IST-2002-507932

ECRYPT

European Network of Excellence in Cryptology

Network of Excellence
Information Society Technologies

D.AZTEC.2

Alternatives to RSA
(Lightweight Asymmetric Cryptography and Alternatives to RSA)

Due date of deliverable: 30. July 2005

Start date of project: 1 February 2004
Duration: 4 years

Lead contractor: Institute for Experimental Mathematics (IEM)

Revision 1.2
Alternatives to RSA

(Lightweight Asymmetric Cryptography and Alternatives to RSA)

Editor
Roberto Avanzi (RUB)

Contributors
Roberto Avanzi (RUB), Gebhard Böckle (IEM), An Braeken (KUL), Carlos Cid (RHUL), Katharina Geissler (BRIS), Rob Granger (BRIS), Tanja Lange (DTU), Arjen K. Lenstra (TUE), Phong Q. Nguyen (ENS), Bart Preneel (KUL), Nicolas Sendrier (INRIA), Nigel. P. Smart (BRIS), Martijn Stam (BRIS), Jacques Stern (ENS), Christopher Wolf (KUL)

Revision 1.2
## Contents

1 Introduction ................................. 1
   References – Introduction .................... 5

2 Curve-Based Cryptosystems .......... 9
   2.1 Introduction .......................... 9
   2.2 Description ........................... 10
      2.2.1 Basic definitions ................. 10
      2.2.2 Group operation .................. 11
         2.2.2.i Elliptic Curves ............... 11
         2.2.2.ii Curves of Larger Genus ....... 12
      2.2.3 Compression ....................... 13
      2.2.4 Use in protocols .................. 13
   2.3 Finding Curves ......................... 14
      2.3.1 Odd characteristic ............... 14
      2.3.2 Even characteristic .............. 14
   2.4 Security ................................ 15
   2.5 Choice of parameter .................... 16
   2.6 Other related systems and special choices ................. 17
   2.7 Conclusions ........................... 17
   References – Curve-based Cryptosystems ................. 18

3 Lattice-based Cryptography ........ 23
   3.1 Introduction .......................... 23
   3.2 Lattice problems ...................... 24
      3.2.1 Definitions ...................... 24
      3.2.2 Algorithmic problems .......... 26
      3.2.3 Complexity results ............. 27
      3.2.4 Algorithmic results .......... 27
   3.3 Lattice-based cryptography .......... 28
      3.3.1 The Ajtai-Dwork cryptosystem .... 28
         3.3.1.i Description ................ 28
3.3.1.ii Security ........................................... 29
3.3.1.iii Cryptanalysis overview ............................ 29
3.3.2 The Goldreich-Goldwasser-Halevi cryptosystem .... 30
3.3.2.i Description ........................................ 30
3.3.2.ii Improvements ..................................... 30
3.3.2.iii Security ......................................... 31
3.3.3 The NTRU cryptosystem ................................ 32
3.4 Conclusions ............................................. 32
References – Lattice-based Cryptography .................. 32

4 The NTRU Public-Key Cryptosystem ......................... 39
4.1 Introduction ............................................ 39
4.2 The NTRU Cryptosystem ................................ 40
4.2.1 Notation ............................................. 40
4.2.2 The NTRU Primitive ................................ 41
4.2.3 Parameter Choices .................................. 46
4.2.4 Implementation Details .............................. 47
4.3 NTRU and Lattice Reduction .............................. 48
4.4 Security ................................................ 50
4.5 Conclusion .............................................. 53
References – NTRU .......................................... 53

5 XTR, Subgroup- and Torus-based Cryptography ............ 55
5.1 Introduction ............................................. 55
5.2 XTR .................................................... 57
5.2.1 Classical XTR ...................................... 57
5.2.1.i Description of XTR ............................... 58
5.2.1.ii Implementation aspects .......................... 59
5.2.1.iii Applications .................................... 61
5.2.2 Subgroup XTR ....................................... 63
5.3 CEILIDH ................................................. 66
5.3.1 The torus \( T_n(\mathbb{F}_p) \) ............................ 67
5.3.2 Rationality of tori over \( \mathbb{F}_q \) ..................... 67
5.3.3 CEILIDH construction

5.3.4 Implementing CEILIDH
  5.3.4.i The Representation $F_1$
  5.3.4.ii The Representation $F_2$
  5.3.4.iii The Representation $F_3$
  5.3.4.iv Exponentiation

5.4 Higher Dimension Tori

5.4.1 Asymptotically Optimal Torus-Based Cryptography

5.4.2 The New Construction
  5.4.2.i Applying the Construction to $T_{30}$
  5.4.2.ii Missing points

5.4.3 Applications

5.4.4 Representations and Algorithms for $T_{30}$
  5.4.4.i Field Representations
  5.4.4.ii Compression and Decompression
  5.4.4.iii Arithmetic Costs
  5.4.4.iv Exponentiation in $T_{30}$

5.4.5 Security

5.5 Conclusions

References – XTR, Subgroup- and Torus-based Cryptography

6 Multivariate Systems

6.1 Hidden Field Equations in Asymmetric Cryptography

6.1.1 Hidden Field Equations
  6.1.1.i Mathematical Background
  6.1.1.ii Variations

6.1.2 Quartz
  6.1.2.i Historical Note
  6.1.2.ii System Parameters
6.1.2.iii System Description ........................................ 96
6.1.2.iv Conclusions ............................................. 99
6.1.3 Attacks ......................................................... 99
   6.1.3.i Kipnis-Shamir: Recover the Private Key .................. 99
   6.1.3.ii Faugère: Fast Gröbner Bases .......................... 99
   6.1.3.iii Secure Versions of Quartz ............................ 100
6.1.4 Conclusions ............................................... 100
6.2 Enhanced TTM Multivariate Signature Schemes .................... 101
   6.2.1 Enhanced TTM ............................................. 102
      6.2.1.i Design Criteria ...................................... 102
      6.2.1.ii Enhanced TTS scheme ................................ 103
      6.2.1.iii Scaled-Up Schemes .................................. 104
      6.2.1.iv Attacks ............................................. 105
   6.2.2 Conclusion .............................................. 106
6.3 Unbalanced Oil and Vinegar Multivariate Signature Scheme ....... 106
   6.3.1 The Unbalanced Oil and Vinegar Schemes .................... 106
      6.3.1.i Security Parameters .................................. 107
      6.3.1.ii Key-size ........................................... 107
      6.3.1.iii Signatures ......................................... 108
      6.3.1.iv UOV/: Restricting to a Subfield ...................... 109
   6.3.2 Cryptanalysis ............................................ 109
      6.3.2.i Kipnis and Shamir Attack ............................. 109
      6.3.2.ii Improved Relinearization Attacks .................... 109
      6.3.2.iii Solving IP ......................................... 110
   6.3.3 Conclusions ............................................. 110
References – Multivariate Cryptosystems .......................... 111
7 Code-based Cryptosystems ........................................ 115
   7.1 Introduction .............................................. 115
   7.2 Description ................................................ 116
   7.3 Security .................................................... 117
      7.3.1 Theoretical security ................................... 117
      7.3.2 Best known attacks .................................... 118
7.3.2.i Decoding attacks ........................................ 118
7.3.2.ii Structural attacks ...................................... 118
7.4 Choosing the parameters .................................... 119
7.5 Other related systems ......................................... 120
7.6 Conclusions .................................................. 120
References – Code-based Cryptosystems ....................... 121

8 Cryptosystems based on Drinfeld Modules .................. 123
8.1 Introduction .................................................. 123
8.2 Drinfeld Modules ............................................. 123
8.3 Cryptosystems based on Drinfeld Modules .................. 124
  8.3.1 A realization and a simple cryptosystem ................. 124
  8.3.2 The general case and cryptanalysis ...................... 125
8.4 Conclusions .................................................. 127
References – Cryptosystems based on Drinfeld Modules .......... 128
1 Introduction

The Current Situation

Public key cryptography is now a well established technology: For example there are millions of fielded products with built-in Rivest-Shamir-Adleman (RSA) cryptography [RSA78]. However, the RSA system is not the only existing cryptosystem, not the only one in use, nor, for many applications, the most efficient cryptosystem available with respect to the security level achieved by a given amount of computational resources.

In the 27 years passed since the introduction of RSA, cryptosystems based on elliptic [Mil85, Kob87] and hyperelliptic [Kob89] curves, algebraic codes [McE78, Nie86], multivariate quadratic equations [MaIm88, Pat96, KPG99], and polynomial factorization such as NTRU [HPS96, HPS98], have been proposed, as well as several others. Many of these cryptosystems have some advantages with respect to RSA, either with respect to speed, or to availability of protocols, or ease of set-up. To regulate the use of some of these cryptosystems there exist established standards, for example for Elliptic Curve Cryptography (ECC) [P1363, FIPS186-2], and for NTRU [CEES02]. Also, cryptographic hardware, including coprocessors for smart cards, for performing the fundamental operations of ECC has already hit the market. In the meantime, the RSA system has been shown to provide less security than initially believed. Still, RSA has remained unchallenged among the public key cryptosystems in terms of dissemination.

The Aging of RSA

As we already mentioned, RSA, as well as systems designed around the discrete logarithm problem (DLP) in the multiplicative group of a finite field, such as the original ElGamal
1. Introduction

cryptosystem \[ElG85\], started quite soon to age. The main reason has been the tremendous progress in integer factoring methods and, because of the the similarities of some of the techniques involved, in solving the DLP in finite fields. In fact, few computational problems have seen successes as spectacular as those in integer factorisation. (For a survey of the history of integer factoring methods and for discrete logarithm computations in finite fields, as well as in varieties, see, for example, Chapters 19, 20, 21, 22 and 25 of [ACD+05].) Recently, a team lead by Jens Franke and Thorsten Kleinjung, on the wave of their successful factorisation of the RSA-576 challenge (announced on December, 3, 2003), has been able to factor the RSA-200 number (announced on May 10, 2005), a 662 bit integer.

The popular 1024-bit key size for RSA keys is becoming the next horizon for researchers in integer factorization, and some hardware designs have been proposed to tackle such problems. The first such design has been Adi Shamir’s TWINKLE \[Sha99\] (“The Weizmann Institute Key Locating Engine”) sieving device. Never built, based on non-conventional electro-optical components for performing analog summation. It was analysed in [LS00]. The second such device was TWIRL (“The Weizmann Institute Relation Locator”) designed by Adi Shamir and Eran Tromer \[ShTr03a, ShTr03b\]. It has been analysed in [LTS+03]. Other devices include Mesh-based Sieving \[GeSt03, GeSt04\] (based on ideas of Bernstein [Ber01] further developed in [LSTT02]) and the recent SHARK \[FKP+05\] device. SHARK applies a butterfly network for communication between the computers and is based on conventional computers such that it might be realizable for 1 million dollars to factor 1024 bit RSA moduli. In February 2005 the SHARCS workshop (\[http://www.sharcs.org\]), the ECRYPT workshop on Special Purpose Hardware for Attacking Cryptographic Systems, was organized. There, a good overview of the state of the art in hardware crackers was given.

The need for ever increasing levels of security together with the improvements of techniques for factoring integers and solving the discrete logarithm problem in finite fields (these are all sub-exponential attacks) has led to the requirements of keys of 1000 bits or even more. On elliptic curve cryptosystems, and in general on cryptosystems designed round the DLP in the Jacobian variety of a hyperelliptic curve of low genus, there are no known sub-exponential attacks. Similar favorable considerations apply to other primitives, such as lattice-based ones. This means that, for example, as security demands increase, the lengths of the key sizes for ECC or for lattice-based cryptography increase much slower than the key sizes for RSA or for cryptosystems based on the DLP in finite fields. Since there is an obvious correlation between key size and performance for a given cryptosystem, it is clear that RSA could soon become impractical, and that alternative systems will offer better performance and security at the same time. As a matter of fact, implementing high-security RSA on embedded systems is nowadays a difficult technological challenge, despite the steady hardware developments of the last years. Even when short key variations are employed, it will be soon no longer possible to consider RSA as a lightweight cryptosystem.

The Need for New Technologies

Indeed, an important requirement for a practical cryptosystem is its speed. Whereas for specific applications a highly secure, yet slow cryptosystems can be used, for a general deployment a system must make reasonable use of the available resources such as memory, speed, bandwidth. Lightweight cryptosystems are therefore desirable. We already know a
few systems that offer excellent performance, such as NTRU and Drinfeld Module based systems, but their security history has been plagued by many accidents and the construction of secure instances is still an active area of research. Recent work seems to suggest that a fast, yet efficient, NTRU-based system is feasible (see Chapter 4). In particular NTRU has the advantage that encoding and decoding takes $O(N^2)$ operations for a message block of length $N$ (comparable to the key size) – and even $O(N \log N)$ operations if $N$ is large enough that the Fast Fourier Transform can be used. This is a major advantage compared to RSA, which needs $O(N^3)$ operations (note, however, that in practice low-exponent RSA is used, which means that decryption and signing are done with $O(N^3)$ operations, but encryption and signature verification is $O(N^2)$), and even compared to ECC (where the complexity is also cubic, but the keys are smaller).

At the moment we do not know whether secure systems based on Drinfeld Modules (see Chapter 8) can be built at all, but, since the deep reasons for the failure of the proposals so far published are not clear, we still decided to include a short treatment of them in this deliverable. In fact, until now Drinfeld modules have been used in practice to disguise a linear algebra cryptosystems, and a more profound use of their structure could find applications. As for cryptosystems based on the conjugally problem in braid groups [Deh97, AAC99, CHK99, DGS03, Deh04], they have been repeatedly broken [HuTa00, Hug02, LeLe02, FGM03] and fixed, until the word and conjugacy problems in the underlying groups has been shown computationally easier than many people had expected, possibly breaking the approach definitively [BKL98, BKL01, Geb03, Ban03]: Thus, they were not considered here.

As recalled in the conclusions to Chapter 3 there has been no massive computer experiment in lattice reduction comparable to what has been done for integer factorization or the elliptic curve discrete logarithm problem – pushing the state-of-the art and developing specific optimisation techniques also for lattice reduction would be a very welcome line of research. For other primitives such as the DLP in Jacobians of low genus hyperelliptic curves this line of research is just beginning to be explored (a taster of this can be seen in the algorithmic improvements described in Chapters 20 and 21 of the already cited handbook [ACD05]). We hope that the present report, as well as the research done by ECRYPT, will foster further similar massive computational attempts, which are necessary to precisely assess the security of practical cryptosystems. Too often the complexity of an attack is simply given by the order or magnitude, in the familiar $O$-notation, yet the constants are difficult to determine but with concrete experiments on instances of the hard problems being tested which are of sizes at least close to those proposed for real-world applications.

Furthermore, there might be other technological advancements lurking in the future that could impair the security of most known cryptosystems. Any advance in quantum computation will lead, via Shor’s algorithm [Sho97], to the need for public key cryptosystems based on problems other than factoring and discrete logarithms in abelian groups. So far there has been no quantum algorithm which significantly improves on the classical algorithms for finding shortest vectors in lattices or for solving any combinatorial NP-hard problem. Hence, a breakthrough in quantum computer technology would make systems designed around the discrete logarithm and factorisation problems obsolete (this includes, among others, RSA and ECC). NTRU, Lattice-based, code-based and multivariate cryptosystems would probably survive, and of these NTRU seems to be the most efficient – however it is not yet clear how to
turn the mathematically secure hard problem into a secure primitive (cf. Chapter 4).

The Scope of the Present Deliverable

This deliverable is devoted to the description of the main alternatives to RSA which are currently known to be secure, at least in principle. Primitives for which no insecurity proof exists, despite difficulties in producing secure instances, are also considered. Lightweight primitives are a central concern of our survey.

For each primitive the underlying mathematical problem is presented, a description of the use of primitive itself is given (possibly in several concrete and practical instances) and the results on their security are reviewed, together with some conclusions.

We begin our survey with curve-based cryptosystems, in Chapter 2. Treatments of lattice-based cryptology and of NTRU (which can be viewed as a special lattice-based cryptosystems) follow in Chapters 3 and 4 respectively. Chapter 5 is devoted to XTR and to Subgroup- and Torus-based cryptography, including the CEILIDH systems and the recent development in higher dimension tori. Multivariate systems are the subject matter of Chapter 6: In particular systems like HFE, Quartz, TTM and Unbalanced Oil and Vinegar Signature schemes for the bulk of the treatment. In Chapter 7 code-based cryptosystems are described, and Drinfeld Modules conclude the report in Chapter 8.

Bibliographic references are given at the end of each Chapter for ease of consultation. The overlap between these bibliographies is minimal.

A Short Summary of the Conclusions

Curve-based cryptosystems are one of the most versatile alternatives to RSA. They are based on very well investigated problems, and their security is fairly well understood. They can be used in a extremely broad range of protocols and offer very good speed. There has been also extensive research in securing implementations of these systems against side channel attacks and fault analysis. Systems based on low genus hyperelliptic curves offer performance often comparable to that of systems based on elliptic curves, and are sometimes faster: This achievement is in part fruit of a previous E.U. funded project, the AREHCC Project (http://www.arehcc.com), specifically devoted to Advanced Research in Elliptic and Hyperelliptic Curve Cryptography – and which also motivated the writing of a large handbook on the subject [ACD+05]. Since hyperelliptic curve cryptosystems are not affected by the ECC patents Certicom and acquired by the NSA, they represent a good choice for PKC infrastructure in the european countries. However, a significant breakthrough in quantum computation would spell doom for all these systems.

We have already mentioned that lattice-based systems and NTRU offer, in principle, very good speed, and are, at least in principle, among the cryptosystems which would survive the development of fairly sized quantum computers, because they are based on problems for which no quantum algorithm is known for the general case. However, it is difficult to create instances which are secure even under a classical computing model, and the complexity of the classical lattice reduction algorithms is not well understood. There is little on the way of
XTR, Subgroup- and Torus-based cryptography is an interesting and elegant attempt to revive classical ElGamal cryptography in finite fields by exploiting groups which possess special properties. Security is fairly well understood, quantum computing would however render them obsolete. Their performance is (for current key sizes) a bit slower than that of elliptic curves, but the existence of sub-exponential classical algorithms for the DLP in finite fields makes these systems shortly-lived anyway. They are easy to implement and would ideally suit embedded devices. Some side-channel-related research has been performed.

Multivariate quadratic systems can be used to construct both secure and efficient public key schemes. Their main problem is the key size, which can easily go to several hundreds of kilobytes. In particular, promising are the Unbalanced Oil and Vinegar (UOV) schemes. The attacks known so far against UOV are basically exponential – in particular they do not fall to the same kind of attacks that have plagues earlier schemes like HFE. Hence, it is necessary a very high workload for breaking systems with reasonably small parameters. This leads to a public key size of 18 kB. There are no known quantum algorithms to break these systems. Very little on implementation issues has been done.

Large key sizes are also a downside of Code-based cryptosystems, but they offer the advantage of providing very short signatures, even though signing is still quite slow. Encryption is very fast and can use a relatively small block size, since all known attacks have a fully exponential complexity in the block length. There are also no known quantum algorithms to break these systems. The implementor can profit from the great experience in the implementation of algebraic codes. Hardware implementations have been done, too (FPGA).

We already mentioned the problems that plague the idea of Drinfeld module based cryptosystems. They could provide fast systems, if fixed, and easy to implement.

The situation is quickly developing on all fronts and there are many open questions. Classical attacks, quantum attacks, implementation issues, correct performance assessments by means of large software experiments and development of cryptoanalytic hardware are all active areas of research. A first attempt to delineate a comparative benchmarking strategy is being done with another ECRYPT deliverable, the Vampire WG1 Report on Performance Benchmarks D.VAM.1 [VAMI]. For the state of crypto after quantum computing, ECRYPT is also organizing a workshop, which is still in quite early stages (see http://postquantum.cr.yp.to/). Also the other areas are subject of ECRYPT research, as testified by the many recently published papers in these fields: for them we refer the reader to the individual chapters.

Roberto Avanzi

References – Introduction

1. Introduction


References – Introduction


1. Introduction


2 Curve-Based Cryptosystems

Contributors: Tanja Lange and Roberto Avanzi

2.1 Introduction

Systems based on the discrete logarithm problem in the Jacobian of curves over finite fields were suggested quite early in the history of public key cryptography: in 1985, Miller [Mil86] and Koblitz [Kob87] proposed elliptic curves and shortly after [Kob89] Koblitz investigated the Jacobian of hyperelliptic curves. Since then a lot of research was devoted to the study of efficient implementations of curve based cryptography and at the same time attempts to break such systems were made.

So far no subexponential algorithm for solving the discrete logarithm problem (DLP) on elliptic (EC) and hyperelliptic curves (HEC) of genus at most 4 was found; For curves defined over a fixed field and large, increasing genus the complexity of solving the discrete logarithm becomes subexponential in the group order [Eng02] by using index calculus methods, for "small", fixed genus the complexity of the methods remains exponential. In particular, for elliptic curves the complexity of the index calculus methods is linear in the group order, hence no better than brute force; for curves of genus 2 the complexity is $O(\sqrt{n})$ where $n$ is the order of the rational point group of the curve, i.e. the same as Pollard’s methods, but the constants are usually much worse in the index calculus; for hyperelliptic genus 3 curves generic algorithms are faster for current parameter sizes. Starting with genus 4 curves one has to take into account a loss of security bit by a constant factor, and therefore one as to increase the field and group sizes.

Therefore, systems based on curves of genus at most 3 (for some applications, genus 4 curves may still prove interesting) offer much shorter key sizes compared to RSA for equal security requirements, e.g. RSA with 1024 bits corresponds to ECC (and HEC with genus 2 or 3) with 160 bits (cf. e.g. [ECR05]).

