3 Repairing the Parameters and a Pitfall

A possible way to repair the scheme consists in increasing the size of the random part (which corresponds to the last c columns in $\boldsymbol{G}_{\text{sec}}$ here). Instead of choosing this part to be of size c as suggested in [LJ12], its size can be increased in order to thwart the algorithm of Subsection 5.1. Let r be the number of random columns we add at the end of the convolutional part, so that the final length of the code is now $n_B + Lc + r$ instead of $n_B + (L+1)c$ as before. If we choose r to be equal to 140, then the aforementioned attack needs about 2^{80} operations before

outputting an element of $\mathscr{C}$ which is the (permuted) subcode corresponding to the last b rows of $\boldsymbol{G}_{\text{sec}}$. As before, let us denote by $\mathscr{C}_s$ the permuted (by $\boldsymbol{P}$) subcode of $\mathscr{C}_{\text{pub}}$ generated by the last $s \times b$ rows of $\boldsymbol{G}_{\text{sec}}$ permuted by $\boldsymbol{P}$. We can use the previous analysis to estimate the complexity of obtaining an element of $\mathscr{C}_s$ by the previous algorithm. We have gathered the results in Table 2.

Table 2. Complexity of obtaining at least one element of $\mathscr{C}_s$ by the algorithm of Subsection 5.1

s	1	5	10	15	20	21	22	25
complexity (bits)	80.4	72.1	65.1	61.0	59.4	59.3	59.4	59.8

We see from this table that in this case the most important threat does not come from finding low weight codewords arising from codewords in $\mathscr{C}_1$, but codewords of moderate weight arising from codewords in $\mathscr{C}_{20}$ for instance. Codewords in this code have average weight $\frac{r+20c}{2} = 370$. This implies that a simple policy for detecting such candidates which consists in keeping all the candidates in the algorithm of Subsection 5.1 that have weight less than this quantity is very likely to filter out the vast majority of bad candidates and keep with a good chance the elements of $\mathscr{C}_{20}$. Such candidates can then be used as explained in Subsection 5.1 to check whether or not they belong to a subcode of large dimension and small support.

There is a simple way for explaining what is going on here. Notice that the rate of $\mathscr{C}$ is equal to $\frac{b}{c+r}$, which is much smaller than the rate of the overall scheme that is close to $\frac{b}{c}$ in this case by the choice of the parameters of the Goppa code. However as s increases, the rate of $\mathscr{C}_s$ gets closer and closer to $\frac{b}{c}$, since its rate is given by $\frac{sb}{sc+r} = \frac{b}{c+r/s}$. Assume for one moment that the rate of $\mathscr{C}_s$ is equal to $\frac{b}{c}$. Then putting $\boldsymbol{G}_{\text{pub}}$ in systematic form (which basically means that we run the aforementioned algorithm with $p = 1$ and $l = 0$) is already likely to reveal most of the support of $\mathscr{C}_s$ by looking at the support of the rows that have weight around $\frac{sc+r}{2}$ (notice that this phenomenon was already observed in [Ove07]). This can be explained like this. We choose $\mathcal{I}$ to be of size k, the dimension of $\mathscr{C}_{\text{pub}}$, and to be an information set for $\mathscr{C}_{\text{pub}}$. Then, because the rate of $\mathscr{C}_s$ is equal to the rate of $\mathscr{C}_{\text{pub}}$, we expect that the size of $\mathcal{I} \cap \mathcal{J}$ (where $\mathcal{J}$ is the support of $\mathscr{C}_s$) has a rather good chance to be of size smaller than or equal to the dimension of $\mathscr{C}_s$. This in turn implies that it is possible to get codewords from $\mathscr{C}_s$ by any choice over the information set $\mathcal{I}$ of weight 1 which is non zero over $\mathcal{I} \cap \mathcal{J}$ (and therefore of weight 1 there). More generally, even if $\mathcal{I} \cap \mathcal{J}$ is slightly bigger than the dimension of $\mathscr{C}_s$ we expect to be able to get codewords in $\mathscr{C}_s$ a soon as p is greater than the Gilbert-Varshamov distance of the restriction $\mathscr{C}'_s$ of $\mathscr{C}_s$ to $\mathcal{I} \cap \mathcal{J}$, because there is in this case a good chance that this punctured code has codewords of weight p. This Gilbert-Varshamov distance will be very small in this case, because the rate of $\mathscr{C}_s$ is very close to 1 (it is expected to be equal to $\frac{\dim(\mathscr{C}_s)}{|\mathcal{I} \cap \mathcal{J}|}$).

Nevertheless, it is clear that it should be possible to set up the parameters (in particular increasing r should do the job) so that existing low weight codeword algorithms should be unable to find these subcodes $\mathscr{C}_s$ with complexity less than some fixed threshold. However, all these codes $\mathscr{C}_s$ have to be taken into account and the attacks on the dual have also to be reconsidered carefully ([LJ12] considered only attacks on the dual aiming at finding the codewords of lowest weight, but obviously the same technique used for finding some of the $\mathscr{C}_s$ will also work for the dual). Moreover, even if by construction the restriction of $\mathscr{C} = \mathscr{C}_1$ to its support should behave as a random code, this is not true anymore for $\mathscr{C}_s$ with s greater than one, due to the convolutional structure. The analysis sketched in Subsection 5.1 should be adapted a little bit for this case and should take into account the improvements over low weight searching algorithms [MMT11, BJMM12]. Finally, setting up the parameters also requires a careful study of the error probability that sequential decoding fails. This whole thread of work is beyond the scope of the present paper.

Acknowledgements. We thank the reviewers for a careful reading of this manuscript which helped us to improve its editorial quality.

References

[BBC08] Baldi, M., Bodrato, M., Chiaraluce, F.G.: A new analysis of the McEliece cryptosystem based on QC-LDPC codes. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 246–262. Springer, Heidelberg (2008)

[BBC+11] Baldi, M., Bianchi, M., Chiaraluce, F., Rosenthal, J., Schipani, D.: Enhanced public key security for the McEliece cryptosystem (2011) (submitted), arxiv:1108.2462v2[cs.IT]

[BBD09] Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.): Post-Quantum Cryptography. Springer (2009)

[BC07] Baldi, M., Chiaraluce, G.F.: Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes. In: IEEE International Symposium on Information Theory, Nice, France, pp. 2591–2595 (March 2007)

[BCGO09] Berger, T.P., Cayrel, P.-L., Gaborit, P., Otmani, A.: Reducing key length of the McEliece cryptosystem. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 77–97. Springer, Heidelberg (2009)

[BJMM12] Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 520–536. Springer, Heidelberg (2012)

[BL05] Berger, T.P., Loidreau, P.: How to mask the structure of codes for a cryptographic use. Designs Codes and Cryptography 35(1), 63–79 (2005)

[BLP11] Bernstein, D.J., Lange, T., Peters, C.: Smaller decoding exponents: ball-collision decoding. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 743–760. Springer, Heidelberg (2011)

[CFS01] Courtois, N., Finiasz, M., Sendrier, N.: How to achieve a McEliece-based digital signature scheme. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 157–174. Springer, Heidelberg (2001)

[CGG+13] Couvreur, A., Gaborit, P., Gauthier, V., Otmani, A., Tillich, J.P.: Distinguisher-based attacks on public-key cryptosystems using Reed-Solomon codes. In: Proceedings of WCC 2013 (to appear, April 2013); see also arxiv

[Dum91] Dumer, I.: On minimum distance decoding of linear codes. In: Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory, Moscow, pp. 50–52 (1991)

[FGO+10] Faugère, J.-C., Gauthier, V., Otmani, A., Perret, L., Tillich, J.-P.: A distinguisher for high rate McEliece cryptosystems. Cryptology ePrint Archive, Report 2010/331 (2010), http://eprint.iacr.org/

[FGO+11] Faugère, J.-C., Gauthier, V., Otmani, A., Perret, L., Tillich, J.-P.: A distinguisher for high rate McEliece cryptosystems. In: Proceedings of the Information Theory Workshop 2011, ITW 2011, Paraty, Brasil, pp. 282–286 (2011)

[FM08] Faure, C., Minder, L.: Cryptanalysis of the McEliece cryptosystem over hyperelliptic curves. In: Proceedings of the eleventh International Workshop on Algebraic and Combinatorial Coding Theory, Pamporovo, Bulgaria, pp. 99–107 (June 2008)

[FOPT10] Faugère, J.-C., Otmani, A., Perret, L., Tillich, J.-P.: Algebraic cryptanalysis of McEliece variants with compact keys. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 279–298. Springer, Heidelberg (2010)

[FS09] Finiasz, M., Sendrier, N.: Security bounds for the design of code-based cryptosystems. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 88–105. Springer, Heidelberg (2009)

[GOT12] Gauthier, V., Otmani, A., Tillich, J.-P.: A distinguisher-based attack on a variant of McEliece's cryptosystem based on Reed-Solomon codes. CoRR, abs/1204.6459 (2012)

[JM96] Janwa, H., Moreno, O.: McEliece public key cryptosystems using algebraic-geometric codes. Designs Codes and Cryptography 8(3), 293–307 (1996)

[KKS97] Kabatianskii, G., Krouk, E., Smeets, B.J.M.: A digital signature scheme based on random error-correcting codes. In: Darnell, M.J. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 161–167. Springer, Heidelberg (1997)

[Kra91] Kravitz, D.: Digital signature algorithm. US patent 5231668 (July 1991)

[LJ12] Löndahl, C., Johansson, T.: A new version of McEliece PKC based on convolutional codes. In: Chim, T.W., Yuen, T.H. (eds.) ICICS 2012. LNCS, vol. 7618, pp. 461–470. Springer, Heidelberg (2012)

[MB09] Misoczki, R., Barreto, P.S.L.M.: Compact mcEliece keys from goppa codes. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 376–392. Springer, Heidelberg (2009)

[McE78] McEliece, R.J.: A Public-Key System Based on Algebraic Coding Theory, pp. 114–116. Jet Propulsion Lab. (1978); DSN Progress Report 44

[MMT11] May, A., Meurer, A., Thomae, E.: Decoding random linear codes in $O(2^{0.054n})$. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 107–124. Springer, Heidelberg (2011)

[MS07] Minder, L., Shokrollahi, A.: Cryptanalysis of the Sidelnikov cryptosystem. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 347–360. Springer, Heidelberg (2007)

[MTSB12] Misoczki, R., Tillich, J.-P., Sendrier, N., Barreto, P.S.L.M.: MDPC-McEliece: New McEliece variants from moderate density parity-check codes. IACR Cryptology ePrint Archive, 2012:409 (2012)

[Nie86] Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Problems of Control and Information Theory 15(2), 159–166 (1986)

[OT11] Otmani, A., Tillich, J.-P.: An efficient attack on all concrete KKS proposals. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 98–116. Springer, Heidelberg (2011)

[OTD10] Otmani, A., Tillich, J.P., Dallot, L.: Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes. Special Issues of Mathematics in Computer Science 3(2), 129–140 (2010)

[Ove07] Overbeck, R.: Recognizing the structure of permuted reducible codes. In: Tillich, J.P., Augot, D., Sendrier, N. (eds.) Proceedings of WCC 2007, pp. 269–276 (2007)

[RSA78] Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21(2), 120–126 (1978)

[Sho97] Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. 26(5), 1484–1509 (1997)

[Sid94] Sidelnikov, V.M.: A public-key cryptosytem based on Reed-Muller codes. Discrete Mathematics and Applications 4(3), 191–207 (1994)

[SS92] Sidelnikov, V.M., Shestakov, S.O.: On the insecurity of cryptosystems based on generalized Reed-Solomon codes. Discrete Mathematics and Applications 1(4), 439–444 (1992)

[Ste88] Stern, J.: A method for finding codewords of small weight. In: Wolfmann, J., Cohen, G. (eds.) Coding Theory 1988. LNCS, vol. 388, pp. 106–113. Springer, Heidelberg (1989)

[UL09] Umana, V.G., Leander, G.: Practical key recovery attacks on two McEliece variants, IACR Cryptology ePrint Archive 509 (2009)

[Wie06] Wieschebrink, C.: Two NP-complete problems in coding theory with an application in code based cryptography. In: 2006 IEEE International Symposium on Information Theory, pp. 1733–1737 (2006)

[Wie10] Wieschebrink, C.: Cryptanalysis of the Niederreiter public key scheme based on GRS subcodes. In: Sendrier, N. (ed.) PQCrypto 2010. LNCS, vol. 6061, pp. 61–72. Springer, Heidelberg (2010)

[WY93] Wei, V.K.-W., Yang, K.: On the generalized Hamming weights of product codes. Trans. Inf. Theory 39(5), 1709–1713 (1993)

Extended Algorithm for Solving Underdefined Multivariate Quadratic Equations

Hiroyuki Miura[1], Yasufumi Hashimoto[2], and Tsuyoshi Takagi[3]

[1] Graduate School of Mathematics, Kyushu University,
744, Motooka, Nishi-ku, Fukuoka, 819-0395, Japan
[2] Department of Mathematical Sciences, University of the Ryukyus,
1, Senbaru, Nishihara, Okinawa 903-0213, Japan
[3] Institute of Mathematics for Industry, Kyushu University,
744, Motooka, Nishi-ku, Fukuoka, 819-0395, Japan

Abstract. It is well known that solving randomly chosen Multivariate Quadratic equations over a finite field (MQ-Problem) is NP-hard, and the security of Multivariate Public Key Cryptosystems (MPKCs) is based on the MQ-Problem. However, this problem can be solved efficiently when the number of unknowns n is sufficiently greater than that of equations m (This is called "Underdefined"). Indeed, the algorithm by Kipnis et al. (Eurocrypt'99) can solve the MQ-Problem over a finite field of even characteristic in a polynomial-time of n when $n \geq m(m+1)$. Therefore, it is important to estimate the hardness of the MQ-Problem to evaluate the security of Multivariate Public Key Cryptosystems. We propose an algorithm in this paper that can solve the MQ-Problem in a polynomial-time of n when $n \geq m(m+3)/2$, which has a wider applicable range than that by Kipnis et al. We will also compare our proposed algorithm with other known algorithms. Moreover, we implemented this algorithm with Magma and solved the MQ-Problem of $m = 28$ and $n = 504$, and it takes 78.7 seconds on a common PC.

Keywords: Multivariate Public Key Cryptosystems (MPKCs), Multivariate Quadratic Equations, MQ-Problem.

1 Introduction

Multivariate Public Key Cryptosystems (MPKCs) are cryptosystems whose security depends on the hardness of solving Multivariate Quadratic equations over a finite field (MQ-Problem). It is known that the MQ-Problem over a finite field is NP-hard [13] when the coefficients are randomly chosen, and no quantum algorithm efficiently solving the MQ-Problem has been presented. Therefore, MPKCs are one of candidates for post quantum cryptographies. For example, the Matsumoto-Imai cryptosystem [16], Hidden Field Equation (HFE) [18], Unbalanced Oil and Vinegar (UOV) [15], and Rainbow [7] are MPKCs. However, the MQ-Problem is efficiently solved under special n and m conditions. In particular, the algorithm by Kipnis et al. [15] can solve the MQ-Problem over a finite field of even characteristic in a polynomial-time of n when $n \geq m(m+1)$. It is also

P. Gaborit (Ed.): PQCrypto 2013, LNCS 7932, pp. 118–135, 2013.

known that the Gröbner basis algorithms [5,10,11] solve the MQ-Problem, and these algorithms are more effective in the Overdefined ($n \ll m$) MQ-Problem [1,2]. Thus, estimating the hardness of the MQ-Problem is important for the security of MPKCs.

The approach by Kipnis et al. diagonalizes the upper left $m \times m$ part of the coefficient matrices, solves linear equations, and reduces the MQ-Problem to find square roots over a finite field. Courtois et al. [6] and Hashimoto [14] modified this algorithm. Although the algorithm by Courtois et al. [6] has a much smaller applicable range, it can solve MQ-Problems over all the finite fields in polynomial-time. Hashimoto's algorithm presented a polynomial-time algorithm that solves those over all finite fields when $n \geq m^2 - 2m^{3/2} + 2m$, which extended the applicable range of that of Kipnis et al. [15]. However, we point out that Hashimoto's algorithm doesn't work efficiently due to some unsolved multivariate equations arisen from the linear transformation (See Appendix A). Recently, Thomae et al. [20] made n smaller than the algorithm by Kipnis et al. by using the Gröbner basis. This algorithm can be used when $n > m$, but it is an exponential-time algorithm.

We will present an algorithm in this paper solves the Underdefined ($n \gg m$) MQ-Problem in a polynomial-time when $n \geq m(m+3)/2$, which is wider than $n \geq m(m+1)$. Moreover, we implemented this algorithm on Magma [4] and solved an MQ-Problem with (n, m) which the algorithm by Kipnis et al. can't be used. We will compare these results with the algorithm by Kipnis et al. [15] and that by Courtois et al. [6].

2 MQ-Problem and Its Known Solutions

In this section we introduce the MQ-Problem and explain some algorithms to solve the Underdefined MQ-Problems.

2.1 MQ-Problem

Let q be a power of prime and k be a finite field of order q. For integers $n, m \geq 1$, denoted by $f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})$ quadratic polynomials of $\boldsymbol{x} = {}^t(x_1, x_2, \ldots, x_n)$ over k.

$$\begin{aligned}
f_1(x_1, ..., x_n) &= \sum_{1 \leq i \leq j \leq n} a_{1,i,j} x_i x_j + \sum_{1 \leq i \leq n} b_{1,i} x_i + c_1 \\
f_2(x_1, ..., x_n) &= \sum_{1 \leq i \leq j \leq n} a_{2,i,j} x_i x_j + \sum_{1 \leq i \leq n} b_{2,i} x_i + c_2 \\
&\vdots \\
f_m(x_1, ..., x_n) &= \sum_{1 \leq i \leq j \leq n} a_{m,i,j} x_i x_j + \sum_{1 \leq i \leq n} b_{m,i} x_i + c_m,
\end{aligned}$$

where $a_{l,i,j}, b_{l,i}, c_l \in k; l = 1, ..., m$. We call it "the MQ-Problem of m equations and n unknowns over finite field k", that the problem tries to find one solution $(x_1, \ldots, x_n) \in k^n$ such that $f_i(x_1, ..., x_n) = 0$ for all $i = 1, \ldots, m$ among the many ones that exist.

2.2 Kipnis-Patarin-Goubin's Algorithm

We explain Kipnis-Patarin-Goubin's Algorithm [15].

Let $n, m \geq 1$ be integers with $n \geq m(m+1)$ and $f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})$ be the quadratic polynomials of $\boldsymbol{x} = {}^t(x_1, x_2, \ldots, x_n)$ over k. Our goal is to find a solution $x_1, x_2, \ldots, x_n$ such that $f_1(\boldsymbol{x}) = 0, f_2(\boldsymbol{x}) = 0, \ldots, f_m(\boldsymbol{x}) = 0$. For $i = 1, \ldots, n$ the polynomials $f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})$ are denoted by

$$f_i(x_1, x_2, \ldots, x_n) = {}^t\boldsymbol{x} F_i \boldsymbol{x} + (\text{linear.})$$

where $F_1, \ldots, F_m$ are $n \times n$ matrices over k.

We also use an $n \times n$ matrix T_t over k to transform all the unknowns, and T_t has the following form.

$$T_t = \begin{pmatrix} 1 & 0 & \cdots & 0 & a_{1,t} & 0 & \cdots & \cdots & 0 \\ 0 & 1 & \ddots & \vdots & a_{2,t} & \vdots & & & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots & \vdots & & & \vdots \\ \vdots & & \ddots & 1 & a_{t-1,t} & \vdots & & & \vdots \\ \vdots & & & 0 & 1 & 0 & & & \vdots \\ \vdots & & & \vdots & a_{t+1,t} & 1 & \ddots & & \vdots \\ \vdots & & & \vdots & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & & \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & a_{n,t} & 0 & \cdots & 0 & 1 \end{pmatrix} \tag{1}$$

where $a_{1,t}, \ldots, a_{t-1,t}, a_{t+1,t}, \ldots, a_{n,t} \in k$.

We want to obtain quadratic equations of the following form.

$$\begin{cases} \sum_{i=1}^{m} \beta_{1,i} x_i^2 - \lambda_1 = 0 \\ \qquad\qquad \vdots \\ \sum_{i=1}^{m} \beta_{m,i} x_i^2 - \lambda_m = 0, \end{cases} \tag{2}$$

where $\beta_{l,i}$ and $\lambda_l \in k$ $(l = 1, \ldots, m)$.

Step 1. Transform $\boldsymbol{x} \mapsto T_2 \boldsymbol{x}$ so that the coefficients of $x_1 x_2$ in f_j $(j = 1, \ldots, m)$ are zero.

$$F_j \mapsto \left(\begin{array}{c|c} * & 0 \\ \hline 0 & * \end{array} \begin{array}{c} \\ \\ * \end{array} \right) \quad (j = 1, \ldots, m)$$

Step 2. Transform $\boldsymbol{x} \mapsto T_3\boldsymbol{x}$ so that the coefficients of x_1x_3, x_2x_3 in f_j $(j = 1, \ldots, m)$ are zero.

$$\left(\begin{array}{c|c} \begin{array}{c|c} * & 0 \\ \hline 0 & * \end{array} & \\ \hline & * \end{array}\right) \mapsto \left(\begin{array}{c|c} \begin{array}{c|c|c} * & 0 & 0 \\ \hline 0 & * & 0 \\ \hline 0 & 0 & * \end{array} & \\ \hline & * \end{array}\right)$$

$$\vdots$$

(We continue similar operations to "**Step $m-1$.**".)

From "**Step 1.**" to "**Step $m-1$.**", we require the condition $n-1 \geq m(m-1)$, i.e., $n \geq m^2 - m + 1$.

Then we can obtain the coefficient matrices of the form

$$\left(\begin{array}{c|c} \begin{array}{ccc} * & & 0 \\ & \ddots & \\ 0 & & * \end{array} & \\ \hline & * \end{array}\right)$$

for each $i = 1, \ldots, m$, and the following quadratic equations.

$$\begin{cases} \sum_{i=1}^{m} \beta_{1,i}x_i^2 + \sum_{i=1}^{m} x_i L_{1,i}(x_{m+1}, \ldots, x_n) + Q_1(x_{m+1}, \ldots, x_n) = 0 \\ \qquad\qquad \vdots \\ \sum_{i=1}^{m} \beta_{m,i}x_i^2 + \sum_{i=1}^{m} x_i L_{m,i}(x_{m+1}, \ldots, x_n) + Q_m(x_{m+1}, \ldots, x_n) = 0 \end{cases} \tag{3}$$

where L's are linear polynomials and Q's are quadratic polynomials in these variables.

Step m. Solve linear equations $\{L_{i,j}(x_{m+1}, \ldots, x_n) = 0\}$ for $i = 1, \ldots, m$, and $j = 1, \ldots, m$, and substitute the solutions $x_{m+1}, \ldots, x_n$ into (3). This system of linear equations has $n - m$ unknowns and m^2 equations, so we can solve if n and m satisfy $n - m \geq m^2$ i.e. $n \geq m(m+1)$. Finally, we obtain quadratic equations of the form (2). Then we can compute the $x_1^2, \ldots, x_m^2$ values easily. The complexity of this algorithm is

$$\begin{cases} O(n^w m(\log q)^2) & (\text{char } k \text{ is } 2) \\ O(2^m n^w m(\log q)^2) & (\text{char } k \text{ is odd}), \end{cases}$$

where $2 \leq w \leq 3$ is the exponent of the Gaussian elimination. This is because this algorithm computes $n \times n$ matrices over finite field $k = \mathrm{GF}(q)$ and solves linear equations to obtain the $x_1^2, \ldots, x_m^2$ values. The complexity of these operations is $O(n^w(\log q)^2)$. When the characteristic of k is odd, the probability of existence of square roots is approximately $1/2$, and we can find a solution with probability of 2^{-m}. Therefore, when the characteristic of k is odd, the complexity of this algorithm is $O(2^m n^w(\log q)^2)$.

2.3 Courtois et al.'s Algorithm

Courtois et al. proposed an algorithm [6] which extend Kipnis-Patarin-Goubin's algorithm when char k is odd, and this algorithm can be applied when the number of equations m and the number of unknowns n satisfy $n \geq 2\frac{m}{7}(m+1)$. This algorithm and Kipnis-Patarin-Goubin's algorithm are very similar until obtain quadratic equations of the form (2). Main idea of this algorithm is to reduce the number of equations and unknowns after they obtain the quadratic equations of the form (2). This algorithm can solve the MQ-Problem of m equations and n unknowns over k in time about $2^{40}(40 + 40/\log q)^{m/40}$.

2.4 Thomae et al.'s Algorithm

Thomae et al. proposed an algorithm [20] which extend Kipnis-Patarin-Goubin's algorithm, and this algorithm can be applied when the number of equations m and the number of unknowns n satisfy $n > m$. Main idea of this algorithm is to make more zero part by using more linear transformations than Kipnis-Patarin-Goubin's algorithm in order to reduce the number of equations and unknowns. This algorithm reduces the MQ-Problem of m equations and n unknowns over finite field k into the MQ-Problem of $(m - \lfloor n/m \rfloor)$ equations and $(m - \lfloor n/m \rfloor)$ unknowns over finite field k. Then this algorithm uses Gröbner basis algorithm, so the complexty of this algorithm exponential-time. Thomae et al. [20] claimed that the MQ-Problem of 28 equations and 84 unknowns over $\mathrm{GF}(2^8)$ has 80-bit security.

3 Proposed Algorithm

We propose an algorithm in this section that solves the MQ-Problem with $n \geq m(m+3)/2$, and explain the analysis of this algorithm.

3.1 Proposed Algorithm

Let $n, m \geq 1$ be integers with $n \geq m(m+3)/2$ and $f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})$ be the quadratic polynomials of $\boldsymbol{x} = {}^t(x_1, x_2, \ldots, x_n)$ over k. Our goal is to find a solution $x_1, x_2, \ldots, x_n$ such that $f_1(\boldsymbol{x}) = 0, f_2(\boldsymbol{x}) = 0, \ldots, f_m(\boldsymbol{x}) = 0$. For $i = 1, \ldots, n$ the polynomials $f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})$ are denoted by

$$f_i(x_1, x_2, \ldots, x_n) = {}^t\boldsymbol{x} F_i \boldsymbol{x} + (\text{linear.})$$

where $F_1, \ldots, F_m$ are $n \times n$ matrices over kD We also use an $n \times n$ matrix T_t over k of the form (1) to transform all the unknowns in "**Step t.**" $(t = 2, \ldots, m)$.

Step 1. Choose $c_i^{(1)} \in k$ $(i = 1, \ldots, m-1)$ so that the $(1,1)$-elements of $F_i - c_i^{(1)} F_m$ are zero, and replace F_i with $F_i - c_i^{(1)} F_m$. If the $(1,1)$-element of F_m is zero, exchange F_m for one of $F_1, \ldots, F_{m-1}$ that satisfies the $(1,1)$-element is not zero.

$$F_1, F_2, \ldots, F_m \mapsto \underbrace{\begin{pmatrix} 0 & \\ & * \end{pmatrix}, \ldots, \begin{pmatrix} 0 & \\ & * \end{pmatrix}}_{m-1}, \begin{pmatrix} * & \\ & * \end{pmatrix}$$

Step 2. (i) Transform $\boldsymbol{x}$ to $T_2\boldsymbol{x}$ so that the coefficients of x_1x_2 in $f_1, f_2, \ldots, f_m$ are zero.

$$\underbrace{\begin{pmatrix} 0 & \\ & * \end{pmatrix}, \ldots, \begin{pmatrix} 0 & \\ & * \end{pmatrix}}_{m-1}, \begin{pmatrix} * & \\ & * \end{pmatrix} \mapsto \underbrace{\left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & * \end{array} *\right), \ldots, \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & * \end{array} *\right)}_{m-1}, \left(\begin{array}{c|c} * & 0 \\ \hline 0 & * \end{array} *\right)$$

After the linear transformation $\boldsymbol{x} \mapsto T_2\boldsymbol{x}$, the coefficient matrices are denoted as

$${}^tT_2F_iT_2 \ (i = 1, 2, \ldots, m).$$

We determine the $a_{1,2}, a_{3,2}, \ldots, a_{n,2}$ values in T_2 by solving the linear equations of coefficients we want to make zero. Note that (1,2)-elements and (2,1)-elements of F_i are not always equal to zero. The picture means that sum of (1,2)-element and (2,1)-element of F_i is equal to zero for each $i = 1, \ldots, m$.

(ii) Choose $c_i^{(2)} \in k$ $(i = 1, \ldots, m-2)$ so that the $(2,2)$-elements of $F_i - c_i^{(2)}F_{m-1}$ are zero, and replace F_i with $F_i - c_i^{(2)}F_{m-1}$. If the $(2,2)$-element of F_{m-1} is zero, exchange F_{m-1} for one of $F_1, \ldots, F_{m-2}$ that satisfies the $(2,2)$-element is not zero.

$$\underbrace{\left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & * \end{array} *\right), \ldots, \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & * \end{array} *\right)}_{m-1}, \left(\begin{array}{c|c} * & 0 \\ \hline 0 & * \end{array} *\right)$$

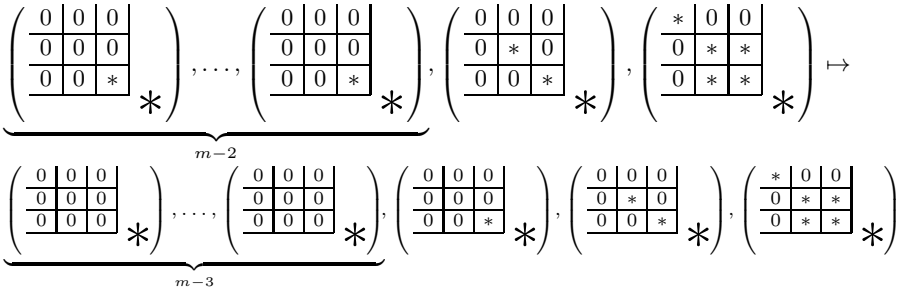
$$\mapsto \underbrace{\left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & 0 \end{array} *\right), \ldots, \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & 0 \end{array} *\right)}_{m-2}, \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & * \end{array} *\right), \left(\begin{array}{c|c} * & 0 \\ \hline 0 & * \end{array} *\right)$$

Step 3. (i) Transform $\boldsymbol{x}$ to $T_3\boldsymbol{x}$ so that the coefficients of x_1x_3 and x_2x_3 in $f_1, f_2, \ldots, f_{m-1}$ and the coefficient of x_1x_3 in f_m are zero.

$$\underbrace{\left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & 0 \end{array} *\right), \ldots, \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & 0 \end{array} *\right)}_{m-2}, \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & * \end{array} *\right), \left(\begin{array}{c|c} * & 0 \\ \hline 0 & * \end{array} *\right)$$

$$\mapsto \underbrace{\left(\begin{array}{cc|c} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline 0 & 0 & * \end{array} *\right), \ldots, \left(\begin{array}{cc|c} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline 0 & 0 & * \end{array} *\right)}_{m-2}, \left(\begin{array}{cc|c} 0 & 0 & 0 \\ 0 & * & 0 \\ \hline 0 & 0 & * \end{array} *\right), \left(\begin{array}{cc|c} * & 0 & 0 \\ 0 & * & * \\ \hline 0 & * & * \end{array} *\right)$$

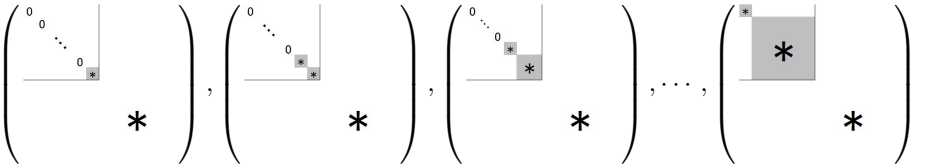
(ii) Choose $c_i^{(3)} \in k$ $(i = 1, \ldots, m-3)$ so that the $(3,3)$-elements of $F_i - c_i^{(3)} F_{m-2}$ are zero, and replace F_i with $F_i - c_i^{(3)} F_{m-2}$. If the $(3,3)$-element of F_{m-2} is zero, exchange F_{m-2} for one of $F_1, \ldots, F_{m-3}$ that satisfies the $(3,3)$-element is not zero.



$\vdots$

(We continue similar operations to "**Step m.**".)

Then we can obtain the coefficient matrices of the form



for each $i = 1, \ldots, m$, and the following quadratic equations.

$$\begin{cases} x_m^2 + \sum\limits_{1 \le i \le m} x_i L_{1,i}(x_{m+1}, \ldots, x_n) + Q_{1,2}(x_{m+1}, \ldots, x_n) = 0 \\ x_{m-1}^2 + x_m^2 + \sum\limits_{1 \le i \le m} x_i L_{2,i}(x_{m+1}, \ldots, x_n) + Q_{2,2}(x_{m+1}, \ldots, x_n) = 0 \\ x_{m-2}^2 + Q_{3,1}(x_{m-1}, x_m) + \sum\limits_{1 \le i \le m} x_i L_{3,i}(x_{m+1}, \ldots, x_n) + Q_{3,2}(x_{m+1}, \ldots, x_n) = 0 \\ \qquad\qquad \vdots \\ x_1^2 + Q_{m,1}(x_2, \ldots, x_m) + \sum\limits_{1 \le i \le m} x_i L_{m,i}(x_{m+1}, \ldots, x_n) + Q_{m,2}(x_{m+1}, \ldots, x_n) = 0 \end{cases} \tag{4}$$

where L's are linear polynomials and Q's are quadratic polynomials in these variables.

Step $m+1$. Solve linear equations $\{L_{i,j}(x_{m+1}, \ldots, x_n) = 0\}$ of $x_{m+1}, \ldots, x_n$ for $i = 1, \ldots, m$ and $j = 1, \ldots, m-i+1$, and substitute the solutions $x_{m+1}, \ldots, x_n$ into (4). If there exists $t = 1, \ldots, m$ such that the (t,t)-elements of $F_1, \ldots, F_{m-t+1}$ are zero, remove $L_{m-t+1,t} = 0$ from the linear systems and choose the $x_{m+1}, \ldots, x_n$ that satisfies $L_{m-t+1,t} \neq 0$.

Finally, we obtain the following quadratic equations.

$$\begin{cases} x_m^2 - \lambda_1 = 0 \\ x_{m-1}^2 + \tilde{Q}_2(x_m) - \lambda_2 = 0 \\ x_{m-2}^2 + \tilde{Q}_3(x_{m-1}, x_m) - \lambda_3 = 0 \\ \qquad\qquad \vdots \\ x_2^2 + \tilde{Q}_{m-1}(x_3, \ldots, x_m) - \lambda_{m-1} = 0 \\ x_1^2 + \tilde{Q}_m(x_2, \ldots, x_m) - \lambda_m = 0 \end{cases}$$

where $\lambda_1, \ldots, \lambda_m \in k$ and $\tilde{Q}$'s are quadratic polynomials in these variables.

We can find a solution for the quadratic equations in the following way. First, we solve the first equation and substitute the solution x_m into the others. Next, we solve the second equation and substitute the solution x_{m-1} into the remaining equations $\cdots$. If there exists $t = 1, \ldots, m$ such that (t, t)-elements of $F_1, \ldots, F_{m-t+1}$ are zero, the $(m - t + 1)$-th equation takes the form of $x_t + Q(x_{t+1}, \ldots, x_m) - \lambda_{m-t+1} = 0$.

3.2 Analysis of Proposed Algorithm

We will explain the required conditions and complexity of the proposed algorithm in this section.

Theorem 3.1. The proposed algorithm works when $n \geq m(m+3)/2$.

Proof. Our algorithm works if we can solve the linear equations.

In "**Step t.**" $(t = 2, \ldots, m)$, the number of linear equations to be solved is

$$(m - t + 1)(t - 1) + \sum_{i=1}^{t-1} i = -\frac{1}{2}\left\{t - \left(m + \frac{3}{2}\right)\right\}^2 + \frac{1}{2}m^2 + \frac{1}{2}m + \frac{1}{8},$$

and the number of unknowns is $n - 1$. Thus, we require $n \geq m(m+1)/2$ until "**Step m.**".

In "**Step $m + 1$.**", the number of linear equations to be solved is

$$\sum_{t=1}^{m}(m - t + 1) = \frac{1}{2}m(m+1),$$

and the number of unknowns is $n - m$. Thus, we require $n \geq m(m+3)/2$.

For these reasons, we found that the proposed algorithm can be applied when $n \geq m(m+3)/2$. □

Lemma 3.2. For $n = m(m+3)/2$, the proposed algorithm succeeds in finding a solution of the MQ-Problem of m equations, n unknowns with probability of approximately

$$\begin{cases} 1 - q^{-1} & (\text{char } k \text{ is } 2) \\ 2^{-m}(1 - q^{-1}) & (\text{char } k \text{ is odd}). \end{cases}$$

Proof. When $n = m(m+3)/2$, we must solve the linear equations that are not underdefined in "**Step $m+1$.**". Then, we fail to solve linear equations with probability of q^{-1}. When the characteristic of k is odd, the probability of existence of square roots over k is approximately $1/2$. Therefore, the success probability of this algorithm is approximately $2^{-m}(1-q^{-1})$ when the characteristic of k is odd. □

Moreover, the proposed algorithm uses only $n \times n$ matrix operations and the calculation of square roots over finite field k, so we obtain the following result concerning the complexity of the proposed algorithm.

Theorem 3.3. The complexity of the proposed algorithm is

$$\begin{cases} O(n^w m(\log q)^2) & (\text{char } k \text{ is } 2) \\ O(2^m n^w m(\log q)^2) & (\text{char } k \text{ is odd}), \end{cases}$$

where $2 \leq w \leq 3$ is the exponent of the Gaussian elimination.

Proof. In this algorithm, we calculate $n \times n$ matrices over finite field $k = \mathrm{GF}(q)$ for about m times. The complexity of this operation is $O(n^w(\log q)^2)$. When the characteristic of k is odd, the probability of existence of square roots is approximately $1/2$, and we can find a solution with probability of 2^{-m}. Therefore, when the characteristic of k is odd, the complexity of this algorithm is $O(2^m n^w m(\log q)^2)$. □

4 Implementations

We implemented the proposed algorithm using Magma [4], and compare the proposed algorithm and other known algorithms in this section. The results depend on the characteristic of k, and we will explain two cases, when the characteristic of k is 2 and an odd prime.

4.1 Parameters and Computational Environments

We chose the n and m parameters in which other algorithms can't be applied, and used homogeneous quadratic polynomials to experiment. We also chose $m = 28$ the same as Thomae et al. [20], and n so that the proposed algorithm can apply. The computer specification and software are listed in Table 1.

Table 1. Computer specifications

OS	CPU	RAM	Software
Windows 7 (64bit)	Intel Core i3 (1.33GHz)	4.00 GB	Magma V2.17-9

4.2 When char k Is 2

These algorithms have the same complexity $O(n^w m(\log q)^2)$, but the proposed algorithm has a wider applicable range than the others. The applicable ranges of the algorithms are drawn in Fig. 1.

Table 2. Applicable ranges of the proposed algorithm and other known algorithms (char k is 2)

	Applicable range	Complexity
Proposed	$n \geq m(m+3)/2$	(poly.)
Kipnis et al. [15]	$n \geq m(m+1)$	(poly.)
Courtois et al. [6]	$n \geq m(m+1)$	(poly.)

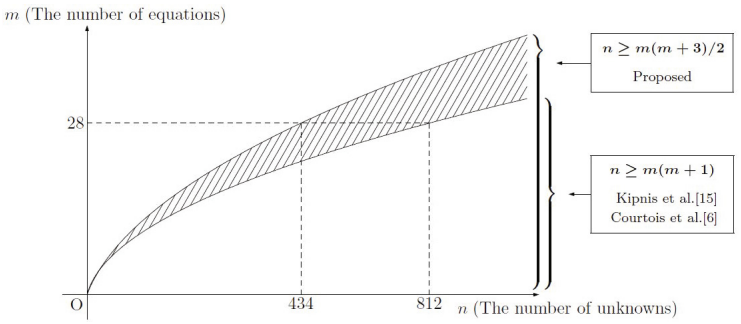


Fig. 1. Applicable range of proposed algorithm and other known algorithms

When $m = 28$, we can reduce the number of unknowns n from 812 to 434. The experimental results in our implementation are in Table. 3.

Table 3. Experimental results (char k is 2)

Field	n	m	Time / a try	Success probability
GF(2^8)	16	4	8.76 (msec.)	99.99 %
	84	11	506.83 (msec.)	100.0 %
	504	28	78.71 (sec.)	100.0 %

4.3 When char k Is Odd

We consider the algorithms by Courtois et al. [6]. Although the former one is polynomial-time of n, but it is not practical because the applicable range is too small. Thus, we compare the proposed algorithm and the latter one by Courtois et al. These algorithms are exponential-time. The applicable ranges of the algorithms are drawn in Fig. 2.

Table 4. Applicable ranges of the proposed algorithm and other known algorithms (char k is odd)

	Applicable range	Complexity
Proposed	$n \geq m(m+3)/2$	(exp.)
Courtois et al. [6]	$n \geq 2^{\frac{m}{7}} m(m+1)$	(poly.)
	$n \geq 2^{\frac{m}{7}} (m+1)$	(exp.)

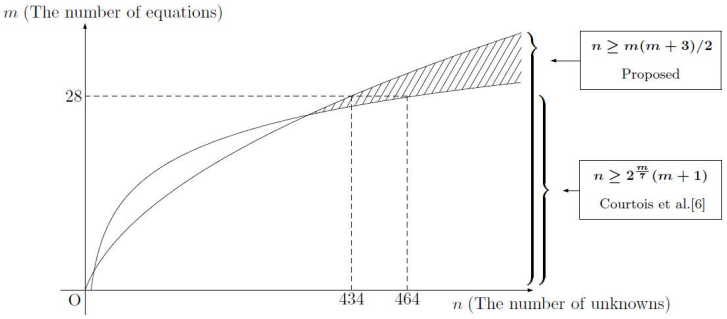


Fig. 2. Applicable range of proposed algorithm and algorithm by Courtois et al.

If $m \geq 27$, we can reduce the number of unknowns n to smaller than that of the algorithm by Courtois et al. The experimental results in our implementation are in Table. 5.

Table 5. Experimental results (char k is odd)

Field	n	m	Time / a try	Success probability
GF(7)	16	4	3.99 (msec.)	11.83 %
	84	11	259.28 (msec.)	0.22 %
	434	28	39.99 (sec.)	0.00 %

The proposed algorithm succeeds in solving the MQ-Problem with probability of 11.83% when $n = 16$ and $m = 4$, and 0.22% when $n = 84$ and $m = 11$. These results follow our success probability estimation, and we can get a similar result when $n = 434$ and $m = 28$ which can't use the algorithm by Courtois et al., and then, the success probability is $(4/7)^{28} \times (6/7) \approx 10^{-6.87} \approx 2^{-22.83}$. We estimate that it takes 1-core PC 9.44 years to solve the MQ-Problem of $n = 434$ and $m = 28$.

5 Conclusion

We presented an algorithm in this paper that can solve the MQ-Problem when $n \geq m(m+3)/2$, where n is the number of unknowns and m is the number of equations. This algorithm makes the range of solvable MQ-Problems wider than that by Kipnis et al. Moreover, we compared this algorithm and other known algorithms, and found that the proposed algorithm is easier to use than the others. In order to demonstrate the effectiveness of the proposed algorithm we implemented it using Magma on a PC. We were able to solve the MQ-Problem of $m = 28$ and n = 504 in 78.7 seconds.

Two open problems remain. The first is to make the applicable range wider and the second is to apply the proposed algorithm to the algorithm developed by Thomae et al. [20].

References

1. Bardet, M., Faugère, J.-C., Salvy, B., Yang, B.-Y.: Asymptotic Behaviour of the Degree of Regularity of Semi-Regular Polynomial Systems, MEGA 2005 (2005), http://www-polsys.lip6.fr/~jcf/Papers/BFS05b.pdf
2. Bettale, L., Faugère, J.-C., Perret, L.: Hybrid Approach for Solving Multivariate Systems over Finite Fields. Journal of Mathematical Cryptology 3, 177–197 (2009)
3. Braeken, A., Wolf, C., Preneel, B.: A Study of the Security of Unbalanced Oil and Vinegar Signature Schemes. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 29–43. Springer, Heidelberg (2005)
4. Computational Algebra Group, University of Sydney. The MAGMA Computational Algebra System for Algebra, Number Theory, and Geometry
5. Courtois, N.T., Klimov, A.B., Patarin, J., Shamir, A.: Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000), http://www.minrank.org/xlfull.pdf
6. Courtois, N.T., Goubin, L., Meier, W., Tacier, J.-D.: Solving Underdefined Systems of Multivariate Quadratic Equations. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 211–227. Springer, Heidelberg (2002)
7. Ding, J., Schmidt, D.: Rainbow, a New Multivariate Polynomial Signature Scheme. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 164–175. Springer, Heidelberg (2005)
8. Ding, J., Gower, J.E., Schmidt, D.S.: Multivariate Public Key Cryptosystems. Springer (2006)

9. Ding, J., Hodges, T.J.: Inverting HFE Systems Is Quasi-Polynomial for All Fields. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 724–742. Springer, Heidelberg (2011)
10. Faugère, J.-C.: "A New Efficient Algorithm for Computing Gröbner bases (F_4)". Journal of Pure and Applied Algebra 139, 61–88 (1999)
11. Faugère, J.-C.: A New Efficient Algorithm for Computing Gröbner bases without reduction to zero (F_5). In: Proceedings of ISSAC 2002, pp. 75–83. ACM Press (2002)
12. Faugère, J.-C., Perret, L.: On the Security of UOV. In: Proceedings of SCC 2008, pp. 103–109 (2008)
13. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H.Freeman (1979)
14. Hashimoto, Y.: Algorithms to Solve Massively Under-Defined Systems of Multivariate Quadratic Equations. IEICE Trans. Fundamentals E94-A(6), 1257–1262 (2011)
15. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced Oil and Vinegar Signature Schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999)
16. Matsumoto, T., Imai, H.: Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 419–453. Springer, Heidelberg (1988)
17. Patarin, J.: Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt '88. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 248–261. Springer, Heidelberg (1995)
18. Patarin, J.: Hidden Field Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 33–48. Springer, Heidelberg (1996)
19. Shor, P.W.: Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In: Proceedings of 35th Annual Symposium on Foundations of Computer Science, pp. 124–134. IEEE Computer Society Press (1994)
20. Thomae, E., Wolf, C.: Solving Underdetermined Systems of Multivariate Quadratic Equations Revisited. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 156–171. Springer, Heidelberg (2012)

Appendix A: Hashimoto's Algorithm

In this appendix we explain Hashimoto's algorithm [14], which claimed that the MQ-Problem of $n \geq m^2 - 2m^{3/2} + 2m$ over all finite fields can be solved in a polynomial-time. The applicable range of Hashimoto's algorithm is wider than that of the algorithm by Kipnis et al. [15]. However, we point out that Hashimoto's algorithm doesn't work efficiently due to some unsolved multivariate equations arisen from the linear transformation.

A.1 Outline

In the following we describe Hashimoto's algorithm which consists of Algorithm A and Algorithm B.

Algorithm A
Let $g(\boldsymbol{x})$ be a quadratic form of unknowns $\boldsymbol{x} = {}^t(x_1, \ldots, x_n)$ over finite field k. We transform $\boldsymbol{x}$ by a linear matrix $U \in k^{n \times n}$. For $a_{2,1}, a_{3,1}, a_{3,2}, \ldots, a_{n,n-1} \in k$ we define U as follows :

$$U = \begin{pmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ a_{2,1} & 1 & 0 & & & \vdots \\ a_{3,1} & a_{3,2} & 1 & \ddots & & \vdots \\ 0 & 0 & a_{4,3} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 1 & 0 \\ 0 & 0 & \cdots & 0 & a_{n,n-1} & 1 \end{pmatrix}.$$

We determine the linear transformation U such that the coefficients of $x_1^2, x_1x_2, x_1x_3, \ldots, x_1x_{n-1}$ in $g(U\boldsymbol{x})$ are all zero in the following way.
Step 1. Calculate $a_{2,1}, a_{3,1}$ such that the coefficient of x_1^2 in $g(U\boldsymbol{x})$ is zero.
Step 2. Calculate $a_{3,2}$ such that the coefficient of x_1x_2 in $g(U\boldsymbol{x})$ is zero.
Step 3. Calculate $a_{4,3}$ such that the coefficient of x_1x_3 in $g(U\boldsymbol{x})$ is zero.

$$\vdots$$

Step $\boldsymbol{n-1}$. Calculate $a_{n,n-1}$ such that the coefficient of x_1x_{n-1} in $g(U\boldsymbol{x})$ is zero.

Algorithm B
Let $n, L, M \geq 1$ be integers that satisfy the following condition :

$$n \geq \begin{cases} 2L & (M = 1) \\ ML - M + L & (1 < M < L) \\ L^2 + 1 & (M = L) \end{cases}. \tag{5}$$

Let $g_1(\boldsymbol{x}), \ldots, g_M(\boldsymbol{x})$ be quadratic forms of $\boldsymbol{x}$ over k such that the coefficients of x_ix_j $(1 \leq i, j \leq L)$ in $g_1(\boldsymbol{x}), \ldots, g_{M-1}(\boldsymbol{x})$ are all zero. Then we can find an

invertible linear transformation U such that the coefficients of $x_i x_j$ $(1 \le i, j \le L)$ in $g_1(U\boldsymbol{x}), \ldots, g_M(U\boldsymbol{x})$ are all zero.

$$\underbrace{{}^t\boldsymbol{x}\begin{pmatrix} O_L & * \\ * & * \end{pmatrix}\boldsymbol{x}, \ldots, {}^t\boldsymbol{x}\begin{pmatrix} O_L & * \\ * & * \end{pmatrix}\boldsymbol{x}}_{M-1}, {}^t\boldsymbol{x}\begin{pmatrix} * & * \\ * & * \end{pmatrix}\boldsymbol{x} \mapsto \underbrace{{}^t\boldsymbol{x}\begin{pmatrix} O_L & * \\ * & * \end{pmatrix}\boldsymbol{x}, \ldots, {}^t\boldsymbol{x}\begin{pmatrix} O_L & * \\ * & * \end{pmatrix}\boldsymbol{x}}_{M}$$

where O_L is $L \times L$ zero matrix. **Step 1.** (i) Using Algorithm A, find a transformation $T_{1,1}$ such that the coefficients of $x_1 x_j$ $(j = 1, \ldots, L-1)$ in $g_M(\boldsymbol{x})$ are zero, and transform $\boldsymbol{x} \mapsto T_{1,1}\boldsymbol{x}$.
(ii) Transform $\boldsymbol{x} \mapsto T_{2,1}\boldsymbol{x}$ such that the coefficients of $x_1 x_L$ in $g_M(\boldsymbol{x})$ and $x_i x_L$ $(i = 1, \ldots, L)$ in $g_1(\boldsymbol{x}), \ldots, g_{M-1}(\boldsymbol{x})$ are all zero.
Step 2. (i) Using Algorithm A, find a transformation $T_{1,2}$ such that the coefficients of $x_2 x_j$ $(j = 2, \ldots, L-1)$ in $g_M(\boldsymbol{x})$ are all zero, and transform $\boldsymbol{x} \mapsto T_{1,2}\boldsymbol{x}$.
(ii) Transform $\boldsymbol{x} \mapsto T_{2,2}\boldsymbol{x}$ such that the coefficients of $x_2 x_L$ in $g_M(\boldsymbol{x})$ and $x_i x_L$ $(i = 2, \ldots, L)$ in $g_1(\boldsymbol{x}), \ldots, g_{M-1}(\boldsymbol{x})$ are all zero.

$$\vdots$$

(We continue similar operations to "**Step $L-1$.**".)