Elliptic curves belong to the most studied cryptosystems and because of their bandwidth and security properties gained a noticeable share in applications: at first with a slow acceptance and in the last years with growing momentum. Special curves for which the Tate pairing can be computed efficiently have recently found a lot of applications, the ECRYPT report on “New Trends in Asymmetric Cryptography” [AZTEC] considers ID-based cryptography, short signatures and searchable encryption to name just a few.

Until recently, hyperelliptic curves have been considered not competitive with respect to elliptic curves [Sma99b] because of the difficulty of finding suitable curves and their poor performance with respect to EC. In the subsequent years the situation changed.

Firstly, it is now possible to efficiently construct genus 2 and 3 HEC whose Jacobian has almost prime order of cryptographic relevance. Over prime fields one can either count points in genus 2 [GaSc04], or use the complex multiplication (CM) method for genus 2 [Mes91, Wen01] and 3 [Wen01].
Secondly, the performance of the HEC group operations has been considerably improved. For genus 2 the first explicit formulae for computing addition and doubling in the group were due to Harley [Har00], and since then there has been a plethora of advancements. The results have been recently summarized in [DoLa05a, DuLa05b], but of course research has not stopped: Recent improvements for genus 3 are in [FWW05] and further implementation improvements for genus 3 and 4 in [AvTh05].

HEC are attractive to designers of embedded hardware since they require smaller field than EC: The order of the Jacobian of a HEC of genus $g$ over a field with $q$ elements is $q^g$. This means that a 160-bit group is given by an EC with $q = 2^{160}$, by an HEC of genus 2 with $q = 2^{80}$, and genus 3 with $q = 2^{53}$.

Most of the major conferences have papers on curve based cryptography and the field is still growing – but the kernel is very stable since years. The state of the art can be found in e. g. [BSS99, HMV03, BSS05, ACD+05].

2.2 Description

For this report we concentrate on elliptic and hyperelliptic curves. Generalizations to other curves are possible but so far no advantage in speed or security was identified.

2.2.1 Basic definitions

Definition 2.2.1 Let $K$ be a field. The projective curve defined by the affine equation
\[ C : y^2 + h(x)y = f(x), f, h \in K[x], \deg(f) = 2g + 1, \deg(h) \leq g \]  
(1)
is a hyperelliptic curve of genus $g$ if there is no point $P \in C(\overline{K})$ over the algebraic closure $\overline{K}$ such that both partial derivatives vanish simultaneously.

According to this definition we subsume elliptic curves as curves of genus 1 under hyperelliptic curves.

Example 2.2.2 An elliptic curve, i.e. a curve of genus 1, over $\mathbb{F}_2$ is given by
\[ y^2 + xy = x^3 + x^2 + 1 \]
as the only point for which the partial derivative with respect to $y$ vanishes is $(0,1)$ and it does not satisfy the partial derivative with respect to $x$.

The group of $K$-rational points on the Jacobian of a curve given by (1) is isomorphic to the ideal class group $\text{Cl}_C(K)$ of the function field $K(C)$. This leads to a compact representation of the group elements.

Definition 2.2.3 (Mumford representation)
Let $C$ be a genus $g$ hyperelliptic curve given by $y^2 + h(x)y = f(x)$, where $h, f \in K[x]$, $\deg f = 2g + 1$, $\deg h \leq g$. Each nontrivial group element $D \in \text{Cl}_C(K)$ can be represented via a unique pair of polynomials $u(x)$ and $v(x)$, $u, v \in K[x]$, where
2.2. Description

(i) $u$ is monic,

(ii) $\deg v < \deg u \leq g$,

(iii) $u \mid v^2 + vh - f$.

This means that a group element needs at most $2g$ field elements to be represented. In particular for an elliptic curve a generic element is represented by $D = [x - a, b]$, and this representation reduced to the usual representation of points via pairs of coordinates; the third condition implies that $P = (a, b)$ is a point on the curve (1). Therefore, each nontrivial element corresponds to an affine point and one can define a group law on the set of points. The zero element is the single point at infinity $P_\infty$. Some formulae for the group operation are given in the next section.

For the group size the Theorem of Hasse-Weil gives the estimate that for a curve $C$ of genus $g$ over a finite field with $q$ elements one has

$$\text{Cl}_C(F_q) = O(q^g)$$

elements in the ideal class group of the curve. This implies that the field size for a curve of genus 2 has half the bitlength compared to an elliptic curve to achieve the same group order.

2.2.2 Group operation

2.2.2.i Elliptic Curves For an elliptic curve of the form

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

the group operation on two points gives the following result: Let $P = (x_P, y_P), Q = (x_Q, y_Q)$, and $P \oplus Q = R = (x_R, y_R)$. Then

$$-P = (x_P, -y_P - a_1x_P - a_3),$$

$$P \oplus Q = (\lambda^2 + a_1\lambda - a_2 - x_P - x_Q, \lambda(x_P - x_Q) - y_P - a_1x_Q - a_3),$$

where

$$\lambda = \begin{cases} 
\frac{y_P - y_Q}{x_P - x_Q} & \text{if } P \neq \pm Q, \\
\frac{3x_P^2 + 2a_2x_P + a_4 - a_1y_P}{2y_P + a_1x_P + a_3} & \text{if } P = Q.
\end{cases}$$

Depending on whether the characteristic of $K$ is even or not different isomorphic transformations of the curve equation are possible. They lead to more zero coefficients in the curve equation, i.e. in characteristic two one can either achieve $a_1 = 1, a_4 = 0$ or $a_1 = 0, a_2 = 0$ and otherwise one gets $a_1 = a_3 = 0$ and for $\text{char}(K) \neq 3$ additionally $a_2 = 0$. For each of these cases specialized addition formulae exist and are faster than the general formulas stated in (2). Additionally, for each case there are other representations of the curve which allow to implement the group arithmetic without inversions. Usual choices are $F_q = \mathbb{F}_p$ a prime field with $p > 2^{160}$ and $F_q = \mathbb{F}_{2^p}$ with $p$ a prime $> 160$ and for these choices the formulas are optimized for efficient scalar multiplication. For an overview we refer to [DoLa05a].
2.2.2.ii Curves of Larger Genus  The following algorithm, due to Cantor [Can87], adds two group elements in the Jacobian variety of a hyperelliptic curve, given as divisors representing the divisor classes associated to the two group elements.

Algorithm 2.2.4 (Cantor’s Algorithm)

In:  \( D_1 = [u_1, v_1], D_2 = [u_2, v_2], C : y^2 + h(x)y = f(x) \)
Out:  \( D = [u, v] \) with \( D = D_1 + D_2 \)

1. \( d \leftarrow \gcd(u_1, u_2, v_1 + v_2 + h) = s_1u_1 + s_2u_2 + s_3(v_1 + v_2 + h) \)
2. \( u \leftarrow \frac{u_1u_2}{d^2} \)
   \( v \leftarrow \frac{s_1u_1v_2 + s_2u_2v_1 + s_3(v_1v_2 + f)}{d} \) mod \( u \)
3. DO
4. \( u \leftarrow \frac{f - v^2 - hv}{u} \), \( v \leftarrow (-h - v) \) mod \( u \)
5. WHILE (\( \deg u > g \))
6. Make \( u \) monic
7. RETURN \([u, v]\)

It consists, essentially, of two parts. In steps 1 to 3 (the so-called “composition step”) the two given elements are combined in their formal sum, which is correct as an element of the divisor class, but the degrees of the polynomials do not usually possess the correct degrees. In the remaining steps (the so-called “reduction step”) this sum is then brought to a form which satisfies all the properties of Definition 2.2.3.

Starting with this algorithm, that operates on polynomials, it is possible to derive explicit formulae that manipulate fields elements directly, as it has been done for elliptic curves. By means of this it is possible to perform many operations (for example, skipping the computation of unused coefficients or coalescing the computation of several field inverses into one by means of a trick attributed to Montgomery). The initiator of this approach Harley [Har00]. This in turn opens the way to specific optimisations of the field arithmetic designed for these formulae (see [Ava04, AvTh05]). A survey of the results until very recently is given in [DuLa05b] but, as already mentioned, research progresses.

Each group operation needs more operations as the genus increases but on the other hand the field size decreases correspondingly. A theoretical comparison depends very much on the relative costs of the field arithmetic. Therefore only implementations can give relevant results, and to obtain fair performance assessments one has to devote particular care to the implementation of the finite fields. For odd characteristic a comparison of software implementation is given in [Ava04]; for characteristic 2, a comparison of elliptic and genus 2 curves is [LaSt05], and a comparison of curves of genus up to 4 is in [AvTh05].
2.2. Description

2.2.3 Compression

On an elliptic curve there exist at most two points for a given \( x \) coordinate \( a \), namely the two solutions of
\[
y^2 + a_1 y + a_3 y = a^3 + a_2 a^2 + a_4 a + a_6.
\]
Therefore, a compressed representation is given by the \( x \)-coordinate and one bit to determine the correct solution of the quadratic equation. For larger genus curves similar ideas apply [HSS01, Sta04, Lan05c] and each element can be represented by the first polynomial \( u(x) \) of degree at most \( g \) and \( g \) bits to determine the correct choices of the coefficients of \( v(x) \) in Definition 2.2.3.

2.2.4 Use in protocols

The use of elliptic curves in protocols is already standardized [P1363], for curves of larger genus the suggestions hold \textit{mutatis mutandis}. In general the difference to generic DL based protocols consists in using specified maps between the group elements and bit strings and applications of compression. We present the ECDSA signature scheme [ANSI] based on elliptic curves as an example. Note that an actual implementation requires to check the correctness of the provided parameters such as that the points are in fact on the curve and that the claimed group order and cofactor are correct. The domain parameter are \( D = (q, FR, a, b, P, \ell, h) \), where \( q \) is the field size, \( FR \) gives information on the field representation, \( a, b \) correspond to \( a_2, a_6 \) in even characteristic and to \( a_4, a_6 \) otherwise, \( P \) is a point on the curve of prime order \( \ell \), and \( n = c\ell \) is the group order, where \( c \) is the cofactor.

Algorithm 2.2.5 (ECDSA – Signature generation)

In: message \( m \), entity \( A \) with private key \( d \) and public key \( Q = [d]P \), domain parameter \( D \).
Out: \( A \)'s signature \((r,s)\) on \( m \).

1. Select a random or pseudorandom integer \( k \), \( 1 \leq k \leq \ell - 1 \).
2. Compute \([k]P = (x_1, y_1)\) and \( r = x_1 \mod \ell \). If \( r = 0 \) goto step 1.
3. Compute \( k^{-1} \mod \ell \).
4. Compute \( e = \text{SHA-1}(m) \).
5. Compute \( e = k^{-1}(e + dr) \mod \ell \). If \( s = 0 \) goto step 1.
6. \( A \)'s signature for the message \( m \) is \((r,s)\).

Clearly the hash function SHA-1 can be replaced by any other hash function.

To verify a signature the following algorithm is used:

Algorithm 2.2.6 (ECDSA – Signature verification)

In: \( A \)'s signature \((r,s)\) on message \( m \) and \( A \)'s public key \( Q = [d]P \), domain parameters \( D \).
Out: Acceptance or Rejection of signature.

1. Verify that \( r \) and \( s \) are integers in the interval \([1, \ell - 1]\).
2. Compute $e = \text{SHA-1}(m)$.

3. Compute $w = s^{-1} \mod \ell$.

4. Compute $u_1 = ew \mod \ell$ and $u_2 = rw \mod \ell$.

5. Compute $X = [u_1]P \oplus [u_2]Q$. If $X = P_\infty$, then reject signature. Otherwise compute $v = x_1 \mod \ell$, where $X = (x_1, y_1)$.

6. Accept the signature if and only if $v = r$.

The scheme works correctly as

$$k \equiv s^{-1}(e + dr) \equiv s^{-1}e + s^{-1}dr \equiv we + wrd \equiv u_1 + u_2d \mod \ell$$

and thus $[u_1]P \oplus [u_2]Q = [(u_1 + u_2d)]P = [k]P$ and so $v = r$ as required.

Note that the $y$ coordinates are not used. In fact, the same signature belongs to two nonces $k$, namely $k$ and $\ell - k$. Security implications of this observation are pointed out in [SPM+02].

2.3 Finding Curves

2.3.1 Odd characteristic

For elliptic curves, Schoof’s algorithm and its improvements [Sch95, LLV05] allows one to efficiently find elliptic curves with group orders of 200 bits and more. It is a point counting algorithm, hence one has to try several curves of approximately the desired order (this can be done in view of the above mentioned Hasse-Weil theorem) and count the number of points on them until one has been found of “almost prime” order, i.e. the order is the product of a large prime number (of approximately the desired number of bits) times a small number. This kind of order factorization is the one suggested by standards.

For larger genus curves point counting in odd characteristic still poses a problem: for a genus 2 curve in the cryptographically relevant range one needs about a week to determine its group order, for larger genera no practical algorithm exists (see the second reference for an overview).

Another possibility is to use the complex multiplication (CM) method [AtMo93, LaZi94, Wen01, FrLa05b] which amounts to determining a suitable order first and then constructing the corresponding curve. It computes curves over finite fields with a known number of points as reductions from curves over number fields by using their invariants. Due to computational restrictions the discriminant of the number field cannot be too large so that these curves are special. So far no attack exploiting this fact is known but for security reasons one should use discriminants larger than 1000.

2.3.2 Even characteristic

Satoh [Sat00] showed how to count the number of points on an elliptic curve in small characteristic by computing a $p$-adic approximation of the Serre-Tate canonical lift and the action of
2.4. Security

Frobenius on this lift. An overview of the many variants and further optimisations of Satoh’s algorithm can be found in [Ver03, LLV05]. Mestre [Mes00] presented a “dual” algorithm using the Arithmetic-Geometric mean and sketched how it could be extended to ordinary hyperelliptic curves [Mes00]. Results by Lercier and Lubicz [LeLu03] show that this algorithm is very efficient as long as the genus is small; this is due to the exponential dependence on the genus – this is not a problem for the cryptographic applications, since, as we mentioned in the introduction, we shall limit ourselves to curves of genus at most 4.

A second strategy is based on the computation of the Frobenius action on $p$-adic cohomology groups. Kedlaya [Ked01] first described such an algorithm for hyperelliptic curves over finite fields of small odd characteristic using Monsky-Washnitzer cohomology. The average running time of the algorithm is $\tilde{O}(g^4 n^3)$ for a hyperelliptic curve of genus $g$ over $\mathbb{F}_p$. Denef and Vercauteren consider various extensions of Kedlaya’s algorithm to various families of curves defined over fields of characteristic 2, and their research recently culminated with a consideration of the case of hyperelliptic curves [DeVe05]. Their algorithm has the same complexity as Kedlaya’s.

A related approach by Lauder and Wan [LaWa02a] is based on Dwork’s $p$-adic proof of the rationality of the zeta function and results in a polynomial time algorithm to compute the zeta function of an arbitrary algebraic variety over a finite field. Despite its polynomial time complexity, first implementations seem to indicate that cryptographical sizes are still out of reach. Several extensions to special types of curves exist. In particular, Lauder and Wan [LaWa02b, LaWa04] developed a point counting method for Artin-Schreier covers based on Dwork-Reich cohomology. Their algorithm is slightly less general than the one by Denef and Vercauteren, but the additional conditions (which mostly boil down to the fact that the polynomials $f$ and $h$ in the definition have both at most simple zeros) should represent no real restriction in practice. For hyperelliptic curves of genus $g$ in characteristic 2, the complexity of the algorithm is not rigorously proven, but it is conjectured to be the same as Kedlaya’s and Denef-Vercauteren’s, i.e. $\tilde{O}(g^4 n^3)$.

2.4 Security

Cryptosystems based on the discrete logarithm problem in groups $G$ can always be attacked by the generic attacks [Ava05a] which run in time $O(\sqrt{G})$, most prominently Pollard’s rho and Kangaroo method and Shank’s Baby-Step Giant-Step algorithm. The possibility of applying index calculus attacks on curve based cryptosystems was first answered positive only for curves of large genus [ADH99]; in 1999 Gaudry [Gau00] showed that “large” is as small as “greater than 4.” The present state of the art is that for curves of genus 3 a double large prime variant is asymptotically faster than the generic attacks but this does not influence the choices for practical systems today [GTT04]. For elliptic curves other approaches for the factor base were investigated but so far this research did not lead to a breakthrough except for some promising trails in small characteristic [Die04, Gau04]. So, to our knowledge there does not exist a subexponential algorithm to solve the DLP on random elliptic and genus 2 curves.

The Tate pairing maps the ideal class group of a hyperelliptic curve to the group of $\ell$-th roots of unity in $\mathbb{F}_q^\times$, where $k$ is minimal such that the $k$-th roots are contained in the field, i.e. $\ell \mid q^k - 1$. If $k$ is small then this map is efficiently computable and since the discrete
logarithm in finite fields is subexponentially computable this transfer leads to an attack for special curves. In [BK98] it is shown that for a random elliptic curves it is very unlikely that the $k$ is small enough to have a more efficient algorithm and for larger genus curves similar estimates hold. Still one should check that $k > 1000$ before applying the curve in a cryptosystem. This is done by testing whether $\ell \mid q^k - 1$ for $k = 1..1000$. A prominent class of curves for which $k$ is always small are supersingular curves.

A very special case are curves over large prime fields $\mathbb{F}_p$ which have order divisible by $p$. These curves are extremely weak as the DLP can be transferred to the additive group of $\mathbb{F}_p$ for elliptic curves or a linear system for larger genera [SaAr98, Sem98, Sma99a, Ruc99]. Such curves must be avoided.

For curves over composite extension fields, Weil descent might be an issue. This is a transfer of the DLP in the Jacobian of $C/\mathbb{F}_{p^m}$ to the DLP in the Jacobian of a curve $D/\mathbb{F}_{p^n}$ of larger genus. In general the genus depends on $m$ and the genus of the original curve as by Hasse-Weil one needs to have $|\text{Cl}_C(\mathbb{F}_{p^m})| \leq |\text{Cl}_D(\mathbb{F}_{p^n})|$ and thus $g_D \geq gc m$. There are curves and fields for which this lower bound is attained or a slightly larger $g_D$ exists. Then the index calculus attacks on the larger genus curve lead to a weakening of the system. Since this attack is hard to exclude as there are many possibilities of constructing curves $D$, most prominently the GHS version [GHS02], one should avoid composite extension degrees. Furthermore, extension degrees which are Mersenne primes lead to small $g_D$, too.

A collection of results on these transfers can be found in [FrLa05a].

### 2.5 Choice of parameter

According to the preceding chapter one should take into account the following choices to obtain a system with $s$ bits security (i.e. the fastest algorithms need $O(2^s)$):

1. $g = 1, 2$ or $g = 3$.
2. $\mathbb{F}_q = \mathbb{F}_p$ with $\log_2 p \geq 2s/g$ and $p$ is prime OR $\mathbb{F}_q = \mathbb{F}_{2^p}$ with $p \geq 2s/g$ and $p$ is prime.
3. Random curve parameters in $\mathbb{F}_q$ up to isomorphic curves, e.g. for $\mathbb{F}_p$ one can choose $h = 0$ and $f_{2g} = 0$ without violating randomness.
4. Determine the group order of $\text{Cl}_C(\mathbb{F}_q)$, if it does not contain a prime factor $\ell$ of order $2^s$ or if $\ell = p$ goto the previous step; eventually one can also change the finite field.
5. Check that $\ell \mid q^k - 1$ for $1 \leq k \leq 1000g$.

To find a generator of the subgroup of order $\ell$ one chooses a random $x$-coordinate or a random polynomial of order 2 respectively and uses the decompression algorithm to retrieve the $y$-coordinate or the polynomial $v$ respectively in case it exists. Otherwise one starts anew with a different choice. Let $c$ be the cofactor $|\text{Cl}_C(\mathbb{F}_q)| = c\ell$. To obtain an element of order $\ell$ one does a scalar multiplication with $c$; if the result is the neutral element of the group one starts with a fresh choice, otherwise the result is a base point of the system.
2.6 Other related systems and special choices

It is possible to make special choices of the parameters that lead to faster scalar multiplication.

Koblitz curves [Kob92, Sol00, Lan05a] are curves defined over a small finite field which are considered over a large extension field. The Frobenius endomorphism can be used to compute $q^d$-folds efficiently and it is possible to extend this idea to general scalar multiplication. On elliptic curves, it is possible to combine the action of the Frobenius with other operations, such as point halving, in order to get further speedups [ACS04, AHP05].

GLV curves [GLV01] also make use of endomorphism but they are defined over large prime fields, e.g. for $p \equiv 1 \mod 4$ the elliptic curve $E : y^2 = x^3 + ax$ has complex multiplication with $\alpha = \sqrt{-1}$, thus if $P = (a, b) \in E$ then also $(-a, \alpha b)$. Again faster scalar multiplication is possible.

Trace zero varieties are similar to Koblitz curves in that they obtain their speed-up from the Frobenius endomorphism. However, the parameters are a moderately large prime field and a small extension $F_q = F_p^d$. For elliptic curves $d = 3$ and 5 were suggested with $p \sim 2s/d$ and in $g = 2$ only $d = 3$ with $p \sim 4s/d$ is recommended (see [AvLa05, Lan04] and the references therein). The computations take place in the subgroup of elements of trace zero, which leads to faster operations and more efficient compression; this idea is similar to the choices of XTR as the subgroup of elements of relative norm 1 in the larger field $F_{q^d}$. A recent suggestion is to apply the same construction to binary elliptic curves [AvCe05] but the security needs to be assessed.

For security reasons we have excluded curves with a small embedding degree $k$. On the other hand, an efficiently computable Tate pairing can be used to define a DL system with pairing. The security requirements are that the DLP in the subgroup of $Cl_C(F_q)$ is hard to solve and that at the same time the DLP in $F_{q^k}^*$ is hard, i.e. $\log_2 \ell \geq 160$ and $k \log_2 q \geq 1024$. Since $\ell \sim q^d$ one obtains that for this minimal security requirement $k = 6$ or 7 is optimal for elliptic curves. For higher security larger embedding degrees are optimal.

It is possible to obtain $k = 6$ for supersingular curves in characteristic 3; for characteristic 2 at most $k = 4$ can be achieved such that the finite field must be chosen larger than usually necessary for the DLP in ECC; for large prime fields even $k = 2$ is the upper bound for supersingular curves. Similar considerations hold for larger genera [Gal01]. Starting with [MNT01] non-supersingular elliptic curves with small embedding degree $k$ were constructed; this approach has been generalized to families of non-supersingular elliptic curves and genus 2 curves.

2.7 Conclusions

Elliptic curves and hyperelliptic curves of genus at most 3 are secure alternatives to RSA: they come with much smaller parameter sizes and are already now faster than RSA on almost all platforms. In the future the performance and bandwidth gap will widen as the parameter for RSA grow much quicker due to the difference of exponential and subexponential attacks. This will soon make RSA no longer viable for embedded applications, and also for server applications the speed difference will be remarkable. In fact, for embedded applications one observes already now a preference for elliptic curves, e.g. the blackberry hand-held device...
uses ECC and the chips containing biometric data in travel documents will base the security on elliptic curve cryptography.

At the moment, elliptic curves are faster than higher genus curves in large characteristic. The results in [Ava04] seem to point to the fact that this gap should decrease as security demands increase. Therefore, this choice is only recommended if patent issues need to be avoided or extreme security levels are to be achieved. In even characteristic there are families of genus 2 and 3 curves which outperform elliptic curves in the computation of scalar multiples. They might be an interesting alternative for hardware or general embedded implementations. For considerations of side-channel attack resistant implementations we refer to [Ava05b, Lan05b, VAM3].

References – Curve-based Cryptosystems


References – Curve-based Cryptosystems


2. Curve-Based Cryptosystems


References – Curve-based Cryptosystems


2. Curve-Based Cryptosystems


3 Lattice-based Cryptography

Contributors: Phong Q. Nguyen and Jacques Stern

3.1 Introduction

Lattices are discrete subgroups of $\mathbb{R}^n$. A lattice has infinitely many $\mathbb{Z}$-bases, but some are more useful than others. The goal of lattice reduction is to find interesting lattice bases, such as ones consisting of reasonably short and almost orthogonal vectors. From the mathematical point of view, the history of lattice reduction goes back to the reduction theory of quadratic forms developed by Lagrange [La1773], Gauss [Ga1801], Hermite [He1850], Korkine and Zolotarev [KZ1872, KZ1873], among others, and to Minkowski’s geometry of numbers [Mi1896]. With the advent of algorithmic number theory, the subject had a revival in 1981 with Lenstra’s celebrated work on integer programming (see [Le81, Le83]), which was, among others, based on a novel lattice reduction technique (which can be found in the preliminary version [Le81] of [Le83]). Lenstra’s reduction technique was only polynomial-time for fixed dimension, which was however enough in [Le81]. That inspired Lovász to develop a polynomial-time variant of the algorithm, which computes a so-called reduced basis of a lattice. The algorithm reached a final form in the seminal paper [LLL82] where Lenstra, Lenstra and Lovász applied it to factor rational polynomials in polynomial time, from which the name LLL comes. Further refinements of the LLL algorithm were later proposed, notably by Schnorr [Sc87, Sc88].

Those algorithms have proved invaluable in many areas of mathematics and computer science (see [Lo86, Ka87b, Va89, GLS93, Co95, La95]). and their relevance to cryptography became soon apparent, as they were used to break schemes based on the knapsack problem (see [Od90, BrOd91]), which had been then considered as alternatives to the RSA cryptosystem [RSA78]. In fact, lattice reduction has since become one of the most popular techniques for breaking cryptographic schemes (see [JoSt98]).

As a matter of fact, applications of lattices to cryptology have been mainly destructive. Interestingly, it was noticed in many cryptanalytic experiments that LLL, as well as other lattice reduction algorithms, behave much more nicely than what was expected from the worst-case proved bounds. This led to a common belief among cryptographers, that lattice reduction is an easy problem, at least in practice.

That belief has recently been challenged by some exciting progress on the complexity of lattice problems, which originated in large part in two seminal papers written by Ajtai in 1996 and in 1997 respectively. Prior to 1996, little was known on the complexity of lattice problems. In his 1996 paper [Aj96], Ajtai discovered a fascinating connection between the worst-case complexity and the average-case complexity of some well-known lattice problems. Such a connection is not known to hold for any other problem in NP believed to be outside P. In his 1997 paper [Aj98], building on previous work by Adleman [Ad95], Ajtai further proved the NP-hardness (under randomized reductions) of the most famous lattice problem, the shortest vector problem (SVP). The NP-hardness of SVP has been a long standing open
problem. Ajtai’s breakthroughs initiated a series of new results on the complexity of lattice problems, which are nicely surveyed by Cai [Ca99, Ca00].

Those complexity results paved the way for constructive applications in cryptology. Several cryptographic schemes based on the hardness of lattice problems were proposed shortly after Ajtai’s discoveries (see [AjDw97, GGH97b, HoPiSi98, CaCu99, Mi99, FiSe99]). Some have been broken, while others seem to resist state-of-the-art attacks, for now. Those schemes attracted interest for at least three reasons:

1. There are very few public-key cryptosystems based on problems different from integer factorization or the discrete logarithm problem;
2. Some of those schemes offered encryption/decryption rates asymptotically higher than classical schemes.
3. There are no currently know quantum algorithms for solving lattice problems that have better complexity than their classical counterparts. Therefore lattice-based schemes might survive the quantum computation era.

One of those schemes, by Ajtai and Dwork [AjDw97], enjoyed a surprising security proof based on worst-case (instead of average-case) hardness assumptions.

Independently of those developments, there has been renewed cryptographic interest in lattice reduction, following a beautiful work by Coppersmith [Co97] in 1996. There certain problems, apparently non-linear, related to the question of finding small roots of low-degree polynomial equations, have been solved. This led to attacks on the RSA [RSA78] cryptosystem with small public or private exponent, as well as to new security proofs [Sh01, Bo01]. Coppersmith’s results differ from “traditional” applications of lattice reduction in cryptanalysis, where the underlying problem is already linear, and the attack often heuristic by requiring (at least) that current lattice reduction algorithms behave ideally, as opposed to what is theoretically guaranteed.

This Chapter of the Deliverable at hand is based on [NgSt01], which is in turn based on [NgSt00]. The rest of the Chapter is organized as follows. In § 3.2, we give basic definitions and results on lattices and their algorithmic problems. In § 3.3, we discuss lattice-based cryptography, somehow a revival for knapsack-based cryptography.

3.2 Lattice problems

3.2.1 Definitions

Recall that a lattice is a discrete (additive) subgroup of \( \mathbb{R}^n \). In particular, any subgroup of \( \mathbb{Z}^n \) is a lattice, and such lattices are called integer lattices. An equivalent definition is that a lattice consists of all integral linear combinations of a set of linearly independent vectors, that is,

\[
L = \left\{ \sum_{i=1}^{d} n_i b_i \mid n_i \in \mathbb{Z} \right\},
\]
where the $b_i$’s are linearly independent over $\mathbb{R}$. Such a set of vectors $b_i$’s is called a lattice basis. All the bases have the same number $\dim(L)$ of elements, called the dimension (or rank) of the lattice since it matches the dimension of the vector subspace $\text{span}(L)$ spanned by $L$.

There are infinitely many lattice bases when $\dim(L) \geq 2$. Any two bases are related to each other by some unimodular matrix (integral matrix of determinant $\pm 1$), and therefore all the bases share the same Gramian determinant $\det_{1 \leq i, j \leq \dim(L)}(b_i, b_j)$. The volume $\text{vol}(L)$ (or determinant) of the lattice is by definition the square root of that Gramian determinant, thus corresponding to the $d$-dimensional volume of the parallelepiped spanned by the $b_i$’s. In the important case of full-dimensional lattices where $\dim(L) = n$, the volume is equal to the absolute value of the determinant of any lattice basis (hence the name determinant). If the lattice is further an integer lattice, then the volume is also equal to the index $[\mathbb{Z}^n : L]$ of $L$ in $\mathbb{Z}^n$.

Since a lattice is discrete, it has a shortest non-zero vector: the Euclidean norm of such a vector is called the lattice first minimum, denoted by $\lambda_1(L)$ or $\|L\|$. Of course, one can use other norms as well: we will use $\|L\|_\infty$ to denote the first minimum for the infinity norm. More generally, for all $1 \leq i \leq \dim(L)$, Minkowski’s $i$-th minimum $\lambda_i(L)$ is defined as the minimum of $\max_{1 \leq j \leq i} \|v_j\|$ over all $i$ linearly independent lattice vectors $v_1, \ldots, v_i \in L$. There always exist linearly independent lattice vectors $v_1, \ldots, v_d$ reaching the minima, that is $\|v_i\| = \lambda_i(L)$. However, surprisingly, as soon as $\dim(L) \geq 4$, such vectors do not necessarily form a lattice basis, and when $\dim(L) \geq 5$, there may not even exist a lattice basis reaching the minima. This is one of the reasons why there exist several notions of basis reduction in high dimension, without any “optimal” one. It will be convenient to define the lattice gap as the ratio $\lambda_2(L)/\lambda_1(L)$ between the first two minima.

Minkowski’s Convex Body Theorem guarantees the existence of short vectors in lattices: a careful application shows that any $d$-dimensional lattice $L$ satisfies $\|L\|_\infty \leq \text{vol}(L)^{1/d}$, which is obviously the best possible bound. It follows that $\|L\| \leq \sqrt[d]{\text{vol}(L)}^{1/d}$, which is not optimal, but shows that the value $\lambda_1(L)/\text{vol}(L)^{1/d}$ is bounded when $L$ runs over all $d$-dimensional lattices. The supremum of $\lambda_1(L)^2/\text{vol}(L)^{2/d}$ is denoted by $\gamma_d$, and called Hermite’s constant\footnote{For historical reasons, Hermite’s constant refers to $\max \lambda_1(L)^2/\text{vol}(L)^{2/d}$ and not to $\max \lambda_1(L)/\text{vol}(L)^{1/d}$.} of dimension $d$, because Hermite was the first to establish its existence in the language of quadratic forms. The exact value of Hermite’s constant is only known for $d \leq 8$. The best asymptotic bounds known for Hermite’s constant are the following ones (see [MiHu73, Chapter II] for the lower bound, and [CoSl98, Chapter 9] for the upper bound):

$$\frac{d}{2\pi e} + \frac{\log(\pi d)}{2\pi e} + o(1) \leq \gamma_d \leq \frac{1.744d}{2\pi e} (1 + o(1)).$$

Minkowski proved more generally:

**Theorem 3.2.1** [Minkowski] For all $d$-dimensional lattices $L$ and all $r \leq d$:

$$\prod_{i=1}^{r} \lambda_i(L) \leq \sqrt[d]{\text{vol}(L)}^{r/d}.$$
with a small error term. This approach can be proved to be rigorous in certain settings, such as when the lattice dimension is fixed and the set is the ball centered at the origin with radius growing to infinity. Thus, one often heuristically approximates the successive minima of a $d$-dimensional lattice $L$ by \( \sqrt{\frac{d}{2\pi}} \text{vol}(L)^{1/d} \). This is of course only an intuitive estimate, which may be far away from the truth.

For any lattice $L$ of $\mathbb{R}^n$, one defines the **dual lattice** (also called **polar lattice** or **polar family**) of $L$ as:

\[
L^* = \{ x \in \text{span}(L) : \forall y \in L, \langle x, y \rangle \in \mathbb{Z} \}.
\]

If \( (b_1, \ldots, b_d) \) is a basis of $L$, then the dual family \( (b_1^*, \ldots, b_d^*) \) is a basis of $L^*$ (the dual family is the unique linearly independent family of $\text{span}(L)$ such that $\langle b_i^*, b_j \rangle$ is equal to 1 if $i = j$, and to 0 otherwise). Thus, $(L^*)^* = L$ and $\text{vol}(L)\text{vol}(L^*) = 1$. The so-called transference theorems relate the successive minima of a lattice and its dual lattice. The first transference theorem follows from the definition of Hermite’s constant:

\[
\lambda_1(L)\lambda_1(L^*) \leq \gamma_d.
\]

A more difficult transference theorem (see [Ba93]) ensures that for all $1 \leq r \leq d$:

\[
\lambda_r(L)\lambda_{d-r+1}(L^*) \leq d.
\]

Both these transference bounds are optimal up to a constant. More information on lattice theory can be found in numerous textbooks, such as [GrLe87, Si89].

### 3.2.2 Algorithmic problems

In the rest of this subsection, we assume implicitly that lattices are rational lattices (lattices in $\mathbb{Q}^n$), and $d$ will denote the lattice dimension.

The most famous lattice problem is the **shortest vector problem** (SVP): given a basis of a lattice $L$, find $u \in L$ such that $\|u\| = \|L\|$ (recall that $\|L\| = \lambda_1(L)$). SVP will denote the analogue for the infinity norm. One defines approximate short vector problems by asking a non-zero $u \in L$ with norm bounded by some approximation factor: $\|u\| \leq f(d)\|L\|$.

The **closest vector problem** (CVP), also called the **nearest lattice point problem**, is a non-homogeneous version of the shortest vector problem: given a basis of a lattice $L$ and a vector $v \in \mathbb{R}^n$ (it does not matter whether $v \in \text{span}(L)$), find a lattice vector minimizing the distance to $v$. Again, one defines approximate closest vector problems by asking $u \in L$ such that for all $w \in L$, $\|u - v\| \leq f(d)\|w - v\|$.

Another problem is the **smallest basis problem** (SBP), which has many variants depending on the exact meaning of “smallest”. The variant currently in vogue (see [Aj96, BlSe99]) is the following: find a lattice basis minimizing the maximum of the lengths of its elements. A more geometric variant asks instead to minimize the product of the lengths (see [GLS93]), since the product is always larger than the lattice volume, with equality if and only if the basis is orthogonal.
3.2. Lattice problems

3.2.3 Complexity results

We refer to Cai [Ca99, Ca00] for an up-to-date survey of complexity results. Ajtai [Aj98] recently proved that SVP is NP-hard under randomized reductions. Micciancio [Mi98a, Mi98b] simplified and improved the result by showing that approximating SVP to within a factor \( \sqrt{2} \) is also NP-hard under randomized reductions. The NP-hardness of SVP under deterministic (Karp) reductions remains an open problem.

CVP seems to be a more difficult problem. Goldreich et al. [GMSS99] recently noticed that CVP cannot be easier than SVP: given an oracle that approximates CVP to within a factor \( f(d) \), one can approximate SVP in polynomial time to within the same factor \( f(d) \). Reciprocally, Kannan proved in [Ka87b, Section 7] that any algorithm approximating SVP to within a non-decreasing function \( f(d) \) can be used to approximate CVP to within \( d^{3/2} f(d)^2 \).

CVP was shown to be NP-hard as early as in 1981 [Emde81] (for a much simpler “one-line” proof using the knapsack problem, see [Mi01b]). Approximating CVP to within a quasi-polynomial factor \( 2^{\log^{1+\epsilon} d} \) is NP-hard [ABSS97, DKS98].

However, NP-hardness results for SVP and CVP have limits. Goldreich and Goldwasser [GoGo98] showed that approximating SVP or CVP to within \( \sqrt{d}/\log d \) is not NP-hard, unless the polynomial-time hierarchy collapses.

Interestingly, SVP and CVP problems seem to be more difficult with the infinity norm. It was shown that SVP\(_{\infty}\) and CVP\(_{\infty}\) are NP-hard in 1981 [Emde81]. In fact, approximating SVP\(_{\infty}/CVP\(_{\infty}\) to within an almost-polynomial factor \( d^{1/\log \log d} \) is NP-hard [Di99]. On the other hand, Goldreich and Goldwasser [GoGo98] showed that approximating SVP\(_{\infty}/CVP\(_{\infty}\) to within \( d/\log d \) is not NP-hard, unless the polynomial-time hierarchy collapses.

We will not discuss Ajtai’s worst-case/average-case equivalence [Aj96, CaNe97], which refers to special versions of SVP and SBP (see [Ca99, Ca00, BlSe99]) such as SVP when the lattice gap \( \lambda_2/\lambda_1 \) is at least polynomial in the dimension.

3.2.4 Algorithmic results

The main algorithmic results are surveyed in [Lo86, Ka87b, Va89, GLS93, Co95, La95, Ca99, Ng99b]. No polynomial-time algorithm is known for approximating either SVP, CVP or SBP to within a polynomial factor in the dimension \( d \). In fact, the existence of such algorithms is an important open problem. The best polynomial-time algorithms achieve only slightly subexponential factors, and are based on the LLL algorithm [LLL82], which can approximate SVP and SBP. However, it should be emphasized that these algorithms typically perform much better than is theoretically guaranteed, on instances of practical interest. Given as input any basis of a lattice \( L \), LLL provably outputs in polynomial time a basis \((b_1, \ldots, b_d)\) satisfying:

\[
\|b_1\| \leq 2^{(d-1)/4} \text{vol}(L)^{1/d}, \quad \|b_i\| \leq 2^{(d-1)/2} \lambda_i(L) \quad \text{and} \quad \prod_{i=1}^d \|b_i\| \leq 2^{d^2/2} \text{vol}(L).
\]
Thus, LLL can approximate SVP to within $2^{(d-1)/2}$. Schnorr [Sc87] improved the bound to $2^{O(d \log \log d/\log d)}$, and Ajtai et al. [AKS01] recently further improved it to $2^{O(d \log \log d/\log d)}$ in randomized polynomial time thanks to a new randomized algorithm to find the shortest vector. In fact, Schnorr defined an LLL-based family of algorithms [Sc87] (named BKZ for blockwise Korkine-Zolotarev) whose performances depend on a parameter called the blocksize. These algorithms use some kind of exhaustive search super-exponential in the blocksize. So far, the best reduction algorithms in practice are variants [ScEu94, ScHo95] of those BKZ-algorithms, which apply a heuristic to reduce exhaustive search. But little is known on the average-case (and even worst-case) complexity of reduction algorithms.

Babai’s nearest plane algorithm [Ba86] uses LLL to approximate CVP to within $2^{d/2}$, in polynomial time (see also [Kl00]). Using Schnorr’s algorithm [Sc87], this can be improved to $2^{O(d \log \log d/\log d)}$ in polynomial time, and even further to $2^{O(d \log \log d/\log d)}$ in randomized polynomial time using [AKS01], due to Kannan’s link between CVP and SVP (see previous subsection). In practice however, the best strategy seems to be the embedding method (see [GGH97b, Ng99a]), which uses the previous algorithms for SVP and a simple heuristic reduction from CVP to SVP. Namely, given a lattice basis $(b_1, \ldots, b_d)$ and a vector $v \in \mathbb{R}^d$, the embedding method builds the $(d+1)$-dimensional lattice (in $\mathbb{R}^{n+1}$) spanned by the row vectors $(b_i, 0)$ and $(v, 1)$. Depending on the lattice, one should choose a coefficient different than 1 in $(v, 1)$. It is hoped that a shortest vector of that lattice is of the form $(v - u, 1)$ where $u$ is a closest vector (in the original lattice) to $v$, whenever the distance to the lattice is smaller than the lattice first minimum. This heuristic may fail (see for instance [Mi98b] for some simple counterexamples), but it can also sometimes be proved, notably in the case of lattices arising from low-density knapsacks.

For exact SVP, the best algorithm known (in theory) is the recent randomized $2^{O(d)}$-time algorithm by Ajtai et al. [AKS01], which improved Kannan’s super-exponential algorithm [Ka83, Ka87a] (see also [He85]). For exact CVP, the best algorithm remains Kannan’s super-exponential algorithm [Ka83, Ka87a], with running time $2^{O(d \log d)}$ (see also [He85] for an improved constant).

### 3.3 Lattice-based cryptography

We review state-of-the-art results on the main lattice-based cryptosystems. To keep the presentation simple, descriptions of the schemes are intuitive, referring to the original papers for more details. Only one of these schemes (the GGH cryptosystem [GGH97b]) explicitly works with lattices.

#### 3.3.1 The Ajtai-Dwork cryptosystem

**3.3.1.i Description.** The Ajtai-Dwork cryptosystem [AjDw97] (AD) works in $\mathbb{R}^n$, with some finite precision depending on $n$. Its security is based on a variant of SVP.

The private key is a uniformly chosen vector $u$ in the $n$-dimensional unit ball. One then...
defines a distribution \( \mathcal{H}_u \) of points \( a \) in a large \( n \)-dimensional cube such that the dot product \( \langle a, u \rangle \) is very close to \( Z \).

The public key is obtained by picking \( w_1, \ldots, w_n, v_1, \ldots, v_m \) (where \( m = n^3 \)) independently at random from the distribution \( \mathcal{H}_u \), subject to the constraint that the parallelepiped \( w \) spanned by the \( w_i \)'s is not flat. Thus, the public key consists of a polynomial number of points close to a collection of parallel affine hyperplanes, which is kept secret.

The scheme is mainly of theoretical purpose, as encryption is bit-by-bit. To encrypt a '0', one randomly selects \( b_1, \ldots, b_m \) in \( \{0, 1\} \), and reduces \( \sum_{i=1}^{m} b_i v_i \mod \) the parallelepiped \( w \). The vector obtained is the ciphertext. The ciphertext of '1' is just a randomly chosen vector in the parallelepiped \( w \). To decrypt a ciphertext \( x \) with the private key \( u \), one computes \( \tau = \langle x, u \rangle \). If \( \tau \) is sufficiently close to \( Z \), then \( x \) is decrypted as '0', and otherwise as '1'. Thus, an encryption of '0' will always be decrypted as '0', and an encryption of '1' has a small probability to be decrypted as '0'. These decryption errors can be removed (see [GGH97a]).