In "**Step t.**-(i), (ii)" $(t = 1, \ldots, L-1)$, we use $n \times n$ matrices $T_{1,t}, T_{2,t}$ which have the following form :

$$T_{1,t} = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ a_{2,1}^{(t)} & 1 & \ddots & & & \vdots \\ a_{3,1}^{(t)} & a_{3,2}^{(t)} & 1 & \ddots & & \vdots \\ 0 & 0 & a_{4,3}^{(t)} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 1 & 0 \\ 0 & 0 & \cdots & 0 & a_{n,n-1}^{(t)} & 1 \end{pmatrix}, \; T_{2,t} = \begin{pmatrix} 1 & 0 & \cdots & 0 & b_{1,L}^{(t)} & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots & b_{2,L}^{(t)} & \vdots & & & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots & \vdots & & & \vdots \\ \vdots & & \ddots & 1 & b_{L-1,L}^{(t)} & \vdots & & & \vdots \\ \vdots & & & 0 & 1 & 0 & & & \vdots \\ \vdots & & & \vdots & b_{L+1,L}^{(t)} & 1 & \ddots & & \vdots \\ \vdots & & & \vdots & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & & \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & b_{n,L}^{(t)} & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

Step L. Transform $\boldsymbol{x} \mapsto T_L\boldsymbol{x}$ such that the coefficients of $x_i x_L$ $(i = 1, \ldots, L)$ in $g_1(\boldsymbol{x}), \ldots, g_M(\boldsymbol{x})$ are all zero, where

$$T_L = \begin{pmatrix} 1 & 0 & \cdots & 0 & a_{1,L}^{(L)} & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots & a_{2,L}^{(L)} & \vdots & & & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots & \vdots & & & \vdots \\ \vdots & & \ddots & 1 & a_{L-1,L}^{(L)} & \vdots & & & \vdots \\ \vdots & & & 0 & a_{L,L} & 0 & & & \vdots \\ \vdots & & & \vdots & a_{L+1,L}^{(L)} & 1 & \ddots & & \vdots \\ \vdots & & & \vdots & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & & \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & a_{n,L}^{(L)} & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

If there is no such transformation, then go back to "**Step $L-1$.**".
Step $L+1$. Return $U = T_L T_{2,L-1} T_{1,L-1} \cdots T_{2,1} T_{1,1}$.

A.2 Analysis of Algorithm B

We find the following facts about Algorithm B.

Lemma A.1. Suppose $L \geq 3$. In "**Step t.**" $t = 1, \ldots, L-2$) of Algorithm B,

$${}^tT_{1,t} \begin{pmatrix} O_{L,L} & * \\ * & * \end{pmatrix} T_{1,t} = \begin{pmatrix} O_{L,L} & * \\ * & * \end{pmatrix}.$$

This lemma shows that the $L \times L$ upper left part of $g_1(\boldsymbol{x}), \ldots, g_{M-1}(\boldsymbol{x})$ remains zero by linear transformation $T_{1,t}$.

Lemma A.2. In "**Step t.**-(ii)" ($t = 1, \ldots, L-1$), the coefficient of x_L^2 in $g_i(\boldsymbol{x})$ ($i = 1, \ldots, M-1$) is

$$\sum_{1 \leq j \leq L-1} a_{j,L} L_{i,j}(a_{L+1,L}^{(t)}, \ldots, a_{n,L}^{(t)}) + Q_i(a_{L+1,L}^{(t)}, \ldots, a_{n,L}^{(t)}).$$

Theorem A.3. In "**Step t.**-(ii)" ($t = 1, \ldots, L-1$), the coefficient of $x_j x_L$ in $g_i(\boldsymbol{x})$ is equal to $L_{i,j}(a_{L+1,L}^{(t)}, \ldots, a_{n,L}^{(t)})$ ($i = 1, \ldots, M-1; j = 1, \ldots, L-1$).

Observation A.4. In "**Step t.**-(ii)" ($t = 1, \ldots, L-1$), we must solve equations

$$\begin{cases} (\text{The coefficient of } x_1 x_L \text{ in } g_1(\boldsymbol{x})) = 0 \\ \qquad\qquad \vdots \\ (\text{The coefficient of } x_{L-1} x_L \text{ in } g_1(\boldsymbol{x})) = 0 \\ (\text{The coefficient of } x_L^2 \text{ in } g_1(\boldsymbol{x})) = 0 \\ (\text{The coefficient of } x_1 x_L \text{ in } g_2(\boldsymbol{x})) = 0 \\ \qquad\qquad \vdots \\ (\text{The coefficient of } x_L^2 \text{ in } g_{M-1}(\boldsymbol{x})) = 0 \\ (\text{The coefficient of } x_t x_L \text{ in } g_M(\boldsymbol{x})) = 0, \end{cases}$$

i.e.,

$$\begin{cases} L_{1,1}(a^{(t)}_{L+1,L},\ldots,a^{(t)}_{n,L})=0 \\ \qquad\qquad\vdots \\ L_{1,L-1}(a^{(t)}_{L+1,L},\ldots,a^{(t)}_{n,L})=0 \\ \sum_{1\le j\le L-1} a_{j,L}L_{i,j}(a^{(t)}_{L+1,L},\ldots,a^{(t)}_{n,L})+Q_i(a^{(t)}_{L+1,L},\ldots,a^{(t)}_{n,L})=0 \\ L_{2,1}(a^{(t)}_{L+1,L},\ldots,a^{(t)}_{n,L})=0 \\ \qquad\qquad\vdots \\ L_{M-1,L-1}(a^{(t)}_{L+1,L},\ldots,a^{(t)}_{n,L})=0 \\ (\text{The coefficient of } x_tx_L \text{ in } g_M(\boldsymbol{x}))=0 \end{cases}$$

Note that we can solve linear equations without the L-th equation

$$\begin{cases} L_{1,1}(a^{(t)}_{L+1,L},\ldots,a^{(t)}_{n,L})=0 \\ \qquad\qquad\vdots \\ L_{1,L-1}(a^{(t)}_{L+1,L},\ldots,a^{(t)}_{n,L})=0 \\ L_{2,1}(a^{(t)}_{L+1,L},\ldots,a^{(t)}_{n,L})=0 \\ \qquad\qquad\vdots \\ L_{M-1,L-1}(a^{(t)}_{L+1,L},\ldots,a^{(t)}_{n,L})=0 \\ (\text{The coefficient of } x_tx_L \text{ in } g_M(\boldsymbol{x}))=0 \end{cases} \tag{6}$$

under the condition (5). However, $Q_i(a^{(t)}_{L+1,L},\ldots,a^{(t)}_{n,L})$ is not equal to zero in general for the solution of equations (6). It means that **Step *t*.**-(ii) fails in the case of $Q_i(a^{(t)}_{L+1,L},\ldots,a^{(t)}_{n,L})\neq 0$.

A.3 Example of Algorithm B

Let $k=\mathrm{GF}(7), n=7, M=2, L=3$. We consider quadratic forms represented by the following matrices.

$$G_1=\begin{pmatrix} 0&0&0&5&2&5&1\\ 0&0&0&2&3&1&2\\ 0&0&0&4&1&6&2\\ 6&5&5&4&1&6&1\\ 1&5&6&5&2&1&3\\ 2&3&5&1&5&3&1\\ 1&4&4&1&4&1&5 \end{pmatrix}, G_2=\begin{pmatrix} 1&3&4&1&5&2&3\\ 6&5&2&1&4&3&0\\ 4&6&4&1&5&0&2\\ 6&1&0&0&3&2&4\\ 4&2&6&6&0&1&3\\ 1&5&4&6&6&3&4\\ 2&5&2&4&6&3&2 \end{pmatrix}.$$

Step 1.-(i) Using Algorithm A, we solve the equations

$$\begin{cases} (\text{The coefficient of } x_1^2 \text{ in } g_2(\boldsymbol{x}))=0 \\ (\text{The coefficient of } x_1x_2 \text{ in } g_2(\boldsymbol{x}))=0, \end{cases}$$

i.e.,

$$\begin{cases} 1 + 2a_{2,1}^{(1)} + a_{3,1}^{(1)} + 5{a_{2,1}^{(1)}}^2 + a_{2,1}^{(1)}a_{3,1}^{(1)} + 4{a_{3,1}^{(1)}}^2 = 0 \\ 6 + a_{3,2}^{(1)} = 0 \end{cases}$$

From these equations, we obtain $(a_{2,1}^{(1)}, a_{3,1}^{(1)}, a_{3,2}^{(1)}) = (2, 5, 1)$. Then,

$$G_1 \mapsto \begin{pmatrix} 0 & 0 & 0 & 1 & 6 & 2 & 1 \\ 0 & 0 & 0 & 6 & 4 & 0 & 4 \\ 0 & 0 & 0 & 4 & 1 & 6 & 2 \\ 6 & 3 & 5 & 4 & 1 & 6 & 1 \\ 6 & 4 & 6 & 5 & 2 & 1 & 3 \\ 5 & 1 & 5 & 1 & 5 & 3 & 1 \\ 1 & 1 & 4 & 1 & 4 & 1 & 5 \end{pmatrix}, G_2 \mapsto \begin{pmatrix} 0 & 1 & 0 & 1 & 3 & 1 & 6 \\ 6 & 3 & 6 & 2 & 2 & 3 & 2 \\ 1 & 3 & 4 & 1 & 5 & 0 & 2 \\ 1 & 1 & 0 & 0 & 3 & 2 & 4 \\ 3 & 1 & 6 & 6 & 0 & 1 & 3 \\ 3 & 2 & 4 & 6 & 6 & 3 & 4 \\ 1 & 0 & 2 & 4 & 6 & 3 & 2 \end{pmatrix}.$$

Step 1.-(ii)

$$\begin{cases} (\text{The coefficient of } x_1x_3 \text{ in } g_1(\boldsymbol{x})) = 0 \\ (\text{The coefficient of } x_2x_3 \text{ in } g_1(\boldsymbol{x})) = 0 \\ (\text{The coefficient of } x_1x_3 \text{ in } g_2(\boldsymbol{x})) = 0 \\ (\text{The coefficient of } x_3^2 \text{ in } g_1(\boldsymbol{x})) = 0, \end{cases}$$

i.e.,

$$\begin{cases} 5b_{5,3}^{(1)} + 2b_{7,3}^{(1)} = 0 \\ 2b_{4,3}^{(1)} + b_{5,3}^{(1)} + b_{6,3}^{(1)} + 5b_{7,3}^{(1)} = 0 \\ 1 + 2b_{4,3}^{(1)} + 6b_{5,3}^{(1)} + 4b_{6,3}^{(1)} = 0 \\ b_{1,3}^{(1)}(5b_{5,3}^{(1)} + 2b_{7,3}^{(1)}) + b_{2,3}^{(1)}(2b_{4,3}^{(1)} + b_{5,3}^{(1)} + b_{6,3}^{(1)} + 5b_{7,3}^{(1)}) + 4{b_{4,3}^{(1)}}^2 \\ \quad +6b_{4,3}^{(1)}b_{5,3}^{(1)} + 2b_{4,3}^{(1)}b_{7,3}^{(1)} + 2{b_{5,3}^{(1)}}^2 + 6b_{5,3}^{(1)}b_{6,3}^{(1)} + 3{b_{6,3}^{(1)}}^2 + 2b_{6,3}^{(1)}b_{7,3}^{(1)} + 5{b_{7,3}^{(1)}}^2 = 0 \end{cases}$$

These multivariate equations are hard to solve.

Quantum Key Distribution in the Classical Authenticated Key Exchange Framework

Michele Mosca[1,2], Douglas Stebila[3], and Berkant Ustaoğlu[4]

[1] Institute for Quantum Computing and Dept. of Combinatorics & Optimization University of Waterloo, Waterloo, Ontario, Canada
[2] Perimeter Institute for Theoretical Physics, Waterloo, Ontario, Canada
mmosca@uwaterloo.ca
[3] Information Security Discipline, Queensland University of Technology, Brisbane, Queensland, Australia
stebila@qut.edu.au
[4] Department of Mathematics, Izmir Institute of Technology, Urla, Izmir, Turkey
bustaoglu@uwaterloo.ca

Abstract. Key establishment is a crucial primitive for building secure channels in a multi-party setting. Without quantum mechanics, key establishment can only be done under the assumption that some computational problem is hard. Since digital communication can be easily eavesdropped and recorded, it is important to consider the secrecy of information anticipating future algorithmic and computational discoveries which could break the secrecy of past keys, violating the secrecy of the confidential channel.

Quantum key distribution (QKD) can be used generate secret keys that are secure against any future algorithmic or computational improvements. QKD protocols still require authentication of classical communication, although existing security proofs of QKD typically assume idealized authentication. It is generally considered folklore that QKD when used with computationally secure authentication is still secure against an unbounded adversary, provided the adversary did not break the authentication during the run of the protocol.

We describe a security model for quantum key distribution extending classical authenticated key exchange (AKE) security models. Using our model, we characterize the long-term security of the BB84 QKD protocol with computationally secure authentication against an eventually unbounded adversary. By basing our model on traditional AKE models, we can more readily compare the relative merits of various forms of QKD and existing classical AKE protocols. This comparison illustrates in which types of adversarial environments different quantum and classical key agreement protocols can be secure.

Keywords: quantum key distribution, authenticated key exchange, cryptographic protocols, security models.

P. Gaborit (Ed.): PQCrypto 2013, LNCS 7932, pp. 136–154, 2013.

1 Introduction

Quantum key distribution (QKD) promises new security properties compared to cryptography based on computational assumptions: two parties can establish a key using a pair of quantum and classical channels, secure against any adversary who is limited solely by the laws of quantum mechanics. Most information-theoretically secure classical[1] cryptographic tasks have limited practicality, so many schemes' security rely on computational assumptions, the most widely used of which—factoring, discrete logarithms—could be efficiently solved by a large-scale quantum computer. As a result, QKD could be an important primitive for cryptography secure against advances in computing technology, provided quantum mechanics remains an accurate description of the laws of nature.

The classical cryptographic literature has extensively studied *authenticated key exchange* (AKE) since the founding of public key cryptography in 1976. After a period of ad hoc security analysis, protocols are now generally analyzed in a security model where an active attacker controls communication and can possibly compromise certain private information; proofs usually consist of probabilistic reductions to computationally hard problems. The seminal work in this area by Bellare and Rogaway [1] was followed by the more modern CK01 [2] and eCK [3] models; an alternative approach to this family of security models is given by Canetti's *universal composability (UC) framework* [4]. Typically in AKE protocols, calculating a secret key is relatively easy, but authentication—ensuring that the key is shared only with only the intended party—requires greater care.

There are many types of QKD protocols, but for our purposes we will divide them into 3 classes: prepare-send-measure protocols, measure-only protocols, and prepare-send-only protocols. The first QKD protocol, now called BB84 [5], is an example of a prepare-send-measure protocol in which Alice randomly prepares one of several quantum states, sends it to Bob, and Bob randomly measures in one of several settings. Ekert [6] proposed an entanglement-based protocol, which is an example of a measure-only protocol: Alice and Bob only randomly measure in one of several settings; the state itself can be prepared by Eve entirely untrusted. Biham et al. [7] proposed a prepare-send-only protocol, in which Alice and Bob each randomly prepare one of several quantum states and send them to Eve, who measures and sends back a classical result. Different versions can be appealing due to ease of implementation, resistance to side-channel attacks on preparing or measuring, or device independence.

Research on QKD security has largely proceeded independent of the aforementioned classical AKE security models. Various proofs of QKD have been given in a stand-alone 2-party setting [8,9,10,11,12,13,14]. This contrasts with the aforementioned security models used in classical AKE protocols, which consider the multi-party, multi-session setting, and consider various types of information leakage or compromise. Existing QKD proofs typically take place under the assumption that classical communication happens over on authentic public

[1] We use the adjective "classical" to mean "non-quantum", so "classical cryptography" means "non-quantum cryptography", not "historical cryptography".

channel. It is generally considered folklore [15,16,17,18] that if QKD was performed using a computationally secure message authentication scheme (such as public key digital signatures), then messages encrypted under the keys output by QKD would be secure provided that the adversary could not break the authentication scheme *before or during* the QKD protocol. This result has only been justified formally in this paper and in our concurrent work by Unruh in the universal composability setting [19].

Contributions. Our goal is to describe the security of quantum key distribution in a security model similar to existing classical authenticated key exchange protocols and compare the relative security properties of various QKD and classical AKE protocols. Our model is explicitly a multi-party model, includes authentication, and allows for either computationally secure or information theoretically secure authentication. We aim to capture two properties: (1) QKD is *immediately secure* against an active adversary who is restricted such that he is unable to break the authentication scheme, and (2) QKD is *long-term secure*, meaning that, if it is secure against an active adversary who is restricted during the run of the protocol to be unable to break the authentication scheme, then it remains secure even when the (classical and quantum) data obtained by the active bounded adversary are later given to an unbounded quantum adversary.

Security model for classical-quantum AKE protocols. We first introduce in Section 2 a multi-party model for analyzing the security of QKD protocols. In our model, which adopts the formalism of Goldberg et al.'s framework for AKE [20], parties consist of a pair of classical and quantum Turing machines, each of which is capable of sending and receiving messages. The adversary controls all communications between parties, but is restricted in its ability to affect communication between a party's classical and quantum devices. The adversary also has the ability to compromise various values used by parties before, during, or after the run of the protocol. As is typical, the adversary's goal is to distinguish the session key of a completed session from a random string of the same length. A novelty of our approach is a new technique for defining matching sessions.

Having defined the adversarial model, we then introduce our two security definitions, *immediate security* against an active, potentially bounded adversary, and *long-term security*, against an adversary who during the run of the protocol may be bounded, but after the protocol completes is unbounded (except by the laws of quantum mechanics). Our model is generic enough to allow the bound on the adversary to be computational—assuming that a particular computational problem is hard—or run-time or memory-bounded [21]. We adapt the long-term security notion of Müller-Quade and Unruh [22] from the classical universal composability framework to our classical-quantum model.

*Security of BB84.*We then proceed in Section 3 to show that the BB84 protocol, when used with a computationally secure classical authentication scheme such as a digital signature, is secure in this model. For the quantum aspects of the proof, we rely on existing proof techniques. This is next extended to provide a proof of the folklore theorem that QKD, when used with computationally secure authentication in a multi-party setting, is information theoretically secure,

provided the adversary did not break the authentication during the run of the protocol. Our argument explicitly identifies which secret information leakage does not affect security either before or after the run of the protocol.

Comparison of quantum and classical AKE protocols. Finally, we use our generic security model to compare in Section 4 the security properties of classical key exchange protocols and examples from each of the three classes of QKD protocols (prepare-send-measure, measure-only, prepare-send-only). This comparison is facilitated by our phrasing of QKD in a security model more closely related to traditional AKE security models, which we can then use to compare the relative powers afforded to the adversary under those models. In particular, our model allows us to compare how different protocols react when the randomness used in the protocol is revealed—or if it is later discovered that bad randomness was used. For example, some classical AKE protocols such as `UP` [23] are secure even if the randomness used for either a party's long-term secret key or ephemeral secret key is revealed *before* the run of the protocol, but the same is not true for the randomness used to pick basis choices in BB84. And the EPR protocol of Ekert is secure even if all of the randomness used by the parties is leaked after the protocol completes, unlike BB84 where data bit choices must remain secret. Since obtaining high quality randomness can be very challenging in practice—requiring either a separate, tested quantum source, or relying on a pseudorandom number generator seeded from a high quality source of entropy—it may be desirable to select a protocol based on the quality of randomness available, and our framework provides a method for comparing protocols along these lines.

Comparison with other frameworks. Our approach to defining security differs from existing work in several essential ways. Stand-alone QKD security definitions do not consider the security in a *multi-party setting*, and also tend to ignore entirely the question of *explicit authentication*, instead assuming an authentic classical channel. It is widely recognized that the authentication can be secure against an unbounded adversary if all classical communication is protected by information-theoretically secure message authentication codes, such as the Wegman-Carter 2-universal hash function [24,25]. However, as mentioned above, the classical AKE experience suggests that it is the authentication part of the overall security definition that is often violated; more so when there is *information leakage* to adversary. With a few exceptions (e.g., [13]), stand-alone definitions also exclude the possibility of the adversary learning private information. The universal composability definition of QKD security of Ben-Or et al. [26] (which is an adaptation of Canetti's UC framework [4] to the quantum setting), notably referenced by Renner in his thesis [14], also brushes aside the possibility of any information being leaked to the adversary and focuses solely on information-theoretic authentication. Other frameworks for composability of quantum protocols have been given [27,28,29,30,31] and applied to other types of cryptographic protocols, but not QKD. Our model, then, is the first to define QKD security in the multi-party setting, with explicit consideration of authentication, allowing leakage of information the adversary. Moreover, it defines both

short-term and long-term security; last but not least our definitions paves way for formally analyzing and comparing both classical and quantum AKE protocols within the same framework. In work concurrent with our, Unruh [19] analyzes the long-term security of QKD in the UC framework.

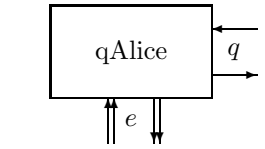
2 QKD Model

Our model begins as an enhancement to the eCK model [3], following the notation of Goldberg et al. [20]. In our model, each party has access to a quantum device. The quantum device may be viewed as limited based on for example current hardware limitations. As usual we consider interactive protocols within a multi-party multi-session setting, where communication is controlled by the adversary. The adversary controls the quantum communication channel between parties, subject to the laws of quantum physics. We also describe how, if at all, the adversary may gain access to secrets used by the parties. We then define secrecy against bounded adversaries and long-term security against unbounded adversaries: the long-term security definition is achieved by having the active bounded short-term adversary output a classical and quantum transcript upon which the unbounded quantum adversary may operate.

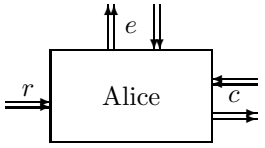
We next formally describe the model. We use k to denote a security parameter. Our description uses qubits but can be generalized to arbitrary-dimension quantum systems.

2.1 Parties and Protocols

A *party* (see also [32, Def. 1.1, bullet 2]) is an interactive classical Turing machine with access to a quantum Turing machine. We refer to this pair jointly as the party.



(a) Quantum Turing machine



(b) Classical Turing machine

Fig. 1. A party's classical and quantum Turing machines

The classical machine can activate the quantum device via a special activation request or receive (via designated activation routines) measurement outcomes from the quantum device. The communication is delivered over a two way classical communication tape (the e-channel in Figure 1(b)). The classical Turing machine has also access to a sequence of random bits – the r-tape in Figure 1(b) – and a separate c-tape over which the party can receive and send other activation requests and messages as specified by designated routines. Similarly, the quantum Turing device can be activated by the classical Turing machine and can receive and send qubits over a designated quantum channel q as in Figure 1(a).

Each party can have associated authenticated public strings (e.g., public keys or identifiers), which are assumed to be distributed over an authenticated channel

to other parties. Furthermore, pairs of parties may possess shared secrets that were distributed confidentially a priori.

A *protocol* is a collection of interactive classical and quantum subroutines that produce a shared secret key between two (or more parties) or output an indicator of an error. The interactions may use messages received on either the classical or quantum channels. The final output of the protocol is made via the classical Turing machine.

A *session* is an execution of the protocol. Sessions are initiated via a special incoming request and upon initiation each one is identified with a unique[2] *session identifier* Ψ chosen by the party at which the session is executed (in which case we say the party *owns* the session). A session that has been initiated but is not yet completed is called *active*. Since sessions are interactive procedures a party may own more than one active session at a given point of time. Each active session has a separate *session state* that stores session-specific classical data.[3]

Upon receiving and sending all protocol messages and performing the required measurements and computations specified by the protocol, the session *completes* by having the classical Turing machine output either an error symbol $\perp$ or a tuple $(sk, pid, \mathbf{v}, \mathbf{u})$. The tuple consists of:

- sk: a session key;
- pid: a party identifier;
- $\mathbf{v}$: a vector $(\mathbf{v}_0, \mathbf{v}_1, \dots)$ where each $\mathbf{v}_i$ is a vector of public values or labels; (For example, $\mathbf{v}_1$ may consist of the public values contributed by party P_1. Including $\mathbf{v}$ in the session output binds the session with the various values used by the parties to compute the session key.)
- $\mathbf{u}$: a vector $(\mathbf{u}_0, \mathbf{u}_1, \dots)$ where each $\mathbf{u}_i$ is vector of a public values or labels; $\mathbf{u}$ is called the *authentication* vector and indicates what the session owner uses to identify its peer pid.

The vectors $\mathbf{v}$ and $\mathbf{u}$ will play an important role in defining freshness.

Definition 1 (Correctness). *A key exchange protocol π is* correct *if, when all protocol messages are relayed faithfully, without changes to content or ordering, the peer parties output the same session key k and the same vector $\mathbf{v}$.*

Memory. A party may hold in its memory several *value pairs* of the form (x, X), generated by some algorithm specified by the protocol, where x is a private value and X is a public value or label. The pair may be a *public key pair*, such as private key x and public key X, or a *labelled private value*, such as a private value x and a unique public label $X = \ell(x)$.

[2] With this definition uniqueness is guaranteed only within a party; globally uniqueness can be guaranteed by requiring the session identifier is the concatenation of the unique party identifier and the party's own session identifier.

[3] While quantum protocols in general may make use of quantum memory for storing quantum states during a session, the current QKD protocols we consider in this paper, such as BB84 or EPR, do not, so we omit this from our model.

There are two classifications of value pairs: *ephemeral* value pairs, which are associated with a particular session Ψ, and *static* value pairs, which can be used across multiple sessions. The party may also have value pairs that have been generated but not yet used. If necessary, different types of key pairs may be permitted, for example, if a protocol uses one type of key pair for digital signatures and another type of key pair for public-key encryption. The protocol specifies an algorithm for generating new pairs.

Classical Turing machine communication. As described above each classical Turing machine has two incoming-outgoing classical communication channels, denoted by e and c in Figure 1(b), over which the classical Turing machine receives activations and submits responses. The responses themselves can be activation requests. Furthermore the classical Turing machine has an input of classical random bits which can be read at will by the Turing machine, denoted by r in Figure 1(b). The following activations of the classical Turing machine are allowed:

- $\mathsf{SendC}(params, pid)$: This activation is received via channel c and directs the party to begin a new key exchange session. A new session is initiated and assigned a unique session identifier Ψ based on protocol-specific public parameters $params$ and an identifier pid of the party with whom to establish the session. The response to this query includes the session identifier Ψ and any protocol-specific outgoing classical message msg' that are sent via the outgoing channel c. If required by the protocol, the Turing machine can send an activation request $\mathsf{C2Q}(m)$ over the e outgoing channel, which may in turn cause that quantum Turing machine to write an output to its q channel as well, or to prepare its measurement device to receive quantum messages.
- $\mathsf{SendC}(\Psi, msg)$: This query models the delivery of classical messages over c-channel. The party's classical Turing machine is activated with session Ψ and classical message msg. It returns any outgoing classical message msg' over the c-channel. If required by the protocol, the Turing machine can send an activation request $\mathsf{C2Q}(m)$ over the e outgoing channel, which may in turn cause that quantum Turing machine to write an output to its q channel as well, or to prepare its measurement device to receive quantum messages.
- $\mathsf{Q2C}(m)$: Upon activation with this query the classical Turing machine activates its most recent session with input m. This query may cause the classical Turing machine to output to its c channel, or send another activation over the e channel.

A protocol may request that the classical Turing machine acts probabilistically, in which case it reads random bits from the r-channel.

Quantum Turing machine communication. Each party's quantum Turing machine has a two-way quantum communication channel, denoted by q in Figure 1(a), over which the machine receives and submits quantum information. The responses themselves can be activation requests. Furthermore the quantum Turing machine has a two-way classical control channel (denoted by e in Figure 1(a)) with which it communicates with the classical Turing machine.

The following activations of the quantum Turing machine are allowed:

- $\mathsf{SendQ}(\rho)$: This query activates the quantum Turing machine with quantum message ρ; it returns any outgoing quantum message ρ' over the q-channel. If required by the protocol, the quantum Turing machine can send an activation request $\mathsf{C2Q}(m)$ over the e outgoing channel (for example, to report any measurement results obtained from measuring ρ), which may in turn cause that classical Turing machine to write an output to its c channel as well.
- $\mathsf{C2Q}(m)$: This query activates the quantum Turing machine with classical control message m, for example to prepare the quantum circuit for execution due to an anticipated SendQ activation. The activation may cause a quantum state to be output over the outgoing quantum channel q as well as a classical message to be returned over classical control channel e.

2.2 Adversarial Model

The *adversary* is, similar to a party, a pair of interactive classical and quantum Turing machines. The adversary's classical Turing machine runs in time at most $t_c(k)$ and has access to a quantum Turing machine with runtime bounded by $t_q(k)$ and memory bounded by $m_q(k)$ qubits; bounds may be unlimited. The adversary takes as its input all public information and may interact with the (honest) parties. Furthermore the adversary can establish corrupted (dishonest) parties which it fully controls. Honest parties cannot distinguish between honest and dishonest parties.

Communication over the parties' classical c-channels is controlled by the adversary. On the classical channels, the adversary can read, copy, reorder, insert, delay, modify, drop or forward messages at will. The sending and receiving parties have no intrinsic mechanism to detect which actions, if any, the adversary performed on the classical messages.

Communication over the parties' quantum q channels is also controlled by the adversary. The adversary's operations on the quantum channels are bound by the laws of quantum mechanics: the delivery of quantum messages can be delayed, modified in order, forwarded, or dropped; the adversary can create new quantum states and perform joint quantum operations on quantum messages received from the parties as well as on the adversary's state. However, due to the laws of quantum mechanics, the adversary cannot necessarily obtain full information about quantum messages from the parties; for example, measurements may irrevocably disturb the state of messages transmitted by the parties, and the adversary may be unable to precisely copy a message due to the no-cloning theorem. We assume communication between the adversary's quantum machine and party's quantum machines is perfect: the adversary can simulate any environmental effect or noise on qubits sent by a party.

Queries. The adversary can direct a party to perform certain actions by sending any of the aforementioned activation queries over party's the c and q channels.

The adversary has neither immediate control and cannot observe the content exchanged between the classical and quantum subcomponents of a party over the e channel, nor has information about the bits obtained from the r-channel. Furthermore, to allow for information leakage the adversary may issue the following queries to parties:

- $\mathsf{RevealNext} \rightarrow X$: This query allows the adversary to activate the classical Turing machine to read input from the r-channel and learn future public values. The activated party generates a new value pair (x, X), records it as unused, and returns the public value X. (This query may be specialized if there are multiple value pair types specified by the protocol.)
- $\mathsf{Reveal}(X) \rightarrow x$: This query allows the adversary to compromise secret values used in the protocol computation.[4] If the party has a value pair (x, X) in its memory, it returns the private value x. $\mathsf{Reveal}(\Psi)$ returns the secret key sk for session Ψ, if it exists; this is often referred to as a RevealSessionKey query.

Where necessary to avoid ambiguity, we use a superscript to indicate the party to whom the query is directed, for example $\mathsf{SendC}^{P_i}(\Psi, msg)$.

Revealing. If (x, X) is a value pair, with public key value or public label X, then the adversary is said to have *revealed the secret for* X if the adversary issued the query $\mathsf{Reveal}(X)$ to a party holding that value pair in its memory. In general, the adversary can reveal the secret for any value X, though this may affect which sessions are fresh.

2.3 Security Definition

For the purpose of defining session key security, the adversary has access to the following additional oracle:

- $\mathsf{Test}(i, \Psi) \rightarrow \kappa$: If party P_i has not output a session key, return $\perp$. Otherwise, choose $b \xleftarrow{\$} \{0, 1\}$. If $b = 1$, then return the session key sk from the output for session Ψ at party P_i. If $b = 0$, return a random bit string of length equal to the length of the session key sk in session Ψ at party P_i. Only one call to the Test query is allowed.

Definition 2 (Fresh session). *A session Ψ owned by an honest party P_i is* fresh *if all of the following occur:*

1. *For every vector $\mathbf{v}_j$ in P_i's output for session Ψ, there is at least one element X in $\mathbf{v}_j$ for which the adversary has not revealed the secret.*
2. *The adversary did not issue $\mathsf{Reveal}(\Psi')$ to any honest party P_j for which Ψ' has the same public output vector as Ψ (including the case where $\Psi' = \Psi$ and $P_j = P_i$).*

[4] Our notation here is altered from that of Goldberg et al. [20], in that we call this query Reveal instead of their original term Partner.

3. At the time of session completion, *for every vector* $\mathbf{u}_j$, $j \geq 1$, *in* P_i*'s output for session* Ψ, *there was at least one element* X *in* $\mathbf{u}_j$ *for which the adversary has not revealed the secret.*

The difference between the first condition (involving $\mathbf{v}$) and the third condition (involving $\mathbf{u}$) is that there are some values ($\mathbf{u}$) that are okay for the adversary to learn after the session completes but not before, whereas there may be other values ($\mathbf{v}$) that he can never learn.

Definition 3 (Security). *Let* k *be a security parameter. An authenticated key exchange protocol is* secure *if, for all adversaries* $\mathcal{A}$ *with classical runtime bounded by* $t_c(k)$, *quantum runtime bounded by* $t_q(k)$, *and quantum memory bounded by* $m_q(k)$, *the advantage of* $\mathcal{A}$ *in guessing the bit* b *used in the* Test *query of a fresh session is negligible in* k; *in other words, the probability that* $\mathcal{A}$ *can distinguish the session key of a fresh session from a random string of the same length is negligible.*

Output vectors. One of the key differences between our model and traditional AKE security models is how we phrase restrictions on what secret values the adversary can learn and when. In the eCK model, for example, a fresh session is defined as one in which the adversary has not learned (a) both the session owner's ephemeral secret key x and long-term secret key a, and (b) both the peer's ephemeral secret key y and long-term secret key b (or just the peer's long-term key if no matching peer session exists). In our model, this could be specified as $\mathbf{v} = (\mathbf{v}_0 = (a, x), \mathbf{v}_1 = (b, y))$.

Since in traditional AKE security models the restriction on values learned is specified in the security model, a new security model is required for each differing combination of learnable values. Though models may often appear similar, they sometimes contain subtle but important formal differences and thus become formally incomparable [33]. The traditional approach of specifying the values that can or cannot be learned in the security definition itself contrasts with our approach—building on that of Goldberg et al. [20]—where the vectors $\mathbf{v}$ and $\mathbf{u}$ in the session output specify what can or cannot be learned. As a result, two protocols with differing restrictions on values that can be learned could both be proven secure in our model and then compared based on which values can or cannot be revealed.

2.4 Long-Term Security

One of the main benefits of quantum key distribution is that it can be secure against unbounded adversaries, but this comes at the cost of being unable to use computationally secure cryptographic primitives such as public key digital signatures for authentication. Definition 3 can be used to analyze QKD when computationally secure cryptographic primitives are used by choosing a $t_c(k)$, $t_q(k)$, and $m_q(k)$ such that the cryptographic primitive is believed secure against

such an adversary. The particular values may be chosen based on known classical algorithms for factoring or discrete logarithms and on present-day limits of quantum devices.

Regardless of the bound on the active adversary, we can still recover a very strong form of long-term security by considering an unbounded quantum Turing machine acting after the protocol has completed. In other words, during the run of the protocol, we assume a bounded adversary as in Definition 3; this bounded active adversary produces some classical and quantum transcript which it provides to the unbounded adversary. This models the real-world scenario of an adversary being somewhat limited by its classical and quantum computing equipment now but later having much more powerful equipment or making an algorithmic breakthrough.

Definition 4 (Long-term security). *An AKE protocol is* long-term secure *if, for all unbounded quantum Turing machines $\mathcal{M}$ acting on a classical and quantum transcript produced by a (bounded) adversary $\mathcal{A}$ in Definition 3, the advantage of $\mathcal{M}$ in guessing the bit b used in the* Test *query of a fresh session is negligible in the security parameter.*

Bounds on devices. If $t_q(k) = m_q(k) = 0$, and Definition 4 is omitted, the model reduces to a classical definition for secure session key establishment. It refines the idea of authentication as the session output can explicitly identify how peers were identified and authenticated. Thus any classical protocol analyzed in [20] can also be analyzed in this model.

This model can be used in conjunction with present limitations of quantum devices. While there are ongoing improvements in controlling quantum systems, at present the number of qubits a device can work with is essentially a small constant compared to classical computers. Thus, using our model with appropriate values of $t_q(k)$ and $m_q(k)$, one can devise efficient protocols that are easy to implement but guarantee unconditional future secrecy. An appropriate assumption on $t_c(k)$—for example that all adversaries with polynomial running time $t_c(k)$ cannot solve a particular hard problem—allow the model to be used as existing classical reductionist security models are used.

Of course, the devices available to the adversary can be made unbounded essentially allowing a complete quantum world. Thus the definitions presented here are suitable for analyzing novel QKD protocols. These alternatives show the wide range of scenarios our definitions incorporate. Due to the unified underlying framework it is easier to compare various protocols and decide which one is the best for the task at hand.

3 BB84

We now turn to the BB84 protocol [5]. We first specify the protocol in the language of the model of Section 2, discuss some aspects of our formulation, and complete the section with a security analysis. Our presentation of BB84

explicitly includes the authentication operations. We choose to focus on authentication using digital signatures, rather than authentication using symmetric key primitives, for several reasons: first, establishment of shared secret keys for authentication is in practice harder than authentic distribution of public keys; and second, the short-term and long-term security properties resulting from the use of public key authentication with QKD are not yet understood.

Definition 5. *Let k be a security parameter. The* BB84 protocol *is defined by having parties responding to activations as follows:*

1. Upon activation $\mathsf{SendC}(\mathtt{start}, \mathtt{initiator}, B)$ the classical Turing machine A does the following:
   (a) create a new session Ψ^A with peer identifier B;
   (b) read n_1 (random) data bits Ψ^A_{dAB} and n_1 (random) basis bits Ψ^A_{bA} from its r-tape;
   (c) send activation $\mathsf{C2Q}(\Psi^A_{bA}, \Psi^A_{dAB})$ on its e-tape, which indicates that the quantum device should encode each data bit from Ψ^A_{dAB} as $|0\rangle$ or $|1\rangle$ if the corresponding basis bit Ψ^A_{bA} is 0, or as $|+\rangle$ or $|-\rangle$ if the corresponding basis bit Ψ^A_{bA} is 1;
   (d) send activation $\mathsf{SendC}(\Psi^A, \mathtt{start}, \mathtt{responder}, A)$ on its c-tape to B.
2. Upon activation $\mathsf{SendC}(\Psi^A, \mathtt{start}, \mathtt{responder}, A)$ the classical Turing machine B does the following:
   (a) create a new session Ψ^B with peer identifier A;
   (b) read n_1 (random) basis bits Ψ^B_{bB} from its r-tape;
   (c) send activation $\mathsf{C2Q}(\Psi^B_{bB})$ on its e-tape, which indicates the quantum device should measure the ith qubit in the $|0\rangle/|1\rangle$ if the ith bit of Ψ^B_{bB} is 0, or in the $|+\rangle/|-\rangle$ basis if ith bit of Ψ^B_{bB} is 1.
3. Upon activation $\mathsf{Q2C}(m)$, the classical Turing machine B does the following:
   (a) set Ψ^B_{dAB} equal to m;
   (b) compute $\sigma \leftarrow \mathsf{Sign}_{pk_B}(\Psi^A, \Psi^B, \Psi^B_{bB}, B)$;
   (c) send activation $\mathsf{SendC}(\Psi^A, \Psi^B, \Psi^B_{bB}, \sigma)$ on its c-tape to A.
4. Upon activation $\mathsf{SendC}(\Psi^A, \Psi^B, \Psi^B_{bB}, \sigma)$, the classical Turing machine A does the following:
   (a) verify σ with pk_B;
   (b) discard all bit positions from Ψ^A_{dAB} for which Ψ^A_{bA} is not equal to Ψ^B_{bB}; assume n_2 such positions remain;
   (c) read n_2 (random) bits Ψ^A_{indAB} from its r-tape; set Ψ^A_{chkAB} to be the substring of Ψ^A_{dAB} for which the bits of Ψ^A_{indAB} are 1, and set Ψ^A_{kAB} to be the substring of Ψ^A_{dAB} for which the bits of Ψ^A_{indAB} are 0; let n_3 denote the length of Ψ^A_{kAB}
   (d) compute $\sigma \leftarrow \mathsf{Sign}_{pk_A}(\Psi^A, \Psi^B, \Psi^A_{bA}, \Psi^A_{indAB}, \Psi^A_{chkAB}, A)$;
   (e) send activation $\mathsf{SendC}(\Psi^A, \Psi^B, \Psi^A_{bA}, \Psi^A_{indAB}, \Psi^A_{chkAB}, \sigma)$ on its c-tape to B.
5. Upon activation $\mathsf{SendC}(\Psi^A, \Psi^B, \Psi^A_{indAB}, \Psi^A_{chkAB}, \sigma)$, the classical Turing machine B does the following:
   (a) verify σ with pk_A;
   (b) discard all bit positions from Ψ^B_{dAB} for which Ψ^A_{bA} is not equal to Ψ^B_{bB}
   (c) set Ψ^B_{chkAB} to be the substring of Ψ^B_{dAB} for which the bits of Ψ^A_{indAB} are 1, and set Ψ^B_{kAB} to be the substring of Ψ^B_{dAB} for which the bits of Ψ^A_{indAB} are 0
   (d) let ϵ be the proportion of bits of Ψ^A_{chkAB} that do not match Ψ^B_{chkAB}; if $\epsilon > 0.061$ then abort;
   (e) compute $\sigma \leftarrow \mathsf{Sign}_{pk_B}(\Psi^A, \Psi^B, \epsilon, B)$;

(f) send activation $\mathsf{SendC}(\Psi^A, \Psi^B, \epsilon, \sigma)$ on its c-tape to A.

6. Upon activation $\mathsf{SendC}(\Psi^A, \Psi^B, \epsilon, \sigma)$, the classical Turing machine A does the following:
   (a) verify σ with pk_B;
   (b) read (random) bits Ψ^A_F from its r-tape to construct a random a 2-universal hash function $F : \{0,1\}^{n_3} \to \{0,1\}^{r'}$ (where $r' = n_3 h(\epsilon) + o(n_3)$) for information reconciliation[5] and compute $F' = F(\Psi^A_{kAB})$;
   (c) read (random) bits $\Psi^A_{P,G}$ from its r-tape to generate a random permutation P on n_3 elements and a 2-universal hash function $G : \{0,1\}^{n_3} \to \{0,1\}^{s'}$ (where $s' = n_3(1 - 3h(\epsilon)) + o(n_3)$) for privacy amplification, respectively; compute $\Psi^A_{skAB} \leftarrow G(P(\Psi^A_{kAB}))$;
   (d) compute $\sigma \leftarrow \mathsf{Sign}_{pk_A}(\Psi^A, \Psi^B, F, F', P, G, A)$;
   (e) send activation $\mathsf{SendC}(\Psi^A, \Psi^B, F, F', P, G, \sigma)$ on its c-tape to B;
   (f) output $(sk = \Psi^A_{skAB}, pid = B, \mathbf{v} = (\mathbf{v}_0 = (\ell(\Psi^A_{dAB})), \mathbf{v}_1 = (\ell(\Psi^A_{bAB})), \mathbf{v}_2 = (\ell(\Psi^B_{dAB})), \mathbf{v}_3 = (\ell(\Psi^B_{bAB})), \mathbf{v}_4 = (\ell(\Psi^A_F)), \mathbf{v}_5 = (\ell(\Psi^A_{P,G}))), \mathbf{u} = (\mathbf{u}_1 = (pk_B)))$ (recall $\ell(\cdot)$ denotes the label describing the corresponding secret value).

7. Upon activation $\mathsf{SendC}(\Psi^A, \Psi^B, F, F', P, G, \sigma)$, the classical Turing machine B does the following:
   (a) verify σ with pk_A;
   (b) use F and F' to correct Ψ^B_{kAB} to $\Psi^B_{kAB'}$;
   (c) compute $\Psi^B_{skAB} \leftarrow G(P(\Psi^B_{kAB'}))$;
   (d) output $(sk = \Psi^B_{skAB}, pid = A, \mathbf{v} = (\mathbf{v}_0 = (\ell(\Psi^A_{dAB})), \mathbf{v}_1 = (\ell(\Psi^A_{bAB})), \mathbf{v}_2 = (\ell(\Psi^B_{dAB})), \mathbf{v}_3 = (\ell(\Psi^B_{bAB})), \mathbf{v}_4 = (\ell(\Psi^A_F)), \mathbf{v}_5 = (\ell(\Psi^A_{P,G})),), \mathbf{u} = (\mathbf{u}_1 = (pk_A)))$.

Remark 1. In the output vector $\mathbf{v}$, the values $\ell(\Psi^A_{bAB})$, $\ell(\Psi^B_{bAB})$, $\ell(\Psi^A_F)$, and $\ell(\Psi^A_{P,G})$ appear as single component vectors. But in step 6(e) the values are broadcast in the clear. This may seem a bit contradictory since, if the adversary has revealed the secret for either of those values (and therefore learns their content), the session is not fresh, but because of the broadcast the adversary *does* in fact learn the values corresponding to the aforementioned labels. The important distinction is *when* the adversary obtains these values, either before or after the protocol commences and measurements are performed. For the adversary to learn these values before parties' measurements, it must reveal the secret for these values, violating session freshness. Learning the values after the session completes is not an issue and the values are given to the adversary "for free", without the need for revealing the secrets.

Remark 2. The output vector $\mathbf{u}$ represents the values which the session owner uses to authenticate its peer. Similar to $\ell(\Psi^A_{bAB})$ the authentication information has to be exclusively available to the alleged peer, but only at the time of protocol execution: they may subsequently be revealed.

Observe that for the BB84 protocol above, Alice's own authentication secret pk_A is not included in her $\mathbf{u}$ or $\mathbf{v}$ vectors. This implies that the protocol is resilient to *key compromise impersonation (KCI) attacks* [35, §2.4.2]: even with Alice's authentication keys no party is able to pretend to be someone other than Alice to Alice.

[5] For details on information reconciliation and privacy amplification, see the full version [34, Appendix A].

3.1 Security of BB84

We now show that the BB84 protocol stated above is a secure (Theorem 1) and long-term-secure (Theorem 2) AKE protocol assuming that the bounded active adversary cannot break the signature scheme.

Theorem 1 (Security of BB84). *Let k be a security parameter. Suppose that the probability ϵ_{sig} that any probabilistic polynomial time classical Turing machine with oracle access to a $(t_q(k), m_q(k))$-bounded quantum Turing machine can break the signature scheme is negligible in k. Then the BB84 protocol is a secure AKE protocol (Definition 3).*

Proof sketch. Our proof combines an existing proof of security by Christandl et al. [36] for the BB84 protocol with the sequence-of-games technique of Shoup [37]. First we show—using techniques from classical reductionist security—that no bounded adversary can (except with negligible probability) successfully tamper with the classical authenticated communication. Then we show—using techniques from QKD security proofs—that the adversary cannot distinguish the key from random. Details appear in the full version [34].

Theorem 2 (Long-term security of BB84). *Let k be a security parameter. Suppose the signature scheme is secure against all bounded adversaries as specified in Theorem 1. Then the BB84 protocol is a long-term secure authenticated key exchange protocol (Definition 4).*

Proof. The argument in fact appears in the proof of Theorem 1. In its proof, the bounds on $t_c(k)$, $t_q(k)$, and $m_q(k)$ and on the adversary are required only for guaranteeing the authenticity and origin of messages in a game hop that assures that the classical authentic communication has not been tampered with. The remainder of the argument is a typical argument for a quantum key distribution scheme, which does not require any bounds on the adversarial power. Since the unbounded adversary runs after the protocol completes, meaning it cannot inject reorder or modify messages in the transcript, therefore the past classical communication remains authentic and the result follows.

4 Comparing Classical and Quantum Key Exchange Protocols

Given the similarity of our model for both classical and quantum AKE protocols to existing classical AKE security models and our model's flexibility in analyzing the security of a variety of protocols, we can use our model to identify qualitative differences between classes of protocols.