### 3.3.1.ii Security

The Ajtai-Dwork [AjDw97] cryptosystem received wide attention due to a surprising security proof based on worst-case assumptions. Indeed, it was shown that any probabilistic algorithm distinguishing encryptions of a '0' from encryptions of a '1' with some polynomial advantage can be used to solve SVP in any \( n \)-dimensional lattice with gap \( \lambda_2/\lambda_1 \) larger than \( n^8 \). There is a converse, due to Nguyen and Stern [NgSt98]: one can decrypt in polynomial time with high probability, provided an oracle that approximates SVP to within \( n^{0.5-\varepsilon} \), or one that approximates CVP to within \( n^{1/33} \). It follows that the problem of decrypting ciphertexts is unlikely to be NP-hard, due to the result of Goldreich-Goldwasser [GoGo98].

Nguyen and Stern [NgSt98] further presented a heuristic attack to recover the secret key. Experiments suggest that the attack is likely to succeed up to at least \( n = 32 \). For such parameters, the system is already impractical, as the public key requires 20 megabytes and the ciphertext for each bit has bit-length 6144. This shows that unless major improvements are found, the Ajtai-Dwork cryptosystem is only of theoretical importance.

### 3.3.1.iii Cryptanalysis overview

At this point, the reader might wonder how lattices come into play, since the description of AD does not involve lattices. Any ciphertext of '0' is a sum of \( v_i \)'s minus some integer linear combination of the \( w_i \)'s. Since the parallelepiped spanned by the \( w_i \)'s is not too flat, the coefficients of the linear combination are relatively small. On the other hand, any linear combination of the \( v_i \)'s and the \( w_i \)'s with small coefficients is close to the hidden hyperplanes. This enables to build a particular lattice of dimension \( n + m \) such that any ciphertext of '0' is in some sense close to the lattice, and reciprocally, any point sufficiently close to the lattice gives rise to a ciphertext of '0'. Thus, one can decrypt ciphertexts provided an oracle that approximates CVP sufficiently well. The analogous version for SVP uses related ideas, but is technically more complicated. For more details, see [NgSt98].

The attack to recover the secret key can be described quite easily. One knows that each \( \langle v_i, u \rangle \) is close to some unknown integer \( V_i \). It can be shown that any sufficiently short

---

3 A variant of AD with less message expansion was proposed in [CaCu99], however without any security proof. It mixes AD with a knapsack.
3. Lattice-based Cryptography

Linear combination of the $v_i$’s give information on the $V_i$’s. More precisely, if $\sum_i \lambda_i v_i$ is sufficiently short and the $\lambda_i$’s are sufficiently small, then $\sum_i \lambda_i V_i = 0$ (because it is a too small integer). Note that the $V_i$’s are disclosed if enough such equations are found. And each $V_i$ gives an approximate linear equation satisfied by the coefficients of the secret key $u$. Thus, one can compute a sufficiently good approximation of $u$ from the $V_i$’s. To find the $V_i$’s, we produce many short combinations $\sum_i \lambda_i v_i$ with small $\lambda_i$’s, using lattice reduction. Heuristic arguments can justify that there exist enough such combinations. Experiments showed that the assumption was reasonable in practice.

3.3.2 The Goldreich-Goldwasser-Halevi cryptosystem

The Goldreich-Goldwasser-Halevi cryptosystem [GGH97b] (GGH) can be viewed as a lattice-analog to the McEliece [McE78] cryptosystem based on algebraic coding theory. In both schemes, a ciphertext is the addition of a random noise vector to a vector corresponding to the plaintext. The public key and the private key are two representations of the same object (a lattice for GGH, a linear code for McEliece). The private key has a particular structure allowing to cancel noise vectors up to a certain bound. However, the domains in which all these operations take place are quite different.

3.3.2.1 Description. The GGH scheme works in $\mathbb{Z}^n$. The private key is a non-singular $n \times n$ integral matrix $R$, with very short row vectors (entries polynomial in $n$). The lattice $L$ is the full-dimensional lattice in $\mathbb{Z}^n$ spanned by the rows of $R$. The basis $R$ is then transformed to a non-reduced basis $B$, which will be public. In the original scheme, $B$ is the multiplication of $R$ by sufficiently many small unimodular matrices. Computing a basis as “good” as the private basis $R$, given only the non-reduced basis $B$, means approximating SBP.

The message space is a “large enough” cube in $\mathbb{Z}^n$. A message $m \in \mathbb{Z}^n$ is encrypted into $c = mB + e$ where $e$ is an error vector uniformly chosen from $\{-\sigma, \sigma\}^n$, where $\sigma$ is a security parameter. A ciphertext $c$ is decrypted as $\lfloor cR^{-1}\rfloor RB^{-1}$ (note: this is Babai’s round method [Ba86] to solve CVP). But an eavesdropper is left with the CVP-instance defined by $c$ and $B$. The private basis $R$ is generated in such a way that the decryption process succeeds with high probability. The larger $\sigma$ is, the harder the CVP-instances are expected to be. But $\sigma$ must be small for the decryption process to succeed.

3.3.2.2 Improvements. In the original scheme, the public matrix $B$ is the multiplication of the secret matrix by sufficiently many unimodular matrices. This means that without appropriate precaution, the public matrix may be as large as $O(n^3 \log n)$ bits. Micciancio [Mi99, Mi01a] therefore suggested to define instead $B$ as the Hermite normal form (HNF) of $R$. Recall that the HNF of an integer square matrix $R$ in row notation is the unique lower triangular matrix with coefficients in $\mathbb{N}$ such that: the rows span the same lattice as $R$, and any entry below the diagonal is strictly less than the diagonal entry in its column. Here, one can see that the HNF of $R$ is $O(n^2 \log n)$ bits, which is much better but still big. When using the HNF, one should encode messages into the error vector $e$ instead of a lattice point, because the HNF is unbalanced. The ciphertext is defined as the reduction of $e$ modulo the HNF.

Footnote 4: A different construction for $R$ based on tensor product was proposed in [FiSe99].
3.3. Lattice-based cryptography

and hence uses less than $O(n \log n)$ bits. One can easily prove that the new scheme (which is now deterministic) cannot be less secure than the original GGH scheme (see [Mi99, Mi01a]).

3.3.2.iii Security. GGH has no proven worst-case/average-case property, but it is much more efficient than AD. Specifically, for security parameter $n$, key-size and encryption time can be $O(n^2 \log n)$ for GGH (note that McEliece is slightly better though), vs. at least $O(n^4)$ for AD. For RSA and El-Gamal systems, key size is $O(n)$ and computation time is $O(n^3)$.

The authors of GGH argued that the increase in size of the keys was more than compensated by the decrease in computation time. To bring confidence in their scheme, they published on the Internet a series of five numerical challenges [GGH], in dimensions 200, 250, 300, 350 and 400. In each of these challenges, a public key and a ciphertext were given, and the challenge was to recover the plaintext.

The GGH scheme is now considered broken, at least in its original form, due to an attack recently developed by Nguyen [Ng99a]. As an application, using small computing power and Shoup’s NTL library [Shoup], Nguyen was able to solve all the GGH challenges, except the last one in dimension 400. But already in dimension 400, GGH is not very practical: in the 400-challenge, the public key takes 1.8 Mbytes without HNF or 124 Kbytes using the HNF.5

Nguyen’s attack used two “qualitatively different” weaknesses of GGH. The first one is inherent to the GGH construction: the error vectors used in the encryption process are always much shorter6 than lattice vectors. This makes CVP-instances arising from GGH easier than general CVP-instances. The second weakness is the particular form of the error vectors in the encryption process. Recall that $c = mB + e$ where $e \in \{\pm \sigma\}^n$. The form of $e$ was apparently chosen to maximize the Euclidean norm under requirements on the infinity norm. However, if we let $s = (\sigma, \ldots, \sigma)$ then $c + s \equiv mB \pmod{2\sigma}$, which allows to guess $m \mod 2\sigma$. Then the original closest vector problem can be reduced to finding a lattice vector within (smaller) distance $e/(2\sigma)$ from $(c - (m \mod 2\sigma)B)/(2\sigma)$. The simplified closest vector problem happens to be within reach (in practice) of current lattice reduction algorithms, thanks to the embedding strategy that heuristically reduces CVP to SVP. We refer to [Ng99a] for more information.

It is easy to fix the second weakness by selecting the entries of the error vector $e$ at random in $\{-\sigma, \ldots, +\sigma\}$ instead of $\{\pm \sigma\}$. However, one can argue that the resulting GGH system would still not be much practical, even using [Mi99, Mi01a]. Indeed, Nguyen’s experiments [Ng99a] showed that SVP could be solved in practice up to dimensions as high as 350, for (certain) lattices with gap as small as 10. To be competitive, the new GGH system would require the hardness (in lower dimensions due to the size of the public key, even using [Mi99]) of SVP for certain lattices of only slightly smaller gap, which means a rather smaller improvement in terms of reduction. Note also that those experiments do not support the practical hardness of Ajtai’s variant of SVP in which the gap is polynomial in the lattice dimension. Besides, it is not clear how to make decryption efficient without a huge secret key (Babai’s rounding requires the storage of $R^{-1}$ or a good approximation, which could be in $[\text{GGH97b}]$ over 1 Mbyte in dimension 400).

5The challenges do not use the HNF, as they were proposed before [Mi99]. Note that 124 Kbytes is about twice as large as McEliece for the recommended parameters.

6In all GGH-like constructions known, the error vector is always at least twice as short.
3. Lattice-based Cryptography

3.3.3 The NTRU cryptosystem

For a description of the NTRU cryptosystems see Chapter 4.

3.4 Conclusions

The LLL algorithm and other lattice basis reduction algorithms have proved invaluable in cryptography. They have become a very popular tool in public-key cryptanalysis. In particular, they play a crucial rôle in several attacks against the RSA cryptosystem. At the same time, a series of complexity results on lattice reduction has given rise to the design of cryptographic schemes based on the hardness of lattice problems. The resulting cryptosystems have enjoyed different fates, but it is probably too early to tell whether or not secure and practical cryptography can be built using hardness of lattice problems. Indeed, several questions on lattices remain open. In particular, we still do not know whether or not it is easy to approximate the shortest vector problem up to some polynomial factor, or to find the shortest vector when the lattice gap is larger than some polynomial in the dimension. Besides, only very few lattice basis reduction algorithms are known, and their behaviour (both complexity and output quality) is still not well understood. And so far, there has not been any massive computer experiment in lattice reduction comparable to what has been done for integer factorization or the elliptic curve discrete logarithm problem. However, there are no currently known quantum algorithms for solving the just mentioned supposedly hard problems that have better complexity than their classical counterparts. Therefore lattice based cryptography could save public key cryptography should a breakthrough in physics research allow the construction of quantum computers. The NTRU cryptosystem (See Chapter 4) can be viewed as a lattice-based cryptosystem in disguise and would therefore also be one of the few cryptosystems to survive quantum computation.

References – Lattice-based Cryptography


3. Lattice-based Cryptography


References – Lattice-based Cryptography


3. Lattice-based Cryptography


3. Lattice-based Cryptography
4

The NTRU Public-Key Cryptosystem

Contributors: Katharina Geissler and Nigel P. Smart

4.1 Introduction

The present chapter is devoted to the NTRU Public-Key Cryptosystem. We present the system and discuss aspects related to its security.

The NTRU cryptosystem was first publicly presented at the rump session of Crypto'96 [HPS98], [HPS96]. Its security is based on the difficulty of a certain polynomial factorisation problem in the polynomial ring \( \mathbb{Z}_q[X]/(X^N - 1) \), where \( N \) denotes the security parameter of the system. In 1997 Coppersmith and Shamir [CS97] showed that this factorisation problem can be reduced into a shortest vector problem in a \( 2N \times 2N \) certain lattice. This lattice has been dubbed the “NTRU lattice” and we shall refer to it as \( L^{NT} \), for the precise definition of this lattice see § 4.3.

Even though the shortest vector problem for an arbitrary lattice is in the worst case a NP hard problem, it is not clear if the problem of finding the shortest vector in the lattice \( L^{NT} \) is NP hard. This has created considerable interest since all other practical public key cryptosystems are either based on the integer factorization problem, e.g. RSA [RSA78] or Rabin [Rab79], or are based a discrete logarithm problem in a finite field or the group of points of an elliptic curve, e.g. ElGamal [ElG85]. Having another hard problem on which to base the security of protocols is important since any advance in factoring or discrete logarithms could leave our current systems exposed. As mentioned in the Introduction to the Deliverable, NTRU is one of the few current public key schemes which would survive the advent of a quantum computer.

However, the most remarkable property of the NTRU Public Cryptosystem is its speed. Encoding and decoding takes \( O(N^2) \) operations for a message block of length \( N \). This is equivalent to the running time of polynomial multiplication in the ring \( \mathbb{Z}_q[X]/(X^N - 1) \). Using the Fast Fourier Transformation it is even possible to achieve asymptotically \( O(N \log N) \) operations. This is a major advantage compared to RSA, which needs \( O(N^3) \) operations. Furthermore other proposed public key cryptosystems with fast encoding and decoding routines (in time \( O(N^2) \)) such as those of McEliece [McE78] or Goldwasser, Goldreich, Halevi [GGH97] possess impractically large keys of length \( O(N^2) \), whereas NTRU’s key length for the public key is \( O(N \log q) \) and \( O(N) \) for the private key. Therefore for small values of \( N \) the NTRU public key cryptosystem can be embedded in small devices, for example mobile phones, smart cards and small information/entertainment appliances and still achieve an acceptable performance level.

The efficiency of encoding and decoding routines is based on the underlying polynomial ring \( \mathbb{Z}_q[X]/(X^N - 1) \) in which arithmetical operations can be carried out with high speed. NTRU is a ring-based public key cryptosystem, because it uses the two ring operations addition and multiplication. Therefore it differs from the most common cryptosystems, which are based on groups and use only group operations for the parameters. As already mentioned
the rich arithmetical structure of the underlying ring is one of the advantages of the NTRU cryptosystem. On the other hand ring structures in cryptography are not as well studied as group theory and security proofs are therefore often easier to handle within groups.

The NTRU encryption function

\[ e = r \ast h + m, \]

which we will study more fully in \S\ 4.2, is an example of a one-way trapdoor function. This function does not on its own however constitute a secure encryption algorithm. This is exactly the case with RSA where the RSA function

\[ c = m^e \pmod{N} \]

forms a one-way trapdoor permutation on \( \mathbb{Z}/N\mathbb{Z} \), yet one cannot use the RSA function as an encryption algorithm on its own. The path from mathematical function to provably secure encryption scheme is what we will consider in this survey article. Whilst we will concentrate on the NTRU scheme, we hope to also explain to a general audience the issues behind a rigorous definition of security and we will point out some of the pitfalls that have befallen cryptographers in the last few years. To fix ideas, and hopefully make this article useful to a wider audience, we will also discuss some of the security definitions in the context of the more familiar RSA function.

The rest of the Chapter is structured as follows. In \S\ 4.2 we discuss the NTRU encryption function in some detail. We discuss the various algorithms and the current recommended parameter choices. In \S\ 4.3 we discuss the link between the NTRU function and lattices, by which we mean \( \mathbb{Z} \)-submodules of \( \mathbb{R}^n \) rather than the structure based on partially ordered sets. In \S\ 4.4 we discuss the security of the NTRU system - and in particular of the implications of the fact that the system might produce valid ciphertexts that will not correctly decrypt to the original plaintext, this paving the way to attacks. Finally, we end with some conclusions.

4.2 The NTRU Cryptosystem

4.2.1 Notation

We denote the ring of integers by \( \mathbb{Z} \) and the integers modulo \( q \) by \( \mathbb{Z}_q \) which we shall assume are represented by elements in the symmetric interval \((-q/2, q/2] \). Let \( N \) be a positive integer, we will identify the set \( \mathbb{Z}^N \) (resp. \( \mathbb{Z}_q^N \)) with the ring of polynomials \( P(N) = \mathbb{Z}[X]/(X^N - 1) \) (resp. \( P_q(N) = \mathbb{Z}_q[X]/(X^N - 1) \)), by

\[ f = (f_0, f_1, \ldots, f_{N-1}) = \sum_{i=0}^{N-1} f_i X^i. \]

Note, the modulus \( q \) will not necessarily be prime, hence \( \mathbb{Z}_q \) is not in general a field.

Two polynomials \( f, g \in P(N) \) are multiplied by the cyclic convolution product since we are working modulo \( X^N - 1 \), an operation which will be denoted by \( \ast \) to distinguish it from the multiplication - in \( \mathbb{Z} \) or \( \mathbb{Z}[X] \). Let \( h = f \ast g \), then the \( k \)th-coefficient \( h_k \) of \( h \) is given by

\[ h_k = \sum_{i=0}^{k} f_i g_{k-i} + \sum_{i=k+1}^{N-1} f_i g_{n+k-i} = (f \ast g)_k \equiv \sum_{i+j=k \mod N} f_i \cdot g_j \ (0 \leq k \leq N). \]
4.2. The NTRU Cryptosystem

This is the ordinary polynomial product in $P_q(N)$, and is both commutative and associative. The symmetric representation of $\mathbb{Z}_q$ ensures that the product of two polynomials with coefficients of small absolute value will again be a polynomial with small coefficients.

The multiplicative group of units in $P_q(N)$ we shall denote by $P_q^*(N)$ and the inverse polynomial of $f \in P_q^*(N)$ is denoted by $f_q^{-1}$.

We will also require a “small” element of $P(N)$ which is relatively prime to $q$, which we shall denote by $p$. Typically $p$ is chosen to be equal to one of $2, 3, \text{ or } 2 + X$.

Reduction modulo $p$ when $p$ is equal to 2 or 3 is conducted in the standard way to produce a representative either in the set $\{0, 1\}$ or the set $\{-1, 0, 1\}$. When $p = 2 + X$ a slightly non-standard reduction is carried out, signified by the use of $p = 2 + X$ rather than $p = X + 2$. By writing $2 + X$ we are signifying that the term 2 is of higher priority than $X$ in the reduction. The reduction of a polynomial modulo $2 + X$ proceeds by rewriting each integer $n = 2a + b$ as $(-X)a + b$. Hence, we rewrite 2 as $-X$ as opposed to the more standard rewriting of $X$ as $-2$. As an example of these two different types of reduction consider

$$X^4 + 6X + 2 \pmod{X + 2} = 6$$
$$X^4 + 6X + 2 \pmod{2 + X} = X^4 + 3(-X)X + (-X)$$
$$= X^4 - 3X^2 - X$$
$$= X^4 + (X^2 - 4X^2) + (X - 2X)$$
$$= X^4 + (X^2 - (-X)^2X^2) + (X - (-X)X)$$
$$= 2X^2 + X$$
$$= -X^3 + X$$
$$= X^3 - 2X^3 + X$$
$$= X^4 + X^3 + X.$$

It is easily seen that reduction modulo $2 + X$ always leads to a polynomial with coefficients in $\{0, 1\}$.

We now define $P_p(N)$ to be the elements in $P(N)$ reduced modulo $p$, the multiplicative group of units in $P_p(N)$ we shall denote by $P_p^*(N)$ and the inverse polynomial of $f \in P_p^*(N)$ is denoted by $f_p^{-1}$.

4.2.2 The NTRU Primitive

We sketch the NTRU cryptosystem, as developed in [HPS98]. The public parameters consist of values for $(N, p, q)$ as above with with $p$ and $q$ relatively prime, and $q$ will always be considerably larger than $p$. The value of $q$ is chosen to lie between $N/2$ and $N$ and can be chosen to aid computation. For example for the “recommended” security parameter $N = 251$ for “standard security” one could choose $q = 128$ or $q = 127$ so as to aid in reduction modulo $q$.

Other required parameters are various pairs of integers $(d_1, d_2)$ which are used to define several families of trinary polynomials of $P_q(N)$ as follows: The notation $L(d_1, d_2)$ is used...
to denote the set of polynomials in $P_q(N)$, with $d_1$ coefficients equal to 1, $d_2$ coefficients equal to $-1$ and all other coefficients set to zero. These sets are used to define three sets of polynomials $L_f$, $L_g$ and $L_r$. The literature contains three common choices for these sets:

**Choice A**
This is the choice used in earlier academic papers on the NTRU system, where

$$L_f = L(d_f, d_f - 1), \quad L_g = L(d_g, d_g) \quad \text{and} \quad L_r = L(d_r, d_r),$$

for certain parameters $d_f, d_g$ and $d_r$ dependent on the security parameter $N$.

**Choice B**
This choice is one adopted in the standard [CEES02]. We have

$$L_f = \{1 + p \cdot f_1 : f \in L(d_f, 0)\},$$
$$L_g = L(d_g, 0),$$
$$L_r = L(d_r, 0),$$

for certain parameters $d_f, d_g$ and $d_r$. Notice, that the choice of $f$ means computing $f_{p^{-1}}$ is particular easy.

**Choice C**
This is also a choice adopted in the standards, but produces polynomials slightly larger than those in Choice B. In this case we have

$$L_f = \{1 + p \cdot (f_1 \cdot f_2 + f_3) : f_i \in L(d_{f,i}, 0)\},$$
$$L_g = L(d_g, 0),$$
$$L_r = \{r_1 \cdot r_2 + r_3 : r_i \in L(d_{r,i}, 0)\},$$

for certain parameters $d_{f,1}, d_{f,2}, d_{f,3}, d_g, d_{r,1}, d_{r,2}$ and $d_{r,3}$.