One of the key differences between existing AKE security models such as CK01 and eCK is what randomness the adversary is allowed reveal—and when—yet still have the protocol be secure. Our framework is more generic: it is not the *model* that specifies which randomness can be revealed but the *protocol itself* in its output vectors $\mathbf{v}$ and $\mathbf{u}$. As a result, we can "compare" protocols by viewing

Table 1. Comparison of security properties of various classical and quantum AKE protocols

Protocol	Signed Diffie–Hellman [2]	UP [23]	BB84 [5]	EPR [6]	BHM96 [7,12]
Protocol type	classical	classical	quantum prepare-send-measure	quantum measure-only	quantum prepare-send-only
Security model	CK01 [2]	eCK [3], this paper	this paper	this paper	this paper
Randomness revealable **before** protocol run?	× static key × ephemeral key	at most 1 of static key, ephemeral key	× static key × basic choice × data bits × info. recon. × priv. amp.	× static key × basis choice × info. recon. × priv. amp.	× static key × basis choice × data bits × info. recon. × priv. amp.
Randomness revealable **after** protocol run?	✓ static key × ephemeral key	at most 1 of static key, ephemeral key	✓ static key ✓ basis choice × data bits ✓ info. recon. ✓ priv. amp.	✓ static key ✓ basis choice ✓ info. recon. ✓ priv. amp.	✓ static key ✓ basis choice × data bits ✓ info. recon. ✓ priv. amp.
Short-term security	computational assumption	computational assumption	computational or inf.-th.	computational or inf.-th.	computational or inf.-th.
Long-term security w/short-term-secure authentication	×	×	✓	✓	✓

them all within our model and then comparing which values are included in the output vector.[6]

Table 1 summarizes the observations of this section. We compare two qualitatively different classical AKE protocols and three qualitatively different QKD protocols: (1) the signed Diffie–Hellman protocol [2] (which can be proven secure in the CK01 model), (2) the UP protocol [23], a variant of the MQV protocol [38] which can be proven secure in the eCK model, (3) the BB84 [5] prepare-send-measure QKD protocol, (4) the EPR [6] (entanglement-based) measure-only QKD protocol, and (5) the BHM96 [7,12] prepare-send-only QKD protocol. Our model is flexible enough to allow all these protocols to be proven secure in it, of course with different cryptographic assumptions, bounds on the adversary, and different output vectors, which we compare in Table 1.

Revealing randomness before the run of the protocol. Some classical AKE protocols, especially eCK-secure protocols such as UP and similar MQV-style protocols, remain secure even if the adversary learns either the ephemeral secret key or the long-term secret key, but not both, before the run of the protocol. This contrasts with all known QKD protocols, where none of the random values—the long-term secret key, the basis choices (for measure protocols), data bits (for prepare protocols), information reconciliation function, or privacy amplification function—can be revealed to the adversary in advance. (This is why all of these values are included individually in the output vector $\mathbf{v}$ in the BB84 specification in Section 3.)

Revealing randomness after the run of the protocol. For classical AKE protocols to remain secure, at least some secret values must not be revealed after the

[6] We note that it has been shown [33] that the CK01 and eCK models are *formally incomparable*, meaning neither can be shown to imply the other.

run of the protocol. For protocols with so-called perfect forward secrecy, such as signed Diffie–Hellman, the parties' long-term secret keys can be corrupted after the run of the protocol, but not the ephemeral secret keys. For eCK-secure protocols such as MQV-style protocols like UP, either the long-term or the ephemeral secret key, but not both, can be revealed before, during, or after the protocol run. For measure-only entanglement-based QKD protocols such as EPR, all random choices made by the parties can be revealed after the run of the protocol: this is because the key bits are not chosen by the parties, nor in fact by the adversary, but are the result of measurements and (after successful privacy amplification) are uncorrelated with any of the input bits of any of the parties, including the adversary. This is not the case for prepare-and-send protocols such as BB84 or BHM96, as the sender randomly chooses data bits which must remain secret.

Short-term and long-term security. Classical AKE protocols can be proven secure only under computational assumptions, and as such only offer short-term security in the sense of Definition 3. Even against an unbounded passive adversary they do not retain any of their secrecy properties. Thus classical AKE protocols are only secure against bounded short-term adversaries; however, they can be compared on the relative strength of the bound on the adversary. This contrasts with QKD protocols. QKD can be shown to be secure against either *unbounded* short-term adversaries, by using information-theoretic authentication, or secure against bounded short-term adversaries when using a computationally secure authentication scheme as we have shown for BB84 in Section 3.1. A key contribution of the model in Section 2 is a formalism which captures the notion that QKD can remain secure against an unbounded adversary after the protocol completes, provided the adversary at the time of the run of the protocol could not break the authentication scheme.

Applications wishing to achieve both long-term security (like QKD) and resistance to randomness revelation (like eCK-secure classical AKE protocols) could do so by running both protocols in parallel for each session, and then combining the keys output by the two protocols together; if combined correctly, the resulting key would provide strong short-term security and strong long-term security. This approach is being used by QKD implementers, such as commercial QKD vendor ID Quantique.[7]

5 Conclusions

We have presented a model for key establishment which incorporates both classical key agreement and quantum key distribution. Our model can accommodate a wide range of practical and theoretical scenarios and can serve as a common framework in which to compare relative security properties of different protocols. A key aspect of our model is that restrictions on values the adversary can compromise are not specified by the model but by the output of the protocol. Using our model, we were able to provide a formal argument for the short-term and

[7] http://www.idquantique.com/images/stories/PDF/cerberis-encryptor/cerberis-specs.pdf

long-term security of BB84 in the multi-user setting while using computationally secure authentication.

The ability to compare various classical and quantum protocols in our model has allowed us to identify an important distinction between existing classical and quantum key exchange protocols. At a high level, classical protocols can provide more assurances against online adversaries who can leak or infiltrate in certain ways, but in the long run may be insecure against potential future advances. Current quantum protocols provide assurances against somewhat weaker online adversaries but retain secrecy indefinitely, even against future advances in computing technology.

Since in our model the relative strength of a fresh session is specified by the conditions given in the output vector, an interesting open problem would be to use our model develop a quantum key distribution protocol which does retain its security attributes in the short- and long-terms even if some random values were known before the run of the protocol. Also of interest is how to best combined keys from both quantum and classical key exchange protocols run in parallel.

Acknowledgements. The authors acknowledge helpful discussions with Norbert Lütkenhaus, Alfred Menezes, and Kenny Paterson.

MM is supported by NSERC (Discovery, SPG FREQUENCY, CREATE), QuantumWorks, MITACS, CIFAR, ORF. IQC and Perimeter Institute are supported in part by the Government of Canada and the Province of Ontario.

References

1. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
2. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
3. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
4. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols (extended abstract). In: Proc. 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 136–145. IEEE Press (2001)
5. Bennett, C.H., Brassard, G.: Quantum cryptography: public key distribution and coin tossing. In: Proc. IEEE International Conf. on Computers, Systems and Signal Processing, pp. 175–179. IEEE (December 1984)
6. Ekert, A.K.: Quantum cryptography based on Bell's theorem. Physical Review Letters 67, 661–663 (1991)
7. Biham, E., Huttner, B., Mor, T.: Quantum cryptographic network based on quantum memories. Physical Review A 54(4), 2651–2658 (1996)
8. Mayers, D.: Quantum key distribution and string oblivious transfer in noisy channels. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 343–357. Springer, Heidelberg (1996)

9. Lo, H.K., Chau, H.F.: Unconditional security of quantum key distribution over arbitrarily long distances. Science 283(5410), 2050–2056 (1999)
10. Biham, E., Boyer, M., Boykin, P.O., Mor, T., Roychowdhury, V.: A proof of the security of quantum key distribution (extended abstract). In: Proc. 32nd Annual ACM Symposium on the Theory of Computing (STOC), pp. 715–724. ACM Press (2000)
11. Shor, P., Preskill, J.: Simple proof of security of the BB84 quantum key distribution protocol. Physical Review Letters 85(2), 441–444 (2000)
12. Inamori, H.: Security of practical time-reversed EPR quantum key distribution. Algorithmica 34(4), 340–365 (2002)
13. Gottesman, D., Lo, H.K., Lütkenhaus, N., Preskill, J.: Security of quantum key distribution with imperfect devices. Quantum Information and Computation 4(5), 325–360 (2004)
14. Renner, R.: Security of Quantum Key Distribution. PhD thesis, Swiss Federal Institute of Technology Zürich (2005)
15. Paterson, K.G., Piper, F., Schack, R.: Quantum cryptography: A practical information security perspective. In: Zukowski, M., Kilin, S., Kowalik, J. (eds.) Proc. NATO Advanced Research Workshop on Quantum Communication and Security. NATO Science for Peace and Security Series, Sub-Series D: Information and Communication Security, vol. 11. IOS Press (2007), http://arxiv.org/abs/quant-ph/0406147
16. Alléaume, R., Bouda, J., Branciard, C., Debuisschert, T., Dianati, M., Gisin, N., Godfrey, M., Grangier, P., Länger, T., Leverrier, A., Lütkenhaus, N., Painchault, P., Peev, M., Poppe, A., Pornin, T., Rarity, J., Renner, R., Ribordy, G., Riguidel, M., Salvail, L., Shields, A., Weinfurter, H., Zeilinger, A.: SECOQC white paper on quantum key distribution and cryptography (January 2007), http://www.arxiv.org/abs/quant-ph/0701168
17. Stebila, D., Mosca, M., Lütkenhaus, N.: The case for quantum key distribution. In: Sergienko, A., Pascazio, S., Villoresi, P. (eds.) QuantumComm 2009. LNICST, vol. 36, pp. 283–296. Springer, Heidelberg (2010)
18. Ioannou, L.M., Mosca, M.: A new spin on quantum cryptography: Avoiding trapdoors and embracing public keys. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 255–274. Springer, Heidelberg (2011)
19. Unruh, D.: Everlasting quantum security. Cryptology ePrint Archive, Report 2012/177 (2012), http://eprint.iacr.org/
20. Goldberg, I., Stebila, D., Ustaoglu, B.: Anonymity and one-way authentication in key exchange protocols. Designs, Codes and Cryptography 67(2), 245–269 (2013)
21. Cachin, C., Maurer, U.: Unconditional security against memory-bounded adversaries. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1297, pp. 292–306. Springer, Heidelberg (1997)
22. Müller-Quade, J., Unruh, D.: Long-term security and universal composability. Journal of Cryptology 23(4), 594–671 (2010)
23. Ustaoglu, B.: Comparing SessionStateReveal and EphemeralKeyReveal for Diffie-Hellman protocols. In: Pieprzyk, J., Zhang, F. (eds.) ProvSec 2009. LNCS, vol. 5848, pp. 183–197. Springer, Heidelberg (2009)
24. Carter, J.L., Wegman, M.N.: Universal classes of hash functions. Journal of Computer and System Sciences 18(2), 143–154 (1979)
25. Wegman, M.N., Carter, J.L.: New hash functions and their use in authentication and set equality. Journal of Computer and System Sciences 22(3), 265–279 (1981)

26. Ben-Or, M., Horodecki, M., Leung, D.W., Mayers, D., Oppenheim, J.: The universal composable security of quantum key distribution. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 386–406. Springer, Heidelberg (2005)
27. Ben-Or, M., Mayers, D.: General security definition and composability for quantum & classical protocols (2004); arXiv:quant-ph/0409062.
28. Fehr, S., Schaffner, C.: Composing quantum protocols in a classical environment. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 350–367. Springer, Heidelberg (2009)
29. Unruh, D.: Simulatable security for quantum protocols arXiv:quant-ph/0409125. Extended abstract published as [31]
30. Unruh, D.: Universally composable quantum multi-party computation (full version) (October 2009); arXiv:0910.2912. Short version published as [31]
31. Unruh, D.: Universally composable quantum multi-party computation. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 486–505. Springer, Heidelberg (2010)
32. Aharonov, D., Ben-Or, M., Eban, E.: Interactive proofs for quantum computations. In: Yao, A.C.C. (ed.) Proc. Innovations in Computer Science (ICS 2010), pp. 453–469 (October 2010)
33. Cremers, C.: Examining indistinguishability-based security models for key exchange protocols: the case of CK, CK-HMQV, and eCK. In: Proc. 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2011), pp. 80–91. ACM (2011)
34. Mosca, M., Stebila, D., Ustaoğlu, B.: Quantum key distribution in the classical authenticated key exchange framework. Cryptology ePrint Archive, Report 2012/361 (2012), `http://eprint.iacr.org/2012/361`, `http://arxiv.iacr.org/2012/361`
35. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment. Springer (2003)
36. Christandl, M., Renner, R., Ekert, A.: A generic security proof for quantum key distribution (February 2004), `http://arxiv.org/abs/quant-ph/0402131v2`
37. Shoup, V.: Sequences of games: A tool for taming complexity in security proofs, `http://www.shoup.net/papers/games.pdf` (2006) (first version appeared in 2004)
38. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.A.: An efficient protocol for authenticated key agreement. Designs, Codes and Cryptography 28(2), 119–134 (2003)

Cryptanalysis of Hash-Based Tamed Transformation and Minus Signature Scheme

Xuyun Nie[1,2,3,4], Zhaohu Xu[1,3], and Johannes Buchmann[2]

[1] School of Computer Science and Engineering,
University of Electronic Science and Technology of China, Chengdu 611731, China
[2] Technische Universität Darmstadt, Department of Computer Science,
Hochschulstraβe 10, 64289 Darmstadt, Germany
[3] Network and Data Security Key Laboratory of Sichuan Province
[4] State Key Laboratory of Information Security, Institute of Information Engineering,
Chinese Academy of Sciences, Beijing 100093, China
xynie@uestc.edu.cn, xzh_tiger@yahoo.cn,
buchmann@cdc.informatik.tu-darmstadt.de

Abstract. In 2011, wang et al. proposed a security enhancement method of Multivariate Public Key Cryptosystems (MPKCs), named Extended Multivariate public key Cryptosystems (EMC). They introduced more variables in an original MPKC by a so-called Hash-based Tamed (HT) transformation in order to resist existing attack on the original MPKC. They proposed Hash-based Tamed Transformation and Minus (HTTM) signature scheme which combined EMC method with minus method. Through our analysis, the HTTM is not secure as they declared. If we can forge a valid signature of the original MPKC-minus signature scheme, we could forge a valid signature of HTTM scheme successfully.

Keywords: Multivariate public key cryptosystem, Minus method, Algebraic attack, Hash-based tamed transformation.

1 Introduction

For last three decades, due to the quantum computer attack [Sho99] on the traditional public key cryptosystems which based on the assumption about the difficulty of certain number theory problems, such as the Integer Prime Factorization Problem or the Discrete Logarithm Problem, people are constantly looking for cryptographic algorithms that can resist quantum computer algorithms attack. Multivariate public key cryptosystem (MPKC) is one of the promising alternatives to resist the quantum computer attack. The security of MPKC relies on the difficulty of solving systems of nonlinear multivariate quadratic (MQ) polynomial equations in a finite field, which is a NP-hard problem in general. However, this does not guarantee that these new cryptosystems are secure. Compared with RSA public key cryptosystems, the computation in MPKC can be very fast because it is operated on a small finite field. By now, there is no quantum computer algorithm to solve MQ problem in polynomial time.

P. Gaborit (Ed.): PQCrypto 2013, LNCS 7932, pp. 155–164, 2013.

The first promising construction of MPKC is the Matsumoto-Imai (MI) scheme [MI88] proposed in 1988. Unfortunately, it was defeated by Patarin in 1995 with the linearization equation method [Pat95]. Since then, many types of MPKCs were proposed such as HFE [Pat96], MFE [WYH06], TTM [Moh99], Rainbow [DS05], TTS [YC05] etc. Also, there are many attack methods proposed, for instance, linearization equation attack [Pat95] [DHN07], XL [CKPS00], Groebner basis [FJ03], differential attack [FGS05] [DFSS07] and so on. In addition to the design of the new systems, many security enhancement methods were proposed to resist existing attack. There are plus/minus, internal perturbation [Ding04], piece-in-hand [TTF04] etc. But some of them are not very successful.

In 2011, Wang et al. proposed a method named Extended Multivariate public key Cryptosystems (EMC)[WZW11]. Given an MPKC cryptosystem, they used a Hash-based Tame (HT) transformation working on the plaintext variables to introduce some new variables in public key to enhance the security of the original MPKC. This made the public key seems more complicated. Combined with HT transformation and minus method, they proposed Hash-based Tamed Transformation and Minus signature scheme. They claimed the HTTM is secure against the existing attacks without losing the efficiency of the original MPKC.

Through analysis, we found that the HT transformation can not really enhance the security of the original MPKC. Given a public key of HTTM signature scheme, if there were an algorithm $\mathcal{A}$ which can be used to forge a valid signature of the original MPKC combined with minus method, there would also exist an algorithm which could be used to forge a valid signature of HTTM. Firstly, we get a new public key which is equivalent to the original MPKC by setting all the new variables which were introduced by HT transformation equal to zero. This step can remove all the new variables and make the HT transformation change to be an affine map. And using the special structure of the HT transformation, we get the value of the matrix D which is a key parameter of the HT transformation. And then, we can derive the relationship between the inverses of two public keys on the same message. At last, given a message to be signed, we use algorithm $\mathcal{A}$ forge a valid signature under the new public key and then forge a valid signature under the HTTM scheme according the relationship derived above.

The paper is organized as follows. We introduce EMC and HTTM scheme in section 2 and present our cryptanalysis in section 3. In section 4, we present a practical attack on an instance of HTTM. Finally, in section 5, we conclude the paper.

2 Hash-Based Tamed Transformation and Minus Signature Scheme

Wang et al. introduced a function named Hash-based Tamed Transformation (HT for short) to enhance the security of MPKC. They used HT transformation on the plaintext variables and put the output into the original MPKC. They called the new scheme Extended Multivariate public key Cryptosystems (EMC).

We use the same notation as in [WZW11]. Let $\mathbb{F}_q$ be a degree k extension of the field $\mathbb{F}_2$, where $q = 2^k$, $\mathbb{F}_q^n$ be n-dimensional vector space over $\mathbb{F}_q$, $\mathbb{F}_{q^n}$ be a degree n extension of the field $\mathbb{F}_q$. Let $H(\cdot)$ be a standard hash function such as SHA-1, $H_k(\cdot)$ be an operation extracting the first k bits of $H(\cdot)$ and mapping the bit string into an element in $\mathbb{F}_q$. Let $a \parallel b$ be concatenation of variables a and b. let δ be the number of extended input variables of public key and μ $(0 \leq \mu < \delta)$ be the number of deleted equations of the central map.

2.1 General Form of MPKC

The general form of MPKC : $P : \mathbb{F}_q^n \to \mathbb{F}_q^n$, $F : \mathbb{F}_q^n \to \mathbb{F}_q^n$

$$y = (y_1, \cdots, y_n) = P(x_1, \cdots, x_n) = T \circ F \circ U(x_1, \cdots, x_n)$$

The public key of MPKC is a set of quadratic polynomials $P(x_1, \ldots, x_n) = (P_1(x_1, \ldots, x_n), \ldots, P_n(x_1, \ldots, x_n))$. The private key are two invertible affine maps T and U. The function F is called the central map of MPKC.

2.2 HT Transformation and EMC

The form of HT transformation is described as follow. $L : F_q^{n+\delta} \to F_q^n$

$$\begin{cases} \begin{pmatrix} h_1 \\ \vdots \\ h_{n-\delta} \end{pmatrix} = A \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_{n-\delta} \end{pmatrix} + \alpha_1 \\ \begin{pmatrix} h_{n-\delta+1} \\ \vdots \\ h_n \end{pmatrix} = \begin{pmatrix} x_{n-\delta+1} \\ \vdots \\ x_n \end{pmatrix} + D \cdot \begin{pmatrix} x_{n+1} \\ \vdots \\ x_{n+\delta} \end{pmatrix} + B \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_{n-\delta} \end{pmatrix} + \alpha_2 \end{cases}$$

where α_1, α_2 are $n - \delta$-dimension vector and δ-dimension vector respectively; $(n-\delta) \times (n-\delta)$ invertible matrix A and full-rank $\delta \times \delta$ diagonal matrix D ; B is a $\delta \times (n-\delta)$ random matrix. The extended variables x_{n+i} $1 \leq i \leq \delta$ are defined by

$$x_{n+i} = H_k(x_1 \parallel x_2 \parallel \cdots \parallel x_{n-\delta+i-1})$$

Hence, $(h_1, \ldots, h_n) = L(x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+\delta})$. Due to its structure, the function L can be easy inverted.

The public key of EMC is the expression of function $\bar{P}$.

$$\bar{P} = (\bar{P}_1, \ldots, \bar{P}_n) = P \circ L = T \circ F \circ U \circ L.$$

To encrypt a plaintext $x = (x'_1, \ldots, x'_n)$, they can firstly computer $x'_{n+i} = H_k(x'_1 \parallel x'_2 \parallel \cdots \parallel x'_{n-\delta+i-1})$ and substitute $x'_1, \ldots, x'_{n+\delta}$ into public key. Then, the ciphertext $y' = (y'_1, \cdots, y'_n)$ can be derived.

To decrypt a valid ciphertext is to computer T^{-1}, F^{-1}, U^{-1}, L^{-1} in turn, that is

$$x = (x'_1, \ldots, x'_n) = L^{-1} \circ U^{-1} \circ F^{-1} \circ T^{-1}(y'_1, \cdots, y'_n).$$

2.3 HTTM Signature Scheme

Wang et al. combined EMC and minus method to construct the HTTM signature scheme. we used same notations above.

Private key. The private keys of the original MPKC scheme (T, U, F and their inverses) plus L and L^{-1}.

Public key

$$\bar{P}^{-}(x_1, \ldots, x_{n+\delta}) = (\bar{P}_1, \ldots, \bar{P}_{n-\mu})$$

which is derived by removing the last μ polynomials in the public key of EMC.

Signing. Let the message be $y' = ({y'}_1, \cdots, {y'}_{n-\mu}) \in \mathbb{F}_q^{n-\mu}$. Then a signer chooses μ random elements $y'_{n-\mu+1}, \cdots, y'_n$, which are appended y' to $y' = ({y'}_1, \cdots, {y'}_n) \in \mathbb{F}_q^n$. To obtain the valid signature x', he (or she) calculates

$$x = (x'_1, \ldots, x'_{n+\delta}) = L^{-1} \circ U^{-1} \circ F^{-1} \circ T^{-1}({y'}_1, \cdots, {y'}_n).$$

Verification. After receiving the message $y' = ({y'}_1, \cdots, {y'}_{n-\mu})$ and its signature $(x'_1, \ldots, x'_{n+\delta})$, the verifier performs the following steps. Firstly, the verifier checks whether or not

$$x'_{n+i} = H_k(x'_1 \parallel x'_2 \parallel \cdots \parallel x'_{n-\delta+i-1}), 1 \leq i \leq \delta.$$

If they are true, then the verifier checks whether or not

$$\bar{P}^{-}(x'_1, \ldots, x'_{n+\delta}) = ({y'}_1, \cdots, {y'}_{n-\mu}).$$

Practical Parameters. They gave an practical scheme of HTTM, named HTTM^{v_1}, in which they chose MI as an original MPKC scheme and $n = 31$, $k = 6$, $\delta = 10$, $\mu = 5$.

See reference [WZW11] for more details.

3 Cryptanalysis of HTTM

Through theoretical analysis, we found that if there exists an algorithm $\mathcal{A}$ can forge a valid signature of the original MPKC-minus signature scheme, we could also forge a valid signature of HTTM. That is, HT transformation cannot enhance the security of MPKC signature scheme.

To show this, we need three propositions.

Proposition 1. Let all terms which contained $x_{n+i} (1 \leq i \leq \delta)$ equal to zero in public key $\bar{P}^{-}$ of a HTTM scheme, we can get a new public key $\bar{P}^{-}_{L'}$, which is equivalent to the public key of the original MPKC-minus signature scheme, where L' is the special case of the function L with matrix $D = 0$.

Proof. Let D be a zero matrix in L, we get the function $L' : F_q^n \to F_q^n$.

$$\begin{cases} \begin{pmatrix} h_1 \\ \vdots \\ h_{n-\delta} \end{pmatrix} = A \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_{n-\delta} \end{pmatrix} + \alpha_1 \\ \begin{pmatrix} h_{n-\delta+1} \\ \vdots \\ h_n \end{pmatrix} = \begin{pmatrix} x_{n-\delta+1} \\ \vdots \\ x_n \end{pmatrix} + B \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_{n-\delta} \end{pmatrix} + \alpha_2 \end{cases}$$

Namely,

$$L'(x_1, \cdots, x_n)^t = \begin{pmatrix} A & O \\ B & I \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix}$$

Note that the function L' is exactly an invertible affine map on $\mathbb{F}_q$. Denote $U_{L'} = U \circ L'$. The $U_{L'}$ is also an invertible affine map on $\mathbb{F}_q$.

So, if we set $D = 0$ in public key of $\bar{P}^-$, we could get a new public key, denoted by $\bar{P}^-_{L'}$:

$$\bar{P}^-_{L'} = M_\mu \circ T \circ F \circ U \circ L^{'} = M_\mu \circ T \circ F \circ U_{L'},$$

where M_μ is the minus function which moves the last μ polynomials in the public key. Clearly, $\bar{P}^-_{L'}$ is equivalent to the public key of the original MPKC-minus signature scheme.

The expression of $\bar{P}^-_{L'}$ can be also derived by setting all terms which contained $x_{n+i} (1 \leq i \leq \delta)$ equal to zero in public key $\bar{P}^-$.

$$\begin{aligned} \bar{P}^-_{L'}(x_1, \cdots, x_n) &= M_\mu \circ T \circ F \circ U_{L'}(x_1, \cdots, x_n) \\ &= M_\mu \circ T \circ F \circ U \circ L'(x_1, \cdots, x_n) \\ &= M_\mu \circ T \circ F \circ U(\begin{pmatrix} A & O \\ B & I \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix}) \\ &= M_\mu \circ T \circ F \circ U(\begin{pmatrix} A & O & O \\ B & I & O \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_{n+\delta} \end{pmatrix} + \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix}) \\ &= M_\mu \circ T \circ F \circ U(\begin{pmatrix} A & O & O \\ B & I & D \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix}) \\ &= M_\mu \circ T \circ F \circ U \circ L(x_1, \cdots, x_n, 0, \cdots, 0) \end{aligned}$$

□

The signatures of a message under $\bar{P}^-$ and $\bar{P}^-_{L'}$, respectively, have following relationship.

Proposition 2. Given a message $y' = (y'_1, \cdots, y'_{n-\mu}) \in \mathbb{F}_q^{n-\mu}$, consider the signatures under $\bar{P}^-$ and $\bar{P}^-_{L'}$, denote them $x' = (x'_1, \cdots, x'_{n+\delta})$ and $x'' = (x''_1, \cdots, x''_n)$, respectively. If we choose the same values of $y'_{n-\mu+1}, \cdots, y'_n$, then $x' = (x'_1, \cdots, x'_{n+\delta})$ and $x'' = (x''_1, \cdots, x''_n)$ satisfy:

(1) $x'_i = x''_i, \quad i = 1, \cdots, n-\delta;$
(2) $x'_{n-\delta+i} = x''_{n-\delta+i} - D[i][i]x'_{n+i}, \quad i = 1, \cdots, \delta$

where $D[i][i]$ be the i^{th} element in the diagonal of matrix D.

Proof. Given a message $y' = (y'_1, \cdots, y'_{n-\mu})$ and randomly chosen the value of $y'_{n-\mu+1}, \cdots, y'_n$, consider its corresponding signatures under $\bar{P}^-$ and $\bar{P}^-_{L'}$. Observing the signature generation process, we found that the only difference in two functions is the difference between L' and L.

Given $(h_1, \ldots, h_n) = (h'_1, \ldots, h'_n)$. Let $(x'_1, \cdots, x'_{n+\delta})$ and $(x''_1, \cdots, x''_n)$ be $(h'_1, \ldots, h'_n)$'s inverse under functions L and L', respectively. Then, we can easily check from the structure of functions L and L':

(1) $x'_i = x''_i, \quad i = 1, \cdots, n-\delta;$
(2) $x'_{n-\delta+i} = x''_{n-\delta+i} - D[i][i]x'_{n+i}, \quad i = 1, \cdots, \delta$

where $D[i][i]$ be the i^{th} element in the diagonal of matrix D. □

Hence, if we can get the value of matrix D, we could forge a valid signature of HTTM after forging a valid signature of $\bar{P}^-_{L'}$ by the algorithm $\mathcal{A}$.

The value of matrix D can be derived from the public key of HTTM efficiently.

Proposition 3. Given a public key of HTTM, $\bar{P}^- = M_\mu \circ T \circ F \circ U \circ L$, we can recover the value of matrix D from it.

Proof. Firstly, we derived the function $\bar{P}^-_{L'} = M_\mu \circ T \circ F \circ U \circ L'$ by setting all terms which contained $x_{n+i} (1 \le i \le \delta)$ equal to zero in public key $\bar{P}^-$.

Comparing $\bar{P}^-$ and $\bar{P}^-_{L'}$, if the inputs of map S in two functions are equal, the outputs are also equal. Hence, we focus on the outputs of L and L' in order to recover the value of matrix D.

Note that, let $x_1 = x_2 = \cdots = x_n = 0$ in L, the output of L will be

$$\begin{cases} \begin{pmatrix} h_1 \\ \vdots \\ h_{n-\delta} \end{pmatrix} = \alpha_1 \\ \begin{pmatrix} h_{n-\delta+1} \\ \vdots \\ h_n \end{pmatrix} = D \cdot \begin{pmatrix} x_{n+1} \\ \vdots \\ x_{n+\delta} \end{pmatrix} + \alpha_2 = \begin{pmatrix} D[1][1]x_{n+1} \\ \vdots \\ D[\delta][\delta]x_{n+\delta} \end{pmatrix} + \alpha_2 \end{cases}$$

while let $x_1 = x_2 = \cdots = x_{n-\delta} = 0$ in L', the output of L' will be

$$\begin{cases} \begin{pmatrix} h_1 \\ \vdots \\ h_{n-\delta} \end{pmatrix} = \alpha_1 \\ \begin{pmatrix} h_{n-\delta+1} \\ \vdots \\ h_n \end{pmatrix} = \begin{pmatrix} x_{n-\delta+1} \\ \vdots \\ x_n \end{pmatrix} + \alpha_2 \end{cases}$$

Thus, if

$$\begin{pmatrix} D[1][1]x_{n+1} \\ \vdots \\ D[\delta][\delta]x_{n+\delta} \end{pmatrix} = \begin{pmatrix} x_{n-\delta+1} \\ \vdots \\ x_n \end{pmatrix},$$

the outputs of L and L' will be equal. Thereby, the outputs of $\bar{P}^-$ and $\bar{P}^-_{L'}$ will be equal.

Due to the observation above, we can recover D by performing the following steps:

(1) For the function $\bar{P}^-$, let $x_1 = x_2 = \cdots = x_n = 0$ and $x_{n+2} = x_{n+3} = \cdots = x_{n+\delta} = 0$, thus the function changes into

$$\bar{P}^-(x_{n+1}) = T^- \circ F \circ S \circ L(\overbrace{0, \cdots, 0}^{n}, x_{n+1}, \overbrace{0, \cdots, 0}^{\delta-1})^T.$$

Taking x_{n+1} over the finite field $\mathbb{F}_q$ and storing all results of the function $\bar{P}^-(x_{n+1})$.

(2) For the function $\bar{P}^-_{L'}$, let $x_1 = x_2 = \cdots = x_{n-\delta} = 0$ and $x_{n-\delta+2} = x_{n-\delta+3} = \cdots = x_{n+\delta} = 0$, thus the function changes into

$$\bar{P}^-_{L'}(x_{n-\delta+1}) = M_\mu \circ T \circ F \circ S \circ L'(\overbrace{0, \cdots 0}^{n-\delta}, x_{n-\delta+1}, \overbrace{0, \cdots, 0}^{\delta-1})^T.$$

(3) Taking $x_{n-\delta+1}$ over the finite field $\mathbb{F}_q$ and comparing the results of $\bar{P}^-_{L'}(x_{n-\delta+1})$ to the results of the function $\bar{P}^-(x_{n+1})$, if there were x'_{n+1} and $x'_{n-\delta+1}$ satisfied $\bar{P}^-(x_{n+1}) = \bar{P}^-_{L'}(x_{n-\delta+1})$, then we have $D[1][1]x'_{n+1} = x'_{n-\delta+1}$, namely, $D[1][1] = {x'_{n+1}}^{-1} x'_{n-\delta+1}$. Similarly, we can get the values of $D[2][2], \cdots, D[\delta][\delta]$.

□

The time-complexity of recovering D is $\delta|\mathbb{F}_q|$ and the space-complexity is $|\mathbb{F}_q|$.

Hence, if there exists an algorithm $\mathcal{A}$ can forge a valid signature of the original MPKC-minus signature scheme, we could also forge a valid signature of HTTM through following steps.

(1) Firstly, we derived the function $\bar{P}^-_{L'} = M_\mu \circ T \circ F \circ U \circ L'$ by setting all terms which contained $x_{n+i} (1 \leq i \leq \delta)$ equal to zero in public key $\bar{P}^-$.

(2) Recovering the value of matrix D following by proposition 3.
(3) Given a message $y' = (y'_1, \cdots, y'_{n-\mu})$, forging a valid signature of $\bar{P}^-_{L'}$ using algorithm $\mathcal{A}$.
(4) Deriving a valid signature corresponding to the message y' of HTTM according to proposition 2.

4 Practical Cryptanalysis of HTTM^{v_1}

In [WZW11], the authors gave a practical example of HTTM, namely HTTM^{v_1}. They chose MI scheme as the original MPKC, namely, the central map of HTTM^{v_1} is

$$Y = \hat{F}(X) = X^{1+q^\theta},$$

and they set $q = 2$, $n = 31$, $k = 6$, $\delta = 10$, $\mu = 5$. They did not give the value of θ HTTM^{v_1}. We set $\theta = 11$ in our cryptanalysis such that $\gcd(q^\theta + 1, q^n - 1) = 1$. The hash function in our experiments is SHA-1.

It is well-known that we can forge a valid signature of MI- signature scheme by differential attack [DFSS07].

After generating a public key of HTTM^{v_1}, we perform the following steps.

Firstly, we set all terms which contained $x_{n+i} (1 \leq i \leq \delta)$ equal to zero in public key $\bar{P}^-$ and get the function $\bar{P}^-_{L'} = M_\mu \circ T \circ F \circ U \circ L'$. The function $\bar{P}^-_{L'}$ is equivalent to MI- scheme.

And then, we recover the value of matrix D according to the proposition 3. We did many computer experiments to verify it. The complexity of this step is that the time-complexity is $\delta|\mathbb{F}_q| = 10 \times 2^6 < 2^{10}$ and the space-complexity is $|\mathbb{F}_q| = 2^6$.

Next, given a message $y' = (y'_1, \cdots, y'_{n-\mu})$, we forge a valid signature of it under the function $\bar{P}^-_{L'}$ by using the same technique as in [DFSS07].

At last, we derive a valid signature of the message y' under the public key of HTTM^{v_1} according to proposition 2.

5 Conclusion

In this paper, we gave a practical cryptanalysis of HTTM scheme. The EMC method did not enhance the security of original MPKC. For HTTM scheme, we could forge a valid signature of it if there were an algorithm that can forge a valid signature of the original MPKC. Although the EMC method did not work, it is an interesting method which is worth further studying.

Acknowledgements. The work of this paper was supported by the National Key Basic Research Program of China (2013CB834203), the Fundamental Research Funds for the Central Universities under Grant ZYGX2010J069, the National Natural Science Foundation of China (No. 61103205).

References

[CKPS00] Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000)

[Ding04] Ding, J.: A new variant of the Matsumoto-Imai cryptosystem through perturbation. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 305–318. Springer, Heidelberg (2004)

[DS05] Ding, J., Schmidt, D.: Rainbow, a new multivariable polynomial signature scheme. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 164–175. Springer, Heidelberg (2005)

[DHN07] Ding, J., Hu, L., Nie, X., Li, J., Wagner, J.: High Order Linearization Equation (HOLE) Attack on Multivariate Public Key Cryptosystems. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 233–248. Springer, Heidelberg (2007)

[DFSS07] Dubois, V., Fouque, P., Shamir, A., Stern, J.: Practical Cryptanalysis of SFLASH. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 1–12. Springer, Heidelberg (2007)

[FGS05] Fouque, P.-A., Granboulan, L., Stern, J.: Differential Cryptanalysis for Multivariate Schemes. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 341–353. Springer, Heidelberg (2005)

[FJ03] Faugère, J.-C., Joux, A.: Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using gröbner bases. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 44–60. Springer, Heidelberg (2003)

[GJ79] Garey, M., Johnson, D.: Computers and intractability, A Guide to the theory of NP-compuleteness. W.H.Freeman (1979)

[MI88] Matsumoto, T., Imai, H.: Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 419–453. Springer, Heidelberg (1988)

[Moh99] Moh, T.: A fast public key system with signature and master key functions. Lecture Notes at EE department of Stanford University (May 1999), `http://www.usdsi.com/ttm.html`

[Pat95] Patarin, J.: Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt '88. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 248–261. Springer, Heidelberg (1995)

[Pat96] Patarin, J.: Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 33–48. Springer, Heidelberg (1996)

[Sho99] Shor, P.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev. 41(2), 303–332 (1999)

[TTF04] Tsujii, S., Tadaki, K., Fujioka, R.: Piece in Hand concept for enhanceing the security of multivariate type pulic key cryptosystem: public key without containing all the information of secret key. IACR eprint 2004/366, `http://eprint.iacr.org`

[W07] Wang, Z.: An Improved Medium-Field Equation (MFE) Multivariate Public Key Encryption Scheme. IIH-MISP (2007), `http://bit.kuas.edu.tw/iihmsp07/acceptedlistgeneralsession.html`

[WYH06] Wang, L.-C., Yang, B.-Y., Hu, Y.-H., Lai, F.: A "Medium-Field" Multivariate Public-Key Encryption Scheme. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 132–149. Springer, Heidelberg (2006)

[WZW11] Wang, H., Zhang, H., Wang, Z., Tang, M.: Extended multivariate public key cryptosystems with secure encryption function. SCIENCE CHINA Information Sciences 54(6), 1161–1171 (2011)

[YC05] Yang, B.-Y., Chen, J.-M.: Building Secure Tame-like Multivariate Public-Key Cryptosystems: The New TTS. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 518–531. Springer, Heidelberg (2005)

A Classification of Differential Invariants for Multivariate Post-quantum Cryptosystems

Ray Perlner[1] and Daniel Smith-Tone[1,2]

[1] National Institute of Standards and Technology,
Gaithersburg, Maryland, USA
[2] Department of Mathematics, University of Louisville,
Louisville, Kentucky, USA
{daniel.smith,ray.perlner}@nist.gov

Abstract. Multivariate Public Key Cryptography(MPKC) has become one of a few options for security in the quantum model of computing. Though a few multivariate systems have resisted years of effort from the cryptanalytic community, many such systems have fallen to a surprisingly small pool of techniques. There have been several recent attempts at formalizing more robust security arguments in this venue with varying degrees of applicability. We present an extension of one such recent measure of security against a differential adversary which has the benefit of being immediately applicable in a general setting on unmodified multivariate schemes.

Keywords: Matsumoto-Imai, multivariate public key cryptography, differential, symmetry.

1 Introduction

Since Peter Shor's discovery of quantum algorithms for factoring and computing discrete logarithms quickly with quantum computers, there has been a growing community with the goal of establishing a replacement for RSA or Diffie-Hellman in the quantum realm. The last two decades have witnessed a great deal of progress towards realizing that quantum computing world, indicating that Shor's discovery is a great deal more than a mathematical curiosity; instead, his discovery marks the need for an eventual paradigm shift in our public key infrastructure.

Multivariate Public Key Cryptography(MPKC) has emerged as one of a few serious candidates for security in the post-quantum world. This emergence is due to several facts. First, the problem of solving a system of quadratic equations is known to be NP-hard, and seems to be hard even in the average case. No great reduction of the complexity of this problem has been found in the quantum model of computing, and, indeed, if this problem is discovered to be solvable in the quantum model, we can solve all NP problems, which seems particularly wishful. Second, multivariate systems are very efficient, often having speeds dozens of times faster than RSA, [1–3]. Finally, several theoretical

P. Gaborit (Ed.): PQCrypto 2013, LNCS 7932, pp. 165–173, 2013.

advances have resulted in the development of modification techniques which allow multiple parameters to be hidden within a system which can be altered to achieve different performance or security properties.

One of the great challenges facing MPKC is the task of establishing reasonable security assurance. Though there have been some recent attempts at forming a new model in which to offer provable security for encryption and signatures, see for example [4, 5], it seems apparent that these models are not as general as we would like or require modifications of realistic protocols to carry their full meaning. The task of quantifying indistinguishability between general classes of systems of multivariate equations seems exceptionally difficult in light of the fact that even with a great deal of structure in the construction of a multivariate cryptosystem, the coefficients can appear to have a uniform distribution. Although history has shown that once a way to distinguish a class of systems of structured multivariate equations from a collection of randomly generated equations is discovered, a method of solving this system is often quickly developed, it is not clear that the techniques for distinguishing such systems are indicative of an underlying theme powerful enough to establish a general method of security proof.

The many cryptanalyses of various big field multivariate cryptosystems have, however, pointed out weaknesses in the predominant philosophy for the construction of such multivariate public key cryptosystems. Several systems, SFLASH, Square, for example, which are based on simple modifications of the prototypical Matsumoto-Imai public key cryptosystem, have been broken by very similar differential attacks exploiting some symmetry which is inherent to the field structure these systems utilize. See [6–9]. Even in the small field milieu, various attacks, for example the oil-vinegar attack, see [10], can be viewed as an attack on differential structure; specifically, discovering a differential invariant.

In [11], a measure of security against attacks exploiting differential symmetry was advanced. This methodology allows one to construct proofs that a cryptosystem is secure against a differential symmetry adversary by classifying the differential symmetric structure of the cryptosystem. By identifying all possible initial general linear differential symmetries possessed by a field map, one can determine which linear relations involving the differential of a public key are accessible to any adversary, and thus guarantee security against such an attack model. Although this result is not as robust as a reduction theoretic proof of security, it has the benefit, first, of being far stronger than the traditional model of checking the vulnerability of new schemes against old attacks, second, of being immediately applicable in the design of cryptosystems, and third, of perhaps being a more realistic goal than that of reduction theoretic proof.

In this article, we introduce a technique which is dual to that of [11] in the sense that it assures security against any first-order differential invariant adversary. Specifically, we establish a model for classifying first-order differential invariants of a field map and apply the model, providing classifications of such invariants for specific cryptosystems. This characterization, in conjunction with an analogous classification in the symmetric setting, provides a model for security

against any first-order differential adversary, and is the first step towards establishing general differential security via an existence criterion. We suggest such an analysis of differential invariant security as a reasonable criterion and pragmatic tool for cryptographers in the development of future multivariate schemes.

The paper is organized as follows. The next section illustrates the ubiquitous nature of the differential attack by recasting the attack on the balanced oil and vinegar scheme in the differential setting. In the following section, we focus on differential invariants, presenting the first-order differential invariant and discussing the technique for realizing the theoretical differential invariant structure of any class of MPKC. The subsequent section restricts the analysis of this space to the case in which the hidden field map of the cryptosystem is a C^* monomial. The differential invariant structure is then determined for projected systems such as the projected SFLASH analogue, pSFLASH. Finally, we review these results and suggest a general model for differential security.

2 Differential Symmetries and Invariants

Differential attacks play a crucial role in multivariate public key cryptography. Such attacks have not only broken many of the so called "big field" schemes, they have directed the further development of the field by inspiring modifiers — Plus (+), Minus (-), Projection (p), Perturbation (P), Vinegar (v) — and the creation of newer more robust techniques.

The differential of a field map, f, is defined by $Df(a,x) = f(a+x) - f(a) - f(x) + f(0)$. The use of this discrete differential appears to occur in very many cryptanalyses of post-quantum multivariate schemes. In fact, we can even consider Patarin's initial attack, in [12], on Imai and Matsumoto's C^* scheme, see [13], as the exploitation of a trivial differential symmetry. Suppose $f(x) = x^{q^\theta+1}$ and let $y = f(x)$. Since the differential of f, Df, is a symmetric bilinear function, $0 = Df(y,y) = Df(y, x^{q^\theta+1}) = yx^{q^{2\theta}+q^\theta} + y^{q^\theta}x^{q^\theta+1} = x^{q^\theta}(yx^{q^{2\theta}} + y^{q^\theta}x)$. Dividing by x^{q^θ} we have Patarin's linear relation, $yx^{q^{2\theta}} = y^{q^\theta}x$; see [12] for details.

Differential methods provide powerful tools for decomposing a multivariate scheme. To illustrate the nearly universal nature of differential attacks, we review the attack of Kipnis and Shamir, see [10], on a non-big-field system, the oil and vinegar scheme. Though they use differing terminology, the attack exploits a symmetry hidden in the differential structure of the scheme.

Recall that the oil and vinegar scheme is based on a hidden quadratic system of equations, $f : k^n \to k^o$, in two types of variables, $x_1, ..., x_o$, the oil variables, and $x_{o+1}, ..., x_{o+v=n}$, the vinegar variables. We focus on the balanced oil and vinegar scheme, in which $o = v$. Let $c_1, ..., c_v$ be random constants. The map f has the property that $f(x_1, ..., x_v, c_1, ..., c_v)$ is affine in $x_1, ..., x_v$. The encryption map, $\overline{f}$ is the composition of f with an n-dimensional invertible affine map, L.

Let O represent the subspace generated by the first v basis vectors, and let V denote the cosummand of O. Notice that the discrete differential given by $Df(a,x) = f(x+a) - f(x) - f(a) + f(0)$ has the property that for all a and x

in O, $Df(a,x) = 0$. Thus for each coordinate, i, the differential coordinate form Df_i can be represented:

$$Df_i = \begin{bmatrix} 0 & Df_{i1} \\ Df_{i1}^T & Df_{i2} \end{bmatrix}.$$

Let M_1 and M_2 be two invertible matrices in the span of the Df_i. Then $M_1^{-1}M_2$ is an O-invariant transformation of the form:

$$\begin{bmatrix} A & B \\ 0 & C \end{bmatrix}.$$

Now the Df_i are not known, but $D(f \circ L)_i = L^T Df_i L$, so the $L^T Df_i L$ are known. Notice that if M is in the span of the Df_i, then $L^T ML$ is in the span of the $L^T Df_i L$. Also, since $(L^T M_1 L)^{-1}(L^T M_2 L) = L^{-1}M_1^{-1}M_2 L$, there is a large space of matrices leaving $L^{-1}O$ invariant, which Kipnis and Shamir are able to exploit to effect an attack against the balanced oil and vinegar scheme; see [10] for details. Making the oil and vinegar scheme unbalanced, see [14], corrects this problem by making any subspace which is invariant under a general product $M_1^{-1}M_2$ very small, see [15].

3 First-Order Differential Invariants

Let $f : k \to k$ be an arbitrary fixed function on k, a degree n extension of the Galois field $\mathbb{F}_q$. Consider the differential $Df(a,x) = f(a+x) - f(a) - f(x) + f(0)$. We can express the differential as an n-tuple of differential coordinate forms in the following way:

$$[Df(a,x)]_i = a^T Df_i x,$$

where Df_i is a symmetric matrix representation of the action on the ith coordinate of the bilinear differential. A first-order differential invariant of f is a subspace $V \subseteq k$ with the property that there exists a $W \subseteq k$ of dimension at most $dim(V)$ for which simultaneously $AV \subseteq W$ for all $A \in Span_i(Df_i)$.

We note that any simultaneous invariant of all $Span_i(Df_i)$ satisfies the above definition, as well the situation for balanced oil and vinegar, in which the invariant was found in the product of an element and an inverse of an element in $Span_i(Df_i)$. A first-order differential invariant is thus a more general construct than a simultaneous invariant among all differential coordinate forms. We present a proof theoretic technique for classifying the first-order differential invariants of such a multivariate map $f : k \to k$ which can specify parameters admitting such invariant structure.

Suppose f has a first-order differential invariant V. Let $V^\perp$ represent the set of all elements x in k such that the dot product $< x, Ay >= 0$ for all $y \in V$ and for all $A \in Span_i(Df_i)$. We should note that in positive characteristic there is a great deal of freedom in membership in $V^\perp$; there is no reason that $V \cap V^\perp$ should be empty in general or even that $V \oplus V^\perp$ be contained in k. Let $M : k \to V$ be an arbitrary linear map. Choosing an arbitrary linear map $M^\perp : k \to V^\perp$

we have the following (non-linear) symmetric relation, a dual expression of the differential invariance:

$$[Df(M^{\perp}a, Mx)]_i = a^T (M^{\perp})^T Df_i Mx = 0,$$

for all i. Thus $Df(M^{\perp}a, Mx)$ is identically zero for all $a, x \in k$.

Consequently, the existence of a first-order differential invariant for a map f implies the existence of a nonlinear symmetry on f, that is, a symmetry induced by linear maps such that the system of equations expressing the symmetric relation are nonlinear in the coefficients of the maps. Note that the converse implication is false, so that having a first-order differential invariant is a stronger property than having this manner of nonlinear differential symmetry. By explicitly constructing the polynomial map $\overline{f}(a, x) = Df(M^{\perp}a, Mx) \equiv 0$ over k^2, we can derive relations permitting the existence of this nonlinear symmetry, and hence the first-order differential invariant.

4 Invariants in the Prototypical Case

As an illustration of this technique we examine the case when $f : k \to k$ is a C^* monomial map. Specifically, we let $f(x) = x^{q^{\theta}+1}$ where $(\theta, [k : \mathbb{F}_q]) = 1$. This case in particular applies to the famously broken, see [9], SFLASH signature scheme, which was constructed by composing f with two affine transformations: $P = T \circ f \circ U$, where T is singular and U is of full rank.