A public key encryption algorithm consists of three sub-procedures: A key generation algorithm, an encryption algorithm and a decryption algorithm. The following paragraphs describe these procedures in the context of the NTRU algorithm.

**Key Creation**
The generation of public/private keys then proceeds as follows:

1. Choose random $f \in L_f$ and $g \in L_g$.
2. Compute $f_{q^{-1}} \in P_q(N)$ and $f_{p^{-1}} \in P_p(N)$. See later for precise details on how this is done in practice.
3. If one of these inverses does not exist choose a new $f$. Otherwise $f$ serves as the secret key.
4. Publish the polynomial
\[ h \equiv p \cdot f_q^{-1} \cdot g \pmod{q} \] (3)
as the public key.

Let \( f = (f_0, f_1, \ldots, f_{N-1}) \in P_q(N) \). Then
\[ f \cdot x^i = (f_{N-i}, f_{N-i+1}, \ldots, f_{N-1}, f_0, \ldots, f_{N-i-1}), \quad (i \in \mathbb{Z}_{>1}) \].

Thus, if \( f \) is the secret key of NTRU, then \( \pm(f \cdot x^i) \) is also a secret key for any integer \( i > 1 \).

Note that for parameter Choice’s B and C we have \( f_p^{-1} = 1 \) and so we do not have to compute this value.

**Encryption**

In NTRU encryption is probabilistic, in the sense that encrypting the same message twice will result in different ciphertexts. To encrypt a plaintext \( m \), which we consider as given by a polynomial in \( P_p(N) \), we perform the following two steps.

1. Choose random \( r \in \mathcal{L}_r \).
2. Compute
\[ e = \mathcal{E}_h(m; r) = r \cdot h + m \pmod{q}. \] (4)

**Decryption**

Given a ciphertext \( e \) and a private key, i.e. \( f_q^{-1} \) and \( f_p^{-1} \), decryption then proceeds as follows:

**Step 1:**
First we recover the value of \( p \cdot r \cdot g + m \cdot f \) as an element of \( P(N) \)
\[ a \equiv e \cdot f \pmod{q} \] (5)
\[ \equiv r \cdot p \cdot f_q^{-1} \cdot g \cdot f + m \cdot f \pmod{q} \]
\[ \equiv p \cdot r \cdot g + m \cdot f \pmod{q} \]

**Step 2:**
Assume that we have now recovered \( a = p \cdot r \cdot g + m \cdot f \) in \( P(N) \). We then switch to reduction modulo \( p \), computing
\[ a \cdot f_p^{-1} \equiv p \cdot r \cdot g \cdot f_p^{-1} + m \cdot f \cdot f_p^{-1} \pmod{p} \] (6)
\[ \equiv m \cdot f \cdot f_p^{-1} \pmod{p} \]
\[ \equiv m \pmod{p} \].

and recover the plaintext \( m \in P_p(N) \). Note that for parameter Choices B and C this calculation simplifies to
\[ a \equiv m \pmod{p}, \]
since \( f_p^{-1} = 1 \).
Notice, that the message space is defined to be $P_p(N)$, whilst the ciphertext lies in $P_q(N)$. In practice $q$ is chosen a lot larger than $p$ and so this leads to a large expansion rate for a message. With typical values for $p$ and $q$ one can have that the ciphertext is seven to eight times larger than the underlying plaintext. A similar expansion happens in practice for RSA, where to encrypt a 128 bit session key one embeds it into an RSA message block of over 1024 bits.

In the originally described NTRU, the public key is set as $h = f_q^{-1} \cdot g \pmod{q}$ and the ciphertext is computed as $e = p \cdot r \cdot h + m \pmod{q}$. However, in later works e.g. [CEES02] the proposers of NTRU changed this to equation’s (3) and (4), with the benefit of saving multiplication with $p$ during encryption. In accordance with recent works we will use (3) and (4) throughout this chapter.

The computation in equation (5) is where the heart of the NTRU algorithm lies. It is important in computing $a$ that we recover $p \cdot r \cdot g + m \cdot f$ precisely, and not just modulo $q$. The following proposition is used by the authors of NTRU to ensure, with high probability, the correctness of the above procedure when one has an integer value of $p$.

**Theorem 4.2.1** For any $\epsilon > 0$ the are constants $\gamma_1, \gamma_2 > 0$ depending on $\epsilon$ and $N$, such that for randomly chosen polynomials $f = (f_0, \ldots, f_{n-1}), g = (g_0, \ldots, g_{n-1}) \in P(N)$, the probability is greater than $1 - \epsilon$ that they satisfy

$$\gamma_1 \|f\|_2 \cdot |g|_2 \leq \|f \cdot g\|_{\infty} \leq \gamma_2 \|f\|_2 \cdot |g|_2.$$ 

The norm $\|f\|_2 := (\sum_{i=0}^{n-1} (f_i - \bar{f})^2)^{1/2}$, with $\bar{f} := 1/n \sum_{i=0}^{n-1} f_i$ denotes a centred $l_2$-norm on $P(N)$ and $\|f\|_{\infty} := \max_{0 \leq i \leq n-1} |f_i| - \min_{0 \leq i \leq n-1} |f_i|$.

For the computation of $p \cdot r \cdot g + m \cdot f$ to be correct we will need $\|p \cdot r \cdot g + f \cdot m\|_{\infty} \leq q$ to recover the correct $a$. The latter is almost always true, if

$$\|p \cdot r \cdot g\|_{\infty} \leq q/4 \text{ and } \|f \cdot m\|_{\infty} \leq q/4.$$ 

Therefore the authors of NTRU suggest, following Theorem 4.2.1, the following parameter choices:

$$\|r\|_2 \cdot |g|_2 \approx q/(4\gamma_2) \text{ and } \|f\|_2 \cdot |m|_2 \approx q/(4\gamma_2).$$

With these parameter choices the probability that Step 1 of the decryption process proceeds correctly is very high. However, it will be shown in § 4.4 that it is still not sufficient to prevent NTRU from attacks which exploit the possibility of a valid encryption not being able to be decrypted (cf. Proos [Pro03]).

Equation (5) of the NTRU primitive provides a nice classification of indecipherable messages: A valid ciphertext $e = r \cdot h + m$ will be indecipherable if and only if at least one coefficient of $p \cdot r \cdot g + m \cdot f$ has a value outside the interval $(-q/2, q/2]$. This problem of indecipherable ciphertexts basically comes down to the fact that knowing a number modulo $q$ does not uniquely determine what it is modulo $p$. 


4.2. The NTRU Cryptosystem

Let \( b = p \star r \star g + m \star f \) be a typical polynomial. Note, we have not reduced \( b \) modulo \( q \). Performing the calculation

\[
b \star f_p^{-1} = p \star r \star g \star f_p^{-1} + m \star f \star f_p^{-1} \pmod{p}
\]
gives us exactly the original message \( m \). Unfortunately during NTRU decryption we do not have \( b \), but only \( b \) modulo \( q \).

If \( a = e \star f = b \pmod{q} \) is not exactly the same as \( b \), then it may no longer be true that \( a \star f_p^{-1} \equiv m \pmod{p} \). Now suppose

\[
b = b_0 + b_1 X + \cdots + b_{N-1} X^{N-1}
\]
and let

\[
\text{Max } b = \max_{1 \leq i \leq N-1} \{ b_i \}, \quad \text{Min } b = \min_{1 \leq i \leq N-1} \{ b_i \}, \quad \text{Spread } b = \text{Max } b - \text{Min } b
\]

Silverman [Sil01] defines two types of NTRU decryption failures.

- First we say that a wrapping failure occurs if either \( \text{Max } b > q/2 \) or \( \text{Min } b \leq -q/2 \).
- Secondly we say a gap failure occurs if \( \text{Spread } b \geq q \).

If a wrapping failure, but not a gap failure occurs, the correct value of \( b \) can be determined by changing the range into which the coefficients are reduced to

\[ [A, A + q), \]
for some value of \( A \), instead of \([-q/2, q/2)\).

In the CEES standards [CEES02], the value of \( A \) given by

\[
A := \left[ \frac{1}{N} \left( p(1) \cdot r(1) \cdot g(1) + I \cdot f(1) + \frac{N}{2} \right) \right] - \frac{q}{2},
\]
is specified, where

\[
I = (a(1) - p(1) \cdot r(1) \cdot g(1)) \cdot f_q^{-1}(1) \pmod{q}.
\]

This value \( A \) corresponds to the average decryption coefficient. If this value of \( A \) and the interval \([A, A + q)\) results in an invalid decryption then the standard proposes one tries to decrypt again using \( A = A \pm 1, A \pm 2 \) etc until successful. Such a procedure will work assuming no gap error occurs. Since wrap errors are more common than gap errors this is a possible solution. However, a timing analysis could reveal the existence of a wrap error since a wrap error will result in the need for further decryptions and so a longer decryption time. In addition one is still left with the problem of gap errors.

We end this short digression on wrap and gap errors by noting that the above trick for dealing with wrap errors by shifting the reduction interval to \([A, A + q)\) also applies when \( p \) is a polynomial. In addition note that the values of \( p(1), g(1), f(1) \) and \( r(1) \) are known to
the decryptor. The value of \( r(1) \) is known exactly to the decryptor since they know how \( r \) is selected. For example if

\[
\begin{align*}
r \in \mathcal{L}_r &= \{ r_1 \ast r_2 + r_3 : r_i \in \mathcal{L}(d_{r,i}, 0) \} \\
r(1) &= r_1(1) \cdot r_2(1) + r_3(1) = d_{r_1}d_{r_2} + d_{r_3}
\end{align*}
\]

then we know

\[
r(1) = r_1(1) \cdot r_2(1) + r_3(1) = d_{r_1}d_{r_2} + d_{r_3}
\]

without knowing the value of \( r \).

The NTRU public key cryptosystem has two apparent disadvantages: Firstly, its expansion rate and secondly its sensitivity to lattice reduction attacks. To overcome the latter problem Hoffstein and Silverman [HS97] proposed in 1997 a non commutative variant of the NTRU public key cryptosystem, which operates on the non commutative polynomial ring of the dihedral \( \mathbb{Z}_q[X,Y]/(X^n - 1, Y^2 - 1, XYXY - 1) \). However, this system was completely broken by Coppersmith [Cop97]. So far, the most significant and successful lattice based attacks came from Coppersmith and Shamir [CS97]. They applied basis reduction techniques to cryptanalyze the scheme, either to discover the original secret key, or an alternative secret key which is equally useful in decoding the ciphertexts. More specifically, they showed that any non-trivial lattice vector of \( L^{NT} \), at most as long as the original secret key (the target vector) can be used for decryption. In addition, if it is possible to find two vectors not longer than 2.5 times the target vector partial information about the plaintext can be obtained and combined to recover the whole message.

As alluded to above there is another difficulty with the NTRU cryptosystem, and many of the suggested padding schemes which we shall discuss later, namely Proos [Pro03] describes several attacks using the fact that NTRU does not provide perfect decryption, due to either wrap or gap errors. This means there exist ciphertexts which can be validly created using the public key but will not correctly decrypt to the original plaintext. These ciphertexts can be used to recover the private key, a topic which we shall turn to later.

4.2.3 Parameter Choices

For the security of NTRU, it is essential that \( p \) and \( q \) have no common factors. For example, if we are given positive integers \( p, q \) with \( p \mid q \) the encrypted message \( e \) in (4) satisfies \( e \equiv m \) (mod \( p \)) and NTRU will become completely insecure. In addition, it is highly advisable to choose \( N \) to be a prime. Firstly, because having \( N \) prime maximises the probability that the private key has an inverse with respect to a specified modulus, but more importantly to avoid so called composite attacks on NTRU (cf. Gentry [Gen01]), which recover the private key.

The following Table 1 gives some recommended values for NTRU parameters, the values for Choice A are taken from academic papers on NTRU whilst those for Choice B and C are taken from the CEES standard [CEES02]. Since it is difficult to predict and analyse all of the kinds of attacks to which a given public key cryptosystem is susceptible, there is no reliable way of measuring the security of such a system. The convention NTRU uses is that the security is defined as \( \sqrt{\#\mathcal{L}(d_g, d_g)} = 1/d_g \sqrt{N/(N - 2d_g)} \). This is based on the time a meet-in-the-middle attack (cf. [HPS96], [SO97]) needs for finding the private key \( g \). Accordingly parameters in Table 1 have been chosen in such a way that the security is
4.2. The NTRU Cryptosystem

4.2. The NTRU Cryptosystem

equivalent to a symmetric key strength of $2^{50}$ (Low Security), $2^{83}$ (Moderate Security), $2^{108}$ (Standard Security) and $2^{285}$ (Highest Security) respectively. However, it has been shown that the suggested $N = 107$ parameters can be broken by lattice attacks in a few hours [MS01]. Thus they are no longer considered secure.

Table 1: Suggested parameters for NTRU

<table>
<thead>
<tr>
<th>Security</th>
<th>Parameters</th>
<th>$N$</th>
<th>$q$</th>
<th>$p$</th>
<th>$d_f$</th>
<th>$d_g$</th>
<th>$d_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Choice A</td>
<td>107</td>
<td>64</td>
<td>3</td>
<td>15</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>Moderate</td>
<td>Choice A</td>
<td>167</td>
<td>128</td>
<td>3</td>
<td>61</td>
<td>20</td>
<td>18</td>
</tr>
<tr>
<td>Moderate</td>
<td>Choice B</td>
<td>139</td>
<td>127</td>
<td>2+$X$</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>Standard</td>
<td>Choice A</td>
<td>251</td>
<td>128</td>
<td>3</td>
<td>50</td>
<td>24</td>
<td>16</td>
</tr>
<tr>
<td>Standard</td>
<td>Choice C</td>
<td>251</td>
<td>128</td>
<td>2+$X$</td>
<td>8</td>
<td>72</td>
<td>8</td>
</tr>
<tr>
<td>High</td>
<td>Choice C</td>
<td>347</td>
<td>128</td>
<td>2+$X$</td>
<td>8</td>
<td>173</td>
<td>8</td>
</tr>
<tr>
<td>Highest</td>
<td>Choice A</td>
<td>503</td>
<td>256</td>
<td>3</td>
<td>216</td>
<td>72</td>
<td>55</td>
</tr>
<tr>
<td>Highest</td>
<td>Choice C</td>
<td>503</td>
<td>256</td>
<td>2+$X$</td>
<td>20</td>
<td>251</td>
<td>14</td>
</tr>
</tbody>
</table>

To simplify our presentation, we will mainly restrict ourselves to parameter Choice A, with $p = 3$, $q$ a power of 2. This means that we know that $f(1) = 1$ and $g(1) = r(1) = 0$. We emphasize, however, that the attacks described later will also work when $q$ is not a power of 2 and can be easily modified to work when the integer $p$ is not 3 as well as for other values for constants $f(1), g(1)$ and $r(1)$.

4.2.4 Implementation Details

The encryption and decryption operations are very elementary with NTRU, bar the possible processing to deal with wrap failures which we discussed above. To encrypt a message the time is dominated by a single convolution product and to decrypt we need to perform two convolution products, which can be reduced to one if we choose a private key of the form

$$f = 1 + p \ast F$$

since then $f_p^{-1} = 1$.

Hence, the only problem with implementing NTRU is likely to be in the key generation procedure. The main problem here is: given a polynomial $f \in \mathcal{L}_f$ compute its inverse $f^{-1}_q \in P_1^*(N)$. An efficient algorithm to do this is given in [Sil99] which uses the almost-inverse algorithm of Schroeppel et. al. [SMOS95].

We are only interested in the case where $q$ is a power of a prime $l$, so we write $q = l^r$. For now assume that we can find the inverse $f_l^{-1}$ of $f$ in $P_1^*(N)$, then using the following Newton iteration we can compute $f^{-1}_q$ in $s := \lfloor \log_2(r) \rfloor$ steps. We repeat for $i = 1, 2, \ldots, s$ the following iteration, starting with $g_0 = f_l^{-1}$

$$g_i \equiv g_{i-1} \ast (2 - f \ast g_{i-1}) \pmod{l^2}.$$
On finishing this iteration we have \( f_q^{-1} = g_s \). The proof is by induction on \( i \). For \( i = 0 \) we have
\[
f \ast g_0 \equiv 1 \pmod{l^2}.
\]
For the induction step, we find
\[
1 - f \ast g_{i+1} \equiv 1 - f \ast (g_i \ast (2 - f \ast g_i)) \\
\equiv 1 - 2f \ast g_i + f^2 \ast g_i^2 \\
\equiv (1 - f \ast g_i)^2 \\
\equiv 0 \pmod{l^{2i+1}}.
\]
All that remains is to compute \( f_l^{-1} \), a task which is achieved using the almost-inverse algorithm. We clearly need to assume that
\[
\gcd(f, X^N - 1) = 1.
\]
The following procedure then computes \( f_l^{-1} \) from \( f \), where we let \( f_i \) denote the \( i \)-th coefficient of \( f \).

1. Set \( k = 0 \), \( b = 1 \), \( c = 0 \), \( g = X^N - 1 \).
2. While \( f_0 = 0 \) set \( f = f/X \), \( c = c \ast X \), \( k = k + 1 \).
3. If \( \deg(f) = 0 \) then set \( b = b/f_0 \pmod{l} \) and output \( X^{N-k} \ast b \).
4. If \( \deg(f) < \deg(g) \) then swap \( f \) and \( g \) and swap \( b \) and \( c \).
5. Set \( u = f_0/g_0 \pmod{l} \)
6. Set \( f = f - u \ast g \pmod{l} \) and \( b = b - u \ast c \pmod{l} \).
7. Return to step 2.

### 4.3 NTRU and Lattice Reduction

Recall the public key is a polynomial \( h \in P_q(N) \) and the private key is essentially given by two "small" polynomials \( f \) and \( g \) such that
\[
h = f^{-1} \ast q \pmod{q}.
\]
In general, elements of the ring \( P_q(N) \) do not have a unique factorisation. Therefore we call two polynomials \( u \) and \( v \) "a" factorisation of \( h \in P_q(N) \), if \( u \ast h \equiv v \) holds in \( P_q(N) \). The security of NTRU is then based on the following complexity assumption:

**Assumption:**
Given a polynomial \( h \in P_q(N) = \mathbb{Z}[X]/(X^N - 1) \) with \( h = f_q^{-1} \ast g \), where the coefficients of the secret keys of \( f \) and \( g \) are small compared to \( q \). For appropriate choices of \( N \) it is hard to recover one of the polynomials \( f \) or \( g \) from \( h \) or find two polynomials \( u, v \) with small...
coefficients such that \( u \ast h \equiv v \pmod{q} \).

There are no statements about the hardness of the above polynomial factorisation problem in complexity theory, but from the following heuristic argument it is very likely that this is a difficult problem: Every polynomial \( u \in P_q(N) \) coprime modulo \( q \) to \( X^N - 1 \), has an inverse in \( P_q(N) \) and therefore gives a solution to the factorisation problem. Thus there are \( |P_q^*(N)| = q^N \) possible factorisations of which only those with small \( l_2 \)-norm are useful for decryption. Up to now, there are no polynomial time algorithms known to solve this problem. But because the secret polynomials \( f \) and \( g \) have small \( l_2 \)-norm lattice based attacks on the public key \( h \) might be a good strategy, if the polynomial factorisation problem can be translated into a lattice problem. So consider the set of vectors

\[
L = \{(u, v) : u \ast h \equiv v \pmod{q}, \ u, v \in \mathbb{Z}^N\} \subseteq \mathbb{Z}^{2N}.
\]

The set \( L \) forms a lattice in \( \mathbb{R}^{2N} \), which clearly contains the vector \((f, g)\). Thus, if we could find a basis for \( L \) then finding short vectors in \( L \) might return \((f, g)\). Coppersmith and Shamir [CS97] developed a basis for what will be called the NTRU Lattice \( L^{NT} \). It is spanned by the row vectors of the \((2N \times 2N)\)-matrix

\[
L^{NT} = \begin{pmatrix}
\lambda & 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{N-1} \\
0 & \lambda & \cdots & 0 & h_1 & h_2 & \cdots & h_0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \lambda & h_{N-1} & h_0 & \cdots & h_{N-2} \\
0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\
\end{pmatrix}, \ (\lambda \in \mathbb{R}).
\]

Notice that this lattice is not the same as \( L \). However, Coppersmith and Shamir [CS97] showed, if \( u = (u_0, \ldots, u_{N-1}), v = (v_0, \ldots, v_{N-1}) \) is an arbitrary factorisation of \( h \in P_q(N) \), then the NTRU Lattice \( L^{NT} \) contains the vector \((\lambda \sigma(u), v)\) with \( \sigma(u) = (u_0, u_{N-1}, u_{N-2}, \ldots, u_2, u_1) \). Thus, in particular \((\sigma(f), g) \in L^{NT} \). Although the vector \((\sigma(f), g)\) is not known to be the shortest vector in the NTRU Lattice, Coppersmith and Shamir [CS97] proved that shorter vectors correspond to alternative private keys. Since current algorithms to find the shortest vector in a given lattice are exponential in time with respect to the dimension, they are (for appropriate parameter choices) no threat for the security of NTRU.

Experiments from various authors seem to agree on how hard it is to recover an NTRU private key from lattice reduction. Table 2 is indicative of the experiments which have been performed, and the extrapolation thereof. Note, that different authors will have slightly different estimates due to differences in machines and in the dates at which the experiments were run. However, all authors figures are in the same region so we can have high confidence of the rough times needed to recover a key using lattice attacks.
4. The NTRU Public-Key Cryptosystem

<table>
<thead>
<tr>
<th>Security Parameters</th>
<th>Breaking Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>2.5 Days</td>
</tr>
<tr>
<td>Moderate</td>
<td>30 Years</td>
</tr>
<tr>
<td>Standard</td>
<td>$10^{14}$ Years</td>
</tr>
<tr>
<td>Highest</td>
<td>$10^{21}$ Years</td>
</tr>
</tbody>
</table>

4.4 Security

The fact that the NTRU system has the possibility of imperfect decryption has led to some interesting issues in the development of the NTRU algorithm.

**Definition 4.4.1** To avoid confusion with other literature we shall use the term public key scheme (rather than use the suffix perfect) to mean the following:

Given a message space $\mathcal{M}$ there is a triple of algorithms, $\Pi = (K, E, D)$, where:

1. $K(1^k)$, the key-generation algorithm, is a probabilistic algorithm which on input of a security parameter $k \in \mathbb{N}$ produces a pair $(pk, sk)$ of matching public and private keys.

2. $E_{pk}(m; r)$, the encryption algorithm, which returns a ciphertext $c \in \mathcal{C} \subseteq \{0, 1\}^*$ corresponding to the plaintext $m \in \mathcal{M}$, using a random bit string $r$ according to the public key $pk$.

3. $D_{sk}(y)$, the decryption algorithm, is a deterministic algorithm which on input of the secret key $sk$ and an arbitrary $y \in \mathcal{C}$ returns a message $x \in \mathcal{M}$ or $\perp$. If $\perp$ is returned, then $y$ is an invalid ciphertext i.e. $y$ is not in the range of $E_{pk}$. In other words there exists no $m \in \mathcal{M}$ and $r$ with $y = E_{pk}(m; r)$.

4. For any $k \in \mathbb{N}$ the following holds: For all $(pk, sk)$ which can be output of $K(1^k)$ and all $m \in \mathcal{M}$ that if $c = E_{pk}(m; r)$, for any $r$, then $D_{sk}(c) = m$.

A few points are worth noting in this definition:

- The security parameter $k$ in the key generation function is the parameter which is usually used to measure the security of the scheme. For example in an RSA scheme $k$ will be the bit length of the modulus, whilst in an elliptic curve based scheme $k$ will be the base-2 logarithm of the size of the elliptic curve group. For NTRU the value of $k$ is given by the parameter $N$. Note however that security parameters of two different schemes are not comparable.

- We allow probabilistic encryption algorithms, for example ElGamal. However, this definition does not exclude non-probabilistic algorithms such as textbook RSA. For non-probabilistic algorithms we set the random string $r$ in encryption to be empty. However,
4.4. Security

we shall see later that non-probabilistic schemes are considered insecure under modern
security definitions.

- The last property guarantees that if a plaintext $m$ is encrypted using $E_{pk}$ and the
resulting ciphertext is subsequently decrypted using $D_{sk}$, then the original plaintext $m$
results. This property holds for all the standard public key algorithms such as textbook
RSA, however it does not hold for NTRU as we have already remarked.

- According to the third property it may happen that for an invalid $y \in C$, i.e. a ciphertext
which cannot be obtained from a valid encryption, the decryption algorithm returns a
message $x \in M$. Thus, following Proos [Pro03], we call a public key scheme restricted,
if for every invalid $y \in C$ the symbol $\perp$ is returned.

If the fourth property is violated, as it is in NTRU, i.e. there exists a public/private key
pair $(pk, sk)$ and a message nonce pair $(m, r)$ for which

$$D_{sk}(E_{pk}(m; r)) \neq m,$$

then the public key cryptosystem is called imperfect. In other words the encryption algorithm
of an imperfect encryption algorithm is not necessarily injective, whereas for a standard
encryption algorithm this is forced by definition (property 4). If we are given an imperfect
scheme, we refer to a valid ciphertext $c = E_{pk}(m; r)$ with $D_{sk}(c) \neq m$ as indecipherable with
respect to $m$, otherwise we call the valid ciphertext decipherable ($D_{sk}(c) = m$).

We can extend the definition of restricted public key scheme to an imperfect public key
scheme in the obvious way.

Proos describes in [Pro03] DCA (decipherable ciphertext attack) attacks against NTRU.
Two DCA attacks against the textbook NTRU primitive are presented, and they are then
extended to the versions of the primitive which make use of paddings in order to fulfill some
stronger security criteria (including indistinguishability of ciphertexts under chosen ciphertext
attacks, see [JJ00]).

We describe the main ideas of the two DCA attacks against the NTRU primitive. Besides
access to the DC oracle the adversary is also given the public key $h$.

For the first attack we assume that the adversary has freedom to choose $m \in P_p(N)$ and
$r \in \mathcal{L}_r$. The adversary then recovers the private key in three stages:

Step 1. Find $(m, r) \in P_p(N) \times \mathcal{L}_r$ which lead to an indecipherable ciphertext $c := E_{pk}(m, r) =
pr \ast g + m \ast f$, e.g. $D_{sk}(c) \neq m$. Thus there exists at least one coefficient of $c$ outside
the interval $[-q/2, q/2]$.

Step 2. Using the $(m, r)$ found in Step 1 one then finds a message $\tilde{m} \in P_p(N)$ with
$D_{sk}(E_{pk}(\tilde{m}, r)) \neq \tilde{m}$ such that if any nonzero bit of $\tilde{m}$ is set to zero then $E_{pk}(\tilde{m}, r)$
is decipherable.

It follows that $\tilde{c} := E_{pk}(\tilde{m}, r) = pr \ast g + \tilde{m} \ast f$ has coefficients in $[-q/2, q/2 + 1]$.
In addition $\tilde{m}$ should satisfy the condition that exactly one of the coefficients of $\tilde{c}$
is in the set \( \{-q/2, q/2 + 1\} \), whereas the rest of the coefficients lies in the interval \([-q/2 + 2, q/2 - 1]\).

The message \( \tilde{m} \) is deduced from \( m \) by successively setting coefficients to zero. If \( \tilde{m} = 0 \) or the condition does not hold, then we return to Step 1.

**Step 3.** The coefficients \( f_{j_i} \) of the private key \( f \) can be determined by setting the \( i \)-th coefficient of \( m \) to \( 1, 0 \) and \( 1 \) respectively and checking whether the corresponding ciphertext \( E_{pk}(\tilde{m}, r) \) is decipherable.

Practical results (cf. [Pro03]) show that one is very likely to find pairs \((m, r)\) which lead to a successful completion of Step 2 and 3.

For the second attack we assume that the adversary has freedom to choose \( m \in P_p(N) \), but \( r \in L_r \) must be selected at random. This time the polynomial \( g \) is recovered. Once \( g \) has been determined the private key \( f \) can be found by solving the system of linear equation

\[
\begin{pmatrix}
h_0 & h_1 & \ldots & h_{N-1} \\
h_1 & h_2 & \ldots & h_0 \\
\vdots & \ddots & \ddots & \vdots \\
h_{N-1} & h_0 & \ldots & h_{N-2}
\end{pmatrix}
\begin{pmatrix}
f_0 \\
f_1 \\
\vdots \\
f_{N-1}
\end{pmatrix}
= \begin{pmatrix}
g_0 \\
g_1 \\
\vdots \\
g_{N-1}
\end{pmatrix} \pmod{q}.
\]

The first stage of this attack is the same as for the last DCA attack, so suppose we are given a tuple \((m, r) \in P_p(N) \times L_r \) which leads to an indecipherable ciphertext.

**Step 2.** Randomly search \( \bar{r} \in L(d_r, d_r) \), such that \((m, \bar{r})\) is indecipherable.

**Step 3.** Analyse the distribution of 1’s and −1’s in all the \( \bar{r} \)'s found in Step 2 to recover the nonzero coefficients of \( g \). If not all nonzero coefficients are found then go back to Step 2.

These attacks can also be used as a basis for attacks against all known NTRU padding schemes (devised to thwart chosen ciphertext attacks) and are effective to different extent. For references on the original NTRU padding schemes (Padding I, II, III) see [HS01, Sil98] and for the new padding schemes (NTRU-OAEP, NTRU-REACT, NTRU-REACT2) proposed by Nguyen and Pointcheval see [Np02].

The attacks in these cases rely on the fact that a validly created ciphertext for any of the padding schemes will decrypt to the original message exactly when the underlying NTRU primitive ciphertext is decipherable. Implementation results [Pro03] show that the attacks against NTRU-REACT are very practical. For the parameter set of the highest security level \((N = 503)\) in Table 1 on average fewer than 30 000 DC oracle calls are required. This corresponds to an average computing time of only a few minutes on a modern PC.

All these attacks show that finding even a single indecipherable ciphertext can lead to the recovery of the private key. Hence, the fact that NTRU exhibits indecipherable ciphertexts
can lead to serious problems with the encryption algorithm, even when padding schemes are used. Hence, one needs to choose parameters for the NTRU system so that the probability of being able to create an indecipherable ciphertext is negligible.

4.5 Conclusion

We have introduced the NTRU encryption function and shown how this on its own does not provide a secure encryption scheme. There is a lot of work still remaining to be done in understanding how padding systems can be applied to mathematical primitives such as NTRU to enable the creation of secure encryption schemes. However, for parameters large enough so that the probability of creating an indecipherable valid ciphertext is negligible, NTRU can lead to a secure system, but further investigations are required.

References – NTRU


4. The NTRU Public-Key Cryptosystem


5

XTR, Subgroup- and Torus-based Cryptography

Contributors: Rob Granger, Arjen K. Lenstra, Martijn Stam, and Roberto Avanzi

5.1 Introduction

The cryptosystems discussed in this Chapter are all based on the DLP in a finite field and are based on the idea of working in a cyclic subgroup $G$ of the multiplicative subgroup of a finite field $\mathbb{F}_{q^n}$, in such a way that the following properties hold

1. Working in $G$ is very efficient, possibly more efficient than in the full group — the subgroup must be small, but also large enough that Pollard rho attacks in it be ineffective.

2. The security of the system should depend also on the difficulty of solving the DL in the full finite field $\mathbb{F}_{q^n}$ — hence the subgroup should not be contained in a proper subfield of $\mathbb{F}_{q^n}$.

3. There should be a compact representation of the elements of $G$ which is much shorter than the representation of all the elements of $\mathbb{F}_{q^n}$ — this is often attained by considering the traces of the elements of $G$ to one or more of the subfields of $\mathbb{F}_{q^n}$. Ideally, $\phi(n)$ elements of $\mathbb{F}_q$ should suffice to represent the elements of $G$, in some cases this seems to be difficult and a few additional bits are necessary.

In fact, when Diffie and Hellman introduced their key agreement scheme in a finite field of prime order, they made the assumption that a birthday attack was the best one can do (the Pohlig-Hellman algorithm [PoHe78] was in submission). Hence it made sense to use a subgroup of size about that of the field itself. Since then, there has been tremendous progress in computing discrete logarithms in the multiplicative group of a finite field. For prime order subgroups the best known attacks today are the birthday attack in the subgroup itself and the Number Field Sieve in the full finite field. It has therefore become common practice to use a subgroup whose cardinality is substantially smaller than the field size, and the issue of compression, i.e. of finding a compact representation for the subgroup elements has become increasingly important.

In a series of papers culminating in [Luc78], Edouard Lucas introduced and explored the properties of certain recurrent sequences that became known as Lucas functions. Since then, generalizations and applications of Lucas functions have been studied (see [Wil72, Wil82, Wil98]), and public key DLP based cryptosystems (such as LUC) have been based on them (see [MuNo81, SmLe93, SmSk94, Wil85, BBL95]). The Lucas functions arise when studying quadratic field extensions. Cryptographic applications of generalizations to cubic and sextic field extensions are given in [GoHa99] and [BPV99, LeVe00a], respectively. The cryptosystem in [LeVe00a] is called XTR and it is, arguably, the most famous cryptosystem of this kind. An approach for constructing a generalization of these cryptosystems to the case of degree 30 extensions is suggested in [BPV99, BHV02]. For XTR and LUC, traces are used to represent...
the elements - XTR was the first instance of a significant compression for the subgroup elements in this line of research, and was an improvement of the system proposed in [BPV99]. The idea of these cryptosystems is to represent certain elements of $F_{q^n}$ (for $n = 2, 6, 30$ respectively) using only $\phi(n)$ elements of $F_q$, and do a variant of the Diffie-Hellman key exchange protocol.

In [BHV02], this approach is generalised in that symmetric functions are proposed in place of the trace (which is the first symmetric function). In fact, by using traces (or other higher symmetric functions) to represent the elements, such systems can be shown to work in a quotient set of the considered subgroup and this can be interpreted as the reason for which the group structure is not directly implemented: instead, by using the symmetric-function based representation of several elements, a chain for computing the desired final result (the symmetric functions of the result of the exponentiation) is given, but simple products of elements of the corresponding group cannot be easily implemented. These representations usually have the distinctive advantage of allowing a very fast arithmetic, but Stam and Lenstra show in [StLe01] that similar (and sometimes better) performance can be obtained by working in a smart way directly in the large field.

Rubin and Silverberg [RuSi03] recast the problem in the language of algebraic tori to construct public key cryptosystems. For any positive integer $n$ one can define an algebraic torus $T_n$ over $F_q$ whose $F_q$-points consist of the elements of $F_{q^n}$ whose norm down to every proper subfield of $F_{q^n}/F$ is one. The systems designed on $T_n$ are thus too on the DLP in a subgroup of $F_{q^n}$ in which the elements can be represented by only $\phi(n)$ elements of $F_{q^n}$. Like [StLe01] and unlike LUC, the original version of XTR, and the conjectured system of [BHV02], they make direct use of the group structure of the torus, hence also of the structure of the large field, thereby allowing more flexibility in designing the cryptographic applications. They showed that if the algebraic torus $T_n$ is rational, the conjectured compression factor $n/\phi(n)$ can in fact be achieved. If $n$ is the product of at most two prime powers then $T_n$ is known to be rational [Kly88, Vos97, Vos98]. Based on the rationality of $T_6$, they developed the CEILIDH public key cryptosystem (Ceilidh, pronounced “Kay-lay”, emphasis on first syllable, is a Scottish Gaelic word that means “gathering”, but the acronym stands for “Compact, Efficient, Improves on LUC, and Improves on Diffie-Hellman”).

However, a problem was found with the case $n = 30 = 2 \cdot 3 \cdot 5$, since it seemed long not possible to represent elements of the interesting subgroup of $F_{q^{30}}$ with just $\phi(30) = 8$ elements of $F_q$ - even though the torus $T_{30}$ is in fact conjectured to be rational, no representation with 8 elements was found. The general case $F_{q^n}$ was considered in [vDiWo04] where the existence of a representation with $\phi(n) + 32$ elements was proved for all $n$ - but for small values like 30 and 210 = $2 \cdot 3 \cdot 5 \cdot 7$ this was definitely not of practical relevance. In the very recent paper [vDGP+05] this “surplus” of elements has been improved from 32 to just 2 elements in the case of $n = 30$ — this represents the first improvement in compression over XTR. Implementation optimisations are also part of that contribution.

None of these systems is really a new primitive and none is based on new allegedly hard problem – on the contrary, they are all based on the primitive underlying the very first public key cryptosystem, the Diffie-Hellman key agreement protocol, one whose security is constantly monitored, and for which there is constant research by some of the finest minds in the cryptographic community. These systems also share the advantages of very fast parameter and key selection (much faster than RSA, orders of magnitude faster than ECC), small key
sizes (much smaller than RSA, comparable with ECC for current security settings), and speed (overall comparable with ECC for current security settings). About these “current security settings”: At the time XTR was introduced, it allowed to use the same bandwidth and performance as ECC cryptosystems despite the fact that ECC has the advantage that the best known attack on them is still birthday paradox based. This meant that, as security requirements increase, ECC would regain a bandwidth – and performance – advantage. The latest development in torus-based cryptography permit another extension of the life of finite field DLP cryptography for the next years. Furthermore, the study of the arithmetic of tori is important because pairing based cryptography requires the computation in tori — although at present no efficient curves with embedding degree 30 are known. Finding them is therefore a very interesting research question.

In §5.2 the original XTR system and its field version are recalled. §5.3 is devoted to Rubin and Silverberg’s compression mechanism CEILIDH [RuSi03], and its development by Granger, Page and Stam [GPS04] into an efficient cryptosystem. In §5.4 we discuss efficient implementation of higher dimensional tori. Finally, in §5.5 we conclude.

5.2 XTR

XTR stands for ‘ECSTR’: Efficient and Compact Subgroup Trace Representation. In its original form it is a cryptographic primitive that makes use of traces to represent and calculate powers of elements of a subgroup of a finite field. As already mentioned in the introduction XTR is not the first method to do so: The LUC cryptosystem used the trace over $\mathbb{F}_p$ to represent elements of the order $p+1$ subgroup of $\mathbb{F}_{p^2}^\times$. Compared to the traditional representation this leads to a factor 2 reduction in the representation size. The variant in [GoHa99] uses the subgroup of order $p^2 + p + 1$ of $\mathbb{F}_{p^3}^\times$ instead, but as a result sizes are reduced by only a factor 1.5. XTR uses the trace over $\mathbb{F}_{p^2}$ to represent elements of the order $p^2 + p + 1$ subgroup of $\mathbb{F}_{p^3}^\times$, thereby achieving a factor 3 size reduction. At the moment it was introduced, computing in XTR was appreciably faster than in the standard representation. The factor 3 size reduction was first achieved – but with much lower performance – in [BPV99].

5.2.1 Classical XTR

In this subsection we describe the original XTR system (with most of the subsequent optimisations) and, to keep the presentation compact, our aim is to show how to implement the XTR-based Diffie-Hellman key agreement protocol and ElGamal encryption. Even though signature schemes are possible (and are efficient), these not included (and also some arithmetic algorithms necessary for their implementation - but not for the implementation of DH and ElGamal schemes - are thus omitted). The aim here is to show the cleverness of the approach and not completeness.

Let $p$ and $q$ be primes with $p \equiv 2 \mod 3$ and $q$ dividing $p^2 - p + 1$ with a small cofactor, and let $g$ be a generator of the order $q$ subgroup of $\mathbb{F}_{p^2}^\times$ contained in $\mathbb{G}_{p^2-p+1}$. The subgroup $\mathbb{G}_{p^2-p+1}$ of $\mathbb{F}_{p^2}^\times$ (and thus $\langle g \rangle$) is interesting for cryptographic purposes because it cannot be embedded in a proper subfield of $\mathbb{F}_{p^2}$.

For $p$ and $q$ of appropriate sizes the discrete logarithm problem in $\langle g \rangle$ is as hard as the
5.2.1.i Description of XTR

Let $p$ and $q$ be primes with $q$ dividing $p^2 - p + 1$. For $g \in \mathbb{F}_{p^2}^\times$, its trace $\text{Tr}_2(g)$ over $\mathbb{F}_{p^2}$ is defined as the sum of the conjugates over $\mathbb{F}_{p^2}$ of $g$:

$$\text{Tr}_2(g) = g + g^{p^2} + g^{p^4} \in \mathbb{F}_{p^2}.$$  

Because the order of $g$ divides $p^6 - 1$ the trace over $\mathbb{F}_{p^2}$ of $g$ equals the trace of the conjugates over $\mathbb{F}_{p^2}$ of $g$:

$$\text{Tr}_2(g) = \text{Tr}_2(g^{p^2}) = \text{Tr}_2(g^{p^4}). \quad (7)$$

In XTR elements of $\mathbb{G}_{p^2-p+1}$ are represented by their trace over $\mathbb{F}_{p^2}$. It follows from (7) that XTR makes no distinction between an element of $\langle g \rangle$ and its conjugates over $\mathbb{F}_{p^2}$. If $g \in \mathbb{G}_{p^2-p+1}$ then its order divides $p^2 - p + 1$, so that

$$\text{Tr}_2(g) = g + g^{p^2} + g^{p^4}$$

since $p^2 \equiv p - 1 \mod (p^2 - p + 1)$ and $p^4 \equiv -p \mod (p^2 - p + 1)$. These relations, together with $\text{Tr}_2(g^{-1}) = \text{Tr}_2(g^p) = \text{Tr}_2(g)^p$ also imply that

$$(X - g)(X - g^{p^2})(X - g^{p^4}) = X^3 - \text{Tr}_2(g)X^2 + \text{Tr}_2(g)^pX - 1.$$  

If $g \in \mathbb{G}_{p^2-p+1}$ then the element $g$, or one of its conjugates, can be retrieved from $c = \text{Tr}_2(g)$ by determining a root of the cubic $X^3 - cX^2 + c^pX - 1$. Lenstra and Verheul also show that any given $c \in \mathbb{F}_{p^2}$ is the trace of some element in $\mathbb{G}_{p^2-p+1}$ if the cubic polynomial is irreducible over $\mathbb{F}_{p^2}$.
Throughout this section, \( c_n \) denotes \( \text{Tr}_2(g^n) \in \mathbb{F}_{p^2} \), for some fixed \( p \) and \( g \) of order \( q \) dividing \( p^2 - p + 1 \) as above. Note that \( c_0 = 3 \) and \( c_1 = c \). Note that

\[
c_{n+m} = c_n c_m - c_p^m c_{n-m} + c_{n-2m},
\]

and efficient implementation of (8) is required to efficiently compute \( c_n \) given \( p \), \( q \), and \( c \).

The next subsubsection explains the cost of the elementary XTR operations in the number of underlying \( \mathbb{F}_p \)-multiplications.

### 5.2.1.ii Implementation aspects

Throughout this subsection, \( c_u \) denotes the trace over \( \mathbb{F}_{p^2} \) of the \( u \)-th power \( g^u \) of \( g \), for some fixed \( p \) and \( g \) of order \( q \) as above. Note that \( c_0 = 3 \). In [LeVe00a, LeVe00b, LeVe01] it is shown how \( p \), \( q \), and \( c_1 \) can be found very quickly (there is no need to find an explicit representation of \( g^2 \)).