Theorem 1. *Let $f : k \to k$ be a C^* monomial map. Then f has no nontrivial first-order differential invariant.*

Proof. Suppose by way of contradiction that f has a first-order differential invariant $\{0\} \subsetneq V \subsetneq k$. Define $V^{\perp} = \{x | < x, Ay >= 0, \forall y \in V \text{ and } \forall A \in Span_i(Df_i)\}$. Then f satisfies the relation $Df(M^{\perp}a, Mx) = 0$ for all $a, x \in k$.

$$\begin{aligned} Df(M^{\perp}a, Mx) &= f(M^{\perp}a + Mx) - f(M^{\perp}a) - f(Mx) + f(0) \\ &= f(\sum_{i=0}^{n-1} m_i^{\perp} a^{q^i} + \sum_{i=0}^{n-1} m_i x^{q^i}) - f(\sum_{i=0}^{n-1} m_i^{\perp} a^{q^i}) - f(\sum_{i=0}^{n-1} m_i x^{q^i}) + f(0) \\ &= (\sum_{i=0}^{n-1} m_i^{\perp} a^{q^i} + \sum_{i=0}^{n-1} m_i x^{q^i})^{q^{\theta}+1} - (\sum_{i=0}^{n-1} m_i^{\perp} a^{q^i})^{q^{\theta}+1} - (\sum_{i=0}^{n-1} m_i x^{q^i})^{q^{\theta}+1} \\ &= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (m_j (m_{i-\theta}^{\perp})^{q^{\theta}} + m_i^{\perp} m_{j-\theta}^{q^{\theta}}) a^{q^i} x^{q^j}. \end{aligned} \tag{1}$$

Since the collection of monomials $\{a^{q^i} x^{q^j}\}$ are algebraically independent, the fact that the above function is identically zero implies that,

$$m_j (m_{i-\theta}^{\perp})^{q^{\theta}} + m_i^{\perp} m_{j-\theta}^{q^{\theta}} = 0,$$

for all $0 \leq i, j \leq n-1$. This fact implies that all 2×2 minors of the following matrix are zero:

$$\begin{bmatrix} m_0 & m_0^{\perp} & m_1 & \cdots & m_{n-1} & m_{n-1}^{\perp} \\ m_{-\theta}^{q^\theta} & (m_{-\theta}^{\perp})^{q^\theta} & m_{1-\theta}^{q^\theta} & \cdots & m_{n-1-\theta}^{q^\theta} & (m_{n-1-\theta}^{\perp})^{q^\theta} \end{bmatrix}.$$

Thus, the rank of this matrix is one, and we have that the second row is a multiple of the first, say $m_i^* = r(m_{i-\theta}^*)^{q^\theta}$, as well as the fact that each column is a multiple of the first, implying, for example, $m_0^{\perp} = sm_0$.

Consequently, for all $0 \leq i \leq n-1$, $m_{i\theta}^* = r^{\frac{q^{i\theta}-1}{q^\theta-1}}(m_0^*)^{q^{i\theta}}$. Moreover, we can specify that $m_{i\theta} = r^{\frac{q^{i\theta}-1}{q^\theta-1}} m_0^{q^{i\theta}}$ and $m_{i\theta}^{\perp} = r^{\frac{q^{i\theta}-1}{q^\theta-1}} s^{q^{i\theta}} m_0^{q^{i\theta}}$, which implies that $m_i^{\perp} = s^{q^i} m_i$ for all $0 \leq i \leq n-1$. Thus

$$\begin{aligned} M^{\perp}x &= \sum_{i=0}^{n-1} m_i^{\perp} x^{q^i} \\ &= \sum_{i=0}^{n-1} m_i s^{q^i} x^{q^i} \\ &= \sum_{i=0}^{n-1} m_i (sx)^{q^i} \\ &= M(sx). \end{aligned} \tag{2}$$

Hence, the fact that $Df(M(sa), Mx) = 0$ for all $a, x \in k$ implies that $Df(Ma, Mx) = 0$ for all $a, x \in k$. This result implies that $dim(Mk) \leq 1$, that is, the dimension of the image of M in k is one, by the following argument.

If $Df(\overline{a}, \overline{x}) = 0$, then $\overline{ax}\left(\overline{x}^{q^\theta-1} + \overline{a}^{q^\theta-1}\right) = 0$, and $\overline{a}^{q^\theta-1} = -\overline{x}^{q^\theta-1}$ implies that $\overline{a}^{q-1} = -\overline{x}^{q-1}$ since $(q^\theta - 1, q^n - 1) = q - 1$. This equation is satisfied exactly when there exists $\alpha \in \mathbb{F}_q$ such that $\overline{a} = \alpha\overline{x}$.

Since this nonlinear differential symmetry exists for any map $g : k \to k$, there exists no nontrivial differential invariant of f.

We can therefore conclude that C^* has no first-order differential invariant weaknesses, even though it is fraught with linear differential symmetric weaknesses. The significance of this result is that we can prove that the cryptosystem in question is secure against all first-order differential invariant adversaries, even those employing attacks yet undiscovered.

5 Invariant Properties under Projection

After SFLASH was broken, it was suggested in [16] that the affine map U be made singular. We continue, establishing security bounds for this suggestion, one of the last unbroken C^* variants, pC^{*-}, or pSFLASH. We recall that in [11] it was

established that pSFLASH with appropriately chosen parameters has no general linear differential symmetries and is thus immune to any type of differential attack relying on the accumulation of linear equations involving the differential of the public key. While it has been established in [17] that the projection in pSFLASH can be removed, the structure when the projection modifier is removed is no longer that of a C^* function; rather, it is an HFE^- scheme. Thus pSFLASH is no more secure than HFE^-, which remains unbroken. For the security details of HFE^-, please see [18].

Theorem 2. *Let $f : k \to k$ be a C^* monomial, and let $\pi : k \to k$ be a linear projection onto a codimension r subspace. Then every nontrivial first-order differential invariant V satisfies $dim(V) \leq dim(V \cap ker(\pi)) + 1$. Consequently, if $r = 1$, there is no nontrivial first-order differential invariant structure beyond the obvious $ker(\pi)$.*

Proof. Let V be a first-order differential invariant of $f \circ \pi$, and let $M : k \to V$ be an arbitrary linear map. Then $\pi \circ M$ is a first-order differential invariant of f, and there exist maps $\overline{M} = \pi \circ M$ and $\overline{M}^{\perp}$ such that:

$$D(f \circ \pi)(M^{\perp}a, Mx) = Df(\pi M^{\perp}a, \pi Mx) = Df(\overline{M}^{\perp}a, \overline{M}x) = 0,$$

for all $a, x \in k$. We note that there are exactly as many possible maps $\overline{M}^{\perp}$ as maps $\pi \circ M^{\perp}$; indeed, the proof of Theorem 1 shows us that $\overline{M}^{\perp}x = \pi \circ M^{\perp}(sx)$ for some s. As in the proof of Theorem 1, $dim(\overline{M}k) \leq 1$, and since π is of codimension r, $dim(Mk) \leq dim(Mk \cap ker(\pi)) + 1$. We note that since any map $g : k \to k$ has this property, $f \circ \pi$ has no nontrivial first-order differential invariant structure beyond $ker(\pi)$.

We can conclude from the above theorem that pSFLASH is secure against any first-order differential invariant adversary.

6 Conclusion

Multivariate public key cryptography has several desirable traits as a potential candidate for post-quantum security. Unfortunately, a standard metric by which we can judge the security of a multivariate scheme has yet to be determined. One consequence of this current status of the field is the similar cryptanalyses of several promising ideas.

We suggest the classification of first-order differential invariants as a second benchmark for the determination of differential security for multivariate public key cryptosystems. We note that while the lack of the symmetric and invariant differential security argument does not imply that a cryptosystem is insecure against a differential adversary, the presence of such an assurance guarantees the resistance against any future first-order differential attack.

The case of pSFLASH is particularly interesting because while retaining the prototypical C^* underlying structure which plagued other variants, the modifications implemented in the scheme seem to perform their intended tasks perfectly.

Most significantly, the projection modifier has provably removed the linear symmetric differential structure, as shown in [11], while retaining the flawless differential invariant structure. On the other hand, the reduction provided by the algorithm in [17] to remove the projection modifier succeeds in transforming pSFLASH into an HFE^- scheme. Although the transformation removes the C^* properties of the core map, it may well prove to be the case that the extra structure the resultant particular HFE^- scheme retains may reveal a weakness. Any new attack on this system will be very exciting, as it will indicate a fundamentally new cryptanalytic technique.

References

1. Chen, A.I.T., Chen, M.S., Chen, T.R., Cheng, C.M., Ding, J., Kuo, E.L.H., Lee, F.Y.S., Yang, B.Y.: Sse implementation of multivariate pkcs on modern x86 cpus. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 33–48. Springer, Heidelberg (2009)
2. Chen, A.I.-T., Chen, C.-H.O., Chen, M.-S., Cheng, C.-M., Yang, B.-Y.: Practical-sized instances of multivariate pKCs: Rainbow, TTS, and ℓIC-derivatives. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 95–108. Springer, Heidelberg (2008)
3. Yang, B.-Y., Cheng, C.-M., Chen, B.-R., Chen, J.-M.: Implementing minimized multivariate PKC on low-resource embedded systems. In: Clark, J.A., Paige, R.F., Polack, F.A.C., Brooke, P.J. (eds.) SPC 2006. LNCS, vol. 3934, pp. 73–88. Springer, Heidelberg (2006)
4. Sakumoto, K., Shirai, T., Hiwatari, H.: On provable security of uov and hfe signature schemes against chosen-message attack. In: [19], pp. 68–82.
5. Huang, Y.J., Liu, F.H., Yang, B.Y.: Public-key cryptography from new multivariate quadratic assumptions. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 190–205. Springer, Heidelberg (2012)
6. Clough, C., Baena, J., Ding, J., Yang, B.-Y., Chen, M.-S.: Square, a New Multivariate Encryption Scheme. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 252–264. Springer, Heidelberg (2009)
7. Baena, J., Clough, C., Ding, J.: Square-vinegar signature scheme. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 17–30. Springer, Heidelberg (2008)
8. Billet, O., Macario-Rat, G.: Cryptanalysis of the square cryptosystems. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 451–468. Springer, Heidelberg (2009)
9. Dubois, V., Fouque, P.-A., Shamir, A., Stern, J.: Practical Cryptanalysis of SFLASH. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 1–12. Springer, Heidelberg (2007)
10. Kipnis, A., Shamir, A.: Cryptanalysis of the oil & vinegar signature scheme. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 257–266. Springer, Heidelberg (1998)
11. Smith-Tone, D.: On the differential security of multivariate public key cryptosystems. In: [19], pp. 130–142.
12. Patarin, J.: Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt '88. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 248–261. Springer, Heidelberg (1995)

13. Matsumoto, T., Imai, H.: Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 419–453. Springer, Heidelberg (1988)
14. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999)
15. Patarin, J.: The oil and vinegar algorithm for signatures. In: Presented at the Dagsthul Workshop on Cryptography (1997)
16. Ding, J., Dubois, V., Yang, B.-Y., Chen, O.C.-H., Cheng, C.-M.: Could SFLASH be repaired? In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 691–701. Springer, Heidelberg (2008)
17. Bettale, L., Faugère, J.C., Perret, L.: Cryptanalysis of multivariate and odd-characteristic hfe variants. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 441–458. Springer, Heidelberg (2011)
18. Ding, J., Kleinjung, T.: Degree of regularity for hfe-. IACR Cryptology ePrint Archive 2011, 570 (2011)
19. Yang, B.-Y. (ed.): PQCrypto 2011. LNCS, vol. 7071. Springer, Heidelberg (2011)

Secure and Anonymous Hybrid Encryption from Coding Theory*

Edoardo Persichetti

University of Warsaw

Abstract. Cryptographic schemes based on coding theory are one of the most accredited choices for cryptography in a post-quantum scenario. In this work, we present a hybrid construction based on the Niederreiter framework that provides IND-CCA security in the random oracle model. In addition, the construction satisfies the IK-CCA notion of anonymity whose importance is ever growing in the cryptographic community.

1 Introduction

A *Hybrid Encryption* scheme is a cryptographic protocol that features both a public-key encryption scheme and a symmetric encryption scheme, the former with the task of encrypting a key for the latter, in charge of encrypting the actual body of the message. The first component is therefore known as *Key Encapsulation Mechanism (KEM)* while the second is called *Data Encapsulation Mechanism (DEM)*. Key feature is that the two parts are independent of one another. The framework was first introduced in a seminal work by Cramer and Shoup [6], along with the corresponding notions of security and an example of a scheme based on the DDH assumptions. In a subsequent work [12], Shoup presents a proposal for an ISO standard on public-key encryption including many different schemes based on the RSA assumptions (RSA-OAEP, RSA-KEM), elliptic curves (ECIES) and Diffie-Hellman (PSEC, ACE). Other schemes based on integer factorization such as EPOC or HIME are also mentioned.

In this paper we present a new KEM construction, based on the Niederreiter framework [9]. The work follows up a suggestion from Bernstein [4] and stems from the RSA-KEM scheme (also known as "Simple RSA" in earlier versions of the paper), and as far as we know is the first proposal for a KEM based on coding theory assumptions. The construction is proved to be CCA secure; moreover, it is shown that, for the resulting Hybrid Niederreiter encryption scheme, it is possible to achieve key-privacy in the IK-CCA sense, as formalized by Bellare et al. in [1]. Key-privacy for coding theory schemes has been studied by Yamakawa et al. in [15], where it is proved that the IND-CPA variant of McEliece by Nojima et al. [10] satisfies the weaker anonymity notion of IK-CPA. To the best of our knowledge, our work is the first code-based construction achieving IK-CCA security.

* European Research Council has provided financial support under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no CNTM-207908.

P. Gaborit (Ed.): PQCrypto 2013, LNCS 7932, pp. 174–187, 2013.

The paper is organized as follows: first, we briefly recall the basic notions of coding theory and the Niederreiter cryptosystem. We then introduce all the definitions and notions of security for KEMs and DEMs, plus other cryptographic tools that we will need for our scheme, such as KDFs and MACs. In Section 3 we introduce the construction and prove its security, then show how to realize an efficient DEM and how to compose the two parts. Anonymity notions and the corresponding result about the Hybrid Niederreiter scheme are presented in Section 4. We conclude in Section 5.

2 Preliminaries

2.1 The Niederreiter Cryptosystem

This cryptosystem was introduced by H. Niederreiter in 1985 [9]. Since it makes use of the parity-check matrix rather than the generator matrix, it is often considered as a "dual" version of the McEliece cryptosystem [7]. Due to space limitations, we leave a detailed description to Appendix A.
The security of the scheme follows from the two following computational assumptions.

Assumption 1 (Indistinguishability). *The $(n-k) \times k$ matrix M output by* KeyGen *is computationally indistinguishable from a uniformly chosen matrix of the same size.*

Assumption 2 (Syndrome Decoding Problem (SDP)). *Let H be a parity-check matrix for a random $[n, k]$ linear code over $\mathbb{F}_q$ and $\boldsymbol{s}$ be chosen uniformly at random in $\mathbb{F}_q^{(n-k)}$. Then it is hard to find a vector $\boldsymbol{e} \in \mathbb{F}_q^n$ with $\mathsf{wt}(\boldsymbol{e}) \leq w$ such that $H\boldsymbol{e}^\tau = \boldsymbol{s}$.*

SDP was proved to be NP-complete in [3].

2.2 Encapsulation Mechanisms and the Hybrid Framework

A key encapsulation mechanism is essentially a public-key encryption scheme (PKE), with the exception that the encryption algorithm takes no input apart from the public key, and returns a pair (K, ψ_0). The string K has fixed length ℓ_K, specified by the KEM, and ψ_0 is an "encryption" of K in the sense that $\mathsf{Dec}_{\mathsf{sk}}(\psi_0) = K$. Formally, a KEM consists of the following three algorithms.

A KEM is required to be *sound* for at least all but a negligible portion of public key/private key pairs, that is, if $\mathsf{Enc}_{\mathsf{pk}}(\ ) = (K, \psi_0)$ then $\mathsf{Dec}_{\mathsf{sk}}(\psi_0) = K$ with overwhelming probability.

The data encapsulation mechanism is a (possibly labeled) symmetric encryption scheme (SE) that uses as a key the string K output by the KEM. In what follows we only discuss, for simplicity, un-labeled DEMs.

Formally, a DEM consists of the following two algorithms.

Table 1. Key Encapsulation Mechanism

KeyGen	A probabilistic key generation algorithm that takes as input a security parameter 1^λ and outputs a public key pk and a private key sk.
Enc	A probabilistic encryption algorithm that receives as input a public key pk and returns a key/ciphertext pair (K, ψ_0).
Dec	A deterministic decryption algorithm that receives as input a private key sk and a ciphertext ψ_0 and outputs either a key K or the failure symbol $\perp$.

Table 2. Data Encapsulation Mechanism

Enc	A deterministic encryption algorithm that receives as input a key K and a plaintext ϕ and returns a ciphertext ψ_1.
Dec	A deterministic decryption algorithm that receives as input a key K and a ciphertext ψ_1 and outputs either a plaintext ϕ or the failure symbol $\perp$.

The security notions are similar to their corresponding ones for PKE and SE schemes (see Appendix B). We present them below.

Definition 1. *The adaptive chosen-ciphertext attack game for a KEM proceeds as follows:*

1. *Query a key generation oracle to obtain a public key* pk.
2. *Make a sequence of calls to a decryption oracle, submitting any string ψ_0 of the proper length. The oracle will respond with $\mathsf{Dec}_{sk}^{KEM}(\psi_0)$.*
3. *Query an encryption oracle. The oracle runs Enc_{pk}^{KEM} to generate a pair $(\tilde{K}, \tilde{\psi}_0)$, then chooses a random $b \in \{0,1\}$ and replies with the "challenge" ciphertext $(K^*, \tilde{\psi}_0)$ where $K^* = \tilde{K}$ if $b = 1$ or K^* is a random string of length ℓ_K otherwise.*
4. *Keep performing decryption queries. If the submitted ciphertext is ψ_0^*, the oracle will return $\perp$.*
5. *Output $b^* \in \{0,1\}$.*

The adversary succeeds if $b^* = b$. More precisely, we define the advantage *of $\mathcal{A}$ against KEM as*

$$\mathsf{Adv}_{KEM}(\mathcal{A}, \lambda) = \left| Pr[b^* = b] - \frac{1}{2} \right|. \tag{1}$$

We say that a KEM is secure if the advantage Adv_{KEM} of any polynomial-time adversary $\mathcal{A}$ in the above CCA attack model is negligible.

Definition 2. *The attack game for a DEM proceeds as follows:*

1. *Choose two plaintexts ϕ_0, ϕ_1 and submit them to an encryption oracle. The oracle will choose a random key K and a random bit $b \in \{0,1\}$ and reply with the "challenge" ciphertext $\psi_1^* = \mathsf{Enc}_K^{DEM}(\phi_b)$.*

2. *Make a sequence of calls to a decryption oracle, submitting any string ψ_1 of the proper length. The oracle will respond with $\mathsf{Dec}_K^{DEM}(\psi_1)$. If the submitted ciphertext is ψ_1^*, the oracle will return $\perp$.*
3. *Output $b^* \in \{0,1\}$.*

The adversary succeeds if $b^* = b$. As above, we define the advantage *of $\mathcal{A}$ against DEM as*

$$Adv_{DEM}(\mathcal{A},\lambda) = \left| Pr[b^* = b] - \frac{1}{2} \right|. \tag{2}$$

We say that a DEM is secure if the advantage Adv_{DEM} of any polynomial-time adversary $\mathcal{A}$ in the above attack model is negligible.

We require that the key K used in $\mathsf{Enc}^{\mathsf{DEM}}$ and $\mathsf{Dec}^{\mathsf{DEM}}$ has the same length ℓ_K as in the KEM. In this case, the mechanisms are said to be *compatible*, and can be composed in the canonical way as shown in Table 3.

Remark 1. An alternative definition of advantage against KEM, or more in general any PKE scheme, is the following:

$$\mathsf{Adv}'_{\mathsf{KEM}}(\mathcal{A},\lambda) = \left| \mathsf{Pr}[b^* = 1 | b = 1] - \mathsf{Pr}[b^* = 1 | b = 0] \right|. \tag{3}$$

The two notions are related in the sense that, for any adversary $\mathcal{A}$, we have $\mathsf{Adv}'_{\mathsf{KEM}}(\mathcal{A},\lambda) = 2 \cdot \mathsf{Adv}_{\mathsf{KEM}}(\mathcal{A},\lambda)$. However, as we will see, the above expression is often more convenient for interpreting the behavior of an adversary in two different attack games, where b is always equal to 0 in one game, and to 1 in the other. This is usually accomplished by replacing a honest encryption with the encryption of a "rubbish" message (commonly a randomly generated string of the proper length), and then analyzing the behavior of the adversary.

Table 3. Hybrid Encryption scheme

K	$\mathsf{K}_{\mathsf{publ}}$ the *public key space.* $\mathsf{K}_{\mathsf{priv}}$ the *private key space.*
P	The set of messages to be encrypted, or *plaintext space.*
C	The set of the messages transmitted over the channel, or *ciphertext space.*
KeyGen	A probabilistic key generation algorithm that takes as input a security parameter 1^λ and outputs a public key $\mathsf{pk} \in \mathsf{K}_{\mathsf{publ}}$ and a private key $\mathsf{sk} \in \mathsf{K}_{\mathsf{priv}}$.
Enc	A probabilistic encryption algorithm that receives as input a public key $\mathsf{pk} \in \mathsf{K}_{\mathsf{publ}}$ and a plaintext $\phi \in \mathsf{P}$. The algorithm invokes $\mathsf{Enc}_{\mathsf{pk}}^{\mathsf{KEM}}(\ )$ and obtains a key/ciphertext pair (K, ψ_0), then runs $\mathsf{Enc}_K^{\mathsf{DEM}}(\phi)$ and gets a ciphertext ψ_1. Finally, it outputs the ciphertext $\psi = (\psi_0 \| \psi_1)$.
Dec	A deterministic decryption algorithm that receives as input a private key $\mathsf{sk} \in \mathsf{K}_{\mathsf{priv}}$ and a ciphertext $\psi \in \mathsf{C}$. The algorithm parses ψ as $(\psi_0 \| \psi_1)$, then decrypts the left part by running $\mathsf{Dec}_{\mathsf{sk}}^{\mathsf{KEM}}(\psi_0)$; it either gets $\perp$ or a key K. In the first case, the algorithm returns $\perp$, otherwise it runs $\mathsf{Dec}_K^{\mathsf{DEM}}(\psi_1)$ and returns either the resulting plaintext ϕ or the failure symbol $\perp$.

It has then been proved that, given a CCA adversary $\mathcal{A}$ for the hybrid scheme (HY), there exist an adversary $\mathcal{A}_1$ for KEM and an adversary $\mathcal{A}_2$ for DEM running in roughly the same time as $\mathcal{A}$, such that for any choice of the security parameter λ we have $\mathsf{Adv}_{\mathsf{HY}}(\mathcal{A}, \lambda) \leq \mathsf{Adv}'_{\mathsf{KEM}}(\mathcal{A}_1, \lambda) + \mathsf{Adv}_{\mathsf{DEM}}(\mathcal{A}_2, \lambda)$. See Cramer and Shoup [6, Th. 5] for a complete proof.

2.3 Other Cryptographic Tools

In this section we introduce other cryptographic tools that we need for our construction. We start with key derivation functions.

Definition 3. *A* Key Derivation Function (KDF) *is a function that takes as input a string $\boldsymbol{x}$ of arbitrary length and an integer $\ell \geq 0$ and outputs a bit string of length ℓ.*

A KDF is modelled as a random oracle, and it satisfies the *entropy smoothing* property, that is, if $\boldsymbol{x}$ is chosen at random from a high entropy distribution, the output of KDF should be computationally indistinguishable from a random length-ℓ bit string.

Intuitively, a good choice for a KDF could be a hash function with a variable (arbitrary) length output, such as the new SHA-3, Keccak [5].

Definition 4. *A* Message Authentication Code (MAC) *is an algorithm that produces a short piece of information* (tag) *used to authenticate a message. A MAC is defined by a function Ev that takes as input a key K of length ℓ_{MAC} and an arbitrary string T and returns a tag to be appended to the message, that is, a string τ of fixed length ℓ_{TAG}.*

Informally, a MAC is similar to a signature scheme, with the difference that the scheme makes use of private keys both for evaluation and verification; in this sense, it could be seen as a "symmetric encryption equivalent" of a signature scheme. The usual desired security requirement is existential unforgeability under chosen message attacks (see Appendix B).

3 The Hybrid Encryption Scheme

3.1 The KEM Construction

The KEM we present here follows closely the Niederreiter framework, and is thus based on the hardness of SDP. Note that, compared to the original Niederreiter scheme, a slight modification is introduced in the decryption process. As we will see later, this is necessary for the proof of security.

If the ciphertext is correctly formed, decoding will always succeed, hence the KEM is perfectly sound. Furthermore, we will see in Section 3.2 that, even if with this formulation $\mathsf{Dec}^{\mathsf{KEM}}$ never fails, there is no integrity loss in the hybrid encryption scheme thanks to the check given by the MAC.

We prove the security of the KEM in the following theorem.

[1] A natural suggestion is for example to set $K = \mathsf{KDF}(\psi_0, \ell_K)$.

Table 4. The Niederreiter KEM

Setup	Fix public system parameters $q, n, k, w \in \mathbb{N}$, then choose a family $\mathcal{F}$ of w-error-correcting $[n, k]$ linear codes over $\mathbb{F}_q$.
KeyGen	Generate at random a code $\mathcal{C} \in \mathcal{F}$ given by its code description Δ and compute its parity-check matrix in systematic form $H = (M\|I_{n-k})$. Publish the public key M and store the private key Δ.
Enc	On input a public key M choose a random $\boldsymbol{e} \in \mathbb{W}_{q,n,w}$, set $H = (M\|I_{n-k})$, then compute $K = \mathsf{KDF}(\boldsymbol{e}, \ell_K)$, $\psi_0 = H\boldsymbol{e}^\mathsf{T}$ and return the key/ciphertext pair (K, ψ_0).
Dec	On input a private key Δ and a ciphertext ψ_0, compute $\mathsf{Decode}_\Delta(\psi_0)$. If the decoding succeeds, use its output $\boldsymbol{e}$ to compute $K = \mathsf{KDF}(\boldsymbol{e}, \ell_K)$. Otherwise, set K to be a string of length ℓ_K determined as a pseudorandom function[1] of ψ_0. Return K.

Theorem 1. *Let $\mathcal{A}$ be an adversary in the random oracle model for the Niederreiter KEM as in Definition 1. Let θ be the running time of $\mathcal{A}$, n_{KDF} and n_{Dec} be two bounds on, respectively, the total number of random oracle queries and the total number of decryption queries performed by $\mathcal{A}$, and set $N = |\mathbb{W}_{q,n,w}|$. Then there exists an adversary $\mathcal{A}'$ for SDP such that $\mathsf{Adv}_{\mathsf{KEM}}(\mathcal{A}, \lambda) \leq \mathsf{Adv}_{\mathsf{SDP}}(\mathcal{A}', \lambda) + n_{\mathsf{Dec}}/N$. The running time of $\mathcal{A}'$ will be approximately equal to θ plus the cost of n_{KDF} matrix-vector multiplications and some table lookups.*

Proof. We replace KDF with a random oracle $\mathcal{H}$ mapping words in $\mathbb{W}_{q,n,w}$ to bit strings of length ℓ_K. To prove our claim, we proceed as follows. Let's call G_0 the original attack game played by $\mathcal{A}$, and S_0 the event that $\mathcal{A}$ succeeds in game G_0. We define a new game G_1 which is identical to G_0 except that the game is halted if the challenge ciphertext $\psi_0^* = H\boldsymbol{e}^{*\mathsf{T}}$ obtained when querying the encryption oracle had been previously submitted to the decryption oracle: we call this event F_1. Since the number of valid ciphertexts is N, we have $\mathsf{Pr}[\mathsf{F}_1] \leq n_{\mathsf{Dec}}/N$. It follows that $\left|\mathsf{Pr}[\mathsf{S}_0] - \mathsf{Pr}[\mathsf{S}_1]\right| \leq n_{\mathsf{Dec}}/N$, where S_1 is the event that $\mathcal{A}$ succeeds in game G_1. Next, we define game G_2 which is identical to G_1 except that we generate the challenge ciphertext ψ_0^* at the beginning of the game, and we halt if $\mathcal{A}$ ever queries $\mathcal{H}$ at $\boldsymbol{e}^*$: we call this event F_2. By construction, since $\mathcal{H}(\boldsymbol{e}^*)$ is undefined, it is not possible to tell whether $K^* = K$, thus we have $\mathsf{Pr}[\mathsf{S}_2] = 1/2$, where S_2 is the event that $\mathcal{A}$ succeeds in game G_2. We obtain that $\left|\mathsf{Pr}[\mathsf{S}_1] - \mathsf{Pr}[\mathsf{S}_2]\right| \leq \mathsf{Pr}[\mathsf{F}_2]$ and we just need to bound $\mathsf{Pr}[\mathsf{F}_2]$.

We now construct an adversary $\mathcal{A}'$ against SDP. $\mathcal{A}'$ interacts with $\mathcal{A}$ and is able to simulate the random oracle and the decryption oracle with the help of two tables T_1 and T_2, initially empty, as described below.

Key Generation: On input the instance $(H, \boldsymbol{s}^*, w)$ of SDP, return the public key $\mathsf{pk} = H$.

Challenge Queries: When $\mathcal{A}$ asks for the challenge ciphertext:

1. Generate a random string K^* of length ℓ_K.
2. Set $\psi_0^* = \boldsymbol{s}^*$.
3. Return the pair (K^*, ψ_0^*).

Random Oracle Queries: Upon $\mathcal{A}$'s random oracle query $\boldsymbol{e} \in \mathbb{W}_{q,n,w}$:

1. Look up $\boldsymbol{e}$ in T_1. If $(\boldsymbol{e}, \boldsymbol{s}, K)$ is in T_1 for some $\boldsymbol{s}$ and K, return K.
2. Compute $\boldsymbol{s} = H\boldsymbol{e}^\intercal$.
3. If $\boldsymbol{s} = \boldsymbol{s}^*$ then $\mathcal{A}'$ outputs $\boldsymbol{e}$ and the game ends.
4. Look up $\boldsymbol{s}$ in T_2. If $(\boldsymbol{s}, K)$ is in T_2 for some K (i.e. the decryption oracle has been evaluated at $\boldsymbol{s}$), return K.
5. Set K to be a random string of length ℓ_K and place the triple $(\boldsymbol{e}, \boldsymbol{s}, K)$ in table T_1.
6. Return K.

Decryption Queries: Upon $\mathcal{A}$'s decryption query $\psi_0 = \boldsymbol{s} \in \mathbb{F}_q^{(n-k)}$:

1. Look up $\boldsymbol{s}$ in T_2. If $(\boldsymbol{s}, K)$ is in T_2 for some K, return K.
2. Look up $\boldsymbol{s}$ in T_1. If $(\boldsymbol{e}, \boldsymbol{s}, K)$ is in T_1 for some $\boldsymbol{e}$ and K (i.e. the random oracle has been evaluated at $\boldsymbol{e}$ such that $\boldsymbol{s} = H\boldsymbol{e}^\intercal$), return K.
3. Generate a random string K of length ℓ_K and place the pair $(\boldsymbol{s}, K)$ in T_2.
4. Return K.

Note that, in both random oracle and decryption queries, we added Step 1 to guarantee the integrity of the simulation, that is, if the same value is queried more than once, the same output is returned.

A fundamental issue is that it is impossible for the simulator to determine if a word is decodable or not. If the decryption algorithm returned $\perp$ if and only if a word was not decodable, then it would be impossible to simulate decryption properly. We have resolved this problem by insisting that the KEM decryption algorithm always outputs a hash value. With this formulation, the simulation is flawless and $\mathcal{A}'$ outputs a solution to the SDP instance with probability equal to $\mathsf{Pr}[\mathsf{F}_2]$. □

3.2 A Standard DEM

A standard way to construct a DEM by means of a SE scheme and a one-time MAC is shown in Table 5.

It is easy to prove that if the underlying components are secure, so is the resulting DEM. In particular it is possible to prove [6, Th. 4] that, for any DEM adversary $\mathcal{A}$, we have $\mathsf{Adv}_{\mathsf{DEM}}(\mathcal{A}, \lambda) \leq \mathsf{Adv}_{\mathsf{FG}}(\mathcal{A}_1, \lambda) + \mathsf{Adv}_{\mathsf{MAC}}(\mathcal{A}_2, \lambda)$, where $\mathcal{A}_1$ and $\mathcal{A}_2$ are, respectively, a find-guess adversary for SE and a one-time existential forgery adversary for MAC, both running in about the same time of $\mathcal{A}$.

Table 5. Standard DEM

Enc	On input a key K and a plaintext ϕ, parse K as $(K_1 \| \| K_2)$ then compute $\psi' = \mathsf{Enc}^{\mathsf{SE}}_{K_1}(\phi)$, set $T = \psi'$ and evaluate $\tau = \mathsf{Ev}(K_2, T)$. Return the ciphertext $\psi_1 = (\psi' \| \| \tau)$.
Dec	On input a key K and a ciphertext ψ_1, parse ψ_1 as $(\psi' \| \| \tau)$ then parse K as $(K_1 \| \| K_2)$, set $T = \psi'$ and apply the MAC algorithm to obtain $\tau' = \mathsf{Ev}(K_2, T)$. If $\tau' \neq \tau$ the verification fails, hence return $\perp$. Otherwise, compute $\phi = \mathsf{Dec}^{\mathsf{SE}}_{K_1}(\psi')$ and return the plaintext ϕ.

3.3 Hybrid Niederreiter

For our purposes, and throughout the rest of this paper, we will think at the DEM as a one-time pad with fixed input/output length m (e.g. 128 or 256 bits), together with a MAC (any of the ISO standards is acceptable). The Hybrid Niederreiter (HN) scheme is simply the composition of the two components, as described in Table 3. Details are presented in Table 6.

Table 6. Hybrid Niederreiter scheme

Setup	Fix public system parameters $q, n, k, w \in \mathbb{N}$, then choose a family $\mathcal{F}$ of w-error-correcting $[n, k]$ linear codes over $\mathbb{F}_q$.
K	$\mathsf{K}_{\mathsf{publ}}$ the set of $(n-k) \times k$ matrices over $\mathbb{F}_q$. $\mathsf{K}_{\mathsf{priv}}$ the set of code descriptions for $\mathcal{F}$.
P	The set of binary strings $\{0,1\}^m$.
C	The set of triples formed by a vector of $\mathbb{F}_q^{(n-k)}$, a bit string of length ℓ, and a tag.
KeyGen	Generate at random a code $\mathcal{C} \in \mathcal{F}$ given by its code description Δ and compute its parity-check matrix in systematic form $H = (M \| I_{n-k})$. Publish the public key $M \in K_{\mathsf{priv}}$ and store the private key $\Delta \in K_{\mathsf{publ}}$.
Enc	On input a public key M and a plaintext $\phi \in \mathsf{P}$, choose a random $\boldsymbol{e} \in \mathbb{W}_{q,n,w}$, set $H = (M \| I_{n-k})$, then compute $K = \mathsf{KDF}(\boldsymbol{e}, m + \ell_{\mathsf{MAC}})$ and $\psi_0 = H\boldsymbol{e}^{\mathsf{T}}$. Parse K as $(K_1 \| \| K_2)$ then compute $\psi' = K_1 \oplus \phi$, set $T = \psi'$ and evaluate $\tau = \mathsf{Ev}(K_2, T)$. Return the ciphertext $\psi = (\psi_0 \| \| \psi' \| \| \tau)$.
Dec	On input a private key Δ and a ciphertext ψ, first parse ψ as $(\psi_0 \| \| \psi_1)$, then compute $\mathsf{Decode}_{\Delta}(\psi_0)$. If the decoding succeeds, use its output $\boldsymbol{e}$ to compute $K = \mathsf{KDF}(\boldsymbol{e}, m + \ell_{\mathsf{MAC}})$. Otherwise, determine K as a pseudorandom function of ψ_0. Parse ψ_1 as $(\psi' \| \| \tau)$ then parse K as $(K_1 \| \| K_2)$, set $T = \psi'$ and apply the MAC algorithm to obtain $\tau' = \mathsf{Ev}(K_2, T)$. If $\tau' \neq \tau$ the verification fails, hence return $\perp$. Otherwise, compute $\phi = K_1 \oplus \psi'$ and return the plaintext ϕ.

4 Anonymity

Anonymity for public-key encryption schemes was first introduced by Bellare et al. in [1] and, as opposed to the classical notions of data-privacy such as indistinguishability, it captures the idea of *key-privacy*. This means that an anonymous PKE scheme does not disclose information about which key, among a set of valid keys, has been used to encrypt. We therefore speak about *indistinguishability of keys*.

Definition 5. *The Indistinguishability of Keys game in the adaptive chosen-ciphertext attack model (IK-CCA) is defined as follows:*

1. *Query a key generation oracle to obtain two public keys* pk_0 *and* pk_1.
2. *Make a sequence of calls to either of the two decryption oracles (corresponding to the two keys), submitting any string* ψ *of the proper length. The oracle will respond respectively with* $\mathsf{Dec}^{PKE}_{sk_0}(\psi)$ *or* $\mathsf{Dec}^{PKE}_{sk_1}(\psi)$.
3. *Choose a plaintext* ϕ^* *and submit it to an encryption oracle. The oracle chooses a random* $b \in \{0,1\}$ *and replies with the "challenge" ciphertext* $\psi^* = \mathsf{Enc}^{PKE}_{pk_b}(\phi^*)$.
4. *Keep performing decryption queries. If the submitted ciphertext (to any of the two decryption oracles) is* ψ^**, return* $\perp$.
5. *Output* $b^* \in \{0,1\}$.

The adversary succeeds if $b^* = b$. *More precisely, we define the* advantage *of* $\mathcal{A}$ *against PKE as*

$$\mathsf{Adv}_{\mathsf{IK\text{-}CCA}}(\mathcal{A}, \lambda) = \left| Pr[b^* = b] - \frac{1}{2} \right|. \tag{4}$$

We say that PKE is secure in this sense if the advantage $\mathsf{Adv}_{\mathsf{IK\text{-}CCA}}$ *of any adversary* $\mathcal{A}$ *in the above CCA attack model is negligible.*

If the attack model does not allow for decryption queries, the security notion is known as IK-CPA.

The above notions for PKE schemes apply also to hybrid encryption schemes. Unlike the case of data-privacy, though, it is not enough to have two anonymous components for the resulting hybrid encryption scheme to be anonymous: a counterexample is given by Mohassel in [8]. The author, however, shows how this can be fixed by using a KEM component that satisfies an additional property called *robustness*. In practice, this requires that a ciphertext does not decrypt to a valid plaintext under distinct private keys.

Now, it is easy to see that plain coding theory schemes are not anonymous: this is immediate for Niederreiter (being deterministic), and it was shown in [15] for McEliece. In the same paper, the authors prove that the randomized version of McEliece by Nojima et al. [10] is IK-CPA secure. We use a similar technique to prove that the Hybrid Niederreiter scheme described in Table 6 is IK-CCA secure.

Theorem 2. *Let $\mathcal{A}$ be an IK-CCA adversary in the random oracle model for the Hybrid Niederreiter scheme. Let θ be the running time of $\mathcal{A}$, n_{KDF} and n_{Dec} be two bounds on, respectively, the total number of random oracle queries and the total number of decryption queries performed by $\mathcal{A}$, and set $N = |\mathbb{W}_{q,n,w}|$. Then there exists an IND-CCA adversary $\mathcal{A}'$ against HN such that $\mathsf{Adv}_{\mathsf{IK\text{-}CCA}}(\mathcal{A}, \lambda) \leq \mathsf{Adv}'_{\mathsf{HN}}(\mathcal{A}', \lambda) + n_{Dec}/2N$. The running time of $\mathcal{A}'$ will be approximately equal to θ plus n_{KDF} matrix-vector multiplications and at most n_{Dec} decryption operations.*

Proof. As in the proof of Theorem 1, we replace KDF with a random oracle $\mathcal{H}$ mapping words in $\mathbb{W}_{q,n,w}$ to bit strings of length ℓ_K. Let's call G_0 the original attack game played by $\mathcal{A}$, and S_0 the event that $\mathcal{A}$ succeeds in game G_0. We define a new game G_1 which is identical to G_0 except that the game is halted if the challenge ciphertext $\psi^* = \mathsf{Enc}^{\mathsf{HN}}_{\mathsf{pk}_b}(\phi^*)$ obtained when querying the encryption oracle had been previously submitted to the decryption oracle: we call this event F_1. Since, for any fixed plaintext, the number of valid ciphertexts is $2N$, we have $\mathsf{Pr}[\mathsf{F}_1] \leq n_{\mathsf{Dec}}/2N$. It follows that $\left|\mathsf{Pr}[\mathsf{S}_1] - \mathsf{Pr}[\mathsf{S}_0]\right| \leq n_{\mathsf{Dec}}/2N$, where S_1 is the event that $\mathcal{A}$ succeeds in game G_1. Next, we define game G_2 which is the same as G_1 apart from the following modification: when $\mathcal{A}$ queries the encryption oracle on an input ϕ^*, a random string $\phi' \in \mathsf{P}$ is generated and returned together with the challenge ψ^*. Since this carries no additional information, we have $\mathsf{Pr}[\mathsf{S}_2] = \mathsf{Pr}[\mathsf{S}_1]$, where S_2 is the event that $\mathcal{A}$ succeeds in game G_2. Finally, we define game G_3 by modifying the encryption oracle such that it replies instead with $\psi^* = \mathsf{Enc}^{\mathsf{HN}}_{\mathsf{pk}_b}(\phi')$. Now, note that the success probability of $\mathcal{A}$ does not change unless it is able to distinguish which of the two plaintexts had been used by the encryption oracle. More precisely, there exists an IND-CCA adversary $\mathcal{A}'$ against HN that uses $\mathcal{A}$ as a subroutine. Since it plays the adaptive chosen-ciphertext game, $\mathcal{A}'$ has access to a decryption oracle $\mathcal{D}$ for its public key pk. The interaction with $\mathcal{A}$ is described below:

Key Generation: On input the public key $\mathsf{pk} = H$, generate a pair $(\mathsf{pk}', \mathsf{sk}') \in \mathsf{K}$ then set $\mathsf{pk}_0 = \mathsf{pk}$ and $\mathsf{pk}_1 = \mathsf{pk}'$ and send $(\mathsf{pk}_0, \mathsf{pk}_1)$ to $\mathcal{A}$.

Challenge Queries: When $\mathcal{A}$ asks for the challenge ciphertext:

1. Receive as input the string ϕ^* from $\mathcal{A}$.
2. Generate a random string $\phi' \in \mathsf{P}$.
3. Submit ϕ^* and ϕ' to the IND-CCA encryption oracle. The oracle will reply with $\psi^* = (\psi_0^* || \psi_1^*) = \mathsf{Enc}^{\mathsf{HN}}_{\mathsf{pk}_0}(\phi_\beta)$, where ϕ_β is equal to ϕ^* if $\beta = 1$ or to ϕ' otherwise. This is the challenge ciphertext for $\mathcal{A}'$.
4. Return ϕ' and ψ^* to $\mathcal{A}$.

Random Oracle Queries: Upon $\mathcal{A}$'s random oracle query $\boldsymbol{e} \in \mathbb{W}_{q,n,w}$:

1. Submit $\boldsymbol{e}$ to $\mathcal{H}$ and get $K = \mathcal{H}(\boldsymbol{e})$.
2. If $H\boldsymbol{e}^\intercal = \psi_0^*$ then $\mathcal{A}'$ decrypts ψ_1^* with K and the game ends.
3. Return $\mathcal{H}(\boldsymbol{e})$.

Decryption Queries: Upon $\mathcal{A}$'s decryption query ψ for sk_b:

1. If $b = 1$, return $\mathsf{Dec}^{\mathsf{HN}}_{\mathsf{sk}'}(\psi)$.
2. If $b = 0$, submit ψ to $\mathcal{D}$ and return $\mathcal{D}(\psi)$.

Observe that, in the attack game that $\mathcal{A}'$ is playing against HN, the value of β is equal to 1 in game G_2, and to 0 in game G_3. Following up from Remark 1, we conclude that $\left|\mathsf{Pr}[\mathsf{S}_3] - \mathsf{Pr}[\mathsf{S}_2]\right| \leq \mathsf{Adv}'_{\mathsf{HN}}(\mathcal{A}', \lambda)$.
Now, since ϕ' is chosen uniformly at random and Assumption 1 holds, the distributions $\{(\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{Enc}^{\mathsf{HN}}_{\mathsf{pk}_0}(\phi'))|(\mathsf{pk}_0, \mathsf{sk}_0) \xleftarrow{\$} \mathsf{K}, (\mathsf{pk}_1, \mathsf{sk}_1) \xleftarrow{\$} \mathsf{K}\}$ and $\{(\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{Enc}^{\mathsf{HN}}_{\mathsf{pk}_1}(\phi'))|(\mathsf{pk}_0, \mathsf{sk}_0) \xleftarrow{\$} \mathsf{K}, (\mathsf{pk}_1, \mathsf{sk}_1) \xleftarrow{\$} \mathsf{K}\}$ are computationally indistinguishable. It follows that $\mathsf{Pr}[\mathsf{S}_3] = 1/2$; hence, $\mathsf{Adv}_{\mathsf{IK\text{-}CCA}}(\mathcal{A}, \lambda) = = \left|\mathsf{Pr}[S_0] - 1/2\right| = \left|\mathsf{Pr}[S_0] - \mathsf{Pr}[S_1]\right| + \left|\mathsf{Pr}[S_1] - \mathsf{Pr}[S_2]\right| + \left|\mathsf{Pr}[S_2] - \mathsf{Pr}[S_3]\right| \leq \mathsf{Adv}'_{\mathsf{HN}}(\mathcal{A}', \lambda) + n_{\mathsf{Dec}}/2N$ as claimed. □

5 Conclusions

In this paper, we have introduced a key encapsulation method based on the Niederreiter cryptosystem. This is the first KEM based directly on a coding theory problem and it enjoys a simple construction and a tight security proof. We have also shown that the Hybrid Niederreiter encryption scheme that makes use of our KEM satisfies the most important notion of anonymity, IK-CCA. Our work builds on the results of [15], and is the first code-based encryption scheme to enjoy IK-CCA security. Future work includes investigating practical applications of our construction, with the aim of an implementation.

References

1. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001)
2. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A Concrete Security Treatment of Symmetric Encryption. In: FOCS, pp. 394–403. IEEE Computer Society (1997)
3. Berlekamp, E., McEliece, R., van Tilborg, H.: On the inherent intractability of certain coding problems. IEEE Transactions on Information Theory 24(3), 384–386 (1978)
4. Bernstein, D.J.: Personal communication (May 2012)
5. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: `http://keccak.noekeon.org/`
6. Cramer, R., Shoup, V.: Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. SIAM J. Comput. 33(1), 167–226 (2004)
7. McEliece, R.: A Public-Key Cryptosystem Based on Algebraic Coding Theory. Technical report, NASA (1978)

8. Mohassel, P.: A Closer Look at Anonymity and Robustness in Encryption Schemes. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 501–518. Springer, Heidelberg (2010)
9. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Problems of Control and Information Theory 15(2), 159–166 (1986)
10. Nojima, R., Imai, H., Kobara, K., Morozov, K.: Semantic security for the McEliece cryptosystem without random oracles. Des. Codes Cryptography 49(1-3), 289–305 (2008)
11. Patterson, N.: The algebraic decoding of Goppa codes. IEEE Transactions on Information Theory 21(2), 203–207 (1975)
12. Shoup, V.: A proposal for an ISO standard for public key encryption (version 2.1). IACR Cryptology ePrint Archive, 112 (2001)
13. Strenzke, F.: A Timing Attack against the Secret Permutation in the McEliece PKC. In: Sendrier, N. (ed.) PQCrypto 2010. LNCS, vol. 6061, pp. 95–107. Springer, Heidelberg (2010)
14. Strenzke, F., Tews, E., Molter, H.G., Overbeck, R., Shoufan, A.: Side Channels in the McEliece PKC. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 216–229. Springer, Heidelberg (2008)
15. Yamakawa, S., Cui, Y., Kobara, K., Hagiwara, M., Imai, H.: On the key-privacy issue of McEliece public-key encryption. In: Boztaş, S., Lu, H.-F(F.) (eds.) AAECC 2007. LNCS, vol. 4851, pp. 168–177. Springer, Heidelberg (2007)

A The Niederreiter Cryptosystem

The scheme we present below is a generalization of the Niederreiter scheme in two senses: first, we extend it to any family of codes with an efficient decoding algorithm; second, we avoid using the outdated method of scrambling matrices S and P (see [9] for details). While the former allows for a wider and safer choice of codes, the latter provides a simpler formulation and resistance to side-channel attacks (Strenzke et al. [13,14]).

In what follows, we consider only families of codes to which is possible to associate an efficient decoding algorithm; we denote this with Decode_{Δ}, where Δ is a description of the selected code that depends on $\mathcal{F}$. For example, in case $\mathcal{F}$ is the family of binary Goppa codes, the associated algorithm is Patterson's algorithm [11] and Δ is given by a Goppa polynomial $g(x)$ and its support $(\alpha_1, \ldots, \alpha_n)$.

B Standard Security Definitions

Definition 6 (IND). *An adversary $\mathcal{A}$ for the indistinguishability (IND) property is a two-stage polynomial-time algorithm. In the first stage, $\mathcal{A}$ takes as input a public key $\mathsf{pk} \in \mathsf{K}_{\mathsf{publ}}$, then outputs two arbitrary plaintexts ϕ_0, ϕ_1. In the second stage, it receives a ciphertext $\psi^* = \mathsf{Enc}_{\mathsf{pk}}(\phi_b)$, for $b \in \{0, 1\}$, and returns a bit b^*. The adversary succeeds if $b^* = b$. More precisely, we define the* advantage *of $\mathcal{A}$ against PKE as*

$$\mathsf{Adv}(\mathcal{A}, \lambda) = \left| \mathsf{Pr}[b^* = b] - \frac{1}{2} \right|. \tag{5}$$

Table 7. The Niederreiter cryptosystem

Setup	Fix public system parameters $q, n, k, w \in \mathbb{N}$, then choose a family $\mathcal{F}$ of w-error-correcting $[n, k]$ linear codes over $\mathbb{F}_q$.
K	$\mathsf{K}_{\mathsf{publ}}$ the set of $(n-k) \times k$ matrices over $\mathbb{F}_q$. $\mathsf{K}_{\mathsf{priv}}$ the set of code descriptions for $\mathcal{F}$.
P	The set $\mathbb{W}_{q,n,w}$ of words of $\mathbb{F}_q^n$ with Hamming weight w.
C	The vector space $\mathbb{F}_q^{(n-k)}$.
KeyGen	Generate at random a code $\mathcal{C} \in \mathcal{F}$ given by its code description Δ and compute its parity-check matrix in systematic form $H = (M\|I_{n-k})$. Publish the public key $M \in \mathsf{K}_{\mathsf{publ}}$ and store the private key $\Delta \in \mathsf{K}_{\mathsf{priv}}$.
Enc	On input a public key $M \in \mathsf{K}_{\mathsf{publ}}$ and a plaintext $\phi = \boldsymbol{e} \in \mathsf{P}$, set $H = (M\|I_{n-k})$, then compute the syndrome $\boldsymbol{s} = H\boldsymbol{e}^{\mathsf{T}}$ and return the ciphertext $\psi = \boldsymbol{s} \in \mathsf{C}$.
Dec	On input the private key $\Delta \in \mathsf{K}_{\mathsf{priv}}$ and a ciphertext $\psi \in \mathsf{C}$, compute $\mathsf{Decode}_{\Delta}(\psi)$. If the decoding succeeds, return its output $\phi = \boldsymbol{e}$. Otherwise, output $\perp$.