The fundamental idea behind XTR is to keep triples of traces throughout the computation, and update all three elements at once. It is an obvious fact that one can obtain any integer \( n \) via a chain starting from 1 where each element is either the double of the previous number or twice the previous number plus one. The number of steps required is \( O(\log n) \). We define triples

\[
S_u = (c_{u+1}, c_u, c_{u-1})
\]

and we start with \( S_1 = (3, c_1, c_2^2 - 2c_1^p) \). If \( 1 = u_1 < u_2 < u_3 < \ldots < u_k = n \) is a chain defined as above we then compute the triples \( S_{u_2}, S_{u_3}, S_{u_4}, S_{u_5}, \ldots, S_{u_k} \) in succession to determine \( c_n \), the central element of the last triple. The key is that the computation of a triple, given the preceding one, \( p \), \( q \) and \( c_1 \), can be done in a very efficient way, as seen in the following facts:

### Facts 5.2.2

We collect here the computational costs of some operations.

1. **Identities involving traces of powers, with \( u, v \in \mathbb{Z} \):**
   - (a) \( c_{-u} = c_{up} = c_u^p \) (so that negations and \( p \)-th powers can be computed for free, cf. 1a).
   - (b) \( c_{u+v} = c_u c_v - c_p^u c_{u-v} + c_{u-2v} \) (which can be computed in four multiplications in \( \mathbb{F}_p \), based on Facts 1a and 1d).
   - (c) If \( c_u = c_1 \), then \( \tilde{c}_v \) denotes the trace of the \( v \)-th power \( g^u \) of \( g \), so that \( c_{uv} = \tilde{c}_v \).

2. **Computing traces of powers, with \( u \in \mathbb{Z} \):**
   - (a) \( c_{2u} = c_u^2 - 2c_u^p \) takes two multiplications in \( \mathbb{F}_p \).
   - (b) \( c_{u+2} = c_1 c_{u+1} - c_1^p c_u + c_{u-1} \) takes four multiplications in \( \mathbb{F}_p \).
   - (c) \( c_{2u-1} = c_{u-1} c_u - c_1^p c_u + c_{u+1} \) takes four multiplications in \( \mathbb{F}_p \).
   - (d) \( c_{2u+1} = c_{u+1} c_u - c_1 c^p u + c_{u-1}^p \) takes four multiplications in \( \mathbb{F}_p \).
The triple $S_{2u-1} = (c_{2(u-1)}, c_{2u-1}, c_{2u})$ can be computed from $S_u$ and $c_1$ by applying Fact 2a twice to compute $c_{2(u-1)}$ and $c_{2u}$ based on $c_{u-1}$ and $c_u$, respectively, and by applying Fact 2c to compute $c_{2u-1}$ based on $S_u = (c_{u-1}, c_u, c_{u+1})$ and $c_1$. This takes eight multiplications in $\mathbb{F}_p$.

The triple $S_{2u+1}$ can be computed in a similar fashion from $S_u$ and $c_1$ in eight multiplications in $\mathbb{F}_p$ (using Fact 2d to compute $c_{2u+1}$). Let $v$ be a non-negative integer, and let $v = \sum_{i=0}^{r-1} v_i 2^i$ be its binary representation, where $v_i \in \{0, 1\}$, $r > 0$, and $v_{r-1} = 1$. It is well known that the $v$th power of an element of, say, a finite field can be computed using the ordinary square and multiply method based on the binary representation of $v$. A very similar iteration can be used to compute $S_{2v+1}$, given $S_1$. As a result there is the following Algorithm

**Algorithm 5.2.3 – XTR single exponentiation** (cf. [LeVe00a, Algorithm 2.3.7]).

Let $S_1, c_1,$ and $v_{r-1}, v_{r-2}, \ldots, v_0 \in \{0, 1\}$ be given, let $y = 1$ and $e = 0$ (so that $2e + 1 = y$; the values $y$ and $e$ are included for expository purposes only). To compute $S_{2v+1}$ with $v = \sum_{i=0}^{r-1} v_i 2^i$ do the following for $i = r - 1, r - 2, \ldots, 0$ in succession:

- If $v_i = 0$, then compute $S_{2y-1}$ based on $S_y$ and $c_1$, replace $S_y$ by $S_{2y-1}$ (and thus $S_{2e+1}$ by $S_{2(2e)+1}$ since if $2e + 1 = y$ then $2(2e) + 1 = 4e + 1 = 2y - 1$), replace $y$ by $2y - 1$, and $e$ by $2e$.

- Else if $v_i = 1$, then compute $S_{2y+1}$ based on $S_y$ and $c_1$, replace $S_y$ by $S_{2y+1}$ (and thus $S_{2e+1}$ by $S_{2(2e+1)+1}$ since if $2e + 1 = y$ then $2(2e + 1) + 1 = 4e + 3 = 2y + 1$), replace $y$ by $2y + 1$, and $e$ by $2e + 1$.

As a result $e = v$. Because both steps maintain the invariant $2e + 1 = y$ the final $S_y$ equals $S_{2v+1}$. Note that $v_{r-1}$, or any other $v_i$, does not have to be non-zero. Both the $v_i = 0$ and $v_i = 1$ cases in the above Algorithm take eight multiplications in $\mathbb{F}_p$. Thus, given an odd positive integer $u < q$ and $S_1$, the triple $S_u = (c_{u-1}, c_u, c_{u+1})$ can be computed in $8 \log_2(u)$ multiplications $\mathbb{F}_p$. The restriction that $u$ is odd and positive is easily removed: if $u$ is even, then first compute $S_{u-1}$ and next apply Fact 2b, and if $u$ is negative, then use Fact 1a.

In [StLe01] the following double exponentiation algorithm is presented.

**Algorithm 5.2.4 – XTR double exponentiation** Let $a, b, c_k, c_{k-\ell}, c_{k-2\ell}$, and $c_\ell$ be given, with $0 < a$ and $b < q$. To compute $c_{bk+al}$ do the following.

1. Let $u = k$, $v = \ell$, $d = b$, $e = a$, $c_u = c_k$, $c_{u-v} = c_{k-\ell}$, $c_{u-2v} = c_{k-2\ell}$, $c_v = c_\ell$, and $f = 0$.

2. As long as $d$ and $e$ are both even, replace $d$ by $d/2$, $e$ by $e/2$ and $f$ by $f + 1$.

3. As long as $d \neq e$ replace $(d, e, u, v, c_u, c_{u+v}, c_{u-2v}, c_v)$ by the 8-tuple given below:

   (a) If $d > e$ then

   i. if $d \leq 4e$, then $(d, e, d-e, u+v, u, c_{u+v}, c_v, c_{v-u}, c_u)$.

   ii. else if $d$ is even, then $(\frac{d-e}{2}, e, 2u, v, c_{2u}, c_{2u-v}, c_{2(u-v)}, c_v)$.

   iii. else if $d \equiv e \text{ mod } 3$ then $(\frac{d-e}{3}, e, 3u, u+v, c_{3u}, c_{2u-v}, c_{u-2v}, c_{u+v})$.

   iv. else if $e$ is even, then $(\frac{d-e}{2}, d, 2u, v, c_{2u}, c_{2u-v}, c_{2(u-v)}, c_u)$.

   v. else $(d$ and $e$ odd), then $(\frac{d-e}{2}, e, 2u, u+v, c_{2u}, c_{u-v}, c_{-2v}, c_{u+v})$.
5.2.1.iii Applications

(b) Else (if $e > d$)
   i. if $e \leq 4d$, then $(d, e - d, u + v, v, c_{u+v}, c_u, c_{u-v}, c_v)$.
   ii. else if $e$ is even, then $(\frac{e}{2}, d, 2v, u, c_{2v}, c_{2v-u}, c_{2(v-u)}, c_u)$.
   iii. else if $e \equiv 0 \mod 3$ then $(\frac{e}{3}, d, 3v, u, c_{3v}, c_{3v-u}, c_{3v-2u}, c_u)$.
   iv. else if $e \equiv d \mod 3$ then $(\frac{e-d}{2}, e, 3v, u + v, c_{3v}, c_{2v-u}, c_{v-2u}, c_{v+u})$.
   v. else if $d$ is even, then $(\frac{d}{2}, e, 2u, v, c_{2u}, c_{2v-u}, c_{u-v}, c_v)$.
   vi. else (d and e odd), then $(\frac{e-d}{2}, d, 2v, u + v, c_{2v}, c_{v-u}, c_{-2u}, c_{u+v})$.

4. Apply Fact 5.2.2(1b) to $c_u, c_{uv}, c_{u-2v}$, and $c_v$, to compute $\tilde{c}_1 = c_{u+v}$.

5. Apply Algorithm “XTR single exponentiation” to $\tilde{S}_1 = (3, \tilde{c}_1, \tilde{c}_1 - 2\tilde{c}_1^2)$, $\tilde{c}_1$, and the binary representation of $d$, resulting in $\tilde{c}_d = c_{d(u+v)}$ (cf. Fact 5.2.2(1c)).

6. Compute $c_{d(u+v)}$ based on $c_{d(u+v)}$ by applying Fact 5.2.2(2a) $f$ times.

Steps 3(a)i and 3(b)i each require a single application of Fact 5.2.2(1b) at the cost of four multiplications in $\mathbb{F}_p$. The remaining steps each require a single application of Fact 5.2.2(1b) and two applications of Fact 5.2.2(2a) at the total cost of 8 multiplications in $\mathbb{F}_p$.

In fact one can use the double exponentiation algorithm to perform even single exponentiations, by suitably splitting the scalar — and in Step 5 one can apply this approach to $\tilde{S}_1$ in order compute $\tilde{c}_d$, resulting in a further speedup.

In [StLe01] experiments are reported that amount to the fact that the expected complexity of the algorithm is around $7.45Q$ multiplications in $\mathbb{F}_p$, where $Q = \lfloor \log_2(q) \rfloor$, with a standard deviation of about $0.2Q$ for a range $40 \leq Q \leq 300$.

The improved single exponentiation based on the double exponentiation algorithm works as follows:

**Algorithm 5.2.5 – Improved XTR single exponentiation.** Let $u$ and $c_1$ be given, with $0 < u < q$. To compute $c_u$, do the following.

1. Let $a = \text{round}(\frac{u - \sqrt{5}}{2}u)$ and $b = u - a$ (where round($x$) is the integer closest to $x$).
   As a result $b/a \approx \varphi = \frac{1 + \sqrt{5}}{2}$, the golden ratio.
2. Let $k = \ell = 1$, $c_k = c_\ell = c_1$, $c_{k-\ell} = c_0 = 3$, $c_{k-2\ell} = c_{-1} = c_1^p$ (cf. Fact 5.2.2(1a)).
3. Apply the XTR double exponentiation algorithm to $a, b, c_k, c_{k-\ell}, c_{k-2\ell}$, $c$, resulting in $c_{b_k+a_\ell} = c_u$, as desired.

5.2.1.iii Applications

As an example, we show XTR cryptographic schemes for confidentiality services. In any cryptosystem that relies on the (subgroup) discrete logarithm problem the ordinary representation of subgroup elements can be replaced by the XTR representation of subgroup elements of a multiplicative group of equivalent security. This section contains a description of some
applications of XTR that provide confidentiality services: Diffie-Hellman key agreement in § 5.2.1.iii.a and ElGamal encryption in § 5.2.1.iii.b. It is possible also to devise signature schemes, which we omit for space reasons. Signature schemes require the computation of the trace of products of elements, and this is in fact tricky since more elements have the same trace and products by them have different traces. A solution (computationally efficient) has been found, and we refer the interested reader to [LeVe00a] for details.

5.2.1.iii.a XTR-DH

Protocol 5.2.6 – XTR-DH key agreement. Let \( p, q, Tr(g) \) be shared XTR public key data. If Alice and Bob want to agree on a secret key \( K \) they do the following.

1. Alice selects a random integer \( a \in [2, q - 3] \), computes

\[
S_a(Tr(g)) = (Tr(g^{a-1}), Tr(g^a), Tr(g^{a+1})) \in \mathbb{F}_{p^2}^3
\]

using the XTR single exponentiation algorithm with \( n = a \) and \( c = Tr(g) \), and sends \( Tr(g^a) \in \mathbb{F}_{p^2} \) to Bob.

2. Bob receives \( Tr(g^a) \) from Alice, selects a random integer \( b \in [2, q - 3] \), computes

\[
S_b(Tr(g)) = (Tr(g^{b-1}), Tr(g^b), Tr(g^{b+1})) \in \mathbb{F}_{p^2}^3
\]

using the XTR single exponentiation algorithm with \( n = b \) and \( c = Tr(g) \), and sends \( Tr(g^b) \in \mathbb{F}_{p^2} \) to Alice.

3. Alice receives \( Tr(g^b) \) from Bob, computes

\[
S_a(Tr(g)^b) = (Tr(g^{(a-1)b}), Tr(g^{ab}), Tr(g^{(a+1)b})) \in \mathbb{F}_{p^2}^3
\]

using the XTR single exponentiation algorithm with \( n = a \) and \( c = Tr(g^b) \), determines the secret key \( K \) based on \( Tr(g^{ab}) \in \mathbb{F}_{p^2} \).

4. Bob computes

\[
S_b(Tr(g)^a) = (Tr(g^{a(b-1)}), Tr(g^{ab}), Tr(g^{a(b+1)})) \in \mathbb{F}_{p^2}^3
\]

using the XTR single exponentiation algorithm with \( n = b \) and \( c = Tr(g^a) \), determines \( K \) based on \( Tr(g^{ab}) \in \mathbb{F}_{p^2} \).

The communication and computational overhead of this XTR-DH key agreement are both about one third of traditional implementations of the Diffie-Hellman protocol that are based on subgroups of multiplicative groups of finite fields, and that achieve the same level of security.
5.2. XTR

5.2.1.iii.b XTR-ElGamal encryption

Protocol 5.2.7 – XTR-ElGamal encryption. Let \( p, q, Tr(g) \) be XTR public key data, either owned (and made public) by Alice or shared by all parties. Furthermore, let \( Tr(g^k) \) be a value computed and made public by Alice, for some integer \( k \) selected (and kept secret) by Alice. Given \((p,q,Tr(g),Tr(g^k))\), Bob can encrypt a message \( M \) intended for Alice as follows.

1. Bob selects at random \( b \in [2, q-3] \) and applies the XTR single exponentiation algorithm to \( n = b \) and \( c = Tr(g) \) to compute

\[
S_b(Tr(g)) = (Tr(g^{b-1}), Tr(g^b), Tr(g^{b+1})) \in \mathbb{F}_{p^2}^3.
\]

2. Bob applies the XTR single exponentiation algorithm to \( n = b \) and \( c = Tr(g^k) \) to compute

\[
S_b(Tr(g^k)) = (Tr(g^{(b-1)k}), Tr(g^{bk}), Tr(g^{(b+1)k})) \in \mathbb{F}_{p^2}^3.
\]

3. Bob determines a symmetric encryption key \( K \) based on \( Tr(g^k) \in \mathbb{F}_{p^2} \).

4. Bob uses an agreed upon symmetric encryption method with key \( K \) to encrypt \( M \), resulting in the encryption \( E \).

5. Bob sends \((Tr(g^b), E)\) to Alice.

Protocol 5.2.8 – XTR-ElGammal decryption. Using her knowledge of \( k \), Alice decrypts the message \((Tr(g^b), E)\) encrypted using XTR-ElGamal encryption 5.2.7 as follows.

1. Alice applies the XTR single exponentiation algorithm to \( n = k \) and \( c = Tr(g^b) \) to compute

\[
S_k(Tr(g^b)) = (Tr(g^{(k-1)b}), Tr(g^{kb}), Tr(g^{(k+1)b})) \in \mathbb{F}_{p^2}^3.
\]

2. Alice determines symmetric encryption key \( K \) based on \( Tr(g^{kb}) \in \mathbb{F}_{p^2} \).

3. Alice uses the agreed upon symmetric encryption method with key \( K \) to decrypt \( E \), resulting in the encryption \( M \).

The communication and computational overhead of XTR-based ElGamal encryption and decryption are both about one third of traditional implementations of the ElGamal encryption and decryption protocols that are based on subgroups of multiplicative groups of finite fields, and that achieve the same level of security.

5.2.2 Subgroup XTR

In [StLe02] Stam and Lenstra show how to compute efficiently in extensions of degrees 2 and 6, so that working directly with the field elements allows performance better to that of the original XTR approach based on traces - and slightly slower than those presented in the previous subsection. The same approach is used as Representation \( F_1 \) for CEILIDH (see
Subsubsection 5.3.4.i), but since CEILIDH goes beyond the “Subgroup XTR” approach, we keep the treatment here contained.

Let $G_t$ denote a subgroup of order $t$ of $\mathbb{F}_p^\times$. The cyclotomic subgroup of $\mathbb{F}_p^\times$ in which one works is then denoted by $G_{p^2-p+1}$ and the LUC cryptosystem is based on $G_{p+1} \subseteq \mathbb{F}_p^\times$. The main results in [StLe02] are that if $p \equiv 2 \mod 9$ then inversion of elements of the XTR subgroup is very cheap, and that squaring is much faster than in the whole field $\mathbb{F}_p$.

Some protocols, for example, are easier.

**Sixth Degree Extensions, the case $p \equiv 2 \mod 9$** In this subsubsection fast exponentiation routines for the group $G_{p^2-p+1} \subset \mathbb{F}_p^\times$, with $p \equiv 2 \mod 9$ are described. Let $f$ be a sixth degree irreducible polynomial over some ground field, with root $z$. We consider the extension induced by $z$ and represented by a polynomial basis $(z, z^2, \ldots, z^6)$ consisting of six consecutive powers of $z$. We recall results about the efficiency of field operations with the representation of the field extension with this bases and its impact on exponentiation.

**Field Arithmetic** Let $p$ be prime with $p \equiv 2 \mod 9$. Then $p$ generates $\mathbb{Z}_p^\ast$ and $\Phi_9(x) = x^6 + x^3 + 1$ is irreducible in $\mathbb{F}_p$. Let $z$ denote a root of $\Phi_9(x)$, then $\Gamma = (z, z^2, \ldots, z^6)$ is a basis for $\mathbb{F}_p$ over $\mathbb{F}_p$.

Let $a = \sum_{i=0}^{5} a_i z^{i+1} \in \mathbb{F}_p$. From $z^n = z^{n \mod 9}$ and thus $z^p = z^2$ it follows with $\Phi_9(z) = 0$ that $a^p = a_4 z + (a_0 - a_3) z^2 + a_5 z^3 + a_1 z^4 - a_3 z^5 + a_2 z^6$. Thus, $p$-th powering costs $A_1$. In a similar way it follows that $p^3$-th powering costs $2A_1$. For multiplication in $\mathbb{F}_p$ Karatsuba’s trick is used, allowing the multiplication to be done with 18 multiplications instead of 36. In fact, one can even reduce the number of modular reductions to just 6. Squaring follows by replacing the 18 multiplications by squarings, but it can be done substantially faster by observing that $G^2 = (G_0 z + G_1 z^2)^2 = (G_0 - G_1)(G_0 + G_1) z^2 + (2G_0 - G_1) G_1 z^3$, with $G_0, G_1 \in \mathbb{F}_p[z]$ of degree two. Computing this requires then a total of 12 multiplications (here, the number of modular reductions needed is just 6, too, as in the multiplication case). For squaring and multiplication several additions in $\mathbb{F}_p$ are necessary, too.

**Lemma 5.2.9** Let $a, b \in \mathbb{F}_p$ with $p \equiv 2 \mod 9$.

1. Computing $a^p$ or $a^{p^5}$ costs one field addition in $\mathbb{F}_p$.
2. Computing $a^{p^2}$, $a^{p^3}$, or $a^{p^4}$ costs two field additions in $\mathbb{F}_p$.

**Subgroup Arithmetic** Let $a = \sum_{i=0}^{5} a_i z^{i+1} \in \mathbb{F}_p$. Membership of one of the three proper subfields of $\mathbb{F}_p$ is characterized by one of the equations $a^{p^i} = a$ for $i = 1, 2, 3$. Specifically, $a \in \mathbb{F}_p$ if and only if $a^p = a$ which is equivalent to the system of linear equations $(a_0, a_1, a_2, a_3, a_4, a_5) = (a_4, a_0 - a_3, a_5, a_1, -a_3, a_2)$. The solution $a_0 = a_1 = a_3 = a_4 = 0$ and
\[ a_2 = a_5 \text{ is not surprising since } 1 + z^3 + z^6 = 0, \text{ so an element } c \in \mathbb{F}_p \text{ takes the form } -cz^3 - cz^6. \]

Similarly, \( a \in \mathbb{F}_p^2 \) if and only if \( a^3 = a \), which is equivalent to \( a = a_2 z^3 + a_5 z^6 \), and \( a \in \mathbb{F}_p^3 \) if and only if \( a^3 = a \) or \( a = (a_3 - a_4)z + (-a_3 + a_4)z^2 + a_5 z^3 + a_3 z^4 + a_4 z^5 + a_5 z^6. \)

Let us turn our attention to the subgroup \( G_{p^2-p+1} \) of \( \mathbb{F}_p^* \). The \( G_{p^2-p+1} \)-membership condition \( a^{p^2-p+1} = 1 \) is equivalent to \( a^p = a \), which can be verified at a cost of, essentially, a single \( \mathbb{F}_p \)-multiplication. From \( a^p = a^{-1} \) it follows that inversion in \( G_{p^2-p+1} \) costs two additions in \( \mathbb{F}_p \).