We say that a PKE scheme enjoys Indistinguishability *if the advantage of any adversary $\mathcal{A}$ over all choices of pk, ψ^* and the randomness used by $\mathcal{A}$ is negligible in the security parameter.*

Definition 7 (IND-CCA). *The attack game for IND-CCA proceeds as follows:*

1. *Query a key generation oracle to obtain a public key pk.*
2. *Make a sequence of calls to a decryption oracle, submitting any string ψ of the proper length (not necessarily an element of C). The oracle will respond with $\mathsf{Dec}_{\mathsf{sk}}(\psi)$.*
3. *Choose $\phi_0, \phi_1 \in \mathsf{P}$ and submit them to an encryption oracle. The oracle will choose a random $b \in \{0, 1\}$ and reply with the "challenge" ciphertext $\psi^* = \mathsf{Enc}_{\mathsf{pk}}(\phi_b)$.*
4. *Keep performing decryption queries. If the submitted ciphertext is $\psi = \psi^*$, the oracle will return $\perp$.*
5. *Output $b^* \in \{0, 1\}$.*

We say that a PKE scheme has Indistinguishability against Adaptive Chosen Ciphertext Attacks (IND-CCA) *if the advantage $\mathsf{Adv}_{\mathsf{CCA}}$ of any IND adversary $\mathcal{A}$ in the CCA attack model is negligible.*

A similar but weaker notion is *Indistinguishability against Chosen Plaintext Attacks (IND-CPA)*. The game proceeds at above, but no decryption queries are allowed.

The equivalent scenario for symmetric schemes is a model called *find-guess* (Bellare et al., [2]). The definition is similar to IND, except that in this case some extra information is needed before producing the response bit. This replaces

the role of the randomness in the adversary since we are now operating with symmetric encryption. The names "find" and "guess" refer to the two stages of the algorithm.

Definition 8 (FG). *An adversary $\mathcal{A}$ for the find-guess (FG) property is a two-stage polynomial-time algorithm. In the first stage (find), $\mathcal{A}$ takes as input a key $\kappa \in \mathsf{K}$, then outputs two arbitrary plaintexts ϕ_0, ϕ_1 along with some extra information ι to be used later. In the second stage (guess), it receives a ciphertext $\psi^* = \mathsf{Enc}_\kappa(\phi_b)$ for $b \in \{0,1\}$, and returns a bit $b^* = \mathcal{A}(\kappa, \psi^*, \iota)$. The adversary succeeds if $b^* = b$. More precisely, we define the* advantage *of $\mathcal{A}$ against SE as*

$$\mathsf{Adv}(\mathcal{A}, \lambda) = \left| \mathsf{Pr}[b^* = b] - \frac{1}{2} \right|. \tag{6}$$

We say that a SE enjoys Find-Guess security *if the probability of success of any adversary $\mathcal{A}$ over all choices of pk, ψ^* and ι is negligible in the security parameter.*

Finally, the following is the most desirable security properties for signature schemes. It challenges an adversary, equipped with a signing oracle, to reproduce at least one valid message/signature pair.

Definition 9 (EUF-CMA). *We define an adversary $\mathcal{A}$ as a polynomial-time algorithm that acts as follows:*

1. *Query a key generation oracle to obtain a verification key vk.*
2. *Make a sequence of calls to a signing oracle, submitting any message $\mu \in \mathsf{M}$. The oracle will reply with $\sigma = \mathsf{Sign}_{\mathsf{sgk}}(\mu)$.*
3. *Output a pair (μ^*, σ^*).*

The adversary succeeds if $\mathsf{Ver}_{\mathsf{vk}}(\mu^*, \sigma^*) = 1$ and $(\mu^*, \sigma^*) \neq (\mu, \sigma)$ for any pair (μ, σ) previously obtained by querying the signing oracle. We say that a signature scheme is Existentially Unforgeable against Chosen Message Attacks (EUF-CMA) *if the probability of success of any adversary $\mathcal{A}$ is negligible in the security parameter.*

Fast Verification for Improved Versions of the UOV and Rainbow Signature Schemes

Albrecht Petzoldt[1], Stanislav Bulygin, and Johannes Buchmann[1,2]

[1] Technische Universität Darmstadt, Department of Computer Science
Hochschulstraße 10, 64289 Darmstadt, Germany
{apetzoldt,buchmann}@cdc.informatik.tu-darmstadt.de

[2] Center for Advanced Security Research Darmstadt - CASED
Mornewegstraße 32, 64293 Darmstadt, Germany
johannes.buchmann@cased.de

Abstract. Multivariate cryptography is one of the main candidates to guarantee the security of communication in the post-quantum era. While multivariate signature schemes are fast and require only modest computational resources, the key sizes of such schemes are quite large. In [14] Petzoldt et al. proposed a way to reduce the public key size of certain multivariate signature schemes like UOV and Rainbow by a large factor. In this paper we show that by using this idea it is possible to speed up the verification process of these schemes, too. For example, we are able to speed up the verification process of UOV by a factor of 5.

Keywords: Multivariate Cryptography, UOV Signature Scheme, Rainbow Signature Scheme, Key Size Reduction, Fast Verification.

1 Introduction

When quantum computers arrive, classical public-key cryptosystems like RSA and ECC will be broken [1]. The reason for this is Shor's algorithm [18] which solves number theoretic problems like integer factorization and discrete logarithms in polynomial time on a quantum computer. So, to guarantee the security of communication in the post-quantum era, we need alternatives to those classical schemes. Besides lattice-, code-, and hash-based cryptosystems multivariate cryptography seems to be a candidate for this.
Additionally to its (believed) resistance against quantum computer attacks, multivariate cryptosystems are very fast, especially for signatures [2,3]. Furthermore they require only modest computational resources, which makes them appropriate for the use on low-cost devices like smartcards and RFID chips. However, multivariate schemes are not widely used yet, mainly because of the large size of their public and private keys.

In [14], [16] and [17] Petzoldt et al. showed different possibilities to decrease the public key size of the Unbalanced Oil and Vinegar (UOV) and Rainbow signature schemes. The key idea is it to insert a highly structured matrix into the coefficient matrix of the public key. Therefore, the coefficient matrix of the

P. Gaborit (Ed.): PQCrypto 2013, LNCS 7932, pp. 188–202, 2013.

public key has the form $M_P = (B|C)$, where B is a matrix of a very special form (e.g. partially circulant or generated by an LFSR) and C is a matrix without visible structure. By doing so, they were able to decrease the public key size of UOV by 86 %, namely from 99.9 kB to 13.4 kB.

In this paper we show that this idea can not only be used to decrease the size of the public key, but also to speed up the verification process. We use the rich structure of the matrix B to reduce the number of field multiplications needed during the verification process by a large factor (for cyclicUOV this factor is about 80 %). We derive our results both theoretically and show them using a C implementation of the schemes.

The structure of this paper is as follows: In Section 2 we give a short overview on multivariate signature schemes and describe the UOV and Rainbow signature schemes. Section 3 reviews the approach of [14] and [16] to create UOV and Rainbow schemes with structured public keys. In Section 4 we demonstrate how we can use this special structure to speed up the verification process of the schemes. In Subsection 4.2 we look hereby on partially cyclic UOV schemes, whereas Subsection 4.3 deals with cyclic versions of Rainbow. Section 5 presents the results of our experiments and Section 6 concludes the paper.

2 Multivariate Public Key Cryptography

The basic idea behind multivariate cryptography is to choose a system $\mathcal{F}$ of m quadratic polynomials in n variables which can be easily inverted (central map). After that one chooses two affine invertible maps $\mathcal{S}$ and $\mathcal{T}$ to hide the structure of the central map. The public key of the cryptosystem is the composed quadratic map $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ which is difficult to invert. The private key consists of $\mathcal{S}$, $\mathcal{F}$ and $\mathcal{T}$ and therefore allows to invert $\mathcal{P}$.

Due to this construction, the security of multivariate cryptography is based on two mathematical problems:

Problem MQ. Solve the system $p^{(1)} = \cdots = p^{(m)} = 0$, where each $p^{(i)}$ is a quadratic polynomial in the n variables $x_1, \ldots, x_n$ with coefficients and variables in $GF(q)$.

The MQ-problem is proven to be NP-hard even for quadratic polynomials over $GF(2)$ [8].

Problem EIP (Extended Isomorphism of Polynomials)**.** Given a class of central maps $\mathcal{C}$ and a map $\mathcal{P}$ expressible as $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$, where $\mathcal{S}$ and $\mathcal{T}$ are affine maps and $\mathcal{F} \in \mathcal{C}$, find a decomposition of $\mathcal{P}$ of the form $\mathcal{P} = \mathcal{S}' \circ \mathcal{F}' \circ \mathcal{T}'$, with affine maps $\mathcal{S}'$ and $\mathcal{T}'$ and $\mathcal{F}' \in \mathcal{C}$.

In this paper we concentrate on the case of multivariate signature schemes. The standard process for signature generation and verification works as shown in Figure 1.

Signature Generation. To sign a document d, we use a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{F}^m$ to compute the value $\mathbf{h} = \mathcal{H}(d) \in \mathbb{F}^m$. Then we compute $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{h})$,

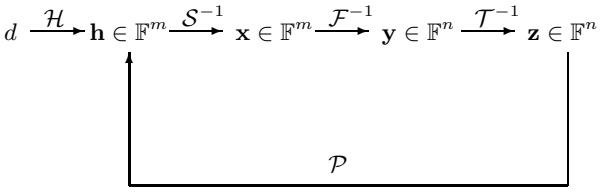


Fig. 1. Signature generation and verification

$\mathbf{y} = \mathcal{F}^{-1}(\mathbf{x})$ and $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$. The signature of the document is $\mathbf{z} \in \mathbb{F}^n$. Here, $\mathcal{F}^{-1}(\mathbf{x})$ means finding one (of the possibly many) pre-image of $\mathbf{x}$ under the central map $\mathcal{F}$.

Verification. To verify the authenticity of a document, one simply computes $\mathbf{h}' = \mathcal{P}(\mathbf{z})$ and the hash value $\mathbf{h} = \mathcal{H}(d)$ of the document. If $\mathbf{h}' = \mathbf{h}$ holds, the signature is accepted, otherwise rejected.

There are several ways to build the central map $\mathcal{F}$ of multivariate schemes. In this paper we concentrate on the so called SingleField constructions. In contrast to BigField schemes like Matsumoto-Imai [11] and MiddleField schemes like ℓiC [6], here all the computations are done in one (relatively small) field. In the following two subsections we describe two well known examples of these schemes in detail.

2.1 The Unbalanced Oil and Vinegar (UOV) Signature Scheme

One way to create an easily invertible multivariate quadratic system is the principle of Oil and Vinegar, which was first proposed by J. Patarin in [13].
Let $\mathbb{F}$ be a finite field. Let o and v be two integers and set $n = o + v$. We set $V = \{1, \ldots, v\}$ and $O = \{v+1, \ldots, n\}$. We call $x_1, \ldots, x_v$ the Vinegar variables and $x_{v+1}, \ldots, x_n$ Oil variables. We define o quadratic polynomials $f^{(k)}(\mathbf{x}) = f^{(k)}(x_1, \ldots, x_n)$ by

$$f^{(k)}(\mathbf{x}) = \sum_{i \in V,\ j \in O} \alpha_{ij}^{(k)} x_i x_j + \sum_{i,j \in V,\ i \leq j} \beta_{ij}^{(k)} x_i x_j + \sum_{i \in V \cup O} \gamma_i^{(k)} x_i + \eta^{(k)} \ (1 \leq k \leq o). \tag{1}$$

Note that Oil and Vinegar variables are not fully mixed, just like oil and vinegar in a salad dressing.

The map $\mathcal{F} = (f^{(1)}(\mathbf{x}), \ldots, f^{(o)}(\mathbf{x}))$ can be easily inverted. First, we choose the values of the v Vinegar variables $x_1, \ldots, x_v$ at random. Therefore we get a system of o linear equations in the o variables $x_{v+1}, \ldots, x_n$ which can be solved e.g. by Gaussian Elimination. If the system does not have a solution, one has to choose other values of $x_1, \ldots, x_v$ and try again.

The public key of the scheme is given as $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$, where $\mathcal{T}$ is an affine map from $\mathbb{F}^n$ to itself. The private key consists of the two maps $\mathcal{F}$ and $\mathcal{T}$ and therefore allows to invert the public key.

Remark. In opposite to other multivariate schemes the second affine map $\mathcal{S}$ is not needed for the security of UOV. So it can be omitted.

In his original paper [13] Patarin suggested to choose $o = v$ (Balanced Oil and Vinegar (OV)). After this scheme was broken by Kipnis and Shamir in [10], it was recommended in [9] to choose $v > o$ (Unbalanced Oil and Vinegar (UOV)). The UOV signature scheme over GF(256) is commonly believed to be secure for $o \geq 28$ equations [19] and $v = 2 \cdot o$ Vinegar variables. For UOV schemes over GF(31) we set $(o, v) = (33, 66)$.

2.2 The Rainbow Signature Scheme

In [4] J. Ding and D. Schmidt proposed a signature scheme called Rainbow, which is based on the idea of (Unbalanced) Oil and Vinegar [9].

Let $\mathbb{F}$ be a finite field and V be the set $\{1, \ldots, n\}$. Let $v_1, \ldots, v_{u+1}, u \geq 1$ be integers such that $0 < v_1 < v_2 < \cdots < v_u < v_{u+1} = n$ and define the sets of integers $V_i = \{1, \ldots, v_i\}$ for $i = 1, \ldots, u$. We set $o_i = v_{i+1} - v_i$ and $O_i = \{v_i + 1, \ldots, v_{i+1}\}$ $(i = 1, \ldots, u)$. The number of elements in V_i is v_i and we have $|O_i| = o_i$. For $k = v_1 + 1, \ldots, n$ we define multivariate quadratic polynomials in the n variables $x_1, \ldots, x_n$ by

$$f^{(k)}(\mathbf{x}) = \sum_{i \in O_l,\ j \in V_l} \alpha_{ij}^{(k)} x_i x_j + \sum_{i,j \in V_l,\ i \leq j} \beta_{ij}^{(k)} x_i x_j + \sum_{i \in V_l \cup O_l} \gamma_i^{(k)} x_i + \eta^{(k)}, \quad (2)$$

where l is the only integer such that $k \in O_l$. Note that these are Oil and Vinegar polynomials with $x_i,\ i \in V_l$ being the Vinegar variables and $x_j,\ j \in O_l$ being the Oil variables.

The map $\mathcal{F}(\mathbf{x}) = (f^{(v_1+1)}(\mathbf{x}), \ldots, f^{(n)}(\mathbf{x}))$ can be inverted as follows. First, we choose $x_1, \ldots, x_{v_1}$ at random. Hence we get a system of o_1 linear equations (given by the polynomials $f^{(k)}$ $(k \in O_1)$) in the o_1 unknowns $x_{v_1+1}, \ldots, x_{v_2}$, which can be solved by Gaussian Elimination. The so computed values of x_i $(i \in O_1)$ are plugged into the polynomials $f^{(k)}(\mathbf{x})$ $(k > v_2)$ and a system of o_2 linear equations (given by the polynomials $f^{(k)}$ $(k \in O_2)$) in the o_2 unknowns x_i $(i \in O_2)$ is obtained. By repeating this process we can get values for all the variables x_i $(i = 1, \ldots, n)$ [1].

The public key of the scheme is given as $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ with two invertible affine maps $\mathcal{S} : \mathbb{F}^m \to \mathbb{F}^m$ and $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$. The private key consists of $\mathcal{S}$, $\mathcal{F}$ and $\mathcal{T}$ and therefore allows to invert te public key.

In the following, we restrict ourselves to Rainbow schemes with two layers (i.e. $u = 2$). For this, $\mathbb{F} = GF(256)$, $(v_1, o_1, o_2) = (17, 13, 13)$ provides 80-bit security under known attacks [15]. For Rainbow schemes over GF(31), we choose $(v_1, o_1, o_2) = (14, 19, 14)$.

[1] It may happen, that one of the linear systems does not have a solution. If so, one has to choose other values of $x_1, \ldots x_{v_1}$ and try again.

3 Improved Versions of UOV and Rainbow

In [14] and [16] Petzoldt et al. presented an approach to create UOV- and Rainbow-based schemes with structured public keys, by which they could reduce the public key size of these schemes by up to 83 %. Due to lack of space we give here only a very brief description and refer to [14] and [16] for the details.

The main idea of the approach is to insert a structured matrix B into the Macauley matrix M_P of the public key. In our case the matrix B is chosen partially circulant, i.e. its rows are given by

$$B[i] = \mathcal{R}^{i-1}(\mathbf{b}) \ (i = 1, \ldots, m), \tag{3}$$

where $\mathbf{b}$ is a randomly chosen vector and $\mathcal{R}^i$ denotes the cyclic right shift by i positions.

To insert this matrix B into M_P, the authors used the relation $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$ between a UOV public and private key, which translates into the matrix equation

$$M_P = M_F \cdot A \tag{4}$$

with a transformation matrix A whose elements are given as quadratic functions in the coefficients of the affine map $\mathcal{T}$. If this matrix is invertible, one can compute the matrix M_F in such a way that M_P has the form $M_P = (B|C)$ with a partially circulant matrix B and a matrix C without visible structure. Figure 2 shows this key generation process graphical form.

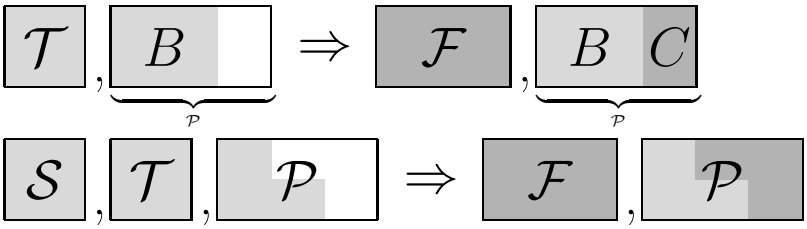


Fig. 2. Alternative key generation for UOV (above) and Rainbow. The light gray parts are chosen by the user, the dark gray parts are computed during the key generation process.

4 The Verification Process

The central part of the verification process for multivariate signature schemes is the evaluation of the public polynomials. Normally this is done as follows: For a given (valid or invalid) signature $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{F}^n$ one first computes an $\frac{(n+1)\cdot(n+2)}{2}$ vector mon, which contains the values of all monomials of degree ≤ 2, i.e.

$$\text{mon} = (z_1^2, z_1 z_2, \ldots, z_n^2, z_1, \ldots, z_n, 1). \tag{5}$$

Then we have

$$\mathcal{P}(\mathbf{z}) = \begin{pmatrix} M_P[1] \cdot \mathrm{mon}^T \\ \vdots \\ M_P[m] \cdot \mathrm{mon}^T \end{pmatrix}, \tag{6}$$

with $M_P[i]$ being the i-th row of the Macauley matrix M_P and $\cdot$ being the standard scalar product.

For schemes with partially cyclic public key, the following strategy seems to be more promising:

4.1 Notations

Let $\mathbf{h} = (h_1, \ldots, h_m)$ be the hash value of the signed message.

The public polynomials can be written as

$$p^{(k)}(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=i}^{n} p_{ij}^{(k)} \cdot x_i x_j + \sum_{i=1}^{n} p_i^{(k)} \cdot x_i + p_0^{(k)} \quad (k = 1, \ldots, m). \tag{7}$$

For $k = 1, \ldots, m$ we define upper triangular matrices $MP^{(k)}$ by

$$MP^{(k)} = \begin{pmatrix} p_{11}^{(k)} & p_{12}^{(k)} & p_{13}^{(k)} & \cdots & p_{1n}^{(k)} & p_1^{(k)} \\ 0 & p_{22}^{(k)} & p_{23}^{(k)} & \cdots & p_{2n}^{(k)} & p_2^{(k)} \\ 0 & 0 & p_{33}^{(k)} & & p_{3n}^{(k)} & p_3^{(k)} \\ \vdots & & \ddots & & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & p_{nn}^{(k)} & p_n^{(k)} \\ 0 & 0 & \ldots & 0 & 0 & p_0^{(k)} \end{pmatrix}. \tag{8}$$

For a (valid or invalid) signature $\mathbf{z} = (z_1, \ldots, z_n)$ of the message we define the extended signature vector

$$\mathrm{sign} = (z_1, \ldots, z_n, 1). \tag{9}$$

With this notation we can write the verification process in the following form

$$\text{accept the signature } \mathbf{z} \iff \mathrm{sign} \cdot MP^{(k)} \cdot \mathrm{sign}^T = h_k \ \forall k \in \{1, \ldots, m\}. \tag{10}$$

In the following two subsections we consider the question how we can evaluate this equation more efficiently for improved versions of UOV and Rainbow.

4.2 cyclicUOV

In the case of cyclicUOV [14], the matrices $MP^{(k)}$ are of the form shown in Figure 3. We have

$$MP_{ij}^{(k)} = MP_{i,j-1}^{(k-1)} \ \forall i = 1, \ldots, v, \ j = i+1, \ldots, n, \ k = 2, \ldots, o. \tag{11}$$

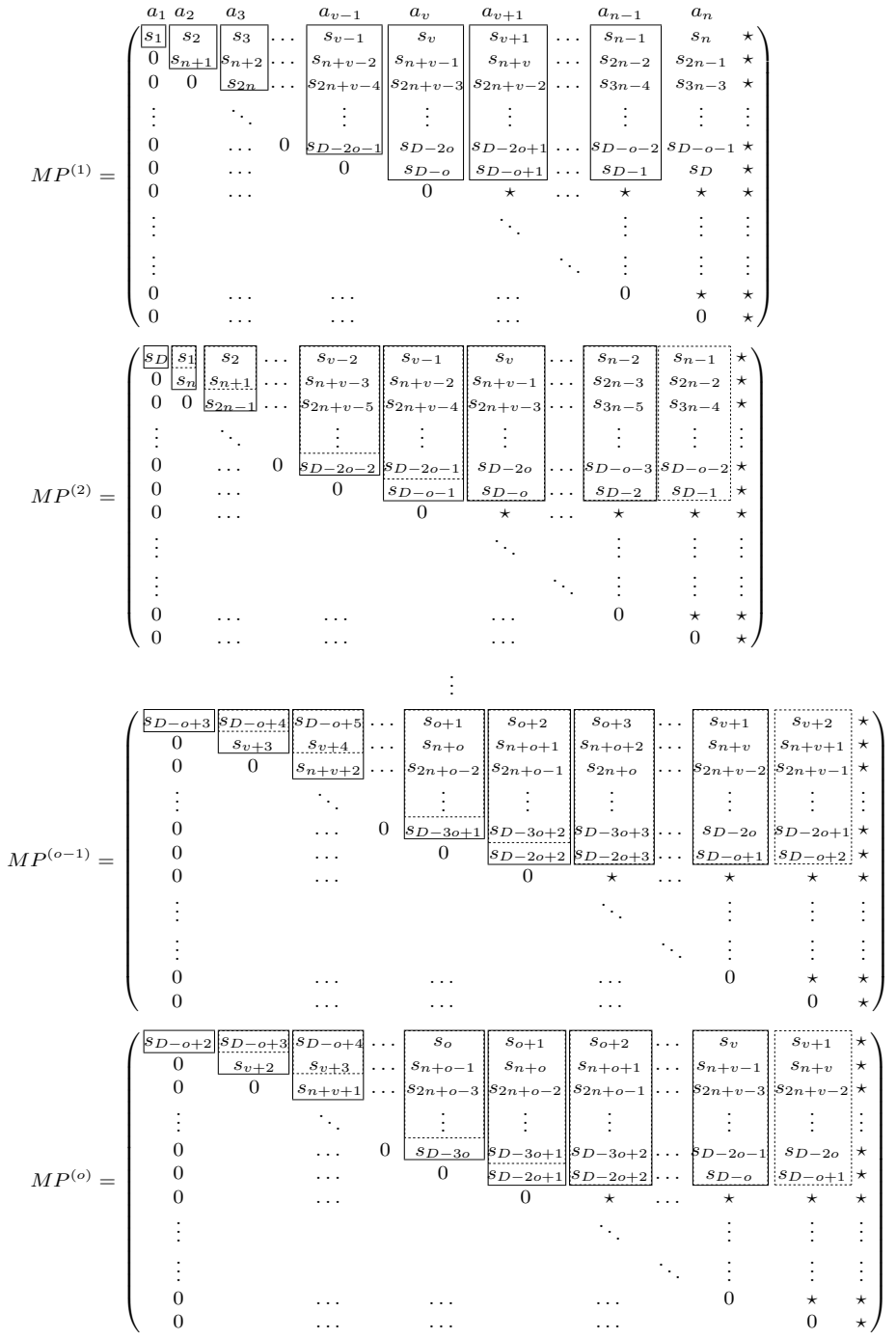


Fig. 3. Matrices $MP^{(i)}$ for cyclicUOV

Therefore we get

$$(\text{sign}_1,\ldots,\text{sign}_i)\cdot\begin{pmatrix} MP^{(k)}_{1,j} \\ MP^{(k)}_{2,j} \\ \vdots \\ MP^{(k)}_{i,j} \end{pmatrix} = (\text{sign}_1,\ldots,\text{sign}_i)\cdot\begin{pmatrix} MP^{(k-1)}_{1,j-1} \\ MP^{(k-1)}_{2,j-1} \\ \vdots \\ MP^{(k-1)}_{i,j-1} \end{pmatrix} \quad \begin{matrix} \forall i = 1,\ldots,v \\ j = i+1,\ldots,n, \\ k = 2,\ldots,o. \end{matrix} \tag{12}$$

The boxes in Figure 3 illustrate this equation. Boxes with continuous lines show the vector $(MP^{(k-1)}_{1,j-1},\ldots,MP^{(k-1)}_{i,j-1})^T$ on the right hand side of the equation, whereas the boxes with dashed lines represent the vector $(MP^{(k)}_{1,j},\ldots,MP^{(k)}_{i,j})^T$ on the left hand side. As one can see, the dashed boxes in the matrix $MP^{(k)}$ are exactly the same as the boxes with continuous lines in the matrix $MP^{(k-1)}$ ($k = 2,\ldots,o$). We can use this fact to speed up the verification process of cyclicUOV by a large factor (see Algorithm 1).

Algorithm 1. Verification process for cyclicUOV

for $i = 1$ to $n-1$ **do** ▷ first polynomial
  $a_i \leftarrow \sum_{j=1}^{\min(i,v)} MP^{(1)}_{ji} \cdot \text{sign}_j$
  $\text{temp}_i \leftarrow a_i$
end for
for $i = v+1$ to $n-1$ **do**
  $\text{temp}_i \leftarrow a_i + \sum_{j=v+1}^{i} MP^{(1)}_{ji} \cdot \text{sign}_j$
end for
$\text{temp}_n \leftarrow \sum_{j=1}^{n} MP^{(1)}_{ji} \cdot \text{sign}_j$
$\text{temp}_{n+1} \leftarrow \sum_{j=1}^{n+1} MP^{(1)}_{j,n+1} \cdot \text{sign}_\text{j}$
$h'_1 \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
for $l = 2$ to o **do** ▷ polynomials $2,\ldots,o$
  $\text{temp}_{n+1} \leftarrow \sum_{j=1}^{n+1} MP^{(l)}_{j,n+1} \cdot \text{sign}_j$
  for $i = n$ to $v+1$ by -1 **do**
    $a_i \leftarrow a_{i-1}$
    $\text{temp}_i \leftarrow a_i + \sum_{j=v+1}^{i} MP^{(l)}_{ji} \cdot \text{sign}_j$
  end for
  for $i = v$ to 2 by -1 **do**
    $a_i \leftarrow a_{i-1} + MP^{(l)}_{ii} \cdot \text{sign}_i$
    $\text{temp}_i \leftarrow a_i$
  end for
  $a_1 \leftarrow MP^{(l)}_{11} \cdot \text{sign}_1$
  $\text{temp}_1 \leftarrow a_1$
  $h'_l \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
end for
if $h_l = h'_l\ \forall l \in \{1,\ldots,o\}$ **then return** "ACCEPT" ▷ TEST
else return "REJECT"
end if

Algorithm 1 works as follows. The first matrix-vector product $\text{sign} \cdot MP^{(1)} \cdot \text{sign}^T$ is computed as for a random polynomial: From step 1 to step 9 we compute the product $\text{sign} \cdot MP^{(1)}$ (the result is written into the vector temp) and step 10 computes the scalar product of temp and sign. Furthermore we compute the vector $a = (a_1, \ldots, a_{n-1})$ which can be used for the computation of $\text{sign} \cdot MP^{(2)}$. In the loop (step 11 to step 24 of the algorithm) we compute the matrix vector products $\text{sign} \cdot MP^{(k)} \cdot \text{sign}^T$ $(k = 2, \ldots, o)$. Step 12 to step 22 computes the vector $\text{temp} = \text{sign} \cdot MP^{(k)}$. We begin with temp_{n+1} and go back to temp_1. In the computation of temp_i $(i = 2, \ldots, n)$ we use the values a_i computed before, since, due to the cyclic structure of the public key, they appear in several of the products $\text{sign} \cdot MP^{(k)}$ (see equation (12)). Furthermore (step 18 and 21) we update the values of the a_i $(i = 1, \ldots, n-1)$ for the use in the next iteration of the loop. Step 23 computes the scalar product of temp and sign. The last three steps (step 25 to 27) use the values h'_l $(l = 1, \ldots o)$ computed in step 10 and 23 to verify the authenticity of the signature.

Computational Effort. Evaluating the system $\mathcal{P}$ in the standard way, one needs

- $\frac{n \cdot (n+1)}{2}$ field multiplications to compute the vector mon (c.f. equation (5))
- and $o \cdot \frac{(n+1) \cdot (n+2) - 2}{2}$ field multiplications to compute the scalar products of equation (6).

Altogether, we need

$$\frac{n+1}{2} \cdot (n \cdot (o+1) + 2 \cdot o) - o \tag{13}$$

field multiplications. Algorithm 1 needs

- in step 2 $\frac{v \cdot (v+1)}{2} + (o-1) \cdot v$ field multiplications,
- in step 6 $\frac{(o-1) \cdot o}{2}$ field multiplications,
- in step 8 n field multiplications,
- in step 9 $n+1$ field multiplications,
- and in step 10 again $n+1$ field multiplications.

Therefore, to compute the value of h'_1, the algorithm needs $\frac{(n+1) \cdot (n+4)}{2}$

In the loop (step 11 to 24) Algorithm 1 needs

- in step 12 $n+1$ field multiplications,
- in step 15 $\frac{o \cdot (o+1)}{2}$ field multiplications,
- in step 18 $v-1$ field multiplications,
- in step 21 1 field multiplication,
- and in step 23 $n+1$ field multiplications.

So, for every iteration of the loop the algorithm needs $2 \cdot (n+1) + v + \frac{o \cdot (o+1)}{2}$ field multiplications.

Altogether, we need therefore

$$(o-1)\cdot\left(2\cdot(n+1)+v+\frac{o\cdot(o+1)}{2}\right)+\frac{(n+1)\cdot(n+4)}{2} \tag{14}$$

field multiplications to evaluate equation (10).

For $\mathbb{F}$ =GF(256), $(o,v)=(28,56)$ this means a reduction of the number of field multiplications needed during the verification process by 80 % or a factor of 5.0. For a UOV scheme over GF(31), $(o,v)=(33,66)$, we get a reduction factor of 5.4.

4.3 cyclicRainbow

The verification process for cyclicRainbow is mainly done as for cyclicUOV. However we have to consider the different structure of the polynomials. For cyclicRainbow, the matrices $MP^{(k)}$ look as shown in Figure 4.

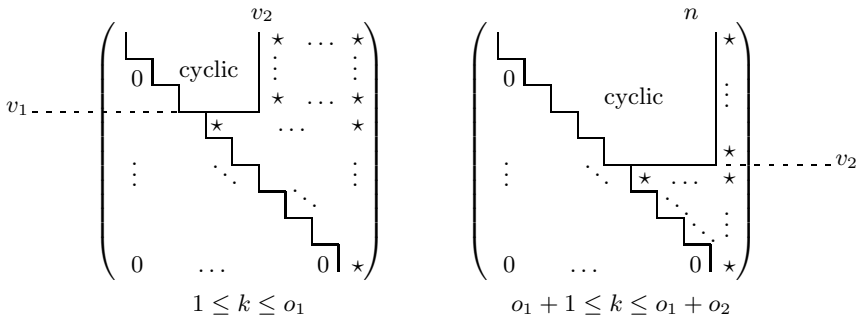


Fig. 4. Matrices $MP^{(k)}$ for cyclicRainbow

So we get for the polynomials $2,\ldots,o_1+1$

$$MP_{ij}^{(k)} = MP_{i,j-1}^{(k-1)} \;\forall i=1,\ldots,v_1,\; j=i+1,\ldots,v_2,\; k=2,\ldots,o_1+1 \tag{15}$$

or

$$(\mathrm{sign}_1,\ldots,\mathrm{sign}_i)\cdot\begin{pmatrix} MP_{1,j}^{(k)} \\ MP_{2,j}^{(k)} \\ \vdots \\ MP_{i,j}^{(k)} \end{pmatrix} = (\mathrm{sign}_1,\ldots,\mathrm{sign}_i)\cdot\begin{pmatrix} MP_{1,j-1}^{(k-1)} \\ MP_{2,j-1}^{(k-1)} \\ \vdots \\ MP_{i,j-1}^{(k-1)} \end{pmatrix} \quad \begin{array}{l}\forall i=1,\ldots,v_1,\\ j=i+1,\ldots,v_2,\\ k=2,\ldots,o_1+1.\end{array} \tag{16}$$

For the polynomials $o_1+2,\ldots,o_1+o_2$ we get

$$MP_{ij}^{(k)} = MP_{i,j-1}^{(k-1)} \;\forall i=1,\ldots,v_2,\; j=i+1,\ldots,n,\; k=o_1+2,\ldots,o_1+o_2 \tag{17}$$

or

$$(\text{sign}_1, \ldots, \text{sign}_i) \cdot \begin{pmatrix} MP_{1,j}^{(k)} \\ MP_{2,j}^{(k)} \\ \vdots \\ MP_{i,j}^{(k)} \end{pmatrix} = (\text{sign}_1, \ldots, \text{sign}_i) \cdot \begin{pmatrix} MP_{1,j-1}^{(k-1)} \\ MP_{2,j-1}^{(k-1)} \\ \vdots \\ MP_{i,j-1}^{(k-1)} \end{pmatrix} \begin{array}{c} \forall i = 1, \ldots, v_2, \\ j = i+1, \ldots, n, \\ k = o_1 + 2, \ldots, o_1 + o_2. \end{array} \tag{18}$$

To cover this fact, we use Algorithm 1 for both groups of polynomials separately (see Algorithm 2 in Appendix A).

Computational Cost. Our algorithm needs

- $\frac{(n+1)\cdot(n+4)}{2}$ field multiplications to evaluate $p^{(1)}$,
- $o_1 \cdot \left(\frac{(n+1)\cdot(n+4)}{2} - \frac{v_2\cdot(v_2+1)}{2} + \frac{o_1\cdot(o_1+1)}{2} + v_1 \right)$ field multiplications to evaluate the polynomials $p^{(2)} \ldots, p^{(o_1+1)}$ and
- $(o_2 - 1) \cdot \left(2 \cdot (n+1) + v_2 + \frac{o_2\cdot(o_2+1)}{2} \right)$ field multiplications to evaluate the polynomials $p^{(o_1+2)}, \ldots, p^{(o_1+o_2)}$.

For the parameters $(q, v_1, o_1, o_2) = (2^8, 17, 13, 13)$, this means a reduction by 56 % or a factor of 2.3 (with respect to the evaluation with the standard approach, see (13)). For a Rainbow scheme over GF(31), $(v_1, o_1, o_1) = (14, 19, 14)$ the reduction factor is 2.2.

5 Experiments

We checked our theoretical results on a straightforward C implementation of our schemes. Table 1 shows the results. The parameters in this table are chosen for 80 bit security.

Table 1. Improved versions of UOV and Rainbow

Scheme	private key size (kB)	hash length (bit)	signature length (bit)	public key size (kB)	public key red. factor	verification time ms	verification time s. u. f.[1]
UOV(31, 33, 66)	102.9	160	528	108.5	-	1.75	-
cyclicUOV(31, 33, 66)	102.9	160	528	17.1	6.3	0.34	5.2
UOV(256, 28, 56)	95.8	224	672	99.9	-	0.98	-
cyclicUOV(256, 28, 56)	95.8	224	672	16.5	6.1	0.20	4.9
Rainbow(31, 14, 19, 14)	17.1	160	256	25.3	-	0.44	-
cyclicRainbow(31, 14, 19, 14)	17.1	160	256	12.0	2.1	0.21	2.1
Rainbow(256, 17, 13, 13)	19.1	208	344	25.1	-	0.26	-
cyclicRainbow(256, 17, 13, 13)	19.1	208	344	9.5	2.6	0.13	2.0

[1] speed up factor of the verification time

The differences between the results of our theoretical analysis (see Section 4) and the actual runtime of the verification process is mainly caused by the heavy use of control structures in Algorithms 1 and 2.

6 Conclusion

In this paper we show a way how the structure in the public keys of cyclic versions of UOV and Rainbow can be used to achieve a significant speed up of the verification process. We propose improved algorithms for the verification process of UOV and Rainbow which run up to 5 times faster than the standard verification algorithm. Future research includes:

- *Use of special processor instructions*
  Like in the paper of Chen et al. [3] we plan to use special processor instructions to speed up our implementations.
- *Implementation in hardware*
  We plan to implement our schemes in hardware (e.g. on FPGA and HSM), which should also decrease the verification time.

Acknowledgements. We want to thank the anonymous reviewers for their comments which helped to improve the paper. The first author is supported by the Horst Görtz Foundation.

References

1. Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.): Post Quantum Cryptography. Springer, Heidelberg (2009)
2. Bogdanov, A., Eisenbarth, T., Rupp, A., Wolf, C.: Time-area optimized public-key engines: $\mathcal{MQ}$-cryptosystems as replacement for elliptic curves? In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 45–61. Springer, Heidelberg (2008)
3. Chen, A.I.-T., Chen, M.-S., Chen, T.-R., Cheng, C.-M., Ding, J., Kuo, E.L.-H., Lee, F.Y.-S., Yang, B.-Y.: SSE implementation of multivariate PKCs on modern x86 CPUs. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 33–48. Springer, Heidelberg (2009)
4. Ding, J., Schmidt, D.: Rainbow, a new multivariable polynomial signature scheme. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 164–175. Springer, Heidelberg (2005)
5. Ding, J., Yang, B.-Y., Chen, C.-H.O., Chen, M.-S., Cheng, C.M.: New Differential-Algebraic Attacks and Reparametrization of Rainbow. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 242–257. Springer, Heidelberg (2008)
6. Ding, J., Wolf, C., Yang, B.-Y.: ℓ-Invertible Cycles for $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic ($\mathcal{MQ}$) Public Key Cryptography. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 266–281. Springer, Heidelberg (2007)
7. Faugère, J.C.: A new efficient algorithm for computing Groebner bases (F4). Journal of Pure and Applied Algebra 139, 61–88 (1999)
8. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company (1979)
9. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced Oil and Vinegar Signature Schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999)

10. Kipnis, A., Shamir, A.: Cryptanalysis of the Oil & Vinegar Signature Scheme. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 257–266. Springer, Heidelberg (1998)
11. Matsumoto, T., Imai, H.: Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 419–453. Springer, Heidelberg (1988)
12. Patarin, J.: Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 33–48. Springer, Heidelberg (1996)
13. Patarin, J.: The oil and vinegar signature scheme. Presented at the Dagstuhl Workshop on Cryptography (September 1997)
14. Petzoldt, A., Bulygin, S., Buchmann, J.: A Multivariate Signature Scheme with a partially cyclic public key. In: Proceedings of SCC 2010, pp. 229–235 (2010)
15. Petzoldt, A., Bulygin, S., Buchmann, J.: Selecting Parameters for the Rainbow Signature Scheme. In: Sendrier, N. (ed.) PQCrypto 2010. LNCS, vol. 6061, pp. 218–240. Springer, Heidelberg (2010)
16. Petzoldt, A., Bulygin, S., Buchmann, J.: CyclicRainbow – A Multivariate Signature Scheme with a Partially Cyclic Public Key. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 33–48. Springer, Heidelberg (2010)
17. Petzoldt, A., Bulygin, S., Buchmann, J.: Linear Recurring Sequences for the UOV Key Generation. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 335–350. Springer, Heidelberg (2011)
18. Shor, P.: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM J. Comput. 26(5), 1484–1509
19. Thomae, E., Wolf, C.: Solving Underdetermined Systems of Multivariate Quadratic Equations Revisited. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 156–171. Springer, Heidelberg (2012)

A Algorithm for the Verification Process of Cyclicrainbow

Algorithm 2 shows the improved verification process for Rainbow schemes with two layers and partially circulant public key. The algorithm can be extended to Rainbow schemes with more than two layers in a natural way.

Algorithm 2. Verification process for cyclicRainbow

for $i = 1$ to $v_2 - 1$ **do** ▷ First polynomial
  $a_i \leftarrow \sum_{j=1}^{\min(i,v_1)} MP_{ji}^{(1)} \cdot \text{sign}_j$
  $\text{temp}_i \leftarrow a_i$
end for
for $i = v_1 + 1$ to $v_2 - 1$ **do**
  $\text{temp}_i \leftarrow a_i + \sum_{j=v_1+1}^{i} MP_{ji}^{(1)} \cdot \text{sign}_j$
end for
for $i = v_2$ to $n + 1$ **do**
  $\text{temp}_i \leftarrow \sum_{j=1}^{i} MP_{ji}^{(1)} \cdot \text{sign}_j$
end for
$h'_1 \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
for $l = 2$ to o_1 **do** ▷ Polynomials 2 to o_1
  for $i = v_2 + 1$ to $n + 1$ **do**
    $\text{temp}_i \leftarrow \sum_{j=1}^{i} MP_{ji}^{(l)} \cdot \text{sign}_j$
  end for
  for $i = v_2$ to $v_1 + 1$ by -1 **do**
    $a_i \leftarrow a_{i-1}$
    $\text{temp}_i \leftarrow a_i + \sum_{j=v+1}^{i} MP_{ji}^{(l)} \cdot \text{sign}_j$
  end for
  for $i = v_1$ to 2 by -1 **do**
    $a_i \leftarrow a_{i-1} + MP_{ii}^{(l)} \cdot \text{sign}_i$
    $\text{temp}_i \leftarrow a_i$
  end for
  $a_1 \leftarrow MP_{11}^{(l)} \cdot \text{sign}_1$
  $\text{temp}_1 \leftarrow a_1$
  $h'_l \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
end for

Algorithm 2. Verification process for cyclicRainbow (cont.)

$\text{temp}_{n+1} \leftarrow \sum_{j=1}^{n+1} MP_{j,n+1}^{(o_1+1)} \cdot \text{sign}_j$ ▷ (o_1+1)-th polynomial
for $i = n$ to $v_2 + 1$ by -1 **do**
  $a_i \leftarrow \sum_{j=1}^{v_2} MP_{ji}^{(o_1+1)} \cdot \text{sign}_j$
  $\text{temp}_i \leftarrow a_i + \sum_{j=v_2+1}^{i} MP_{ji}^{(o_1+1)} \cdot \text{sign}_j$
end for
for $i = v_2$ to $v_1 + 1$ by -1 **do**
  $a_i \leftarrow a_{i-1} + \sum_{j=v_1+1}^{i} MP_{ji}^{(o_1+1)} \cdot \text{sign}_j$
  $\text{temp}_i \leftarrow a_i$
end for
for $i = v_1$ to 2 by -1 **do**
  $a_i \leftarrow a_{i-1} + MP_{ii}^{(o_1+1)} \cdot \text{sign}_i$
  $\text{temp}_i \leftarrow a_i$
end for
$a_1 \leftarrow MP_{11}^{(o_1+1)} \cdot \text{sign}_1$
$\text{temp}_1 \leftarrow a_1$
$h'_{o_1+1} \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
for $l = o_1 + 2$ to $o_1 + o_2$ **do** ▷ Polynomials $o_1 + 2$ to $o_1 + o_2$
  $\text{temp}_{n+1} \leftarrow \sum_{j=1}^{n+1} MP_{j,n+1}^{(l)} \cdot \text{sign}_j$
  for $i = n$ to $v_2 + 1$ by -1 **do**
    $a_i \leftarrow a_{i-1}$
    $\text{temp}_i \leftarrow a_i + \sum_{j=v_2+1}^{i} MP_{ji}^{(l)} \cdot \text{sign}_j$
  end for
  for $i = v_2$ to 2 by -1 **do**
    $a_i \leftarrow a_{i-1} + MP_{ii}^{(l)} \cdot \text{sign}_i$
    $\text{temp}_i \leftarrow a_i$
  end for
  $a_1 \leftarrow MP_{11}^{(l)} \cdot \text{sign}_1$
  $\text{temp}_1 \leftarrow a_1$
  $h'_l \leftarrow \sum_{j=1}^{n+1} \text{temp}_j \cdot \text{sign}_j$
end for
if $h_l = h'_l \; \forall l \in \{1, \ldots, m\}$ **then return** "ACCEPT" ▷ TEST
else return "REJECT"
end if

The Hardness of Code Equivalence over $\mathbb{F}_q$ and Its Application to Code-Based Cryptography

Nicolas Sendrier[1] and Dimitris E. Simos[1,2]

[1] INRIA Paris-Rocquencourt
Project-Team SECRET
78153 Le Chesnay Cedex, France
[2] SBA Research
1040 Vienna, Austria
{nicolas.sendrier,dimitrios.simos}@inria.fr,
dsimos@sba-research.org

Abstract. The code equivalence problem is to decide whether two linear codes over $\mathbb{F}_q$ are identical up to a linear isometry of the Hamming space. In this paper, we review the hardness of code equivalence over $\mathbb{F}_q$ due to some recent negative results and argue on the possible implications in code-based cryptography. In particular, we present an improved version of the three-pass identification scheme of Girault and discuss on a connection between code equivalence and the hidden subgroup problem.

Keywords: Code Equivalence, Isometry, Hardness, Zero-Knowledge Protocols, Quantum Fourier Sampling, Linear Codes.

1 Introduction

The purpose of this work is to examine the applications of the worst-case and average-case hardness of the Code Equivalence problem to the field of code-based cryptography. The latter problem is, given the generator matrices of two q-ary linear codes, how hard is it to decide whether or not these codes are identical up to an isometry of Hamming space? The support splitting algorithm ($\mathcal{SSA}$) [28] runs in polynomial time for all but a negligible proportion of all linear codes, and solves the latter problem by recovering the isometry when it is just a permutation of the code support.

The McEliece public-key cryptosystem [23] and Girault's zero-knowledge protocol [17], both candidates for post-quantum cryptography, are related to the hardness of permutationally equivalent linear codes. For the McEliece cryptosystem, the $\mathcal{SSA}$ is able to detect some weak keys but a polynomial attack is infeasible due to the large number of possible private keys. However, the security of Girault's zero-knowledge protocol is severely weakened and cannot longer be used with random codes but only with weakly self-dual codes (the hard instances of $\mathcal{SSA}$).

P. Gaborit (Ed.): PQCrypto 2013, LNCS 7932, pp. 203–216, 2013.

Recently in [29], the worst-case and average-case hardness of code equivalence over $\mathbb{F}_q$ was studied and it was shown that in practice, $\mathcal{SSA}$ could be extended for $q \in \{3, 4\}$, and similarly solve all but an exponentially small proportion of the instances in polynomial time, when isometries are under consideration. However, for any fixed $q \geq 5$, the problem seems to be intractable for almost all instances.

In light of these new results, we repair Girault's zero-knowledge protocol over $\mathbb{F}_q$, when $q \geq 5$, by showing that random codes are again a viable option. Moreover, the context of the framework built in [11] suggests that codes with large automorphism groups resist quantum Fourier sampling as long as permutation equivalence is considered. We examine whether it is possible to extend these results, when a more general notion of code equivalence over $\mathbb{F}_q$ is taken into account, in particular when the equivalence mapping is an isometry and not just a permutation of the code support.

The paper is structured as follows. In section 2 we define the different notions of equivalence of linear codes over $\mathbb{F}_q$ when isometries are considered, while in section 3 we formally define the Code Equivalence problem and present a thorough analysis of its hardness. In section 4 we review the protocol of Girault together with its weakness and repair its security using results based on the hardness of code equivalence, while in the last section we elaborate on the connection between code equivalence over $\mathbb{F}_q$ and the quantum Fourier sampling.

2 Equivalence of Linear Codes over $\mathbb{F}_q$

Code equivalence is a basic concept in coding theory with several applications in code-based cryptography; the McEliece public-key cryptosystem [23], Girault's identification scheme [17] and the CFS signature scheme [10], to name a few. The notion of equivalence of linear codes used in code-based cryptography usually involves only permutations as the code alphabet is the binary field. However, this is by far the case in coding theory where for a more general notion of equivalence all isometries of the Hamming space have to be included. In this section, we review the concept of what it means for codes to be "essentially different" by considering the metric Hamming space together with its isometries, which are the maps preserving the metric structure. This in turn will lead to a rigorous definition of equivalence of linear codes and as we shall see later on may provide additional applications in cryptography. In fact, we will call codes isometric if they are equivalent as subspaces of the Hamming space.

Let $\mathbb{F}_q$ be a finite field of cardinality $q = p^r$, where the prime number p is its characteristic, and r is a positive integer. As usual, a linear $[n, k]$ code C is a k-dimensional subspace of the finite vector space $\mathbb{F}_q^n$ and its elements are called codewords. We consider all vectors, as row vectors. Therefore, an element v of $\mathbb{F}_q^n$ is of the form $v := (v_1, \ldots, v_n)$. It can also be regarded as the mapping v from the set $\mathcal{I}_n = \{1, \ldots, n\}$ to $\mathbb{F}_q$ defined by $v(i) := v_i$. The Hamming distance (metric) on $\mathbb{F}_q^n$ is the following mapping,

$$d : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{N} : (x, y) \mapsto d(x, y) := \mid \{i \in \{1, 2, \ldots, n\} \mid x_i \neq y_i\} \mid .$$

The pair $(\mathbb{F}_q^n, d)$ is a metric space, called the Hamming space of dimension n over $\mathbb{F}_q$, denoted by $H(n,q)$. The Hamming weight $w(x)$ of a codeword $x \in C$ is simply the number of its non-zero coordinates, i.e. $w(x) := d(x,0)$.

It is well-known due to a theorem of MacWilliams that any isometry between linear codes preserving the weight of the codewords induces an equivalence for codes [22]. Therefore, two codes C, C' are of the same quality if there exists a mapping $\iota : \mathbb{F}_q^n \mapsto \mathbb{F}_q^n$ with $\iota(C) = C'$ which preserves the Hamming distance, i.e. $d(\upsilon, \upsilon') = d(\iota(\upsilon), \iota(\upsilon'))$, for all $\upsilon, \upsilon' \in \mathbb{F}_q^n$. Mappings with the latter property are called the isometries of $H(n,q)$, and the two codes C and C' will be called isometric. Clearly, isometric codes have the same error-correction capabilities, and obvious permutations of the coordinates are isometries. We write $\mathcal{S}_n$ for the symmetric group acting on the set $\mathcal{I}_n$, equipped with the composition of permutations.

Definition 1. *Two linear codes $C, C' \subseteq \mathbb{F}_q^n$ will be called permutationally equivalent*[1]*, and will be denoted as $C \overset{\mathbf{PE}}{\sim} C'$, if there exists a permutation $\sigma \in \mathcal{S}_n$ that maps C onto C', i.e. $C' = \sigma(C) = \{\sigma(x) \mid x = (x_1, \ldots, x_n) \in C\}$ where $\sigma(x) = \sigma(x_1, \ldots, x_n) := (x_{\sigma^{-1}(1)}, \ldots, x_{\sigma^{-1}(n)})$.*

Note also that the use of σ^{-1} in the index is consisted as we have $\sigma(\pi(C)) = \sigma \circ \pi(C)$. This can easily be seen by considering $x \in C$, and $\sigma, \pi \in \mathcal{S}_n$ such that $\sigma(\pi(x)) = \sigma((x_{\pi^{-1}(i)})_{i \in \mathcal{I}_n})$. Let $y_i = x_{\pi^{-1}(i)}$, $i \in \mathcal{I}_n$. Then $\sigma(\pi(x)) = \sigma((y_i)_{i \in \mathcal{I}_n}) = (y_{\sigma^{-1}(i)})_{i \in \mathcal{I}_n} = (x_{\pi^{-1}\sigma^{-1}(i)})_{i \in \mathcal{I}_n} = (x_{(\sigma\pi)^{-1}(i)})_{i \in \mathcal{I}_n} = \sigma \circ \pi(x)$.

Moreover, there is a particular subgroup of $\mathcal{S}_n$ that maps C onto itself, the permutation group of C defined as $\mathrm{PAut}(C) := \{C = \sigma(C) \mid \sigma \in \mathcal{S}_n\}$. $\mathrm{PAut}(C)$ always contains the identity permutation. If it does not contain any other element, we will say that it is trivial.

Recall, that we defined two codes to be isometric if there exists an isometry that maps one into another. Isometries that are linear[2], are called linear isometries. Therefore, we can obtain a more general notion of equivalence for codes induced by linear isometries of $\mathbb{F}_q$. Moreover, it can be shown that any linear isometry between two linear codes $C, C' \subseteq \mathbb{F}_q^n$ can always be extended to an isometry of $\mathbb{F}_q^n$ [6].

The group of all linear isometries of $H(n,q)$ corresponds to the semidirect product of $\mathbb{F}_q^{*n}$ and $\mathcal{S}_n$, $\mathbb{F}_q^{*n} \rtimes \mathcal{S}_n = \{(\upsilon; \pi) \mid \upsilon : \mathcal{I}_n \mapsto \mathbb{F}_q^*, \pi \in \mathcal{S}_n\}$, called the monomial group of degree n over $\mathbb{F}_q^*$, where the multiplication within this group is defined by

$$(\upsilon; \pi)(\upsilon'; \pi') = (\upsilon \upsilon'_\pi, \pi\pi') \quad \text{and} \quad (\upsilon \upsilon'_\pi)_i := \upsilon_i \upsilon'_{\pi^{-1}(i)} \tag{1}$$

where $\mathbb{F}_q^*$ denotes the multiplicative group of $\mathbb{F}_q$. Hence, any linear isometry ι can be expressed as a pair of mappings $(\upsilon; \pi) \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$. Note that, some authors [6,14,16], describe this group as the wreath product $\mathbb{F}_q^* \wr_n \mathcal{S}_n$. The action

[1] This definition can also met as permutationally isometric codes in the literature, see [6].

[2] For all $u, v \in \mathbb{F}_q^n$ we have $\iota(u+v) = \iota(u) + \iota(v)$ and $\iota(0) = 0$.

of the latter group in an element of $\mathbb{F}_q^n$ is translated into an equivalence for linear codes.

Definition 2. *Two linear codes $C, C' \subseteq \mathbb{F}_q^n$ will be called linearly or monomially equivalent, and will be denoted as $C \overset{\mathbf{LE}}{\sim} C'$, if there exists a linear isometry $\iota = (\upsilon; \sigma) \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ that maps C onto C', i.e. $C' = (\upsilon; \sigma)(C) = \{(\upsilon; \sigma)(x) \mid (x_1, \ldots, x_n) \in C\}$ where $(\upsilon; \sigma)(x_1, \ldots, x_n) := (\upsilon_1 x_{\sigma^{-1}(1)}, \ldots, \upsilon_n x_{\sigma^{-1}(n)})$.*

If $q = p^r$ is not a prime, then the Frobenius automorphism $\tau : \mathbb{F}_q \to \mathbb{F}_q, x \mapsto x^p$ applied on each coordinate of $\mathbb{F}_q^n$ preserves the Hamming distance, too. Moreover, for $n \geq 3$, the isometries of $\mathbb{F}_q^n$ which map subspaces onto subspaces are exactly the semilinear mappings[3] of the form $(\upsilon; (\alpha, \pi))$, where $(\upsilon; \pi)$ is a linear isometry and α is a field automorphism, i.e. $\alpha \in \mathrm{Aut}(\mathbb{F}_q)$ (c.f. [6,21]). All these mappings form the group of semilinear isometries of $H(n, q)$ which is isomorphic to the semidirect product $\mathbb{F}_q^{*n} \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)$, where the multiplication of elements is given by

$$(\upsilon; (\alpha, \pi))(\varphi; (\beta, \sigma)) := (\upsilon \cdot \alpha(\varphi_\pi); (\alpha\beta, \pi\sigma)) \tag{2}$$

Moreover, there is a description of $\mathbb{F}_q^{*n} \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)$ as a generalized wreath product $\mathbb{F}_q^* \wr_n (\mathrm{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)$, see [6,15,21]. Clearly, the notion of semilinear isometry which can be expressed as a group action on the set of linear subspaces gives rise to the most general notion of equivalence for linear codes.

Definition 3. *Two linear codes $C, C' \subseteq \mathbb{F}_q^n$ will be called semilinearly equivalent, and will be denoted as $C \overset{\mathbf{SLE}}{\sim} C'$, if there exists a semilinear isometry $(\upsilon; (\alpha, \sigma)) \in \mathbb{F}_q^{*n} \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)$ that maps C onto C', i.e. $C' = (\upsilon; (\alpha, \sigma))(C) = \{(\upsilon; (\alpha, \sigma))(x) \mid (x_i)_{i \in \mathcal{I}_n} \in C\}$ where $(\upsilon; (\alpha, \sigma))(x_1, \ldots, x_n) = (\upsilon_1 \alpha(x_{\sigma^{-1}(1)}), \ldots, \upsilon_n \alpha(x_{\sigma^{-1}(n)}))$.*

Finally, we can define the monomial group of C as $\mathrm{MAut}(C) := \{C = (\upsilon; \sigma)(C) \mid (\upsilon; \sigma) \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n\}$ and the automorphism group of C as $\mathrm{Aut}(C) := \{C = (\upsilon; (\alpha, \sigma))(C) \mid (\upsilon; (\alpha, \sigma)) \in \mathbb{F}_q^{*n} \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)\}$ where their elements map each codeword of C to another codeword of C, under the respective actions of the involved groups. For more details, on automorphism groups of linear codes we refer to [20]. In addition, we remark the following:

1. When $\mathbb{F}_q = \mathbb{F}_2$ the group of linear isometries of $H(n, 2)$ is isomorphic to $\mathcal{S}_n$, therefore all notions of equivalence are the same.
2. The group of semilinear isometries of $H(n, q)$ is the same as the group of linear isometries if and only if q is a prime (since $\mathrm{Aut}(\mathbb{F}_q)$ is trivial if and only if q is a prime). Therefore, semilinear equivalence reduces to linear equivalence for prime fields, and is different for all other cases.

[3] $\sigma : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is semilinear if there exists $\alpha \in \mathrm{Aut}(\mathbb{F}_q)$ such that for all $u, v \in \mathbb{F}_q^n$ and $k \in \mathbb{F}_q$ we have $\sigma(u + v) = \sigma(u) + \sigma(v)$ and $\sigma(ku) = \alpha(k)\sigma(u)$.

3 The Code Equivalence Problem

For efficient computation of codes we represent them with generator matrices. A $k \times n$ matrix G over $\mathbb{F}_q$, is called a generator matrix for the $[n, k]$ linear code C if the rows of G form a basis for C, so that $C = \{xG \mid x \in \mathbb{F}_q^k\}$. In general, a linear code possess many different bases, and it is clear from linear algebra that the set of all generator matrices for C can be reached by $\{SG \mid S \in \mathrm{GL}_k(q)\}$, where $\mathrm{GL}_k(q)$ is the group of all $k \times k$ invertible matrices over $\mathbb{F}_q$.

For any $\sigma \in \mathcal{S}_n$ associate by $P_\sigma = [p_{i,j}]$ the $n \times n$ matrix such that $p_{i,j} = 1$ if $\sigma(i) = j$ and $p_{i,j} = 0$ otherwise, therefore P_σ is a permutation matrix. Note that, the action of $\sigma \in \mathcal{S}_n$ on $x \in \mathbb{F}_q^n$ agrees with the ordinary matrix multiplication. The permutation matrices form a subgroup of $M_n(q)$, the set of all $n \times n$ monomial matrices over $\mathbb{F}_q$, that is, matrices with exactly one nonzero entry per row and column from $\mathbb{F}_q$. If $M = [m_{i,j}] \in M_n(q)$, then $M = DP$, where P is a permutation matrix and $D = [d_{i,j}] = \mathrm{diag}(d_1, \ldots, d_n)$ is a diagonal matrix with $d_i = d_{i,i} = m_{i,j}$ if $m_{i,j} \neq 0$ and $d_{i,j} = 0$ if $i \neq j$. There is an isomorphism between diagonal matrices and $\mathbb{F}_q^{*n}$, therefore we associate $D_\upsilon = \mathrm{diag}(\upsilon_1, \ldots, \upsilon_n)$ for $\upsilon = (\upsilon_i)_{i \in \mathcal{I}_n} \in \mathbb{F}_q^{*n}$. Hence, we can map any linear isometry $(\upsilon; \sigma) \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ to a monomial matrix $M_{(\upsilon;\sigma)} = D_\upsilon P_\sigma \in M_n(q)$, and this mapping is an isomorphism between $\mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ and $M_n(q)$. Therefore, we can express the equivalence between linear codes in terms of their generator matrices.

Problem 1. Given two $k \times n$ matrices G and G' over $\mathbb{F}_q$, whose rows span two $[n, k]$ linear codes C and C' over $\mathbb{F}_q$, does there exist $S \in \mathrm{GL}_k(q)$ and a monomial matrix $M_{(\upsilon;\sigma)} = D_\upsilon P_\sigma \in M_n(q)$ such that $G' = SGD_\upsilon P_\sigma$?

We will refer to the decidability of the previous problem, as the Linear Code Equivalence problem. The Semi-Linear Code Equivalence problem can be defined analogously by permitting the application of a field automorphism in the columns of the scrambled generator matrix. In particular, we define the following problem.

Problem 2. Given two $k \times n$ matrices G and G' over $\mathbb{F}_q$, whose rows span two $[n, k]$ linear codes C and C' over $\mathbb{F}_q$, does there exist $S \in \mathrm{GL}_k(q)$, a monomial matrix $M_{(\upsilon;\sigma)} = D_\upsilon P_\sigma \in M_n(q)$ and a field automorphism $\alpha \in \mathrm{Aut}(\mathbb{F}_q)$ such that $G' = S\alpha(GD_\upsilon P_\sigma)$?

Finally, we review the hardness of the code equivalence problem, therefore we deem necessary to briefly mention the most significant results in terms of complexity, for deciding it, and algorithms, for solving it.

When the linear isometry $(\upsilon; \sigma)$ is just a permutation, i.e. D_υ is equal to I_n, we will call problem 1, as the Permutation Code Equivalence problem. The latter problem, was introduced in [26], who showed that if $\mathbb{F}_q = \mathbb{F}_2$ then it is harder than the Graph Isomorphism, there exists a polynomial time reduction, but not NP-complete unless P = NP. A different proof of this reduction is also given in [21]. Recently, the reduction of [26] was generalized in [18] over any field $\mathbb{F}_q$, hence Permutation Code Equivalence is harder than the Graph

Isomorphism, for any field $\mathbb{F}_q$. The latter problem, has been extensively studied for decades, but until now there is no polynomial-time algorithm for solving all of its instances. Clearly, (Semi)-Linear Code Equivalence for any $\mathbb{F}_q$ cannot be easier than the Graph Isomorphism, since it contains the Permutation Code Equivalence as a subproblem.

Last but not least, we would like to mention that the McEliece public-key cryptosystem [23] is related to the hardness of permutationally equivalent binary linear codes. Towards this direction, another important complexity result was shown in [11], that the Hidden Subgroup problem also reduces to Permutation Code Equivalence for any field $\mathbb{F}_q$.

The Support Splitting Algorithm can be used as an oracle to decide whether two binary codes are permutationally equivalent [28], as well as to retrieve the equivalence mapping. Other notable algorithms for code equivalence can be found in [3,7,13]. The main idea of $\mathcal{SSA}$ is to partition the support $\mathcal{I}_n$ of a code $C \subseteq \mathbb{F}_2^n$, into small sets that are fixed under operations of $\mathrm{PAut}(C)$. The algorithm employs the concept of invariants and signatures, defined in [28]. Invariants are mappings such that any two permutationally equivalent codes take the same value, while signatures depends on the code and one of its positions.

Definition 4. *A signature S over a set F maps a code $C \subseteq \mathbb{F}_q^n$ and an element $i \in \mathcal{I}_n$ into an element of F and is such that for all $\sigma \in \mathcal{S}_n$, $S(C,i) = S(\sigma(C), \sigma(i))$. Moreover, S is called discriminant for C if there exist $i, j \in \mathcal{I}_n$ such that $S(C,i) \neq S(C,j)$ and fully discriminant if this holds $\forall\ i, j \in \mathcal{I}_n$.*

The fundamental idea of $\mathcal{SSA}$ is to be able to find a distinct property for the code and one of its positions, and thus by labeling them accordingly it is possible to recover the permutation between equivalent codes.

The main difficulty of the algorithm, is to obtain a fully discriminant signature, for as many codes as possible. In [28] it was shown that such a signature, can be built from the weight enumerator of the hull of a code C, denoted by $\mathcal{H}(C)$, and defined as the intersection of the code with its dual, $\mathcal{H}(C) = C \cap C^\perp$ [2], because the hull commutes with permutations[4], $\mathcal{H}(\sigma(C)) = \sigma(\mathcal{H}(C))$, and therefore is an invariant for permutation equivalence. The (heuristic) complexity of $\mathcal{SSA}$ for an $[n,k]$ code C is $\mathcal{O}(n^3 + 2^h n^2 \log n)$ where h is the dimension of the hull [24,28]. The first term is the cost of the Gaussian elimination needed to compute the hull. The second term is the (conjectured) number of refinements, $\log n$, multiplied by the cost one refinement (n weight enumerators of codes of dimension h and length n). Moreover, the cost of computing the weight enumerator of an $[n,h]$ code over $\mathbb{F}_q$ is proportional to nq^h operations in $\mathbb{F}_q$ [28].

In practice, for random codes, the hull has a small dimension with overwhelming probability [27] and the dominant cost for the average case is $\mathcal{O}(n^3)$. Note that, the worst case occurs when the hull dimension is maximal; weakly self-dual

[4] No such property exists in general for linear codes when (semi)-linear equivalence is considered, because the dual of equivalent codes do not remain equivalent with the same isometry as the original codes.

codes ($C \subset C^{\perp}$) are equal to their hulls. Then the algorithm becomes intractable with a complexity equal to $\mathcal{O}(2^k n^2 \log n)$.

Reduction of Linear Code Equivalence to Permutation Code Equivalence was made possible via the introduction of the closure of a linear code in [29]. A similar approach was given in [30].

Definition 5. *Let $\mathbb{F}_q = \{a_0, a_1, \ldots, a_{q-1}\}$, with $a_0 = 0$, and a linear code $C \subseteq \mathbb{F}_q^n$. Define $\mathcal{I}_{q-1}^{(n)}$ as the cartesian product of $\mathcal{I}_{q-1} \times \mathcal{I}_n$. The closure $\widetilde{C}$ of the code C is a code of length $(q-1)n$ over $\mathbb{F}_q$ where,*

$$\widetilde{C} = \{(a_k x_i)_{(k,i)\in\mathcal{I}_{q-1}^{(n)}} \mid (x_i)_{i\in\mathcal{I}_n} \in C\}.$$

Clearly, we see that every coordinate of the closure $\widetilde{C}$, corresponds to a coordinate position of a codeword of C multiplied by a nonzero element of $\mathbb{F}_q$. Since, the index $(k, i) \in \mathcal{I}_{q-1}^{(n)}$ of a position of a codeword of the closure means that $k \in \mathcal{I}_{q-1}$ and $i \in \mathcal{I}_n$, we have taken into account every possible multiplication of x_i with nonzero elements of $\mathbb{F}_q$. The fundamental property of the closure is realised in the following theorem, first given in [29].

Theorem 1. *Let $C, C' \subseteq \mathbb{F}_q^n$. Then C and C' are linearly equivalent, i.e. $C \overset{\mathbf{LE}}{\sim} C'$, if and only if $\widetilde{C}$ and $\widetilde{C'}$ are permutationally equivalent, i.e. $\widetilde{C} \overset{\mathbf{PE}}{\sim} \widetilde{C'}$.*

Theorem 1 is of great importance, because it realizes a reduction from the Linear Code Equivalence problem to the Permutation Code Equivalence problem. Thus, we are able to decide if the codes C and C' are linearly equivalent by checking their closures for permutation equivalence. Moreover, if the closures are permutation equivalent then there exists an algorithmic procedure that allows the retrieval of the initial isometry between C and C' by considering that a signature for an extension of $\mathcal{SSA}$ can be built from the weight enumerator of the $\mathcal{H}(\widetilde{C})$.

Unfortunately, it turns out that the closure $\widetilde{C}$ is a weakly self-dual code for every $q \geq 5$, considering both Euclidean and Hermitian duals, which are exactly the hard instances of $\mathcal{SSA}$ [29]. Moreover, for $\mathbb{F}_3$ and $\mathbb{F}_4$ equipped with the Euclidean and Hermitian inner product, respectively, the distribution of the dimension of $\mathcal{H}(\widetilde{C})$ follows the distribution of the dimension $\mathcal{H}(C)$, since the closure has the same dimension as C, and will be on average a small constant, [27], except in the cases where C is also a weakly self-dual code. Therefore, the Linear Code Equivalence problem can be decided (and solved) in polynomial time using $\mathcal{SSA}$ only in $\mathbb{F}_3$ and $\mathbb{F}_4$, as long as the hull of the given code is small (the worst-case being a weakly self-dual code). However, for $q \geq 5$ its complexity growth becomes exponential for all instances. Moreover, it was conjectured in [29], that for $q \geq 5$, Code Equivalence is hard for almost all instances. This argument, was further supported by some impossibility results on the Tutte polynomial of a graph which corresponds to the weight enumerator

of a code [34]. To conclude with, we would like to make clear that the hardness of the code equivalence arises from the absence of an easy computable invariant not the inexistence of an algorithm.

Table 1. Heuristic complexity for $\mathcal{SSA}$ and its extension over $\mathbb{F}_q$

Algorithm	Field (alphabet)	Random codes (average-case)	Weakly self-dual codes (worst-case)
$\mathcal{SSA}$	$\mathbb{F}_2$	$\mathcal{O}(n^3)$	$\mathcal{O}(2^k n^2 \log n)$
$\mathcal{SSA}$ extension	$\mathbb{F}_3$	$\mathcal{O}(n^3)$	$\mathcal{O}(3^k n^2 \log n)$
$\mathcal{SSA}$ extension	$\mathbb{F}_4$	$\mathcal{O}(n^3)$	$\mathcal{O}(2^{2k} n^2 \log n)$
$\mathcal{SSA}$ extension	$\mathbb{F}_q$, $q \geq 5$	$\mathcal{O}(q^k n^2 \log n)$	$\mathcal{O}(q^k n^2 \log n)$

4 Zero-Knowledge Protocols

A central concept in cryptography is zero-knowledge protocols. These protocols allow a prover to convince a verifier that it knows a secret without the verifier learning any information about the secret. In practice, this is used to allow one party to prove its identity to another by proving it has a particular secret. For a protocol to be zero-knowledge, no information can be revealed no matter what strategy a so-called cheating verifier, simply cheater, follows when interacting with the prover. Therefore, an important question is what happens to these protocols when the cheater is a quantum computer. Are there any zero-knowledge protocols sufficient to withstand such a powerful cheater in a post-quantum era?

In this section, we deal with protocols based on a particular type of alternative cryptography originating from error-correcting codes, called code-based cryptography. In this emerging field of cryptography the underlying hard problems which pose as its security assumptions, decoding in a random linear code and recovering the code structure, does not seem so far to be susceptible to attacks mounted by quantum computers [24]. In addition, as we shall mention in the following section there is a negative result regarding the connection between coding theory and the Hidden Subgroup problem, which is the starting point for designing efficient quantum algorithms.

The idea of using error-correcting codes for identification schemes is due to Harari [19], followed by Stern (first protocol) [31] and Girault [17]. Harari's protocol was broken and the security of Girault's one was severely weakened (we shall explain this shortly after) while the protocol of Stern was five-pass and unpractical. At Crypto'93, Stern proposed a new scheme [32], which is one of the main references in this area. Recently, there has been an upsurge on designing identification schemes mainly due to the work of several researchers [8,9,1], where their efforts concentrated on both reducing the communication cost and the probabibility of someone impersonating an honest prover.

4.1 Girault's Three-Pass Identification Scheme

Girault's identification scheme is a three-pass one with a cheating probability of 1/2 (compared to the usual 2/3 of Stern's protocol), and has the additional advantage that all computations are performed on the standard model instead of the random oracle model since there is no involvement of a hash function in the committments of the protocol. However, this advantage comes with a cost. At each round of the protocol a large number of bits has to be transmitted, which render the scheme unpractical. Its principle is as follows: Let H be an $(n-k)\times n$ matrix over the binary field $\mathbb{F}_2$ common to all users. Each prover $\mathcal{P}$ has an n-bit word e of small weight w randomly chosen by him and a public identifier $He = s$. Clearly, when H is a parity-check matrix of a linear code, computing e from H and s comes to finding a word of given small weight and given syndrome s, a well-known NP-hard problem. When $\mathcal{P}$ needs to authenticate to a verifier $\mathcal{V}$ as the owner of s, then $\mathcal{P}$ and $\mathcal{V}$ interact through the following scheme.

Step 1: $\mathcal{P}$ picks a random $n \times n$ permutation matrix P and a random $k \times k$ non-singular matrix S. P computes $H' = SHP$ and $s' = Ss$, and sends H' and s' to $\mathcal{V}$.
Step 2: $\mathcal{V}$ generates a random bit $c \in \{0, 1\}$ and sends it to $\mathcal{P}$.
Step 3a: If $c = 0$, $\mathcal{P}$ replies by delivering S, P to $\mathcal{V}$, who checks that $SHP = H'$ and $Ss = s'$.
Step 3b: If $c = 1$, $\mathcal{P}$ replies by delivering $e' = P^{-1}e$ to $\mathcal{V}$, who checks that the weight of e' is w and $H'e' = s'$.

The protocol is a multi-round one as it has to be repeated t times to reach a security level of $1 - (1/2)^t$ and was proved to be zero-knowledge on [17]. Its security is based on the hardness of two well-known problems in coding theory. The first one is the Binary Syndrome Decoding problem shown to be NP-complete in the worst case [5], but it is also widely believed that for the average case it still remains hard. The other assumption is related to the hardness of the Permutation Code Equivalence problem over the binary field since from the knowledge of H and H' someone must not be able to recover the scrambing matrix S and the permutation matrix P, as this would lead to information leakage about the secret key of $\mathcal{P}$. However, as we extensively discussed on section 3, $\mathcal{SSA}$ can recover the matrix P in (almost) polynomial time when the underlying code is chosen at random (see also the complexity figures in table 3), and then using elementary linear algebra the matrix S can also be found.

Still, the protocol can be used with weakly self-dual codes, the instances of Permutation Code Equivalence that the growth of $\mathcal{SSA}$ becomes exponential, however there is no significant advantage on decoding with self-dual codes and in addition this restrict too much the possibilities for the public key.

4.2 Improved Version of the Girault Protocol

We now consider, Girault's identification scheme in a q-ary setting. That is, the underlying finite field, will no longer be the binary field but the field $\mathbb{F}_q$

with q elements. For the security assumptions of the scheme we first have to consider syndrome decoding over $\mathbb{F}_q$. We define the decisional version of the q-ary Syndrome Decoding problem, below,

Problem 3. Given an $m \times n$ matrix H over $\mathbb{F}_q$, a target vector $s \in \mathbb{F}_q^m$ and an integer $w > 0$ does there exist a vector $x \in \mathbb{F}_q^n$ of weight $\leq w$ such that $Hx = s$?

which was also proven to be NP-complete in [4]. There are two main families of algorithms for solving the latter problem: Information Set Decoding (ISD) and (Generalized) Birthday algorithm (GBA). ISD has the lowest complexity of the two, and in a recent work [25] the complexity of a generalization of Stern's algorithm from [33] is analyzed which permits the decoding of linear codes over arbitrary finite fields $\mathbb{F}_q$. For a general treatment of the topic we refer to [24], while for the security of the scheme it is sufficient to consider that all known decoding attacks have an exponential cost on the code length.

Moreover, in an attempt to repair the security of the scheme we consider the (semi)-linear code equivalence instead of the permutation code equivalence, depending on whether $\mathbb{F}_q$ is a prime field or not. As one of the purposes of this paper, is to state the implications of the hardness of the (Semi)-Linear Code equivalence problem for designing cryptographic primitives, we choose the parameter q to be at least equal to 5, since we strongly believe that a random instance of the latter problem is hard for these cases (see also section 3).

The starting point of this improved version of Girault's scheme is the same as in the original one, with the exception that all operations now occur over $\mathbb{F}_q$, $q \geq 5$. Let H be an $(n-k) \times n$ matrix over $\mathbb{F}_q$ common to all users. Each prover $\mathcal{P}$ has an n-bit word e of small weight w randomly chosen by him and a public identifier $He = s$. As before, when a prover $\mathcal{P}$ needs to authenticate to a verifier $\mathcal{V}$ as the owner of s, then $\mathcal{P}$ and $\mathcal{V}$ interact through the following protocol.

Improved Version of Girault Identification Scheme

Key Generation: Random $[n, k]$ linear code with an $(n-k) \times n$ parity-check matrix H over $\mathbb{F}_q$
- **Private key:** A word $e \in \mathbb{F}_q^n$ of small weight w
- **Public key:** A public identifier $s \in \mathbb{F}_q^{n-k}$ such that $He = s$

Commitments:
- $\mathcal{P}$ picks a random $n \times n$ monomial matrix M, a random $k \times k$ non-singular matrix S and a field automorphism α of $\mathbb{F}_q$.
- $\mathcal{P}$ computes the commitments $s' = Ss$ and $H' = S\alpha(HM)$.
- $\mathcal{P}$ sends s' and H' to $\mathcal{V}$.

Challenge: $\mathcal{V}$ chooses randomly $c \in \{0, 1\}$ and sends it to $\mathcal{P}$.

Response:
- If $c = 0$ then $\mathcal{P}$ replies by delivering α, S, M to $\mathcal{V}$
- If $c = 1$ then $\mathcal{P}$ replies by delivering $e' = \alpha^{-1}(M^{-1}e)$ to $\mathcal{V}$

Verification:
- If $c = 0$ then $\mathcal{V}$ checks that $S\alpha(HM) = H'$ and $Ss = S'$.
- If $c = 1$ then $\mathcal{V}$ checks that the weight of e' is w and $H'e' = s'$.

The scheme is again a three-pass one and has to be repeated t times to reach a security level of $1-(1/2)^t$. The completeness, soundness and zero-knowledge of the scheme is a straight-forward verification of the proofs given by Girault in the original version [17], by replacing the permutation matrix P with the monomial matrix M and the field automorphism α (for non-prime fields) and therefore we avoid repeating them here to save space. Note that, the scheme is again usable in the standard model in contrast to the usual random oracle model.

We would like also to remark, that this q-ary version of Girault's scheme can be used again with the family of random linear codes for any field $\mathbb{F}_q$, $q \geq 5$. Moreover, we choose to commit the monomial matrix M instead of its (unique) factorization to a diagonal matrix D and a permutation matrix P (see also section 3) to reduce the (already) large cost of communication at each round (since we transmit matrices) as much as possible. A promising approach to circumvent this drawback could be to employ random structured codes as the public keys such as quasi-cyclic (QC) codes, similar to the work carried out in [1]. Although, there is no obvious advantage for an adversary mounting decoding attacks on QC codes, their rich structure may lead to structural attacks even when semi-linear code equivalence is considered (even though we are unaware of such kind of attacks) and a more careful analysis is required before proposing any specific parameters for the scheme.

5 A Note about Code Equivalence over $\mathbb{F}_q$ and Quantum Fourier Sampling

In [11], it was shown that permutation code equivalence over $\mathbb{F}_q$ has a direct reduction to a nonabelian Hidden Subgroup problem (HSP). It was further shown in the same paper that McEliece-type cryptosystems with certain conditions on the permutation automorphism groups of the underlying linear codes used as private keys, as is the case of rational Goppa codes, resist precisely the attacks to which the RSA and ElGamal cryptosystems are vulnerable, namely those based on generating and measuring coset states. This fact, eliminated the approach of strong Fourier sampling on which almost all known exponential speedups by quantum algorithms are based. In addition, these negative results have been extended in [12] for the case of Reed-Muller codes, which correspond to the particular case of the Sidelnikov cryptosystem.

There are two main questions arising from this framework: Whether there are any other families of codes suitable for cryptographic applications and what happens when we consider a more general notion of code equivalence over $\mathbb{F}_q$. We will investigate these matters, after briefly mentioning the conditions needed for the results of [11,12].

Recall from [11] that a linear code C is HSP-hard if strong quantum Fourier sampling, reveals negligible information about the permutation $\sigma \in \mathcal{S}_n$ of permutationally equivalent codes, i.e. $C' = \sigma(C)$. Moreover, the support of a permutation $\sigma \in \mathcal{S}_n$ is the number of points that are not fixed by σ, and the minimal degree of a subgroup $H \leq \mathcal{S}_n$ is the smallest support of any non-identity $\pi \in H$.

Theorem 2 (Theorem 1, [12]). *Let C be a q-ary $[n, k]$ linear code such that $q^{k^2} \leq n^{0.2n}$. If $|\mathrm{PAut}(C)| \leq e^{o(n)}$ and the minimal degree of $|\mathrm{PAut}(C)|$ is $\Omega(n)$ then C is HSP-hard.*

We now consider $\mathbb{F}_q$ to be a prime field (hence $\mathrm{Aut}(\mathbb{F}_q)$ is trivial) and the monomial group $\mathrm{MAut}(C)$ of a code $C \subseteq \mathbb{F}_q^n$ for the notion of linear code equivalence. Clearly, if the permutation part of $\mathrm{MAut}(C)$ satisfies the conditions of theorem 2, so does its closure $\widetilde{C}$ (see definition 5) which is a code of length $(q-1)n$ over the same field. Recall that two codes are linearly equivalent if and only if their closures are permutationally equivalent (c.f. theorem 1). In other words, the instances of codes that are HSP-hard for the Permutation Code Equivalence problem remain HSP-hard for the Linear Code Equivalence. This remark, would further imply that someone could design a McEliece-type cryptosystem by considering a monomial transformation of the private key instead of just a permutation without having to worry about attacks originating from the quantum Fourier sampling, based on rational Goppa codes over $\mathbb{F}_q$ for instance. However, we should note that these results apply only to high-rate $[n, k]$ codes over $\mathbb{F}_q$ (as $q^{k^2} \leq n^{0.2n}$ must be satisfied).

6 Conclusion

In this paper, we presented an analysis of the hardness of the Code Equivalence problem over $\mathbb{F}_q$ when the equivalence mapping is an isometry and not just a permutation of the code support. The hardness of the latter problem is of great importance when designing cryptographic primitives, such as public-key cryptosystems and identification schemes in the field of code-based cryptography. We stated the weaknesses of such an identification scheme (Girault's zero-knowledge protocol), and presented an improved version which relies on exactly these instances of the code equivalence that the problem is believed to be hard on average. Finally, we showed that some negative results regarding the possibility of attacking McEliece-type cryptosystems with quantum algorithms based on Fourier sampling apply also for other notions of code equivalence, besides the permutation equivalence, subject to certain conditions on the underlying family of codes used as private keys.

Acknowledgments. The work of the second author was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme. This Programme is supported by the Marie Curie Co-funding of Regional, National and International Programmes (COFUND) of the European Commission. In addition, the work of the second author has been supported by the Austrian Research Promotion Agency under grant 258376 and the Austrian COMET Program (FFG).

References

1. Aguilar, C., Gaborit, P., Schrek, J.: A new zero-knowledge code based identification scheme with reduced communication. In: 2011 IEEE Information Theory Workshop (ITW), pp. 648–652 (2011)
2. Assmus, E.F.J., Key, J.D.: Designs and their Codes. Cambridge Tracts in Mathematics, vol. 103. Cambridge University Press (1992), second printing with corrections (1993)
3. Babai, L., Codenotti, P., Grochow, J.A., Qiao, Y.: Code equivalence and group isomorphism. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, pp. 1395–1408. SIAM (2011)
4. Barg, S.: Some new NP-complete coding problems. Probl. Peredachi Inf. 30, 23–28 (1994)
5. Berlekamp, E., McEliece, R., van Tilborg, H.: On the inherent intractability of certain coding problems (corresp.). IEEE Transactions on Information Theory 24, 384–386 (1978)
6. Betten, A., Braun, M., Fripertinger, H., Kerber, A., Kohnert, A., Wassermann, A.: Error-Correcting Linear Codes: Classification by Isometry and Applications. Algorithms and Computation in Mathematics, vol. 18. Springer, Heidelberg (2006)
7. Bouyukliev, I.: About the code equivalence. Ser. Coding Theory Cryptol. 3, 126–151 (2007)
8. Cayrel, P.L., Gaborit, P., Girault, M.: Identity-based identification and signature schemes using correcting codes. In: Augot, D., Sendrier, N., Tillich, J.P. (eds.) Workshop on Coding and Cryptography - WCC 2007, pp. 69–78. INRIA (2007)
9. Cayrel, P.-L., Véron, P., El Yousfi Alaoui, S.M.: A zero-knowledge identification scheme based on the q-ary syndrome decoding problem. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 171–186. Springer, Heidelberg (2011)
10. Courtois, N.T., Finiasz, M., Sendrier, N.: How to achieve a McEliece-based digital signature scheme. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 157–174. Springer, Heidelberg (2001)
11. Dinh, H., Moore, C., Russell, A.: McEliece and niederreiter cryptosystems that resist quantum fourier sampling attacks. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 761–779. Springer, Heidelberg (2011)
12. Dinh, H., Moore, C., Russell, A.: Quantum fourier sampling, code equivalence, and the quantum security of the mceliece and sidelnikov cryptosystems. Tech. rep. (2011), also available as arXiv:1111.4382v1
13. Feulner, T.: The automorphism groups of linear codes and canonical representatives of their semilinear isometry classes. Adv. Math. Commun. 3, 363–383 (2009)
14. Fripertinger, H.: Enumeration of linear codes by applying methods from algebraic combinatorics. Grazer Math. Ber. 328, 31–42 (1996)
15. Fripertinger, H.: Enumeration of the semilinear isometry classes of linear codes. Bayrether Mathematische Schriften 74, 100–122 (2005)
16. Fripertinger, H., Kerber, A.: Isometry classes of indecomposable linear codes. In: Giusti, M., Cohen, G., Mora, T. (eds.) AAECC 1995. LNCS, vol. 948, pp. 194–204. Springer, Heidelberg (1995)
17. Girault, M.: A (non-practical) three-pass identification protocol using coding theory. In: Seberry, J., Pieprzyk, J. (eds.) AUSCRYPT 1990. LNCS, vol. 453, pp. 265–272. Springer, Heidelberg (1990)

18. Grochow, J.A.: Matrix lie algebra isomorphism. Tech. Rep. TR11-168, Electronic Colloquium on Computational Complexity (2011), also available as arXiv:1112.2012, IEEE Conference on Computational Complexity (2012) (to appear)
19. Harari, S.: A new authentication algorithm. In: Wolfmann, J., Cohen, G. (eds.) Coding Theory 1988. LNCS, vol. 388, pp. 91–105. Springer, Heidelberg (1989)
20. Human, W.C.: Codes and groups. In: Pless, V., Human, W.C. (eds.) Handbook of Coding Theory, pp. 1345–1440. Elsevier, North-Holland (1998)
21. Kaski, P., Östergård, P.R.J.: Classification Algorithms for Codes and Designs. Algorithms and Computation in Mathematics, vol. 15. Springer, Heidelberg (2006)
22. MacWilliams, F.J.: Error-correcting codes for multiple-level transmission. Bell. Syst. Tech. J. 40, 281–308 (1961)
23. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. Tech. Rep. DSN Progress Report 42-44, California Institute of Technology, Jet Propulsion Laboratory, Pasadena, CA (1978)
24. Overbeck, R., Sendrier, N.: Code-based cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 95–145. Springer (2009)
25. Peters, C.: Information-set decoding for linear codes over $\mathbb{F}_q$. In: Sendrier, N. (ed.) PQCrypto 2010. LNCS, vol. 6061, pp. 81–94. Springer, Heidelberg (2010)
26. Petrank, E., Roth, R.M.: Is code equivalence easy to decide? IEEE Trans. Inform. Theory 43, 1602–1604 (1997)
27. Sendrier, N.: On the dimension of the hull. SIAM J. Discrete Math. 10(2), 282–293 (1997)
28. Sendrier, N.: Finding the permutation between equivalent linear codes: The support splitting algorithm. IEEE Trans. Inform. Theory 26, 1193–1203 (2000)
29. Sendrier, N., Simos, D.E.: How easy is code equivalence over $\mathbb{F}_q$? In: WCC 2013: Proceedings of the 8th International Workshop on Coding and Cryptography (preprint 2012) (to appear, 2013), `https://www.rocq.inria.fr/secret/PUBLICATIONS/codeq3.pdf`
30. Skersys, G.: Calcul du groupe d'automorphisme des codes. Détermination de l' equivalence des codes. Thèse de doctorat, Université de Limoges (October 1999)
31. Stern, J.: An alternative to the fiat-shamir protocol. In: Quisquater, J.J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 173–180. Springer, Heidelberg (1990)
32. Stern, J.: A new identification scheme based on syndrome decoding. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 13–21. Springer, Heidelberg (1994)
33. Stern, J.: A method for finding codewords of small weight. In: Wolfmann, J., Cohen, G. (eds.) Coding Theory 1988. LNCS, vol. 388, pp. 106–113. Springer, Heidelberg (1989)
34. Vertigan, D.: Bicycle dimension and special points of the Tutte polynomial. Journal of Combinatorial Theory, Series B 74, 378–396 (1998)

Timing Attacks against the Syndrome Inversion in Code-Based Cryptosystems*

Falko Strenzke

Cryptography and Computeralgebra, Department of Computer Science,
Technische Universität Darmstadt, Germany
fstrenzke@crypto-source.de

Abstract. In this work we present the first practical key-aimed timing attack against code-based cryptosystems. It arises from vulnerabilities that are present in the inversion of the error syndrome through the Extended Euclidean Algorithm that is part of the decryption operation of these schemes. Three types of timing vulnerabilities are combined to a successful attack. Each is used to gain information about the secret support, which is part of code-based decryption keys: The first allows recovery of the zero-element, the second is a refinement of a previously described vulnerability yielding linear equations, and the third enables to retrieve cubic equations.

Keywords: side channel attack, timing attack, post quantum cryptography, code-based cryptography.

1 Introduction

The McEliece PKC [1] and Niederreiter [2] Cryptosystems, built on error correcting codes, are considered immune to quantum computer attacks [3], and thus are of interest as candidates for future cryptosystems in high security applications. Accordingly, they have received growing interest from researchers in the past years and been analyzed with respect to efficiency on various platforms [4–8]. Furthermore, a growing number of works has investigated the side channel security of code-based cryptosystems [9–14].

Side channel security is a very important implementation aspect of any cryptographic algorithm. A side channel is given when a physical observable quantity that is measured during the operation of a cryptographic device, allows an attacker to gain information about a secret that is involved in the cryptographic operation. The usual observables used in this respect are the duration of the operation (timing attacks [15]), or the power consumption as a function over the time (power analysis attacks[16]).

So far, timing attacks against the decryption operation of the McEliece PKC targeting the plaintext have been developed [10, 12, 14]. In [11], a timing attack

* To the most part, this work was done in the author's private capacity, a part of the work was done at Cryptography and Computeralgebra, Department of Computer Science, Technische Universität Darmstadt, Germany

P. Gaborit (Ed.): PQCrypto 2013, LNCS 7932, pp. 217–230, 2013.

is proposed that targets the secret support that is part of the private key in code-based cryptosystems. From the time taken by the solving of the key equation the attacker learns linear equations about the support in this attack. But that work suffers from two major limitations: Neither is the information that is gained in itself sufficient for a practical attack, nor was the attack actually implemented.

This work extends on the analysis given in [11] in multiple ways: first of all, we find that a control flow ambiguity causing leakage in terms of the linear equations is manifest already in the syndrome inversion preceding the solving of the key equation in the decryption operation, and consequently the countermeasure proposed in that work is insufficient. We also show that there exists a timing side channel vulnerability in the syndrome inversion that allows the attacker to gain knowledge of the zero-element of the secret support. As an extension resp. generalization of the attack yielding linear equations, we derive a practical timing attack that lets the attacker gain cubic equations.

We then describe how to efficiently use these three vulnerabilities to build a practical attack that recovers the private key entirely. Lastly, we give results for practical executions of the timing attack on a personal computer.

2 Preliminaries

In this work, we give a brief description of the McEliece PKC, and stress those features of the decryption algorithm, that are necessary to understand the timing attack presented in this paper. A more detailed description and security considerations can be found e.g. in [17].

Goppa Codes. Goppa codes [18] are a class of linear error correcting codes. The McEliece PKC makes use of irreducible binary Goppa codes, so we will restrict ourselves to this subclass and to code lengths that are powers of two.

Definition 1. *Let the polynomial* $g(Y) = \sum_{i=0}^{t} g_i Y^i \in \mathbb{F}_{2^m}[Y]$ *be monic and irreducible over* $\mathbb{F}_{2^m}[Y]$*, and let* m*,* t *be positive integers. Then* $g(Y)$ *is called a* Goppa polynomial *(for an irreducible binary Goppa code).*

Then an irreducible binary Goppa code is defined as $\mathcal{C}(g(Y)) = \{\boldsymbol{c} \in \mathbb{F}_2^n | S_{\boldsymbol{c}}(Y) := \sum_{i=0}^{n-1} \frac{c_i}{Y-\alpha_i} = 0 \bmod g(Y)\}$*, where* $n = 2^m$*,* $S_{\boldsymbol{c}}(Y)$ *is the syndrome of* $\boldsymbol{c}$*,* $\Gamma = (\alpha_i | i = 0, \ldots, n-1)$*, the* support *of the code, where the* α_i *are pairwise distinct elements of* $\mathbb{F}_{2^m}$*, and* c_i *are the entries of the vector* $\boldsymbol{c}$*.*

The code defined in such way has length n, dimension $k \geq n - mt$ – however we restrict us to $k = n - mt$ in this work – and can correct up to t errors.

As for any linear error correcting code, for a Goppa code there exists a generator matrix $G \in \mathbb{F}_2^{k \times n}$ and a parity check matrix $H \in \mathbb{F}_2^{mt \times n}$ [19]. Given these matrices, a message $\boldsymbol{m} \in \mathbb{F}_2^k$ can be encoded into a codeword $\boldsymbol{c}$ of the code by computing $\boldsymbol{c} = \boldsymbol{m}G$, and the syndrome $\boldsymbol{s} \in \mathbb{F}_2^{mt}$ of a (potentially distorted) codeword can be computed as $\boldsymbol{s} = \boldsymbol{c}H^T$. Here, we do not give the formulas for the computation of these matrices as they are of no importance for the understanding of the attack developed in this work. The interested reader, however, is referred to [19].

Overview of the McEliece PKC. In this section we give a brief overview of the McEliece PKC. The McEliece *secret key* consists of the Goppa polynomial $g(Y)$ of degree t and the support $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$, i.e. a permutation of $\mathbb{F}_{2^m}$, together they define the secret code $\mathcal{C}$. The *public key* is given by the public $n \times k$ generator matrix $G_p = SG$ over $\mathbb{F}_2$, where G is a generator matrix of the secret code $\mathcal{C}$ and S is a non-singular $k \times k$ matrix over $\mathbb{F}_2$, the purpose of which is to bring G_p into reduced row echelon form, i.e. $G_p = [\mathbb{I}|G_2]$, which results in a more compact public key [4].

Note that in the original definition of the McEliece PKC [1], the support is chosen to be in lexicographical ordering, but instead the public key is chosen as $G_p = SGP$, where P is a random permuation matrix. These two descriptions are completely equivalent: P corresponds to the permutation that has to be applied to a lexicographical ordered support Γ to produce the randomized secret support as it is defined above. Accordingly, the attack described in this work that attacks the secret support can alternatively be seen as an attack against the secret permutation.

The *encryption* operation allows messages $\boldsymbol{m} \in \mathbb{F}_2^k$. A random vector $\boldsymbol{e} \in \mathbb{F}_2^n$ with hamming weight $\mathrm{wt}(\boldsymbol{e}) = t$ has to be created. Then the ciphertext is computed as $\boldsymbol{z} = \boldsymbol{m}G_p + \boldsymbol{e}$.

The Decryption is given in Algorithm 1. It makes use of the error correction algorithm, given by the Patterson Algorithm [20], shown in Algorithm 2. In Step 1 of this algorithm, the syndrome vector is computed by multiplying the ciphertext by the parity check matrix, and then turned into the syndrome polynomial $S(Y)$ by interpreting it as an $\mathbb{F}_{2^m}^t$ element and multiplying it with the vector of powers of Y. The Patterson Algorithm furthermore uses an algorithm for finding roots in polynomials over $\mathbb{F}_{2^m}$ (root_find()), and the Extended Euclidean Algorithm (EEA) for polynomials with a break condition based on the degree of the remainder, given in Algorithm 3. The root finding can e.g. be implemented as an exhaustive search on $\mathbb{F}_{2^m}$. Please note that all polynomials appearing in the algorithms have coefficients in $\mathbb{F}_{2^m}$.

The Niederreiter PKC [2] is a cryptosystem that is slightly different from the McEliece PKC, however there also an error vector is chosen during the encryption and decryption features the syndrome decoding. Since these features are, as we shall see, the preconditions for our attack, it is equally applicable to the Niederreiter PKC.

In the following, we turn to those details, that are relevant for the side channel issues we are going to address in Section 3. Please note that the error locator polynomial $\sigma(Y)$, which is determined in Step 4 of Algorithm 2, has the following form:

$$\sigma(Y) = \prod_{j \in \mathcal{E}} (Y - \alpha_j) = \sum_{i=0}^{t} \sigma_i Y^i. \tag{1}$$

where $\mathcal{E}$ is the set of those indexes i, for which $e_i = 1$, i.e. those elements of $\mathbb{F}_{2^m}$ that correspond to the error positions in the error vector. The determination of the error vector in Step 6 of Algorithm 2 makes use of this property. Accordingly, $\deg(\sigma(Y)) = \mathrm{wt}(\boldsymbol{e})$ if $\mathrm{wt}(\boldsymbol{e}) \leq t$ holds.

Algorithm 1. The McEliece Decryption Operation

Require: the McEliece ciphertext $\boldsymbol{z} \in \mathbb{F}_2^n$
Ensure: the message $\boldsymbol{m} \in \mathbb{F}_2^k$
1: $\boldsymbol{e} \leftarrow \text{err\_corr}(\boldsymbol{z}, g(Y))$
2: $\boldsymbol{m}' \leftarrow \boldsymbol{z} + \boldsymbol{e}$
3: $\boldsymbol{m} \leftarrow$the first k bits of $\boldsymbol{m}'$
4: return $\boldsymbol{m}$

Algorithm 2. The McEliece error correction with the Patterson Algorithm (err_corr$(\boldsymbol{z}, g(Y))$)

Require: the distorted code word $\boldsymbol{z} \in \mathbb{F}_2^n$, the secret Goppa polynomial $g(Y)$ and secret support $\Gamma = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$
Ensure: the error vector $\boldsymbol{e} \in \mathbb{F}_2^n$
1: $S(Y) \leftarrow \boldsymbol{z} H^\top \left(Y^{t-1}, \cdots, Y, 1\right)^\top$
2: $\tau(Y) \leftarrow \sqrt{S^{-1}(Y) + Y} \bmod g(Y)$
3: $(a(Y), b(Y)) \leftarrow \text{EEA}\left(\tau(Y), g(Y), \lfloor \frac{t}{2} \rfloor\right)$
4: $\sigma(Y) \leftarrow a^2(Y) + Y b^2(Y)$
5: $\mathcal{E} = \{E_0, \ldots, E_{t-1}\} \leftarrow \text{rootfind}(\sigma(Y))$ // if α_i is a root, then $\mathcal{E}$ contains i
6: $\boldsymbol{e} \leftarrow \boldsymbol{v} \in \mathbb{F}_2^n$ with $v_i = 1$ if and only if $i \in \mathcal{E}$
7: **return** $\boldsymbol{e}$

3 Analysis of Timing Side Channels in the Syndrome Inversion

We now explain three different vulnerabilities present in the syndrome decoding. To this end, we first explore certain properties of the syndrome inversion by the use of the EEA during the code-based decryption operation.

3.1 Properties of the Syndrome Inversion

The syndrome polynomial is defined as

$$S(Y) \equiv \sum_{i=1}^{w} \frac{1}{Y \oplus \epsilon_i} \equiv \frac{\Omega(Y)}{\sigma(Y)} \bmod g(Y) \tag{2}$$

Here, w is the Hamming weight of the error vector $\boldsymbol{e}$ and the $\{\epsilon_i | i \in \{1, \ldots w\}\}$ denote the support elements associated with the indexes of those bits in the error vector having value one in arbitrary ordering, i.e., for instance, if the bits found at the index j and k in the error vector have value one, then $\epsilon_1 = \alpha_j$, $\epsilon_2 = \alpha_k$ and so on. The identification of the error locator polynomial $\sigma(Y)$ in the denominator is simply a result of the form of the common denominator of all sum terms. In the McEliece PKC Decryption, during the error correction, Alg. 2, Step 2, $S^{-1}(Y)$ is computed by invoking Alg. 3 as EEA$(g(Y), S(Y), 0)$. But it is known that in case of $w \leq t/2$ instead it is possible to find $\sigma(Y)$ already at this

Algorithm 3. The Extended Euclidean Algorithm (EEA($r_{-1}(Y)$, $r_0(Y)$, d))

Require: the polynomials $r_{-1}(Y)$ and $r_0(Y)$, with $\deg(r_0(Y)) < \deg(r_{-1}(Y))$
Ensure: two polynomials $r_M(Y)$, $b_M(Y)$ satisfying $r_M(Y) = b_M(Y)r_0(Y) \bmod r_{-1}(Y)$ and $\deg(r_0(Y)) \leq \lfloor \deg(r_{-1})/2 \rfloor$

1: $b_{-1} \leftarrow 0$
2: $b_0 \leftarrow 1$
3: $i \leftarrow 0$
4: **while** $\deg(r_i(Y)) > d$ **do**
5: $\quad i \leftarrow i + 1$
6: $\quad (q_i(Y), r_i(Y)) \leftarrow r_{i-2}(Y)/r_{i-1}(Y)$ // polynomial division with quotient q_i and remainder r_i
7: $\quad b_i(Y) \leftarrow b_{i-2}(Y) + q_i(Y)b_{i-1}(Y)$
8: **end while**
9: $M \leftarrow i$
10: **return** $(r_M(Y), b_M(Y))$

stage by invoking Alg. 3 as EEA$(g(Y), S(Y), \lfloor t/2 \rfloor - 1)$, i.e. with $r_{-1}(Y) = g(Y)$ and $r_0(Y) = S(Y)$ and breaking once $\deg(r_i(Y)) \leq (t/2) - 1$. Then, it returns $\delta\sigma(Y) = b_M(Y)$ and furthermore $\delta\Omega(Y) = r_M(Y)$, where $\delta \in \mathbb{F}_{2^m}$ and M is the number of iterations performed by the EEA [21].

Given this form of the $S(Y)$, we can make a statement about the maximally possible number of iterations in the EEA used to compute $S^{-1}(Y) \equiv \sigma(Y)/\Omega(Y) \bmod g(Y)$. As already mentioned, the actual invocation of the syndrome inversion is EEA$(g(Y), S(Y), 0)$. But the above explained fact that we could stop at $\deg(r_i(Y)) \leq (t/2) - 1$ means that there is one iteration in the EEA where $r_i(Y) = \delta\Omega(Y)$ and $b_i(Y) = \delta\sigma(Y)$, in case of $w \leq (t/2) - 1$.

Theorem 1. *Assume a Goppa Code defined by $g(Y)$ and Γ. When Alg. 3 is invoked as* EEA$(g(Y), S(Y), 1)$ *with* $S(Y) \equiv \frac{\Omega(Y)}{\sigma(Y)} \bmod g(Y)$*, and the error vector $\boldsymbol{e}$ corresponding to $S(Y)$ satisfies* $\mathrm{wt}(\boldsymbol{e}) \leq (\deg(g(Y))/2) - 1$*, then for the number of iterations in Alg. 3 we find:*

$$M \leq M_{\max} = \deg(\Omega(Y)) + \deg(\sigma(Y))$$

Proof. Regard the iteration where $r_j(Y) = \delta\Omega(Y)$ and $b_j(Y) = \delta\sigma(Y)$. Since according to Alg. 3 the degree of $b_j(Y)$, starting from zero, increases at least by one in each iteration, we find $j \leq \deg(\sigma(Y))$. From here on, the degree of $r_j(Y) = \delta\Omega(Y)$ is decreased by at least one in each subsequent iteration down to $\deg(r_M(Y)) = 0$, i.e. $M - j \leq \deg(\Omega(Y))$, giving $M = M - j + j \leq \deg(\Omega(Y)) + \deg(\sigma(Y))$.

Because in the following we are only interested in the derivation of equations of the form $\sigma_i = 0$ for a specific value of i, we will ignore the constant δ from here on.

3.2 Linear Equations from $w = 4$ Error Vectors

We now investigate the effect of the above results for the case where ciphertexts created with error vectors of Hamming weight four are input to the decryption operation.

In the case of $w = 4$ the syndrome polynomial is of the form:

$$S(Y) \equiv \frac{\Omega(Y)}{\sigma(Y)} \equiv \sum_{i=1}^{4} \frac{1}{Y \oplus \epsilon_i} \equiv \frac{\sigma_3 Y^2 \oplus \sigma_1}{Y^4 \oplus \sigma_3 Y^3 \oplus \sigma_2 Y^2 \oplus \sigma_1 Y \oplus \sigma_0} \bmod g(Y), \quad (3)$$

where $\epsilon_i \in \mathbb{F}_{2^m}, i \in 1, \ldots, 4$ denote the four elements of the support associated with the error positions. Furthermore, in the right hand side of Eq. (3), which is found by bringing all four sum terms to their common denominator, we have

$$\sigma_3 = \epsilon_1 \oplus \epsilon_2 \oplus \epsilon_3 \oplus \epsilon_4.$$

With the aim of finding a timing vulnerability revealing certain coefficients of $\sigma(Y)$ and thus information about the secret support, we now analyze the connection between the number of iterations and their complexity on the one hand and the degree of $\Omega(Y)$ on the other. Regarding $\Omega(Y)$ for the case $w = 4$ we find that the coefficient to the highest power of Y is given by $\sigma_3 = \epsilon_1 \oplus \epsilon_2 \oplus \epsilon_3 \oplus \epsilon_4$. If $\sigma_3 = 0$, then the degree of $\Omega(Y)$ is zero, otherwise it is two. This means that in the case of $\sigma_3 = 0$ the maximal number of iterations in the inversion is four, in contrast to six in the general case. Table 1 gives an overview of the individual iterations in the syndrome inversion EEA when $w = 4$, where it is assumed that for each iteration $\deg(q_i(Y)) = 1$, i.e. the case where the maximal number of iterations $M_{\max}$is executed. In the majority of the cases $M_{\max}$ iterations occur, i.e. six when $\deg(\Omega(Y)) = 2$ and four when $\deg(\Omega(Y)) = 0$. But with probability about $1/n$ in each iteration a larger degree of the quotient polynomial $q_i(Y)$ occurs, accordingly then $M < M_{\max}$. With the aim of assessing the reliability of the differences in running time allowing to identify the case $\deg(\Omega(Y)) = 0$, we examine whether $M < M_{\max}$ might lead to timings for $\deg(\Omega(Y)) = 2$ as low as for $\deg(\Omega(Y)) = 0$. We immediately find that the fifth iteration, which is only executed in the case $\sigma_3 = 0$, features a much more complex multiplication $q_5(Y)b_4(Y)$ than all the other iterations.

The control flow for the second EEA invocation, i.e. the solving of the key equation, for the case $w = 4$ has been analyzed in [11], there it is shown that in the case of $\sigma_3 = 0$ the number of iterations N is zero, whereas in the case $\sigma_3 \neq 0$ it is one. In that work, a countermeasure is proposed that removes the possibility to exploit the according timing differences in the second EEA invocation. However, due the fact that, as shown above, timing differences reveal $\sigma_3 = 0$ already in the syndrome inversion EEA, the countermeasure proposed in [11] is insufficient.

Experimental results confirm that taken together, the timing differences emerging in both EEA applications, i.e. the syndrome inversion and the key equation solving, actually allow for reliable distinction of $\deg(\Omega(Y))$ being zero or non-zero, and thus the attacker is able to learn linear equations of the form $\sigma_3 = \sum_{i=1}^{4} \epsilon_i = 0$. Remember that through the choice of the error vector during

encryption, he chooses the indexes j_i with $i = 1, \ldots, 4$ of the support elements $\alpha_{j_i} = \epsilon_i$ according to the definition of the ϵ_i notation for the support elements.

Table 1. Overview of the iterations in the syndrome inversion EEA for Hamming weight four error vectors. If $\deg(\Omega(Y)) = 2$, $M_{\max}$, the maximal number of iterations is six, otherwise, if $\deg(\Omega(Y)) = 0$, we have $M_{\max} = 4$.

i	$\deg(q_i(Y))$	$\deg(b_i(Y))$	$\deg(r_i(Y))$
1	1	1	t-1
2	1	2	t-2
3	1	3	t-3
4	1	**4**	**2 (or 0)**
5	t - 5	t - 1	1
6	1	t	0

3.3 Cubic Equations from $w = 6$ Error Vectors

The vulnerability found for $w = 4$ error vectors can be generalized to any even value of w. For the attack that is subject of this work, we also employ the case $w = 6$. There, we find that the syndrome polynomial according to Eq. (2) is of the form

$$S(Y) \equiv \frac{\Omega(Y)}{\sigma(Y)} \equiv \frac{\sigma_5 Y^4 \oplus \sigma_3 Y^2 \oplus \sigma_1}{Y^6 \oplus \sigma_5 Y^5 \oplus \sigma_4 Y^4 \oplus \sigma_3 Y^3 \oplus \sigma_2 Y^2 \oplus \sigma_1 Y + \sigma_0} \bmod g(Y), \tag{4}$$

where

$$\sigma_3 = \sum_{j=3}^{6} \sum_{k=1}^{j-1} \sum_{l=1}^{k-1} \epsilon_j \epsilon_k \epsilon_l, \tag{5}$$

$$\sigma_5 = \sum_{i=1}^{6} \epsilon_i. \tag{6}$$

As in case of $w = 4$, $\deg(\Omega(Y)) = 0$ implies zero iterations in the key equation EEA. Furthermore, it is again the most complex iteration of the syndrome inversion EEA that is skipped if $\deg(\Omega(Y)) = 0$. The difference to $w = 4$ is that here two coefficients of $\sigma(Y)$, i.e σ_3 and σ_5, have to be zero for this to happen.

Thus from detecting $\deg(\Omega(Y)) = 0$ the attacker can learn the equations $\sigma_3 = 0$ and $\sigma_5 = 0$. However, since from the vulnerability presented in Sec. 3.2 it is already possible for the attacker to learn linear equations about the secret support, the value of the "$w = 6$" vulnerability lies in the equation $\sigma_3 = 0$, which can be learned through a timing side channel analogously to the case "$w = 4$".

3.4 The Zero Element of the Support from $w = 1$ Error Vectors

For $w = 1$ the whole control flow in Patterson's Algorithm is very simple and unambiguous on a high level: $S(Y) \equiv \frac{1}{Y \oplus \epsilon_1} \bmod g(Y)$, $S^{-1}(Y) = Y \oplus \epsilon_1$,

Algorithm 4. Polynomial Division poly_div$(n(Y), d(Y))$

Require: the polynomials $n(Y), d(Y)$ with $\deg(n(Y)) \geq \deg(d(Y))$
Ensure: two polynomials $s(Y)$, $q(Y)$ with $q(Y)d(Y) + s(Y) = n(Y)$ and $\deg(s(Y)) < \deg(d(Y))$

```
s_{-1}(Y) ← n(Y)
s_0(Y) ← d(Y)
q_0(Y) ← 0
i ← 0
while deg(s_i(Y)) ≥ deg(d(Y)) do
    i ← i + 1
    a_i ← s_{i-2,deg(s_{i-2}(Y))} / s_{i-1,deg(s_{i-1}(Y))}
    f_i ← deg(s_{i-2}(Y)) − deg(s_{i-1}(Y))
    q_i(Y) = q_{i-1} + a_i Y^{f_i}
    s_i ← s_{i-2}(Y) − a_i s_{r-1}(Y) Y^{f_i}
end while
return (q_i(Y), s_i(Y))
```

$\tau(Y) = \sqrt{\epsilon_1}$, $a(Y) = \tau(Y)$, $b(Y) = 1$, $\sigma(Y) = Y \oplus \epsilon_1$. The polynomial inversion is, according to Theorem 1, performed in exactly one iteration. But there is an ambiguous control flow within the polynomial division given in Alg. 4, that is executed within this EEA iteration: We find $q_1(Y) = Y$ because there is no alternative to $\deg(S(Y)) = t - 1$. In Alg. 4, $s_{i,j}$ denotes the coefficient to Y^j in $s_i(Y)$. If $\epsilon_1 = 0$, then the division has to stop at this point. Otherwise, a second iteration is performed giving $q_2(Y) = Y \oplus \epsilon_1$. Thus, if the timing difference resulting from the different number of iterations in the division is detectable, the index of z of the secret support element $\alpha_z = 0$ can be found.

4 Combining the "$w = 1$", "$w = 4$", and "$w = 6$" Vulnerabilities to a practical Attack

In this section we explain the construction of a practical attack based on the vulnerabilities shown in Sections 3.2, 3.3 and 3.4.

4.1 Description of the Attack Procedure

Step 1. By performing the respective queries on the decryption device with "$w = 4$" error vectors, a rank $n - m - 1$ linear equation system is build. The experimental results from [11] already showed that this is the maximal rank that can be achieved from the linear equations. Afterwards, the index of the zero element, α_z is determined through the "$w = 1$" vulnerability. In the majority of the cases, this information increases the rank of the equation system to $n - m$. In the rare cases when the rank remains at $n - m - 1$, the attack's on-line and off-line complexity is increased by a factor of n.

In the following, we assume that we have an equation system of rank $n - m$. This is the highest possible rank for a homogeneous linear equation system

describing a permutation of $\mathbb{F}_{2^m}$, since there must be m linearly independent basis elements. Accordingly, by bringing the linear equation system into reduced row echelon form, we find the elements associated with the m rightmost columns must be a basis $\{\beta_i\}$:

$$\begin{array}{cccccccc|ccc}
\alpha_0 & \alpha_1 & \dots & \alpha_i & \dots & \alpha_{n-m-3} & \alpha_{n-m-2} & \beta_0 & \dots & \beta_{m-1} \\
\hline
1 & 0 & \dots & 0 & \dots & 0 & 0 & X & \dots & X \\
\vdots & & & & & & & & & \\
0 & 0 & \dots & 1 & \dots & 0 & 0 & X & \dots & X \\
\vdots & & & & & & & & & \\
0 & 0 & \dots & 0 & \dots & 0 & 1 & X & \dots & X
\end{array}$$

Step 2. At this point for each element α_i we know the corresponding B_i with $\alpha_i = \sum_{j \in B_j} \beta_i$, i.e. its representation in the chosen basis. If the values of all basis elements β_i were known, then the values of all α_i would be set as well and the support was recovered. Accordingly, the next step in the attack is to collect cubic equations according to Eq. (5) in a way that allows for efficient guessing resp. solving for the values of the β_i. To this end, the first set C_1 of "$w = 6$" equations is created by the employment of error vectors involving error positions corresponding to ϵ_i, $i = 1, \dots, 6$, where the following conditions hold:

1. $\epsilon_i \in \mathrm{span}(\{\beta_{\mathrm{s}_1}, \beta_{\mathrm{g}_1}, \beta_{\mathrm{g}_2}, \beta_{\mathrm{g}_3}\})$. These are four arbitrarily chosen basis elements, where β_{s_1} denotes the one to be solved for in the resulting equation according to Eq. (5). The reason for this initial set of basis elements having cardinality four is that this is the lowest cardinality allowing to satisfy all the conditions in the following items.
2. $\sum_{i=1}^{6} \epsilon_i = 0$. This qualifies the error vector for the possibility of $\deg(\Omega(Y)) = N = 0$ according to Eq. (6) in the sense that $\sigma_5 = 0$ is already ensured. As a result, in contrast to the case of random $w = 6$ error vectors that have a probability for $\deg(\Omega(Y)) = N = 0$ in the domain of $1/n^2$, for these candidates this probability is about $1/n$.
3. Exactly two of the ϵ_i contain β_{s_1}. The reason for this constraint is to keep the process of solving, the details of which we shall see shortly, as simple as possible. Specifically, the twofold occurrence of β_{s_i} leads to a quadratic equation for β_{s_i}.

Candidate error vectors $\boldsymbol{e}$ that meet these conditions are used to build ciphertexts which are input to the decryption device; and from the timing of the decryption, it is inferred whether actually $\deg(\Omega(Y)) = N = 0$ occurred. The number of such equations to be collected for one β_{s_i} is given by c_i, which is a parameter for the attack.

After c_1 equations are found for β_{s_1}, the second set of equations is build in the same way as the first, the only differences being that the basis element to be solved for now is $\beta_{s_2} \notin \{\beta_{s_1}, \beta_{g_1}, \beta_{g_2}, \beta_{g_3}\}$, and first condition becomes $\epsilon_i \in \mathrm{span}(\{\beta_{\mathrm{s}_1}, \beta_{\mathrm{g}_1}, \beta_{\mathrm{g}_2}, \beta_{\mathrm{g}_3}, \beta_{\mathrm{s}_2}\})$. In this manner successively sets of cubic equations for $m-3$ different β_{s_i} are collected until the equations in the last set involve all β_i.

Step 3. In this step the solving resp. guessing is performed. Let those two ϵ_i that contain β_{s_i} according to the third condition in Step 4 always be ϵ_1 and ϵ_2. From the conditions given in Step 4, and Eq. (5) we have for each β_{s_i} a quadratic equation

$$a\beta_{s_i}^2 + b\beta_{s_i} + c = 0, \tag{7}$$

with $a = \sum_{j=3}^{6} \epsilon_i$, $b = (\epsilon_1+\epsilon_2)a$, $c = (\epsilon_1-\beta_{s_i})(\epsilon_2-\beta_{s_i})a+(\epsilon_1+\epsilon_2)\sum_{j=4}^{6}\sum_{k=3}^{j-1}\epsilon_j\epsilon_k$ $+\sum_{j=5}^{6}\sum_{k=4}^{j-1}\sum_{l=3}^{k-1}\epsilon_j\epsilon_k\epsilon_l$ (for clarity, in these formulas we provide "+" and "−" even though both amount to "⊕"). Such a quadratic equation has two solutions for β_{s_i}.

The solving is performed as follows: enumerate the initial guesses, i.e. all the possible combinations of the values for $\beta_{g_1}, \beta_{g_2}, \beta_{g_3}$. Here, and for the subsequent guesses, since we are looking for linearly independent $\mathbb{F}_{2^m}$ elements, it holds that

$$\beta_{g_i} \notin \mathrm{span}(\{\beta_{\mathrm{g_1}}, \ldots, \beta_{\mathrm{g_{i-1}}}\}), \tag{8}$$

where we imply the convention $\beta_{s_i} = \beta_{g_{i+3}}$.

For each such combination of values for $\beta_{g_1}, \beta_{g_2}, \beta_{g_3}$ the roots of each equation in C_1 are potential candidates for the value of β_{s_1}. However, additionally to the restriction from Eq. (8), those roots that are found only for a subset of C_1 are discarded. This is wherein the value of a choice $c_i = |C_i| > 1$ lies. The larger the c_i are chosen, the higher is the on-line effort of the attack (more cubic equations have to be collected), but the off-line effort is reduced as the number possible solutions for each β_{s_i} is decreased.

The remaining roots are iterated over to find the possible solutions for β_{s_2} by solving the equations in C_2, which in turn are used to compute the possible values of β_{s_3}, etc. Whenever in such a chain of guesses a solution for all β_{s_i} is found, a guess for the whole support $\Gamma = (\alpha_i | \alpha_i = \sum_{j\in B_i} \beta_j)$ is implied, which has to be checked by a means of key recovery, as described in [13].

4.2 Experimental Results

We conducted the attack with the following measurement setup on an Intel Core 2 Duo x86 platform: from the attack program, the decryption function was called with the attack ciphertexts as input, and the decryption time was measured with the CPU's cycle counter.

Because the cycle counts measured for a deterministic operation of the duration of a code-based decryption vary considerably on such CPUs, a specific strategy has to be used to identify positives, by which we refer to $\epsilon_1 = 0$ for $w = 1$ and $\deg(\Omega(Y)) = 0$ for $w = 4$ and $w = 6$, i.e. those cases that yield an equation for the attacker. Specifically, an approximate model for cycle counts on modern x86 CPUs like the Core2 Duo is a hypothetical constant cycle count associated with the operation which is increased by a random delay on every execution of an operation. Because in all three different attack types the positives, from the algorithmic point of view, are executed faster than the negatives, the following classification strategy can be used: Prior to the attack a training

phase is carried out where the minimal cycle counts for positives are determined as well as the minimal cycle counts for negatives (using a different secret key than during the attack). Then the border below which an operation is classified as a positive during the attack is set as the mean of these two values. We refer to the distance between the minimal cycle counts for positives and the minimal cycle counts for negatives as the cycles gap. Clearly, a larger such gap increases the probability for finding positives. Furthermore, the above approximate model for the cycle counts on the employed CPU is lacking other effects that could be observed in our experiments: during the execution of the attack the previously determined maximal and minimal cycle counts for the two classes of operations seem to be subject to an "upwards drift", i.e. they tend to successively increase over time but sometimes also drop again approximately to the initial levels after some time.

Table 2 summarizes the results for single attack runs with different code parameters. The rows labeled "cycles gap ..." indicate the above discussed gaps. We found that gaps of a couple of hundreds cycles that are characteristic for the $w = 1$ vulnerability tend to cause problems in the detection of positives, i.e. in some runs due to the mentioned drift of the cycle counts the zero support element could not be determined, while the considerably larger gaps for the $w = 4$ and $w = 6$ vulnerabilities allow for reliable detection of positives.

The rows labeled "number of queries ..." show the number of decryption operations that had to be executed with ciphertexts created with error vectors of the respective weight in the course of a single run of the attack.

"number of final verifications" is the number of the guesses for the complete support that are output by the attack. We did not implement an actual verification, but simply compare the guess for the support with the correct support Γ. As already mentioned, in [13] the procedure that had to be used in a real life attack is described. It involves only some linear algebra operations on the public key and the invocation of an EEA and would not perceptibly increase the time for solving, given the small numbers of such final verifications occurring in the attacks.

The time for the solving step is given in the last row. From the theory, one expects an increase of the solving time by a factor of about eight for each increase of m by one. The reason is that the number of initial guesses, i.e. the number of combinations of values that can be chosen for β_{g_1}, β_{g_2} and β_{g_3} is roughly n^3, and all $\mathbb{F}_{2^m}$ operations, including the solving of the quadratic equations [22] Eq. (7), are done with the help of lookup tables, and thus execute in constant time.

The number of equations gathered per β_{s_i} were chosen as $C_1 = 1$, $C_2 = 2$, $C_3 = 4$, i.e. chosen as the double of the previous count, up to a maximal value of 16, i.e. $C_i = 16$ for $i \geq 5$.

As previously mentioned, in the rare cases where the knowledge about the zero-element of the support does not increase the rank of the equation system, Steps 2 and 3 would have to be repeated about up to n times, for these cases stronger hardware would be needed to keep the solving time in reasonable margins.

Table 2. Experimental results for single runs of the attack. Refer to the text for explanations

	$m = 9,\ t = 33$	$m = 10,\ t = 40$
cycles gap $w = 1$	≈ 400	≈ 600
cycles gap $w = 4$	$\approx 13,000$	$\approx 19,000$
cycles gap $w = 6$	$\approx 17,000$	$\approx 23,000$
number of queries for $w = 1$ (Step 1)	3,575,494	11,782,695
number of queries for $w = 4$ (Step 1)	1,517,253	2,869,424
number of queries for $w = 6$ (Step 2)	374,927	1,837,125
number of final verifications (Step 3)	$\approx 8,000$	$\approx 2,000$
running time for solving on 1 GHz x86 CPU (Step 3)	3h	28h

5 Conclusion

The results of this work show that timing attacks based on control flow vulnerabilities in the syndrome inversion and the key equation EEA are a threat to the confidentiality of the secret key. In the chosen measurement setup, the attack has been proved to be practical. Apart from the recovery of the zero-element of the support, the cycles gaps between the controls flows that have to be distinguished are rather large, and thus remote timing attacks seem feasible too. If the zero-element remains unknown, the on-line and off-line attack complexity can still be managed with appropriate hardware.

The question of countermeasures against this attack has not been explicitly addressed in this work, but two possibilities seem to suggest themselves: the first would be similar to the countermeasures given in [12], where "premature" abortion of the key equation solving EEA is prevented by enforcing the "missing" iterations. This however is a delicate undertaking, as even the smallest timing differences have to be prohibited and thus the complexity of the individual iterations must be accounted for (consider for instance the "$w = 1$ attacks" from Section 3.4).

The second option would be stronger in this respect: there, we alter the cryptosystem's parameter specification: during the encryption, only $t - 1$ errors are added, and prior to the standard decryption operation, another "bit flip error" is applied, the position of which should be the same for repeated decryptions of a certain ciphertext but otherwise appear as random, and thus should be pseudorandomly derived from the ciphertext and a constant secret value (for instance a hash value of the secret key). This approach would guarantee a pervasive alteration of the decryption operation, however it demands an increase of security parameters to compensate for the lower error weight used during encryption.

References

1. McEliece, R.J.: A public key cryptosystem based on algebraic coding theory. DSN Progress Report 42-44, 114–116 (1978)
2. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Problems Control Inform. Theory 15(2), 159–166 (1986)

3. Bernstein, D.J., Buchmann, J., Dahmen, E.: Post Quantum Cryptography. Springer Publishing Company, Incorporated (2008)
4. Biswas, B., Sendrier, N.: McEliece Cryptosystem Implementation: Theory and Practice. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 47–62. Springer, Heidelberg (2008)
5. Heyse, S.: Low-Reiter: Niederreiter Encryption Scheme for Embedded Microcontrollers. In: Sendrier, N. (ed.) PQCrypto 2010. LNCS, vol. 6061, pp. 165–181. Springer, Heidelberg (2010)
6. Eisenbarth, T., Güneysu, T., Heyse, S., Paar, C.: MicroEliece: McEliece for Embedded Devices. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 49–64. Springer, Heidelberg (2009)
7. Shoufan, A., Wink, T., Molter, G., Huss, S., Strenzke, F.: A Novel Processor Architecture for McEliece Cryptosystem and FPGA Platforms. In: Proceedings of the 2009 20th IEEE International Conference on Application-Specific Systems, Architectures and Processors, ASAP 2009, pp. 98–105. IEEE Computer Society, Washington, DC (2009)
8. Strenzke, F.: A Smart Card Implementation of the McEliece PKC. In: Samarati, P., Tunstall, M., Posegga, J., Markantonakis, K., Sauveron, D. (eds.) WISTP 2010. LNCS, vol. 6033, pp. 47–59. Springer, Heidelberg (2010)
9. Molter, H.G., Stöttinger, M., Shoufan, A., Strenzke, F.: A Simple Power Analysis Attack on a McEliece Cryptoprocessor. Journal of Cryptographic Engineering (2011)
10. Strenzke, F., Tews, E., Molter, H., Overbeck, R., Shoufan, A.: Side Channels in the McEliece PKC. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 216–229. Springer, Heidelberg (2008)
11. Strenzke, F.: A Timing Attack against the Secret Permutation in the McEliece PKC. In: Sendrier, N. (ed.) PQCrypto 2010. LNCS, vol. 6061, pp. 95–107. Springer, Heidelberg (2010)
12. Shoufan, A., Strenzke, F., Molter, H., Stöttinger, M.: A Timing Attack against Patterson Algorithm in the McEliece PKC. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 161–175. Springer, Heidelberg (2010)
13. Heyse, S., Moradi, A., Paar, C.: Practical power analysis attacks on software implementations of mceliece. In: Sendrier, N. (ed.) PQCrypto 2010. LNCS, vol. 6061, pp. 108–125. Springer, Heidelberg (2010)
14. Avanzi, R., Hoerder, S., Page, D., Tunstall, M.: Side-channel attacks on the mceliece and niederreiter public-key cryptosystems. J. Cryptographic Engineering 1(4), 271–281 (2011)
15. Kocher, P.C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
16. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
17. Engelbert, D., Overbeck, R., Schmidt, A.: A Summary of McEliece-Type Cryptosystems and their Security. Journal of Mathematical Cryptology 1(2), 151–199 (2006)
18. Goppa, V.D.: A new class of linear correcting codes. Problems of Information Transmission 6, 207–212 (1970)

19. MacWilliams, F.J., Sloane, N.J.A.: The theory of error correcting codes. North Holland (1997)
20. Patterson, N.: Algebraic decoding of Goppa codes. IEEE Trans. Info. Theory 21, 203–207 (1975)
21. Sugiyama, Y., Kasahara, M., Hirasawa, S., Namekawa, T.: A method for solving key equation for decoding goppa codes. Information and Control 27(1), 87–99 (1975)
22. Biswas, B., Herbert, V.: Efficient Root Finding of Polynomials over Fields of Characteristic 2. In: WEWoRK (2009), hal.inria.fr/hal-00626997/PDF/tbz.pdf

Simple Matrix Scheme for Encryption

Chengdong Tao[1], Adama Diene[2], Shaohua Tang[3], and Jintai Ding[4,⋆]

[1] South China University of Technology, China
chengdongtao2010@gmail.com
[2] Department of Math. Sciences, UAE University - Al-Ain, United Arab Emirates
adiene@uaeu.ac.ae
[3] South China University of Technology, China
csshtang@gmail.com
[4] University of Cincinnati, Ohio, USA and ChongQing University,China
jintai.ding@gmail.com

Abstract. There are several attempts to build asymmetric pubic key encryption schemes based on multivariate polynomials of degree two over a finite field. However, most of them are insecure. The common defect in many of them comes from the fact that certain quadratic forms associated with their central maps have low rank, which makes them vulnerable to the MinRank attack. We propose a new simple and efficient multivariate pubic key encryption scheme based on matrix multiplication, which does not have such a low rank property. The new scheme will be called Simple Matrix Scheme or ABC in short. We also propose some parameters for practical and secure implementation.

Keywords: Multivariate Public Key Cryptosystem, Simple Matrix Scheme, MinRank Attack.

1 Introduction

Public key cryptography plays an important role in secure communication. The most widely used nowadays are the number theoretical based cryptosystems such as RSA, DSA, and ECC. However, due to Shor's Algorithm, such cryptosystems would become insecure if a large Quantum computer is built. Recent progress made in this area makes this threat realer than ever before. Moreover, the computing capacity of these Number Theoretic based systems is proved to be limited. These are some reasons which motivate researchers to develop a new family of cryptosystems that can resist quantum computers attacks and that are more efficient in terms of computation. Researchers usually use Post Quantum Cryptography (PQC) to denote this new family.

Multivariate public key cryptosystems (MPKC) belong to the PQC family. If well designed, they can be a good candidate for PQC. The public key of an MPKC is a system of multivariate polynomials, usually quadratic, over a finite field. The security of MPKCs is based on the knowledge that solving a set of

⋆ Corresponding author.

P. Gaborit (Ed.): PQCrypto 2013, LNCS 7932, pp. 231–242, 2013.

multivariate polynomial equations over a finite field, in general, is proven to be an NP-hard problem [9]. In fact quantum computers do not appear to have an advantage when dealing with NP-hard problems. However, this does not guarantee that these cryptosystems are secure. The first such practical system was proposed in 1988 by Matsumoto and Imai with their scheme called C* or MI. Nonetheless, Jacques Patarin proved it insecure using linearization equations attack a few years later [18].

In [5], the authors showed that the rank of the quadratic form associated to the central map of C* is only two and therefore the private key could be also recovered with the help of the MinRank Attack.

In [19] Patarin extended the C* scheme by using a new central map to construct a new encryption scheme called Hidden Field Equations (HFE). But Kipnis and Shamir found a way to recover the private keys using the MinRank Attack [13]. Furthermore, it is showed in [8] that inverting HFE is quasi-polynomial if the size of the field and the degree of the HFE polynomials are fixed.

In [15], T.T. Moh proposed a multivariate asymmetric encryption scheme called TTM.

But again, it was broken by exploiting the fact that some quadratic form associated to the central map is of low rank [3].

In the last two decades, many other MPKCs have been proposed for encryption but almost all of them are proven to be insecure and many of them share a common defect; that is some quadratic forms associated to their central maps have low rank and therefore are vulnerable to the MinRank Attack. In consequence, for a MPKC to be secure, it is necessary that all quadratic forms associated with the central map have a rank high enough.

This paper will propose a new multivariate public key scheme for encryption having the property that the quadratic forms associated to the central map do not have a low rank but a rank related to a certain parameter n. The scheme is constructed using some simple matrix multiplications and it will be called Simple Matrix encryption scheme or ABC in short.

This paper is organized as follows. In Section 2 we give an illustration of the MinRank attack using HFE. In Section 3, we describe the construction of the ABC scheme. The security analysis is presented in Section 4. Section 5 shows a practical implementation of the ABC scheme while Section 6 discusses the efficiency and Section 7 concludes the paper.

2 MinRank Attack

The MinRank attack is a cryptanalysis tool that can be used to recover the secret key of MPKCs whose quadratic form associated to the central map is of low rank. In this section, we give an illustration by describing the MinRank attack on the HFE scheme. The attack was first performed by Kipnis and Shamir [13] who showed that the security of HFE can be reduced to a MinRank problem.

2.1 The HFE Scheme

The HFE cryptosystem was proposed by Jacques Patarin in [19]. It can be described as follow. Let $q = p^e$, where p is a prime and $e \geq 1$. Let K be an extension of the finite field $k = \mathbb{F}_q$ of degree n. Clearly, $K \cong k^n$.

Let $\phi : K \to k^n$ be the k-linear isomorphism map between the finite field K and the n-dimensional vector space k^n. The central map of HFE is a univariate polynomial $P(x)$ of the following form

$$P(x) = \sum_{i=0}^{r-1} \sum_{j=0}^{r-1} p_{ij} x^{q^i+q^j} \in K[x],$$

where $p_{ij} \in K$ and r is a small constant chosen in a way such that $P(x)$ can efficiently inverted. The public key is given to be

$$\bar{F} = T \circ \phi \circ P \circ \phi^{-1} \circ S,$$

where $T : k^n \longrightarrow k^n$ and $S : k^n \longrightarrow k^n$ are two invertible linear transformations and the private key consist of T, P and S.

2.2 MinRank Attack on HFE

In [14], Kipnis and Shamir showed that the public key $\bar{F}$ and the transformations S, T, T^{-1} can be viewed as maps G^*, S^*, T^*, T^{*-1} over K. More precisely,

$$S^*(x) = \sum_{i=0}^{n-1} s_i x^{q^i}, \qquad T^{*-1}(x) = \sum_{i=0}^{n-1} t_i x^{q^i}.$$

and $G^*(x) = T^*(P(S^*(x)))$. We can express $G^*(x)$ in the form:

$$G^*(x) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} g_{ij} x^{q^i+q^j} = \underline{x} G \underline{x}^t,$$

where $\underline{x} = (x, x^q, \ldots, x^{q^{n-1}})$ is a vector over K, $\underline{x}^t$ is the transposition of $\underline{x}$ and $G = [g_{ij}]$ is a matrix over K. The identity $T^{*-1}(G^*(x)) = P(S^*(x)$ implies that

$$G' = \sum_{i=0}^{n-1} t_k G^{*k} = W P W^t,$$

where $P = [p_{ij}]$ over K, G^{*k} and W are two matrices over K whose repective (i, j) entries are $g_{i-k,j-k}^{q^k}$ and $s_{i-j}^{q^i}$, with $i-k$, $j-k$ and $i-j$ computed modulo n. Since the rank of WPW^t is not more than r, recovering $t_0, t_1, \ldots, t_{n-1}$ can be reduced to solving a MinRank problem, that is, to find $t_0, t_1, \ldots, t_{n-1}$ such that

$$Rank(\sum_{i=0}^{n-1} t_k G^{*k}) \leq r.$$

Methods to solve the MinRank problem for small r can be found in [11]. Once the values $t_0, t_1, \ldots, t_{n-1}$ are found, T and S will be then easily computed. Therefore, the key point to attack HFE is to solve the MinRank problem.

The Kipnis-Shamir attack was improved by Courtois using a different method to solve the MinRank problem [3]. However, Ding et al. showed that the original Kipnis-Shamir attack and the improvement of Courtois are not valid in [4]. Later, Faugère et al. proposed a more comprehensive improvement of the Kipnis-Shamir attack against HFE [2].

3 Construction of ABC Cryptosystem

Let $n, m, s \in \mathbb{Z}$ be integers satisying $n = s^2$ and $m = 2n$. For a given integer s, let k^s denote the set of all s-tuples of elements of k. We denote the plaintext by $(x_1, x_2, \ldots, x_n) \in k^n$ and the ciphertext by $(y_1, y_2, \ldots, y_m) \in k^m$. The polynomial ring with n variables in k will be denoted by $k[x_1, \ldots, x_n]$. Let $\mathcal{L}_1 : k^n \to k^n$ and $\mathcal{L}_2 : k^m \to k^m$ be two linear transformations, i.e.

$$\mathcal{L}_1(x) = L_1 x \quad \text{and} \quad \mathcal{L}_2(y) = L_2 y,$$

where L_1 and L_2 are respectively an $n \times n$ matrix and an $m \times m$ matrix with entries in k, $x = (x_1, x_2, \ldots, x_n)^t$, $y = (y_1, y_2, \ldots, y_m)^t$, and t denote the matrix transposition.

The Central map Let

$$A = \begin{pmatrix} x_1 & x_2 & \cdots & x_s \\ x_{s+1} & x_{s+2} & \cdots & x_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(s-1)s+1} & x_{(s-1)s+2} & \cdots & x_{s^2} \end{pmatrix}; \quad B = \begin{pmatrix} b_1 & b_2 & \cdots & b_s \\ b_{s+1} & b_{s+2} & \cdots & b_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ b_{(s-1)s+1} & b_{(s-1)s+2} & \cdots & b_{s^2} \end{pmatrix};$$

and $C = \begin{pmatrix} c_1 & c_2 & \cdots & c_s \\ c_{s+1} & c_{s+2} & \cdots & c_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ c_{(s-1)s+1} & c_{(s-1)s+2} & \cdots & c_{s^2} \end{pmatrix}$ be three $s \times s$ matrices, where $x_i \in k$, b_i and c_i are randomly chosen as linear combination of elements from the set $\{x_1, \ldots, x_n\}$, where $i = 1, 2, \ldots, n$. Define $E_1 = AB$, $E_2 = AC$ and let $f_{(i-1)s+j}$ and $f_{s^2+(i-1)s+j} \in k[x_1, \ldots, x_n]$ be respectively the (i, j) element of E_1 and E_2 $(i, j = 1, 2, \ldots, s)$. Then we obtain with this notation m polynomials $f_1, f_2, \ldots, f_m$, and we define the central map to be

$$\mathcal{F}(x_1, \ldots, x_n) = (f_1(x_1, x_2, \ldots, x_n), \ldots, f_m(x_1, x_2, \ldots, x_n)).$$

We note that for any $1 \leq i \leq m$, the rank of the quadratic form f_i which is associated with the central map $\mathcal{F}$ is close to or equal to $2s$. Define

$$\bar{\mathcal{F}} = \mathcal{L}_2 \circ \mathcal{F} \circ \mathcal{L}_1 = (\bar{f}_1, \bar{f}_2, \ldots, \bar{f}_m),$$

where $\mathcal{L}_1 : k^n \to k^n$ and $\mathcal{L}_2 : k^m \to k^m$ are as above, $\bar{f}_i \in k[x_1, \ldots, x_n]$ are m multivariate polynomials of degree two. The secret key and the public key are given by:

Secret Key The secret key is made of the following two parts:

1) The invertible linear transformations $\mathcal{L}_1, \mathcal{L}_2$.
2) The coefficients of x_i of the elements in matrices B, C.

Public Key The public key is made of the following two parts:

1) The field k, including the additive and multiplicative structure;
2) The maps $\bar{\mathcal{F}}$ or equivalently, its m total degree two components

$$\bar{f}_1(x_1, x_2, \ldots, x_n), \ldots, \bar{f}_m(x_1, x_2, \ldots, x_n) \in k[x_1, \ldots, x_n].$$

Encryption
Given a message $x_1, x_2, \ldots, x_n$, the corresponding ciphertext is

$$(y_1, y_2, \ldots, y_m) = \bar{\mathcal{F}}(x_1, x_2, \ldots, x_n).$$

Decryption
To decrypt the ciphertext $(y_1, y_2, \ldots, y_m)$, one need to perform the following steps:

1 Compute $(\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_m) = \mathcal{L}_2^{-1}(y_1, y_2, \ldots, y_m)$.
2 Put

$$E_1 = \begin{pmatrix} \bar{y}_1 & \bar{y}_2 & \cdots & \bar{y}_s \\ \bar{y}_{s+1} & \bar{y}_{s+2} & \cdots & \bar{y}_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{y}_{(s-1)s+1} & \bar{y}_{(s-1)s+2} & \cdots & \bar{y}_{s^2} \end{pmatrix};$$

$$E_2 = \begin{pmatrix} \bar{y}_{s^2+1} & \bar{y}_{s^2+2} & \cdots & \bar{y}_{s^2+s} \\ \bar{y}_{s^2+s+1} & \bar{y}_{s^2+s+2} & \cdots & \bar{y}_{s^2+2s} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{y}_{s^2+(s-1)s+1} & \bar{y}_{s^2+(s-1)s+2} & \cdots & \bar{y}_{2s^2} \end{pmatrix}.$$

Since $E_1 = AB, E_2 = AC$, we consider the following cases:

(i) If E_1 is invertible, then $BE_1^{-1}E_2 = C$. We have n linear equations with n unknowns $x_1, \ldots, x_n$.

(ii) If E_2 is invertible, but E_1 is not invertible, then $CE_2^{-1}E_1 = B$. We also have n linear equations with n unknowns $x_1, \ldots, x_n$.

(iii) If both E_1 and E_2 are not invertible but A is invertible, then $A^{-1}E_1 = B$, $A^{-1}E_2 = C$. We interpret the elements of A^{-1} as the new variables W_i and we end up with $m = 2n$ linear equations in m unknowns. Then we eliminate the new variables to derive n linear equations in the x_i.

(iv) If A is a singular matrix and the rank of A is $n - r$, then there exits a nonsingular matrix W such that $WA = \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}$, where I is a $(n - r) \times (n - r)$ identity matrix, 0 is a zero matrix. Let $W = \begin{pmatrix} W_1 & W_2 \\ W_3 & W_4 \end{pmatrix}$, $B = \begin{pmatrix} B_1 & B_2 \\ B_3 & B_4 \end{pmatrix}$, $C = \begin{pmatrix} C_1 & C_2 \\ C_3 & C_4 \end{pmatrix}$, $E_1 = \begin{pmatrix} E_{11} & E_{12} \\ E_{13} & E_{14} \end{pmatrix}$, $E_2 = \begin{pmatrix} E_{21} & E_{22} \\ E_{23} & E_{24} \end{pmatrix}$, where $W_1, B_1, C_1, E_{11}, E_{21}$ are a $(n-r) \times (n-r)$ matrices. Since $WE_1 =$

$WAB, WE_2 = WAC$, that is $W_1E_{11} + W_2E_{13} = B_1$, $W_1E_{12} + W_2E_{14} = B_2$, $W_1E_{21} + W_2E_{23} = C_1$, $W_1E_{22} + W_2E_{24} = C_2$.
We interpret the elements of W_1, W_2 as the new variables and we end up with $2s(s-r)$ linear equations in $s(s-r)+n$ unknowns. Then we eliminate the $s(s-r)$ elements of W_1, W_2 in these equations. If these $2s(s-r)$ linear equations are independent, we gain $n-sr$ linear equations with the variables $x_1, x_2, ..., x_n$.

The dimension of the solution space of the linear equations with the variables $x_1, x_2, ..., x_n$ is in general very small. Solving this system by Gaussian elimination enables us to eliminate most of the unknowns, say Z of them. Then we write these Z variables as linear combinations of the remaining unknown variables and then substitute them into the central equations. We then obtain a new system of equations of degree two in the remaining $n-Z$ unknowns which can be easily solved since the number of variables of this new system of equations is very small. Sometimes we may have more than one solution, but the probability is very small.

3 Compute the plaintext $(x_1, x_2, \ldots, x_n) = \mathcal{L}_1^{-1}(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$.

Our experiments show that even if A is a singular matrix, decryption remains successful as long as the rank of A is no less than $s-2$. When the rank of A is less than $s-2$, decryption may fail. Let $r > 0$ be the rank of A, then the number of $s \times s$ matrix of rank r over k is $\frac{q^{r(r-1)/2} \prod_{i=s-r+1}^{s} (q^i-1)^2}{\prod_{i=1}^{r} (q^i-1)}$, thus for any $s \times s$ matrix A, the probability of A of rank r is $\frac{q^{r(r-1)/2} \prod_{i=s-r+1}^{s} (q^i-1)^2}{q^{s^2} \prod_{i=1}^{r} (q^i-1)}$. Therefore, the probability of A of rank less than r is $1 - \sum_{j=r}^{s} \frac{q^{j(j-1)/2} \prod_{i=s-j+1}^{s} (q^i-1)^2}{q^{s^2} \prod_{i=1}^{j} (q^i-1)}$. For example, let $q = 2^8, s = 8$, then the probability of A of rank less than 6 is about 2.125919×10^{-22}, thus, in this case, the probability of decryption failure is about 2.125919×10^{-22}. This means that we can adjust the parameters to make sure that decryption will not be a problem.

4 Security Analysis

In this section, we will study the security of the ABC scheme in order to able us to choose the appropriate parameters for a secure encryption.

4.1 High Order Linearization Equation Attack

Linearization equation attack was first discussed in [18] to attack MI [16]. Later, high order linerlization equation attack was proposed to attack MFE cryptosystem [6]. We use this method to attack our scheme. Since $BE_1^{-1}E_2 = C$ (the case

where $CE_2^{-1}E_1 = B$ is similar), there exists polynomial g_1, with $deg(g_1) \leq s$, such that $Bg_1(E_1)E_2 = Cdet(E_1)$. Therefore, the plaintext and the ciphertext satisfy the equation:

$$\sum_{i_0=1}^{n} \sum_{i_1,\ldots,i_s=1}^{m} \mu_{i_0,i_1,\ldots,i_s} x_{i_0} y_{i_1} \cdots y_{i_s} + \\ + \sum_{i_0=1}^{n} \sum_{i_1,\cdots,i_{s-1}=1}^{m} \nu_{i_0,i_1,\ldots,i_{s-1}} x_{i_0} y_{i_1} \cdots y_{i_{s-1}} + \cdots + \\ + \sum_{i_0=1}^{n} \gamma_{i_0} x_{i_0} + \sum_{i_1=1}^{m} \xi_{i_1} y_{i_1} + \theta = 0,$$

which means that we derive linearization equations with order $n+1$. The coefficients $\mu_{i_0,i_1,\ldots,i_s}, \nu_{i_0,i_1,\cdots,i_{s-1}}, \ldots, \gamma_{i_0}, \xi_{i_1}, \theta$ are variables taking value in k. The number of variables is

$$n \sum_{j=0}^{s} \binom{m}{j} + m + 1 = n \binom{m+s}{s} + m + 1.$$

Using the public key we can generate many plaintext-ciphertxet pairs. By substituting these plaintext-ciphertxet pairs into the equations, we have $n\binom{m+s}{s}+m+1$ linear equations with $n\binom{m+s}{s}+m+1$ variables. However, the computation complexity of solving this linearization equation is $\left(n\binom{m+s}{s}+m+1\right)^{\omega}$, where $\omega = 3$ in the usual Gaussian elimination algorithm and $\omega = 2.3766$ in improved algorithm which is impractical for a bit size greater than or equal to 64. Note here that the computation complexity is even high in the case where E_1 and E_2 are not invertible.

4.2 Rank Attack

There are two different methods of using the rank attack. The first one is called MinRank attack or Low Rank attack and an illutration was discussed in section 2. The other one is called the High Rank Attack. We will look at these two attacks against the ABC scheme. For the MinRank attack, let us assume without lost of generality that the public key polynomials and the secret polynomials are homogeneous quadratic polynomials. Let $\mathcal{L}_1, \mathcal{L}_2$ be two invertible linear transformations. Let $\bar{Q}_1, \bar{Q}_2, \ldots, \bar{Q}_m$ be the symmetric matrices associated with the public key quadratic polynomials and $Q_1, Q_2, \ldots, Q_m$ be the symmetric matrices associate with the secret key quadratic polynomials. Clearly, the rank of Q_i is bounded by $2s$. With the MinRank attack, one tries to find $(t_1, t_2, \ldots, t_m) \in k^m$ such that the rank of the linear combinations $\sum_{i=1}^{m} t_i \bar{Q}_i$ is no more than $2s$. In order to find such a linear combination, one can choose any vector $\upsilon \in k^n$ and try to solve the equations $(\sum_{i=1}^{m} t_i \bar{Q}_i)\upsilon = 0$ with the unknowns $t_1, \ldots, t_m$. After

finding at least one linear combination of this form, attacker can recover $\mathcal{L}_2$. The attacker can recover $\mathcal{L}_1$ and $Q_1, \ldots, Q_m$ when $\mathcal{L}_2$ is known. More detail about the MinRank attack can be found in [3,10]. The complexity of this attack against the ABC scheme is $O(q^{\lceil \frac{m}{n} \rceil 2s} m^3)$.

For the High Rank Attack, we form an arbitrary linear combinations $Q = \sum_{i=1}^{m} \alpha_i \bar{Q}_i$, then we find $V = Ker(Q)$. If Q have a nontrivial kernel, set $\sum_{i=1}^{m} \lambda_i \bar{Q}_i V = 0$ and check if the solution set $\hat{V}$ of λ_i has a dimension $n - 2s$. This attack uses about $O(n^6 q^{2s})$ field multiplications. Moreover, note that for every vector v of dimension n, there exists a linear combination of the 2^n secret polynomials that yields zero with probability roughly $1 - \frac{1}{q^n}$. So we are faced with a lot of parasitic solutions, which have to be ruled out at the end. Also as it was mentioned earlier the rank of the Q_i is associated with $2\sqrt{n}$ which means that the complexity of the rank attack may not be polynomial time in the number of variables. These facts prove that the Rank attack is really inefficient against the ABC scheme.

4.3 Algebraic Attack

Let $\bar{f}_1(x_1, \ldots, x_n), \ldots, \bar{f}_m(x_1, \ldots, x_n) \in k[x_1, \ldots, x_n]$ be the public key polynomials. Let $y_1, y_2, \ldots, y_m$ be the ciphertext. We try to solve the system of equations

$$\begin{cases} \bar{f}_1(x_1, x_2, \ldots, x_n) = y_1; \\ \bar{f}_2(x_1, x_2, \ldots, x_n) = y_2; \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ \bar{f}_m(x_1, x_2, \ldots, x_n) = y_m, \end{cases}$$

directly by Gröbner bases or XL method and its variations Mutant XL algorithm[25][26][27].

We carried out a number of experiments with MAGMA [1], which contains an efficient implementation of F4 algorithm [9] for computing Gröbner bases. Table 1 shows the results of our experiments to attack an instance of ABC scheme in a finite field k of 3 elements.

Table 1. Result of experiments with direct attack using MAGMA(2.12-16) on a 1.80GHz Intel(R) Atom(TM) CPU

n	9	16	25
time(s)	0.016	3.494	17588.380
memory(MB)	3.4	8.1	1111.7
degree of regularity	4	5	6

As the table 1 shows, the time and memory complexity increase as n grows. Also the degree of regularity increases as n grows which indicated that complexity is exponential.

4.4 Special Attacks

In terms of the design, one may think that maybe we can choose B and C such that their entries are randomly selected sparse linear functions or even monomials, which will allow us to have smaller secret key. However in the case of using only monomials, there is a possible new risk, namely there is a possibility that the central map polynomials are so sparse that they may have hidden UOV structures, that is there are no quadratic terms of a set of variables in the central map polynomials. One may then use UOV Reconciliation attack to find such structure [23][24]. It is not a good ideal to use monomials for B and C, such a distinguished feature is in general not desired. But in the case of general B and C such a feature does not exist. It is an open interesting problem to find out what really happens in the case of sparse B and C.

On the other hand, one may say that how about making A also more general, namely entries are selected as random linear functions. It is clear this is not needed since a linear transformation will easily remove such a feature. Using a matrix A of variables and L_1 is equivalent to using a matrix A of linear functions, without any transformation L_1. In the case of A also more general, one may consider certain tensor related attack, but we cannot see yet any effective way to do so.

5 A Practical Implementation for Encryption

For a practical implementation, we let k be the finite field of $q = 2^8$ elements and $n = 64$. In this case, the plaintext consist of the message $(x_1, \ldots, x_{64}) \in k^{64}$. The public map is $\bar{F} : k^{64} \to k^{128}$ and the central map is $F : k^{64} \to k^{128}$.

The public key consists of 128 quadratic polynomials with 64 variables. The number of coefficients for the public key polynomials is

$$128 \times 66 \times 65/2 \in \{274560, \text{or about } 280KB \text{ of storage}\}.$$

The private key consists of the coefficients of the x_i of the entries of the matrices B and C. and the two linear transformations $\mathcal{L}_1, \mathcal{L}_2$. The total size is about $30KB$.

The size of a document is $8n = 8 \times 64 = 512bits$ and the total size of the ciphertext is $1024bits$.

Based on the preceding discussion in section 4, security level for this implementation is lager than 2^{86}. Using odd characteristic field may be good to resist algebraic attack, but it requires more storage.

6 Efficiency of ABC Scheme

In this section, we will compare the efficiency of decryption in ABC scheme with HFE challenge 1 by Patarin [19]. This HFE was broken using algebraic attack [13]. In this HFE scheme, J.Patarin chose the parameters as follow: $q = 2, n = 80$,

the degree of central map is 96. Let $P(x)$ be the central map of HFE, the main computation of decryption is to solve the equation $P(x) = y$ over the finite field of 2^{80} elements. In [20], J.Patarin estimated that the complexity of solving this equation is about $O(d^2n^3)$ or $O(dn^3+d^3n^2)$–depending on the chosen algorithms, where d is the degree of $P(x)$. Thus the decryption process needs about 6.4×10^9 times field multiplication over the finite field of 2^{80} elements.

For the proposed parameters of the ABC scheme above, $q = 2^8, n = 64$ and $m = 128$, the steps of decryption were presented in section 3. The computation of step 1) and step3) of decryption are very fast. The main computation of decryption is step 2), solving a set of linear equations. Therefore, we only need about $128^3 = 2^{21} \approx 2.1 \times 10^6$ times field multiplications over the finite field of 2^8 elements for decryption. It is much faster than HFE scheme.

7 Conclusion

In this paper, we propose a new multivariate algorithm for encryption called ABC. A highlight of ABC scheme is that all the quadratic forms associated with the central map are not of low rank but related to some variable integer n. Therefore, it is immune to the MinRank Attack. Another highlight of ABC scheme is that the computation of decryption is very fast, because the main computation is to solve certain linear equations. However we still cannot show that ABC is provably secure.

Acknowledgment. The authors are grateful to the referees for constructive suggestion on special attack and the design which helped to improve the paper. The work was partially supported by **Charles Phelps Taft Foundation** and the NSF of China under the grant #60973131 and the National Natural Science Foundation of China under grant No. U1135004.

References

1. Bosma, W., Cannon, J.J., Playoust, C.: The Magma algebra system I: the user language. J. Symb. Comput. 24(3-4), 235–265 (1997)
2. Bettale, L., Faugère, J.-C., Perret, L.: Cryptanalysis of multivariate and odd-characteristic HFE variants. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 441–458. Springer, Heidelberg (2011)
3. Goubin, L., Courtois, N.T.: Cryptanalysis of the TTM cryptosystem. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 44–57. Springer, Heidelberg (2000)
4. Ding, J., Schmidt, D., Werner, F.: Algebraic attack on HFE revisited. In: Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 215–227. Springer, Heidelberg (2008)
5. Ding, J., Gower, J., Schmidt, D.: Multivariate Public Key Cryptography. Advances in Information Security series. Springer, Heidelberg (2006)

6. Ding, J., Yang, B.-Y., Chen, C.-H.O., Chen, M.-S., Cheng, C.-M.: New Differential-Algebraic Attacks and Reparametrization of Rainbow. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 242–257. Springer, Heidelberg (2008)
7. Ding, J., Hu, L., Nie, X., Li, J., Wagner, J.: High Order Linearization Equation (HOLE) Attack on Multivariate Public Key Cryptosystems. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 233–248. Springer, Heidelberg (2007)
8. Ding, J., Hodges, T.J.: Inverting HFE systems is quasi-polynomial for all fields. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 724–742. Springer, Heidelberg (2011)
9. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases (F4). J. Pure Appl. Algebra 139, 61–88 (1999)
10. Faugère, J.-C., Levy-dit-Vehel, F., Perret, L.: Cryptanalysis of minRank. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 280–296. Springer, Heidelberg (2008)
11. Kipnis, A., Shamir, A.: Cryptanalysis of the Oil & Vinegar Signature Scheme. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 257–267. Springer, Heidelberg (1998)
12. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced Oil and Vinegar Signature Schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999)
13. Kipnis, A., Shamir, A.: Cryptanalysis of the HFE public key cryptosystem by relinearization. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 19–30. Springer, Heidelberg (1999)
14. Lidl, R., Niederreiter, H.: Finite Fields. Encyclopedia of Mathematics and its applications, vol. 20. Cambridge University Press
15. Moh, T.T.: A fast public key system with signature and master key functions. In: Proceedings of CrypTEC 1999, International Workshop on Cryptographic Techniques and E-Commerce, pp. 63–69. Hong-Kong City University Press (July 1999), http://www.usdsi.com/cryptec.ps
16. Matsumoto, T., Imai, H.: Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 419–453. Springer, Heidelberg (1988)
17. Patarin, J.: The Oil and Vinegar Signature Scheme. Presented at the Dagstuhl Workshop on Cryptography (September 1997) (transparencies)
18. Patarin, J.: Cryptoanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 248–261. Springer, Heidelberg (1995)
19. Patarin, J.: Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 33–48. Springer, Heidelberg (1996)
20. Rivest, R., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21(2), 120–126
21. Shor, P.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing 26(5), 1484–1509 (1997)
22. Wang, L.-C., Yang, B.-Y., Hu, Y.-H., Lai, F.: A "Medium-Field" Multivariate Public-Key Encryption Scheme. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 132–149. Springer, Heidelberg (2006)

23. Ding, J., Yang, B.-Y., Chen, C.-H.O., Chen, M.-S., Cheng, C.-M.: New differential-algebraic attacks and reparametrization of rainbow. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M., et al. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 242–257. Springer, Heidelberg (2008)
24. Thomae, E.: A Generalization of the Rainbow Band Separation Attack and its Applications to Multivariate Schemes. IACR Cryptology ePrint Archive (2012)
25. Buchmann, J.A., Ding, J., Mohamed, M.S.E., et al.: MutantXL: Solving multivariate polynomial equations for cryptanalysis. Symmetric Cryptography, 09031 (2009)
26. Mohamed, M.S.E., Mohamed, W.S.A.E., Ding, J., Buchmann, J.: *MXL2*: Solving polynomial equations over GF(2) using an improved mutant strategy. In: Buchmann, J., Ding, J., et al. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 203–215. Springer, Heidelberg (2008)
27. Mohamed, M.S.E., Cabarcas, D., Ding, J., Buchmann, J., Bulygin, S.: MXL3: An efficient algorithm for computing gröbner bases of zero-dimensional ideals. In: Lee, D., Hong, S., et al. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 87–100. Springer, Heidelberg (2010)

Multivariate Signature Scheme Using Quadratic Forms

Takanori Yasuda[1], Tsuyoshi Takagi[2], and Kouichi Sakurai[1,3]

[1] Institute of Systems, Information Technologies and Nanotechnologies
[2] Institute of Mathematics for Industry, Kyushu University
[3] Department of Informatics, Kyushu University

Abstract. Multivariate Public Key Cryptosystems (MPKC) are candidates for post-quantum cryptography. MPKC has an advantage in that its encryption and decryption are relatively efficient. In this paper, we propose a multivariate signature scheme using quadratic forms. For a finite dimensional vector space V, it is known that there are exactly two equivalence classes of non-degenerate quadratic forms over V. We utilize the method to transform any non-degenerate quadratic form into the normal form of either of the two equivalence classes in order to construct a new signature scheme in MPKC. The signature generation of our scheme is between eight and nine times more efficient more than the multivariate signature scheme Rainbow at the level of 88-bit security. We show that the public keys of our scheme can not be represented by the public keys of other MPKC signature schemes and this means our scheme is immune to many attacks that depend on the form of the central map used by these schemes.

Keywords: Multivariate Public Key Cryptosystem, Digital signature, Rainbow, Post-quantum cryptography.

1 Introduction

Multivariate Public Key Cryptosystems (MPKC) [9] can be potentially applied to post-quantum cryptography. MPKC can be used for encryption and digital signature, and its encryption and decryption processes (and signature generation and verification) are relatively efficient in comparison with RSA and elliptic curve cryptography[8]. The security of MPKC depends on the difficulty of solving a system of multivariate polynomials that form its secret key and public key, and the security of MPKC depends on the difficulty in solving a system of multivariate polynomials. At present, the most efficient way to solve a system of multivariate polynomials is to compute the Gröbner basis. The attacks against this method are called direct attacks, and they are applicable against any MPKC scheme. For UOV[18] and Rainbow[10] signature scheme, the direct attacks determine their security level.

In this paper, we propose a new signature scheme. It is known that there are two isometry classes of non-degenerate quadratic forms on a vector space with

P. Gaborit (Ed.): PQCrypto 2013, LNCS 7932, pp. 243–258, 2013.

a prescribed dimension[32]. We use a computational method whereby any non-degenerate quadratic form is transformed into either of the canonical forms of two classes of the signature generation of our scheme. We estimate the efficiency of the signature generation in terms of the number of multiplications of base field. The signature generation scheme consists of two affine transformations and the inverse computation of the central map. The inverse computation of the central map of our scheme is cheaper than the two affine transformations. We compare of the efficiency of our signature generation with that of Rainbow. We choose the parameters of both schemes under the assumption that the security level of these schemes against direct attacks are same, As a result, we find that the signature generation of our scheme is between eight and nine times more efficient than Rainbow at the level of 88-bit security.

A lot of MPKC signature schemes have been proposed: however, not much is known about relations between the different schemes (*ref.* [20]). For example, it is still an open problem as to whether the public key of the Matsumoto-Imai scheme can be expressed as a public key of Rainbow. Our scheme uses two systems of multivariate polynomials. These systems have a property whereby the regions of their values are exclusive. In particular, the two system are not surjective. On the other hand, schemes that have already proposed (e.g. UOV and Rainbow) use only one system of multivariate polynomials. Moreover, their system is surjective. Accordingly, the public key of our scheme is not able to be expressed in terms of the public keys previously proposed schemes. As far as we know, this is the first report of a public key of a scheme that can not be expressed by using the public keys of other schemes.

We can explain the importance of the public key of a scheme not being expressed by the keys of other schemes by using an example. For Rainbow, UOV attack, UOV-Reconciliation attack, Rainbow-band-separation attack, etc., have been proposed against Rainbow. These attacks all transform the public key into a central map of Rainbow. If the public key can be transformed into the central map, the signature can be forged using the same method as the signature generation of Rainbow. However, it can be proved that these attacks can not be applied to our scheme. The public key of our scheme can not be transformed into the central maps of Rainbow. Of course, attacks that are independent of the signature scheme like as direct attacks can be launched against our scheme. In addition, there is a possibility that there is an attack which works well against our scheme. In fact, MinRank attack and a method for solving Isomorphisms of Polynomials can be applied to our scheme. The analysis of these attacks will be tackled more elaborately in the future.

2 Construction of Signature Scheme in MPKC

A lot of MPKC signature schemes e.g., UOV[18] and Rainbow[10], have been proposed.

The security of MPKC is based on the difficulty of solving the *MQ problem*. An MQ problem is to find a solution of the following system of quadratic polynomials with n variables and m polynomials.

$$\begin{cases} a_{11}^{(1)}x_1^2 + a_{12}^{(1)}x_1x_2 + \ldots + c^{(1)} = 0, \\ a_{11}^{(2)}x_1^2 + a_{12}^{(2)}x_1x_2 + \ldots + c^{(2)} = 0, \\ \qquad \vdots \\ a_{11}^{(m)}x_1^2 + a_{12}^{(m)}x_1x_2 + \ldots + c^{(m)} = 0. \end{cases}$$

Here, the coefficients $a_{ij}^{(k)}$ belong to a finite field K. For large m and n and if the coefficients are chosen randomly, the MQ problem is considered to be NP-hard[13].

MPKC aims to design secure encryption and signature schemes by using a system of quadratic multivariate polynomials as the public key. To design a MPKC scheme, we start by constructing a secret key from a system of multivariate polynomials which is easy to solve, and next, we transform the secret key into public key by using affine transformations. Note that not every system of multivariate polynomials can be used in MPKC. More concretely, the secret and public keys are constructed as follows.

Secret Key: A system g of multivariate polynomials with n variables and m polynomials satisfying the following condition, two affine transformations $L : K^m \to K^m$, $R : K^n \to K^n$.

Condition: For any $\mathbf{c} \in K^m$, we can efficiently compute $\mathbf{x} \in K^n$ such that $g(\mathbf{x}) = \mathbf{c}$.

Public Key: A system of multivariate polynomials defined by $f = L \circ g \circ R$.

g appearing in the secret key is regarded as a map $K^n \to K^m$. This map is called the *central map* of this scheme. The signature generation and verification are as follows:

Signature Generation: Let $\mathbf{M} \in K^m$ be a message. Compute $\mathbf{A} = L^{-1}(\mathbf{M})$, $\mathbf{B} = g^{-1}(\mathbf{B})$ and $\mathbf{C} = R^{-1}(\mathbf{B})$ in this order. $\mathbf{C}$ is a signature

Verification: If $F(\mathbf{C}) = \mathbf{M}$, the signature is accepted: it is rejected otherwise.

It is natural to choose a surjective map as a central map of a signature scheme because for any possible message, the corresponding signature must be generated. In fact, all of the previously proposed signature schemes use surjective map.

3 Quadratic Forms

In this section, we summarize the fundamental facts about quadratic forms that are necessary for our scheme. The details of quadratic forms and their properties is refered to [32].

3.1 Definition and Facts

Let K be a finite field with odd order q. Let V be an r-dimensional vector space over K.

Definition 1. *A map $q : V \to K$ is said to be a quadratic form if the following is satisfied:*

1. $q(ax) = a^2 q(x)$ $(a \in K,\ x \in V)$,
2. $V \times V \ni (x, y) \mapsto q(x + y) - q(x) - q(y)$ *is bilinear.*

For a quadratic form q on V, there is an $r \times r$-matrix $A = (a_{ij})_{ij}$ such that

$$q(x + y) - q(x) - q(y) = \mathbf{x}A\mathbf{y}^T \ (x, y \in V),$$

where $\mathbf{x}, \mathbf{y}$ are row vectors in K^r corresponding x, y, respectively. The matrix A is called a matrix expression of the quadratic form q. Note that a matrix expression of q is not unique. In fact, $\frac{1}{2}(A + A^T)$ is also a matrix expression of q. Since this matrix is symmetric, we will take a symmetric matrix as a matrix expression of q. Conversely, for a $r \times r$-symmetric matrix A,

$$q(x) = \frac{1}{2}\mathbf{x}A\mathbf{x}^T$$

is a quadratic form on V.

It looks like a quadratic form corresponds to a one-to-one and onto symmetric matrix. However, this is not true because the choice of matrix expression of a quadratic form is not determined under the condition that the matrix must be symmetric. In fact, the choice depend on the choice of the basis of V over K. Accordingly, we define the concept of equivalence of quadratic forms as follows.

Definition 2. *Let q_1 and q_2 be quadratic forms on V. Let A_1 and A_2 be matrix expressions of q_1 and q_2, respectively. We say that q_1 and q_2 are isometric if there is an $r \times r$-regular matrix C such that*

$$A_1 = CA_2C^T.$$

The relation of isometricity is independent of the choice of the basis of V, and hence, it is an equivalence relation.

Definition 3. *Let q be a quadratic form on V that can be expressed as a matrix A. We say that q is non-degenerate if* $\det A \neq 0$.

This definition is independent of the choice of the basis of V.

Lemma 1. *Any quadratic form q with more than* 1 *dimension, can represents any element of K.*

The following classification theorem is for quadratic forms over finite fields.

Theorem 1 (Classification theorem). *Let V be a vector space of dimension r over K. Let q be a non-degenerate quadratic form on V. Set a non-square δ in K. Then q is isometric to either of the following two quadratic forms:*

$$q_1(x) = \mathbf{x}A_1\mathbf{x}^T, \quad q_\delta(x) = \mathbf{x}A_\delta\mathbf{x}^T,$$

$$A_1 = \mathrm{I}_r(: \textit{identity matrix}), \ A_\delta = \left(\begin{array}{ccc|c} 1 & & & \\ & \ddots & & \\ & & 1 & \\ \hline & & & \delta \end{array}\right).$$

Here, q_1 and q_δ are not isomorphic.

3.2 Idea Behind Our Scheme

Let us explain how Theorem 1 is applied to our scheme. The following is a corollary of the theorem.

Corollary 1. *Let A be an $r \times r$-symmetric matrix with $\det A \neq 0$. Then, there is an η in $\{1, \delta\}$ and an $r \times r$-regular matrix C such that*

$$CAC^T = A_\eta.$$

(Note that C is not uniquely determined.) If η' is another η in $\{1, \delta\}$ there is no $r \times r$-symmetric matrix C' such that

$$C'AC'^T = A_{\eta'}.$$

Next, let us consider the case that a quadratic form is degenerate. In this case, there is a regular matrix B such that

$$BAB^T = \left(\begin{array}{ccc|c} * & \cdots & * & 0 \\ \vdots & \ddots & \vdots & \vdots \\ * & \cdots & * & \vdots \\ \hline 0 & \cdots & \cdots & 0 \end{array}\right).$$

Lemma 1 and induction on r enable us to prove the following.

Proposition 1. *Let A be an $r \times r$-symmetric matrix with $\det A = 0$. Then for each $\eta = 1, \delta$, there is an $r \times r$-matrix C,*

$$CAC^T = A_\eta.$$

Theorem 1 and Corollary 1 can be rewritten in terms of a system of multivariate polynomials. The input and output of the system are expressed as matrices. In fact, for an $r \times r$-matrix variable X, the system is described as

$$f_1(X) = XA_1X^T, \quad f_\delta(X) = XA_\delta X^T$$

The scalar variables of f_1 and f_δ consists of the components of the matrix variable X. Let x_{ij} $(1 \leq i, j \leq r)$ be r^2 variables, and $X = (x_{i,j})$. f_1 and f_δ can be regarded as polynomials with respect to r^2 variables x_{ij}. Since the output matrices of f_1 and f_δ are symmetric, both f_1 and f_δ can be regarded as consisting of $r(r+1)/2$ polynomials.

Theorem 1 and Corollary 1 can be rewritten using f_1 and f_2.

Proposition 2. *For any $w \in K^{r(r+1)/2}$, either of the two systems of multivariate polynomials,*

$$f_1(X) = w, \quad f_\delta(X) = w \tag{1}$$

has a solution. (The solution is not uniquely determined.) Moreover, if the symmetric matrix w is regular, then only one of the above equations has a solution.

This proposition guarantees that when f_1, and f_δ are used in a signature system, there exists its signature for any message. The method of computing (1) will be explained later.

Example. Let us consider the case of $r = 2$ as an example. For a $\alpha \in K^\times$, we define a quadratic multivariate polynomial $f_\alpha : K^4 \to K^3$ by

$$\begin{aligned} &f_\alpha(x_{11}, x_{12}, x_{21}, x_{22}) \\ &\quad = (x_{11}^2 + x_{12}^2\alpha,\ x_{11}x_{21} + x_{12}x_{22}\alpha,\ x_{21}^2 + x_{22}^2\alpha) \end{aligned}$$

This map is not surjective, and from Theorem 1 and Proposition 1, we have

1. For $\mathbf{w} = (w_1, w_2, w_3) \in K^3$, if $w_1w_3 - w_2^2 \in \alpha \cdot \{b^2 \mid b \in K\}$, $f_\alpha(X) = \mathbf{w}$ has a solution.
2. Otherwise, $f_\alpha(X) = \mathbf{w}$ has no solution.

In particular, if $\delta \in K^\times$ is a non-square and $\alpha = 1, \delta$, f_1 and f_δ are not surjective. However, the union of the regions of values of the two maps coincides with whole K^3 (Prop. 2), that is, for any $\mathbf{w} = (w_1, w_2, w_3) \in K^3$, either of the following has a solution.

$$\begin{cases} f_1(X) = \mathbf{w} \\ f_\delta(X) = \mathbf{w}. \end{cases}$$

3.3 Computing the Inverse of f_1, f_δ

We will explain how to compute X by assuming that $f_1(X) = \mathbf{w}$ has a solution $X \in K^{r^2}$ for $\mathbf{w} \in K^{r(r+1)/2}$. Since a solution of $f_\delta(X) = \mathbf{w}$ can be computed in the similar fashion, we will explain only the case in which $f_1(X) = \mathbf{w}$.

Case of $r = 2$. The case of $r = 1$ is easy. ($X = \sqrt{\mathbf{w}}$ is a solution.) Thus, we will start with the case of $r = 2$. In this case, any $\mathbf{w}(\neq 0) \in K^3$ can be expressed by one of the following three 2×2-symmetric matrices:

$$\begin{pmatrix} 0 & b \\ b & 0 \end{pmatrix}, \ \begin{pmatrix} a & b \\ b & c \end{pmatrix} \ (a \neq 0), \ \begin{pmatrix} a & b \\ b & c \end{pmatrix} \ (c \neq 0).$$

We can diagonalize these matrices using the following operation (at most twice):

1. $\begin{pmatrix} 1/2 & 1 \\ -1/2 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1/2 & 1 \\ -1/2 & 1 \end{pmatrix}^T = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$,
2. $\begin{pmatrix} 1 & 0 \\ a^{-1}b & -1 \end{pmatrix} \begin{pmatrix} a & b \\ b & c \end{pmatrix} \begin{pmatrix} 1 & 0 \\ a^{-1}b & -1 \end{pmatrix}^T = \begin{pmatrix} a & 0 \\ 0 & c - a^{-1}b^2 \end{pmatrix}$ $(a \neq 0)$,
3. $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a & b \\ b & c \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}^T = \begin{pmatrix} c & b \\ b & a \end{pmatrix}$ $(c \neq 0)$.

Let

$$\begin{pmatrix} a' & 0 \\ 0 & c' \end{pmatrix} \quad (a' \neq 0) \tag{2}$$

be the diagonalized matrix. In addition, we assume that $c' \neq 0$. Accordingly, there is a $d \in K$ such that $c'd^2 = a'^{-1}$. (In the case of f_δ, $c'd^2 = a'^{-1}\delta$.) d can be computed efficiently by precomputing of the square roots of the elements of K. We have

$$\begin{pmatrix} 1 & 0 \\ 0 & d \end{pmatrix} \begin{pmatrix} a' & 0 \\ 0 & c' \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & d \end{pmatrix}^T = \begin{pmatrix} a' & \\ & a'^{-1} \end{pmatrix}.$$

Since for any a', we can precompute a matrix $C_{a'}$ such that

$$C_{a'} \begin{pmatrix} a' & 0 \\ 0 & a'^{-1} \end{pmatrix} C_{a'}^T = \begin{pmatrix} 1 & \\ & 1 \end{pmatrix}, \tag{3}$$

(2) can be transformed into A_1. In the case that $c' = 0$, from (3), we have

$$(\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} C_{a'}^{-1}) A_1 (\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} C_{a'}^{-1})^T = \begin{pmatrix} a' & 0 \\ 0 & 0 \end{pmatrix}.$$

In either case, we can make an equation of the form,

$$(D_l \cdots D_1) A_1 (D_l \cdots D_1)^T = \mathbf{w}$$

and, $\mathbf{x} = D_l \cdots D_1$ is a solution. Here, $D_1, \ldots, D_l$ are the transformation matrices described above.

Case of $r \geq 3$. Let $r \geq 3$. $\mathbf{w}(\neq 0) \in K^{r(r+1)/2}$ can be expressed as an $r \times r$-symmetric matrix

$$\begin{pmatrix} a & * & \cdots & * & b & * & \cdots \\ * & & & & * & & \\ \vdots & & \ddots & & \vdots & & \\ * & & & & * & & \\ b & * & \cdots & * & c & & \\ * & & & & & & \\ \vdots & & & & \ddots & & \end{pmatrix}. \tag{4}$$

We can apply the above operations in the case of two dimensions to the 2×2-matrix composed by rows and columns including a, b and c:

$$\begin{pmatrix} a & b \\ b & c \end{pmatrix},$$

By iterating these operations, the matrix (4) can be transformed into

$$\left(\begin{array}{c|ccc} a & 0 & \cdots & 0 \\ \hline 0 & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & * & \cdots & * \end{array}\right).$$

We can apply the same operation to the $(r-1) \times (r-1)$-miner matrix. Induction shows that the matrix (4) can be diagonalized. The diagonal matrix can be transformed into A_1 in the similar fashion as in the case of two dimensions. Consequently, we obtain a solution X of $f_1(X) = \mathbf{w}$. This solution is not unique.

4 Our Scheme

Let $n = r^2$, $m = r(r+1)/2$. The key generation, signature generation and verification of our scheme are as follows:

• **Key generation**

Secret key. The secret key consists of a non-square $\delta \in K$, a $r \times r$-regular matrix B, and two randomly chosen affine transformations $L : K^m \to K^m$ and $R : K^n \to K^n$.

Public key. The public key consists of the composite maps $F_1 = L \circ f_1 \circ R$, $F_\delta = L \circ f_{\delta,B} \circ R : K^n \to K^m$, where $f_{\delta,B}(X) = XBA_\delta B^T X^T$.

• **Signature Generation.** Let $\mathbf{M} \in K^m$ be a message. To generate a signature $\mathbf{S}$ from $\mathbf{M}$, first compute $\mathbf{M}' = L^{-1}(\mathbf{M})$, After that, compute an $r \times r$-matrix $\mathbf{S}'$ such that

$$\mathbf{M}' = \mathbf{S}' A_1 \mathbf{S}'^T \text{ or } \mathbf{M}' = \mathbf{S}' A_\delta \mathbf{S}'^T$$

In the former case, compute $\mathbf{S} = R^{-1}(\mathbf{S}')$. In the latter case, compute $\mathbf{S} = R^{-1}(\mathbf{S}'B^{-1})$. $\mathbf{S}'$ is computed using the improved algorithm described above. $L^{-1}(\mathbf{M})$ and $R^{-1}(\mathbf{S}')$ can be easily computed since L and R are affine transformations.

• **Verification** If $F_1(\mathbf{S}) = \mathbf{M}$ or $F_\delta(\mathbf{S}) = \mathbf{M}$, the signature is accepted. Otherwise, it is rejected.

5 The Security of Our Scheme

5.1 Application of Attacks against Rainbow

Here, we show whether the following famous attacks against Rainbow can be launched to our scheme.

- direct attacks ([2,35])
- MinRank attack ([15,34,4])
- HighRank attack ([15,11,25])
- UOV attack ([19,18])
- Rainbow-Band-Separation (RBS) attack ([11,24])
- UOV-Reconciliation (UOV-R) attack ([11,24])

Direct Attacks. For a message $\mathbf{M}$, if $\mathbf{C}'$ is found such that $F(\mathbf{C}') = \mathbf{M}$, then one can forge a signature for $\mathbf{M}$. (F is the public key.) Since this equation involves multivariate polynomial equations, it can be solved by computing gröbner basis[2,35]. Direct attacks depend on the message. Since they are attacks against the MQ problem on which the security of MPKC is based, these attacks are applicable to any MPKC scheme, including ours.

Other Attacks against Rainbow. Beside the direct ones, there are 5 attacks that transform the public key into a central map of Rainbow. These attacks are not effective against our scheme. For these attacks to be applicable to our scheme, the public key of our scheme must be able to transformed into the central map of Rainbow. Since the central map of Rainbow is surjective, the public key of our scheme must also be surjective. However, the public key of our scheme is not surjective, so this is a contradiction that proves such attacks can not be launched.

5.2 Attacks Applicable to Only Our Scheme

As we explained before, the attacks in which the public map is reduced to a central map surjective can not be applied to our scheme. However, there are attacks of other type. Here, we explain two attacks applicable to our scheme.

MinRank Attack. This attack is different from the MinRank attack against Rainbow explained above. The quadratic polynomials composing the central maps of our scheme correspond to matrices of rank r. This property can be applied to an attack against our scheme. We denote by $\mathcal{A}$ the space spanned by the square matrices associated to quadratic parts of components of the public key. Most of elements have rank r^2 by randomness of the affine transformations in the secret key. However, the matrices associated to components of the central map have rank r, and r is the minimal rank among elements in $\mathcal{A}$. On the other hand, a matrix of minimal rank in $\mathcal{A}$ can be found by solving MinRank problem[21]. Therefore, this method reveals the secret key of our scheme. The complexity of solving MinRank problem is estimated by [4]

$$q^r \cdot m(n^2/2 - m^2/6)\ \mathbf{m},$$

where m, n are the number of equations and variables, respectively, and $\mathbf{m}$ means the multiplication in K.

Isomorphism of Polynomials Problem. Isomorphism of Polynomials(IP) Problem[28] is related to an attack against our scheme. The IP problem is the following: given multivariate polynomial maps F and G, find affine transformations A and B such that $F = A \circ G \circ B$ (if they exist). There are several papers which treat IP problem and its variants[6,7,27]. Since our scheme uses special central maps, f_1 and $f_{\delta,B}$, the problem for finding the sceret key of our scheme can be replaced by the IP problem in the case that F and G are the public and secret key, respectively. Patarin estimated the complexity of solving general IP problem by $\mathcal{O}(q^{3n/2})$ for any system of n quadratic equations with n variables, (and by $\mathcal{O}(q^{n/2})$ for a system of Matsumoto-Imai scheme)[28].

6 Efficiency of Signature Generation

In this section, we estimate the efficiency of signature generation of our scheme in terms of the number of multiplications of the base field. After that, we compare it with the efficiency of signature generation of Rainbow.

6.1 Efficiency of Signature Generation of Our Scheme

In our scheme, the number of variables is $n = r^2$ and the number of polynomials is $m = r(r+1)/2$. The signature generation is computed in the following order: For $\mathbf{M} \in K^m$, (i) $\mathbf{A} = L^{-1}(\mathbf{M})$ is computed for an affine isomorphism $L : K^m \to K^m$ in the secret key, (ii) a solution $\mathbf{B}$ of $f_1(\mathbf{X}) = \mathbf{A}$ or $f_\delta(\mathbf{X}) = \mathbf{A}$ is computed by regarding $\mathbf{A}$ as $r \times r$-symmetric matrix, . (iii) if $f_\delta^{-1}(\mathbf{A})$ has a solution, $\mathbf{B}$ is replaced by $\mathbf{B}B^{-1}$. (iv) $\mathbf{C} = R^{-1}(\mathbf{B})$ is computed for an affine isomorphism $R : K^n \to K^n$ in the secret key.

The number of multiplications of the base field for $\mathbf{A} = L^{-1}(\mathbf{M})$ and $\mathbf{C} = R^{-1}(\mathbf{B})$ is

$$L^{-1} : \quad \left(\frac{r(r+1)}{2}\right)^2, \quad R^{-1} : \quad r^4. \tag{5}$$

The $r \times r$-symmetric matrix $\mathbf{A}$ in the computation of (ii) is diagonalized. The following computation of transformation is dominant.

$$\left(\begin{array}{c|ccc} 0\cdots & 0 & \cdots & 0 \\ \vdots\ \ddots & & \ddots & \vdots \\ \hline 0 & a & & b \\ & \ddots & \ddots & \ddots \\ \vdots & b & & c \\ 0\cdots & & \ddots & \ddots \end{array}\right) \rightarrow \left(\begin{array}{c|ccc} 0\cdots & 0 & \cdots & 0 \\ \vdots\ \ddots & & \ddots & \vdots \\ \hline 0 & a & & 0 \\ & \ddots & \ddots & \ddots \\ \vdots & 0 & & c-b^2/a \\ 0\cdots & & \ddots & \ddots \end{array}\right)$$

($a \neq 0$). This computation is executed using the transformation **2** in § 3.3. The number of a's is equal to r because a is on the diagonal, The number of b's is equal to the number of positions in the upper triangle of the matrix. The transformation **2** in § 3.3 needs $3r$ multiplications of the base field. Therefore, the total number of multiplications is equal to $3r^2(r-1)/2$.

We want to use the same matrix transformation for $a = 0$, too. To do this, we can use the transformation **1**,**3** in § 3.3, in order to transform the matrix such that a non-zero element appears at the position of a.

The transformation **3** needs no multiplications of K. The transformation **1** needs $6r$ multiplications. Since the number of a's is equal to r, the total number of multiplications is equal to $6r^2$.

The above computation complete the diagonalization. After that, the diagonal components have to be transformed into 1. For each diagonal component, the computation to transform them into 1 needs $12r$ multiplications. The number of components that have to be transformed into 1 is equal to $r-1$. The total number of multiplications is at most $12r(r-1)$. Therefore the total number of multiplications in the computation of (ii) is at most

$$\frac{3r^2(r-1)}{2} + 6r^2 + 12r(r-1) = \frac{3r^3 - 33r^2 - 24r}{2}. \tag{6}$$

r^2 multiplication is needed in the computation of $\mathbf{B}B^{-1}$ in (iii). Consequently, we have the following theorem.

Theorem 2. *In the signature generation of our scheme, the number of multiplication of K needs at most*

$$\frac{5}{4}r^4 + 2r^3 - \frac{61}{4}r^2 - 12r.$$

The term contributing to (ii) is cubic orders of r. On the other hand, the terms contributing to (i) and (iv) are quartic orders of r. Therefore, the inverse of the central map is computed using fewer multiplications than those of affine transformations.

6.2 Comparison of Efficiencies

Here, we compare the efficiency of the signature generation of our scheme and with that of Rainbow. We will not describe the scheme of Rainbow in any detail, save for the notation that is necessary to describe the number of multiplications of K.

Let n, t be natural numbers (t is called the layer number.) Let $v_1, \ldots, v_{t+1}$ be natural numbers satisfying

$$0 < v_1 < v_2 < \cdots < v_t < v_{t+1} = n.$$

For $i = 1, \ldots, t$, we write $o_i = v_{i+1} - v_i$ and $m = n - v_1$. This defines Rainbow with a parameter $(v_1, o_1, \ldots, o_t)$[10],

$$\text{Rainbow}(K; v_1, o_1, \ldots, o_t).$$

This is a signature scheme using a system of multivariate polynomials with n variables and m polynomials.

The signature of Rainbow$(K; v_1, o_1, \ldots, o_t)$ is generated as follows: For a message $\mathbf{M} \in K^m$, (R1) compute $\mathbf{A} = L^{-1}(\mathbf{M})$, (R2) $\mathbf{B} = G^{-1}(\mathbf{A})$, and (R3) $\mathbf{C} = R^{-1}(\mathbf{B})$. Here, G is the central map of Rainbow, and L and R are affine transformations. Therefore, the number of multiplications of $\mathbf{A} = L^{-1}(\mathbf{M})$ and $\mathbf{C} = R^{-1}(\mathbf{B})$ are the same as those of the affine transformations of our scheme. The number of multiplications in the computation of (R2) is estimated as follows:

$$\sum_{h=1}^{t} \left(\frac{o_h v_h^2}{2} + \frac{o_h^3}{3} + (v_h + 1)o_h^2 + \frac{3o_h v_h}{2} + \frac{v_h(v_h + 1)}{2} - \frac{o_h}{3} \right).$$

The $o_h v_h^2/2$ part is the cost for setting the linear equations to compute the inverse of the central map of Rainbow. The $o_h^3/3$ part is the cost for solving the linear equations with the $o_h \times o_h$-matrix. Using these results, we can compare the efficiency of signature generation. A signature generated with two layers and v_1, o_1, o_2 whose values are almost the same is often used in Rainbow[9,24]. Therefore, we can choose $v_1 = o_1 = o_2$ for simplicity. Moreover, we can determine other parameters based on the security against direct attacks. For our scheme to be more secure than Rainbow against direct attacks, it is sufficient that the number of polynomials of our scheme equals the number of polynomials of Rainbow, and the number of variables of our scheme is in fact greater than or equal to the number of variables of Rainbow. We can choose the parameter r of our scheme, and the parameters of Rainbow as $v_1 = o_1 = o_2 = r(r + 1)/4$. The number of polynomials of these schemes are the same. Here, we have assumed that $r(r + 1)/4$ is an integer. The number of variables of our scheme is equal to r^2, and that of Rainbow is equal to $3r(r + 1)/4$. If $r \geq 3$ then the condition placed upon the number of variables is satisfied. We call this Rainbow, Rainbow A, and we compare it with our scheme.

The number of multiplications of K in the computation of (R2) is estimated as follows.

$$\frac{37}{348}V^3 + \frac{1}{2}V^2 + \frac{5}{24}V \quad (V = r(r+1)).$$

We needs the following multiplications in (R1) and (R3).

$$\left(\frac{3}{4}V\right)^2 + \left(\frac{1}{2}V\right)^2 = \frac{17}{16}V^2 \quad (V = r(r+1)),$$

respectively. The total number of multiplications in the signature generation of Rainbow A is equal to

$$\frac{37}{384}r^3(r+1)^3 + \frac{25}{16}r^2(r+1)^2 + \frac{5}{24}r(r+1).$$

Comparing this with Theorem 2, we can see that the orders of r of these schemes are different. For our scheme, the maximal order is 4: on the other hand, it is 6 for Rainbow A. To compute the inverse of the central map of Rainbow A, we need to solve linear equations of size $O(r^2)$; the cost is $O(r^6)$. On the other hand, to compute the inverse of the central map of our scheme, we need to diagonalize a matrix of size $O(r)$; the cost is $O(r^3)$. In addition, the cost of computing of affine transformations is $O(r^4)$. That is, the affine transformations are more computationally expensive than that the inverse of the central map. As r grows, the signature generation of our scheme becomes more efficient than that of Rainbow A. The tables below compares the efficiencies of these signature generations as well as the secret key and public key lengths for $r = 8$ and 11. The efficiency of the signature generation is estimated in terms of the number of multiplications of K. The secret key and public key lengths are represented by the number of elements of K. Tables 1 and 2 indicate that the signature generation of our scheme is eight to nine times more efficient than that of Rainbow A. The secret key of our scheme is shorter than that of Rainbow A; however, the public key of our schemed is longer.

From [24], Rainbow A with $K = GF(31)$ and $r = 8$ has the same security level against direct attacks as a symmetric key with 88-bits. In the case of $K = GF(31)$ and $r = 11$, Rainbow A with $r = 6$ is more secure against direct attacks than a symmetric key with 140-bits. In this case, our scheme will have higher security level against direct attacks than 88-bits if $r = 8$, and 140-bits if $r = 11$. Note that in our scheme, the order of K may have to be lager than 31 to be secure against MinRank attack. Concretely, K needs an order more than 2048 for $r = 8$, and with more than 6781 for $r = 11$.

Table 1. Efficiencies of Our Scheme and Rainbow ($r = 8$)

	Our scheme	Rainbow A
m	36	36
n	64	54
Efficiency of the signature generation	5072	44079
Secret key length	5532	38520
Public key length	154440	55400

Table 2. Efficiencies of Our Scheme and Rainbow ($r = 11$)

	Our scheme	Rainbow A
m	66	66
n	121	99
Efficiency of the signature generation	18733	248864
Secret key length	19305	219120
Public key length	990396	333300

7 Concluding Remarks

We proposed a new construction of Rainbow using quadratic forms. Our scheme uses a non-surjective multivariate map as a central map. Since previously proposed signature schemes in MPKC use surjective multivariate maps, the public key of our scheme can not be described by the public keys of other signature schemes in MPKC. The signature generation of our scheme is eight to nine times more efficient than that of Rainbow at the level of 88-bit security.

In the future, we will analyze the security of our scheme more elaborately.

Acknowledgements. This work was partially supported by the Japan Science and Technology Agency (JST) Strategic Japanese-Indian Cooperative Programme for Multidisciplinary Research Fields, which aims to combine Information and Communications Technology with Other Fields. The first author is supported by Grant-in-Aid for Young Scientists (B), Grant number 24740078. The authors would like to thank Prof. Tsutomu Matsumoto for helpful comments on Graph Isomorphism. Dr. Yasufumi Hashimoto read carefully the preliminary version of this paper, and pointed out that MinRank attack can be applied to our scheme. The authors would like to thank him.

References

1. Anshel, I., Anshel, M., Goldfeld, D.: An Algebraic Method for Public-Key Cryptography. Math. Res. Lett. 6(3-4), 287–291 (1999)
2. Bernstein, D.J., Buchmann, J., Dahmen, E.: Post Quantum Cryptography. Springer, Heidelberg (2009)

3. Berger, T.P., Cayrel, P.-L., Gaborit, P., Otmani, A.: Reducing Key Length of the McEliece Cryptosystem. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 77–97. Springer, Heidelberg (2009)
4. Billet, O., Gilbert, H.: Cryptanalysis of Rainbow. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 336–347. Springer, Heidelberg (2006)
5. Boneh, D., Durfee, G.: Cryptanalysis of RSA with Private Key d Less Than $N^{0.292}$. IEEE Trans. Inform. Theory 46(4), 1339–1349 (2000)
6. Bouillaguet, C., Fouque, P.-A., Véber, A.: Graph-Theoretic Algorithms for the Isomorphism of Polynomials Problem. IACR Cryptology ePrint Archive Report 2012/607
7. Bouillaguet, C., Faugère, J.-C., Fouque, P.-A., Perret, L.: Practical Cryptanalysis of the Identification Scheme Based on the Isomorphism of Polynomial with One Secret Problem. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 473–493. Springer, Heidelberg (2011)
8. Chen, A.I.-T., Chen, M.-S., Chen, T.-R., Cheng, C.-M., Ding, J., Kuo, E.L.-H., Lee, F.Y.-S., Yang, B.-Y.: SSE Implementation of Multivariate PKCs on Modern x86 CPUs. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 33–48. Springer, Heidelberg (2009)
9. Ding, J., Gower, J.E., Schmidt, D.S.: Multivariate Public Key Cryptosystems. Advances in Information Security, vol. 25. Springer (2006)
10. Ding, J., Schmidt, D.: Rainbow, a New Multivariable Polynomial Signature Scheme. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 164–175. Springer, Heidelberg (2005)
11. Ding, J., Yang, B.-Y., Chen, C.-H.O., Chen, M.-S., Cheng, C.-M.: New Differential-Algebraic Attacks and Reparametrization of Rainbow. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 242–257. Springer, Heidelberg (2008)
12. Farb, B., Dennis, K.: Noncommutative Algebra. In: Graduate Texts in Mathematics. Springer (1993); ACNS 2008
13. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman & Co., Ltd. (1979)
14. Galbraith, S.D., Ruprai, R.S.: Using Equivalence Classes to Accelerate Solving the Discrete Logarithm Problem in a Short Interval. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 368–383. Springer, Heidelberg (2010)
15. Goubin, L., Courtois, N.T.: Cryptanalysis of the TTM Cryptosystem. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 44–57. Springer, Heidelberg (2000)
16. Hashimoto, Y., Sakurai, K.: On Construction of Signature Schemes based on Birational Permutations over Noncommutative Rings. In: Proceedings of the 1st International Conference on Symbolic Computation and Cryptography (SCC 2008), pp. 218–227 (2008)
17. Ko, K.H., Lee, S.-J., Cheon, J.H., Han, J.W., Kang, J.-S., Park, C.-S.: New Public-Key Cryptosystem Using Braid Groups. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 166–183. Springer, Heidelberg (2000)
18. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced Oil and Vinegar Signature Schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999)
19. Kipnis, A., Shamir, A.: Cryptanalysis of the Oil & Vinegar Signature Scheme. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 257–266. Springer, Heidelberg (1998)

20. Lin, D., Faugère, J.-C., Perret, L., Wang, T.: On Enumeration of Polynomial Equivalence Classes and Their Application to MPKC. Cryptology ePrint Archive: Report 2011/055
21. Faugère, J.-C., Levy-dit-Vehel, F., Perret, L.: Cryptanalysis of MinRank. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 280–296. Springer, Heidelberg (2008)
22. van Oorschot, P.C., Wiener, M.J.: Parallel Collision Search with Cryptanalytic Applications. Journal of Cryptology 12, 1–28 (1999)
23. Petzoldt, A., Bulygin, S., Buchmann, J.: A Multivariate Signature Scheme with a Partially Cyclic Public Key. In: Proceedings of the Second International Conference on Symbolic Computation and Cryptography (SCC 2010), pp. 229–235 (2010)
24. Petzoldt, A., Bulygin, S., Buchmann, J.: Selecting Parameters for the Rainbow Signature Scheme. In: Sendrier, N. (ed.) PQCrypto 2010. LNCS, vol. 6061, pp. 218–240. Springer, Heidelberg (2010)
25. Petzoldt, A., Bulygin, S., Buchmann, J.: CyclicRainbow – A Multivariate Signature Scheme with a Partially Cyclic Public Key. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 33–48. Springer, Heidelberg (2010)
26. Petzoldt, A., Bulygin, S., Buchmann, J.: Linear Recurring Sequences for the UOV Key Generation. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 335–350. Springer, Heidelberg (2011)
27. Faugère, J.-C., Perret, L.: Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 30–47. Springer, Heidelberg (2006)
28. Patarin, J., Goubin, L., Courtois, N.T.: Improved Algorithms for Isomorphisms of Polynomials. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 184–200. Springer, Heidelberg (1998)
29. Petzoldt, A., Thomae, E., Bulygin, S., Wolf, C.: Small Public Keys and Fast Verification for $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Public Key Systems. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 475–490. Springer, Heidelberg (2011)
30. Pollard, J.M.: Monte Carlo Methods for Index Computation mod p. Mathmatics of Computation 143(32), 918–924 (1978)
31. Rai, T.S.: Infinite Gröbner Bases and Noncommutative Polly Cracker Cryptosystems. PhD Thesis, Virginia Polytechnique Institute and State Univ. (2004)
32. Scharlau, W.: Quadratic and Hermitian Forms. Springer (1987)
33. Wiener, M.J.: Cryptanalysis of Short RSA Secret Exponents. IEEE Trans. Inform. Theory 36(3), 553–558 (1990)
34. Yang, B.-Y., Chen, J.-M.: Building Secure Tame-like Multivariate Public-Key Cryptosystems: The New TTS. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 518–531. Springer, Heidelberg (2005)
35. Yang, B.-Y., Chen, J.-M.: All in the XL Family: Theory and Practice. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 67–86. Springer, Heidelberg (2005)
36. Yang, B.-Y., Chen, J.-M.: All in the XL Family: Theory and Practice. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 67–86. Springer, Heidelberg (2005)
37. Yasuda, T., Sakurai, K.: A security analysis of uniformly-layered Rainbow — Revisiting Sato-Araki's Non-commutative Approach to Ong-Schnorr-Shamir Signature Towards PostQuantum Paradigm. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 275–294. Springer, Heidelberg (2011)

Author Index