Computing \( a^p a - a^p = \sum_{i=0}^{5} v_i z^{i+1} \) symbolically produces

\[
\begin{align*}
    v_0 &= a_1^2 - a_0 a_2 - a_4 - a_4^2 + a_3 a_5; \\
    v_1 &= -a_0 + a_1 a_2 + a_3 - 2a_0 a_3 + a_3^2 - a_2 a_4 - a_1 a_5; \\
    v_2 &= -a_0 a_1 + a_3 a_4 - a_5 - 2a_2 a_5 + a_2^2; \\
    v_3 &= -a_1 - a_2 a_3 + 2a_1 a_4 - a_4^2 - a_0 a_5 + a_3 a_5; \\
    v_4 &= a_0^2 + a_1 a_2 + a_3 - 2a_0 a_3 - a_4 a_5; \\
    v_5 &= -4a + a_2^2 - a_1 a_3 - a_0 a_4 + a_3 a_4 - 2a_2 a_5.
\end{align*}
\]

If \( a \in G_{p^2-p+1} \), then \( v_i = 0 \) for \( 0 \leq i < 6 \) and the resulting six relations can be used to significantly reduce the cost of squaring in \( G_{p^2-p+1} \). Let \( V = (v_0, v_1, \ldots, v_5) \) be the vector consisting of the \( v_i \)'s. Then for any \( 6 \times 6 \)-matrix \( M \), we have that \( a^2 + \Gamma \cdot (M \cdot V^T) = a^2 \) if \( a \in G_{p^2-p+1} \), because in that case \( V \) is the all-zero vector. Carrying out this computation symbolically, involving the expressions for the \( v_i \)'s for a particular choice of \( M \) yields a method for computing squares in \( G_{p^2-p+1} \) using only 6 multiplications, which is optimal.

Making use of the structure of the subgroup and writing down explicitly all relations involved in a squaring, one can conclude that a squaring in \( G_{p^2-p+1} \) can be done with six multiplications (and modular reductions) with some additions. The following Lemma collects the results about subgroup membership and some computations which are easy in subgroups.

**Lemma 5.2.10** Let \( G_{p^2-p+1} \) be the order \( p^2 - p + 1 \) subgroup of \( \mathbb{F}_p^* \) with \( p \equiv 2 \mod 9 \) and let \( a = a_0 z + a_1 z^2 + \cdots + a_5 z^6 \in \mathbb{F}_p^6 \) with \( \Phi_9(z) = 0 \).

1. The element \( a \) is in \( \mathbb{F}_p \) if and only if \( a = a_2 z^3 + a_2 z^6 \).
2. The element \( a \) is in \( \mathbb{F}_p^2 \) if and only if \( a = a_2 z^3 + a_5 z^6 \).
3. The element \( a \) is in \( \mathbb{F}_p^3 \) if and only if \( a = (a_3 - a_4)z + (-a_3 + a_4)z^2 + a_5 z^3 + a_3 z^4 + a_4 z^5 + a_5 z^6. \)
4. The element \( a \) is in \( G_{p^2-p+1} \) if and only if in relations (9) \( v_i = 0 \) for \( 0 \leq i < 6 \). This can be checked at a cost of essentially 18 multiplications in \( \mathbb{F}_p \), where in fact only 6 modular reductions need to be performed.
5. Computing \( a^{-1} \) for \( a \in G_{p^2-p+1} \) costs two additions.
6. Computing \( a^2 \) for \( a \in G_{p^2-p+1} \) costs essentially 6 multiplications in \( \mathbb{F}_p \).
Subgroup Exponentiation  Exponentiation can be performed using all usual algorithms based on addition chains.

For a single exponentiation we have to compute $a^m$, where $m$ has roughly the same bitlength $k$ as $q$. For the case $q|(p^2 - p + 1)$ the Frobenius endomorphism will save us some work, since $m$ can quickly be written as $m \equiv m_1 + m_2p \mod q$ with $m_1$ and $m_2$ of bitlength $k=2$ according to the following Lemma:

**Lemma 5.2.11** Let $q|(p^2 - p + 1)$ and let $n \in \mathbb{Z}$. Then there exist $n_1, n_2 \in \mathbb{Z}$ such that $n_1 + n_2p \equiv n \mod q$ and $|n_1|, |n_2| < 2\sqrt{q}$.

A proof can be found in [Sta03]. The case $q|(p^2 + 1)$ can be dealt with similarly. A more general and thorough treatment is given by Sica et al. [SMQ03]

Hence $a^m$ can be rewritten as $a^{m_1}(a^p)^{m_2}$. This double exponentiation can be computed using the Joint Sparse Form, resulting in a cost of about six $\mathbb{F}_p$-multiplications per exponent bit.

A double exponentiation $a^m b^n$, with $\log m \approx \log n$ and $m$ as above, can be rewritten as $a^{m_1}(a^p)^{m_2} b^{n_1}(b^p)^{n_2}$ with $\approx k/2$-bit $m_1$, $m_2$, $n_1$, and $n_2$. This quadruple exponentiation can be computed using the JSF resulting in a total of about 9 multiplications in $\mathbb{F}_p$ per exponent bit.

Combination of these observations leads to the following theorem.

**Theorem 5.2.12** Let $p$ and $q$ be primes with $q|(p^2 - p + 1)$, $p \equiv 2 \mod 9$, and $[\log_2 q] = k$. Let $a, b$ be in the order $q$ subgroup $G_q$ of $\mathbb{F}_p^*$ and $m, n \in \mathbb{Z}_q$. Under reasonable assumptions:

1. computing $a^m$ costs on average $12 + 6k$ multiplications in $\mathbb{F}_p$, and
2. computing $a^m b^n$ costs on average $24 + 9k$ multiplications in $\mathbb{F}_p$, and
3. computing $a^m$ costs on average $2.93(k - 10)$ multiplications in $\mathbb{F}_p$ based on 31 precomputed values for Pippengers algorithm (with $w = k/10$ and $t = 5$).

Application of the above computational techniques to the implementation of all the cryptographic protocols that can normally based on the discrete logarithm primitive is trivial and therefore not explained here.

5.3 CEILIDH

CEILIDH is essentially a compression/decompression mechanism for representing elements of the cyclotomic group $G_{p^2 - p + 1} \subset \mathbb{F}_p^*$. It is based on the observation that the field elements of $\mathbb{F}_p^*$ lying in this subgroup can be viewed as the $\mathbb{F}_p$-rational points on the algebraic torus $T_6$ [RuSi03]. Using this perspective, Rubin and Silverberg showed that by exploiting birational maps from $T_6$ to two-dimensional affine space, one can efficiently represent its elements with just two elements of $\mathbb{F}_p$, matching the compression afforded by XTR.
5.3. CEILIDH

5.3.1 The torus $T_n(\mathbb{F}_p)$

Throughout, let $\mathbb{F}_p$ be the prime field consisting of $p$ elements. Let $\phi$ be the Euler $\phi$-function, and let $\Phi_n$ be the $n$-th cyclotomic polynomial. We write $G_{p,n}$ for the subgroup of $\mathbb{F}_p^*$ of order $\Phi_n(p)$, and let $A^n(\mathbb{F}_p)$ denote $n$-dimensional affine space over $\mathbb{F}_p$, i.e. the variety whose points lie in $\mathbb{F}_p^n$.

More formally, one can define algebraic tori as follows:

**Definition 5.3.1** Let $k = \mathbb{F}_p$ and $L = \mathbb{F}_p^n$. Define the torus $T_n$ to be the intersection of the kernels of the norm maps $N_{L/F}$, for all subfields $k \subset F \subseteq L$:

$$T_n(k) := \bigcap_{k \subseteq F \subseteq L} \text{Ker}[N_{L/F}].$$

The dimension of $T_n$ is $\phi(n)$. Since $T_n(\mathbb{F}_p)$ is a subgroup of $\mathbb{F}_p^*$, the group operation is just ordinary multiplication in the larger field. The following lemma provides some essential properties of $T_n$ [RuSi03].

**Lemma 5.3.2**

1. $T_n(\mathbb{F}_p) \cong G_{p,n}$.

2. $\#T_n(\mathbb{F}_p) = \Phi_n(p)$.

3. If $h \in T_n(\mathbb{F}_p)$ is an element of prime order not dividing $n$, then $h$ does not lie in a proper subfield of $\mathbb{F}_p^n/\mathbb{F}_p$.

When $T_n$ is rational, elements can be represented by just $\phi(n)$ elements of $\mathbb{F}_p$, and hence a compression factor of $n/\phi(n)$ is achieved over the usual representation. $T_n$ is known to be rational when $n$ is either a prime power, or is a product of two prime powers, and it is conjectured to be rational for all $n$. For current key size recommendations, this would have interesting cryptographic applications for $n = 30$ and $n = 210$, which would give compression ratios of $3 \frac{3}{4}$ and $4 \frac{7}{8}$ respectively, assuming good parameters can be found efficiently.

5.3.2 Rationality of tori over $\mathbb{F}_q$

In order to compress elements of the variety $T_n$, we make use of rationality, for particular values of $n$. The rationality of $T_n$ means there exists a birational map from $T_n$ to $\phi(n)$-dimensional affine space $A^{\phi(n)}$. This allows one to represent nearly all elements of $T_n(\mathbb{F}_q)$ with just $\phi(n)$ elements of $\mathbb{F}_q$, providing an effective compression factor of $n/\phi(n)$ over the embedding of $T_n(\mathbb{F}_q)$ into $\mathbb{F}_q^n$. Since $T_n$ has dimension $\phi(n)$, this compression factor is optimal. As already recalled, $T_n$ is conjectured to be rational for all $n$.

Formally, rationality can be defined as follows.

**Definition 5.3.3** Let $T_n$ be an algebraic torus over $\mathbb{F}_q$ of dimension $d = \phi(n)$, then $T_n$ is said to be rational if there is a birational map $\rho : T_n \rightarrow A^{\phi(n)}$ defined over $\mathbb{F}_q$. 
This means that there are subsets $W \subset T_n$ and $U \subset \mathcal{A}^{\phi(n)}$, and rational functions $\rho_1, \ldots, \rho_{\phi(n)} \in \mathbb{F}_p(x_1, \ldots, x_n)$ and $\psi_1, \ldots, \psi_n \in \mathbb{F}_q(y_1, \ldots, y_{\phi(n)})$ such that $\rho = (\rho_1, \ldots, \rho_{\phi(n)}): W \to U$ and $\psi = (\psi_1, \ldots, \psi_n): U \to W$ are inverse isomorphisms. Furthermore, the differences $T \setminus W$ and $\mathcal{A}^{\phi(n)} \setminus U$ should be algebraic varieties of dimension $\leq (d - 1)$, which implies that $W$ (resp. $U$) is ‘almost the whole’ of $T$ (resp. $\mathcal{A}^{\phi(n)}$).

5.3.3 CEILIDH construction

Much of this section is a slightly abridged version of the original description of CEILIDH [RuSi03].

Fix $x \in \mathbb{F}_p^2 \setminus \mathbb{F}_p$, so $\mathbb{F}_p^2 = \mathbb{F}_p(x)$, and let $\{\alpha_1, \alpha_2, \alpha_3\}$ be a basis for $\mathbb{F}_p^3$ over $\mathbb{F}_p$. Then $\{\alpha_1, \alpha_2, \alpha_3, x\alpha_1, x\alpha_2, x\alpha_3\}$ is a basis for $\mathbb{F}_p^6$ over $\mathbb{F}_p$. Let $\sigma \in \text{Gal}(\mathbb{F}_p^6/\mathbb{F}_p)$ be the element of order two. Define $\psi_0 : \mathcal{A}^3(\mathbb{F}_p) \to \mathbb{F}_p^6$ by

$$\psi_0(u_1, u_2, u_3) = \frac{\gamma + x}{\gamma + \sigma(x)},$$

where $\gamma = u_1 \alpha_1 + u_2 \alpha_2 + u_3 \alpha_3$. Then $N_{\mathbb{F}_p^6/\mathbb{F}_p}(\psi_0(u)) = 1$ for every $u = (u_1, u_2, u_3)$. Let $U = \{u \in \mathcal{A}^3 : N_{\mathbb{F}_p^6/\mathbb{F}_p}(\psi_0(u)) = 1\}$. By (10), $\psi_0(u) \in T_6(\mathbb{F}_p)$ if and only if $u \in U$, so restricting $\psi_0$ to $U$ gives a morphism $\psi_0 : U \to T_6$. It follows from Hilbert’s Theorem 90 that every element of $T_6(\mathbb{F}_p) \setminus \{1\}$ is in the image of $\psi_0$, and so $\psi_0$ defines an isomorphism

$$\psi_0 : U \cong T_6 \setminus \{1\}.$$

The equation defining $U$ is a quadratic hypersurface in $u_1, u_2, u_3$. Fix a point $a = (a_1, a_2, a_3) \in U(\mathbb{F}_p)$. By adjusting the basis $\{\alpha_1, \alpha_2, \alpha_3\}$ of $\mathbb{F}_p^6$ if necessary, one can assume without loss of generality that the tangent plane at $a$ to the surface $U$ is just the plane $u_1 = a_1$. If $(v_1, v_2) \in \mathbb{F}_p \times \mathbb{F}_p$, then the intersection of $U$ with the line $a + t(1, v_1, v_2)$ consists of two points, namely $a$ and a point of the form $a + \frac{1}{f(v_1, v_2)}(1, v_1, v_2)$ where $f(v_1, v_2) \in \mathbb{F}_p[v_1, v_2]$ is an explicit polynomial independent of $p$. The map that takes $(v_1, v_2)$ to the latter point is a birational isomorphism

$$g : \mathcal{A}^2 \setminus V(f) \cong U \setminus \{a\},$$

where $V(f)$ denotes the subvariety of $\mathcal{A}^2$ defined by $f(v_1, v_2) = 0$. Thus $\psi_0 \circ g$ defines an isomorphism

$$\psi : \mathcal{A}^2 \setminus V(f) \cong T_6 \setminus \{1, \psi_0(a)\}.$$

For the inverse isomorphism, suppose that $\beta = \beta_1 + \beta_2 x \in T_6(\mathbb{F}_p) \setminus \{1, \psi_0(a)\}$ with $\beta_1, \beta_2 \in \mathbb{F}_p^3$. One can check that $\beta_2 \neq 0$, and if $\gamma = (1 + \beta_1)/\beta_2$, then $\gamma/\sigma(\gamma) = \beta$. Write $(1 + \beta_1)/\beta_2 = u_1 \alpha_1 + u_2 \alpha_2 + u_3 \alpha_3$ with $u_i \in \mathbb{F}_p$, and define

$$\rho(\beta) = \left(\frac{u_2 - a_2}{u_1 - a_1}, \frac{u_3 - a_3}{u_1 - a_1}, \frac{u_1 - a_1}{u_1 - a_1}\right).$$

Then $\rho : T_6(\mathbb{F}_p) \setminus \{1, \psi_0(a)\} \cong \mathcal{A}^2(\mathbb{F}_p) \setminus V(f)$ is the inverse isomorphism of $\psi$, and hence we have an efficient compression and decompression mechanism for all (bar two) elements of $T_6(\mathbb{F}_p)$. 


5.3. CEILIDH

5.3.4 Implementing CEILIDH

The above construction is quite general. Restricting to a particular field representation gives
exact formulae: those given below are based on Example 11 of [RuSi03]. In practice however
one would like to use the methods described in Section 5.2 for fast field arithmetic. The
following is based on the work of Granger, Page and Stam [GPS04], and provides the necessary
maps and arithmetic for a fully optimised implementation.

The four representations \( F_1, F_2, F_3, A^2 \) and the isomorphisms between them which consti-
tute GPS implementation of CEILIDH, may be depicted as follows:

\[
\begin{align*}
F_1 & \xrightarrow{\sigma} F_2 \xrightarrow{\tau} F_3 \xrightarrow{\rho} A^2(F_p). \\
& \text{(11)}
\end{align*}
\]

For \( F_1 \) we give a brief description of the arithmetic, and for \( F_2 \) and \( F_3 \) we give full details of
all operations. In Lemma 3 we provide a simple cost analysis where \( M, A, \) and \( I \) represent
the cost of an \( F_p \) multiplication, addition, and inversion respectively. In \( F_p \), we assume that a
subtraction amounts to the same as an addition, and also that squaring costs the same as a
multiplication, since the former operation is seldom used.

For \( (n,p) = 1 \), let \( \zeta_n \) denote a primitive \( n \)-th root of unity mod \( p \), and as in XTR let \( p \equiv 2 \mod 9 \) throughout \( (p \equiv 5 \mod 9 \) is equally valid).

5.3.4.i The Representation \( F_1 \)

A full derivation of the results of this section can be found in [StLe02], some details have
been given in §5.2.2. Let \( z = \zeta_9 \), so that \( F_{p^6} = F_p(z) \), and let our basis for \( F_{p^6} \) be
\( \{ z, z^2, z^3, z^4, z^5, z^6 \} \). For the costs of the operations such as multiplication, squaring, etc... in
\( G_{p,6} \), see §5.2.2.

5.3.4.ii The Representation \( F_2 \)

Let \( x = \zeta_3 \) and \( y = \zeta_9 + \zeta_9^{-1} \). Then \( F_{p^3} = F_p(y) \), and \( F_{p^3} = F_{p^3}(x) \). The bases we use are
\( \{ 1, y, y^2 - 2 \} \) for \( F_{p^3} \), and \( \{ 1, x \} \) for the degree two extension. We now describe the basic
arithmetic in each of these extensions.

\[ \text{F}_{p^3} \text{ Frobenius :} \]

For our basis, since \( p \equiv 2 \mod 9 \), the Frobenius map gives \( y^p = y^2 - 2 \), and \( (y^2 - 2)^p = -y - (y^2 - 2) \). Hence for \( a = a_0 + a_1 y + a_2 (y^2 - 2) \), \( a^p = a_0 - a_2 y + (a_1 - a_2)(y^2 - 2) \).

\[ \text{F}_{p^3} \text{ Multiplication :} \]

Let \( a = a_0 + a_1 y + a_2 (y^2 - 2) \), \( b = b_0 + b_1 y + b_2 (y^2 - 2) \). Then \( ab = (a_0 b_0 + 2a_1 b_1 + 2a_2 b_2 - a_1 b_1 - a_2 b_1) + (a_0 b_1 + a_1 b_0 + a_1 b_2 + a_2 b_2) y + (a_0 b_2 + a_2 b_0 + a_1 b_1 - a_2 b_2)(y^2 - 2) \). Precompute
\( t_{00} = a_0 b_0 \), \( t_{11} = a_1 b_1 \), \( t_{22} = a_2 b_2 \), and \( t_{01} = (a_0 + a_1)(b_0 + b_1) \), \( t_{12} = (a_1 - a_2)(b_2 - b_1) \),
\( t_{20} = (a_2 - a_0)(b_0 - b_2) \). Then \( ab = (t_{00} + t_{11} + t_{22} - t_{12}) + (t_{01} + t_{12} - t_{00}) y + (t_{20} + t_{00} + t_{11})(y^2 - 2) \).

\[ \text{F}_{p^3} \text{ Inversion :} \]

Usually, to invert an element in an extension of a prime field one must either use a basic GCD
algorithm, or one of many suggestions based on exponentiating to a power one less than the
group order [ItTs88]. However, since the extension degree is small, we can perform inversion directly: one uses the multiplication formula and sets the result to the identity, i.e., one solves

\[
\begin{pmatrix}
2a_1 - a_2 & 2a_2 - a_1 \\
a_1 + a_2 & a_1 - a_2
\end{pmatrix}
\begin{pmatrix}
b_0 \\
b_1
\end{pmatrix}
= \begin{pmatrix} 1 \\ 0 \end{pmatrix}
\]

for \( b \). This gives

\[
\begin{pmatrix}
b_0 \\
b_1
\end{pmatrix} = \frac{1}{\Delta}
\begin{pmatrix}
-a_0^2 + a_1^2 + a_2^2 - a_1 a_2 \\
-a_0^2 + a_0 a_1 - 2a_1 a_2 \\
-a_0^2 + a_0 a_1 - 2a_1 a_2
\end{pmatrix},
\]

where \( \Delta = -a_0^2 + a_1^2 + a_2^2 + 3a_0 a_2^2 + 3a_0 a_1^2 + 3a_1 a_2^2 - 6a_1 a_2^2 - 3a_0 a_1 a_2 \). Computing \( t_{00} = a_0^2 \), \( t_{11} = a_1^2 \), \( t_{22} = a_2^2 \), \( t_{01} = a_0 a_1 \), \( t_{12} = a_1 a_2 \), \( t_{20} = a_2 a_0 \), \( t_{02} = a_0 + a_1 + a_2 \) and \( t = t_{12}(a_0 + a_1) \), then \( \Delta = t_{012}^2 - t_{00} (3 t_{012} - a_0) - 9 t \). To finish, we perform one \( \mathbb{F}_p \) inversion and obtain \( a^{-1} \) equals

\[
\Delta^{-1}((t_{11} + t_{22} - t_{00} - t_{12}) + (t_{01} - 2t_{12} + t_{22}) y + (t_{20} + t_{22} - t_{11}))(y^2 - 2)).
\]

**F\(_p^6\) Frobenius**: Let \( c = c_0 + c_1 x \). Then \( c^p = (c_0 + c_1 x)^p = (c_0^p - c_1^p) - c_1 x \).

**F\(_p^6\) Multiplication**: For \( c = c_0 + c_1 x \), \( d = d_0 + d_1 x \), we have \( cd = (c_0 d_0 - c_1 d_1) + (c_0 d_1 + c_1 d_0 - c_1 d_1) x \). If we compute \( t_{00} = c_0 d_0 \), \( t_{11} = c_1 d_1 \), and \( t_{01} = (c_0 + c_1)(d_0 + d_1) \), then \( cd = (t_{00} - t_{11}) + (t_{01} - t_{00} - 2t_{11}) x \).

**F\(_p^6\) Squaring**: \( c^2 = (c_0 + c_1 x)^2 = (c_0^2 - c_1^2) + c_1 (2 c_0 - c_1) x \). We compute \( t_{01} = (c_0 + c_1)(c_0 - c_1) \), giving \( c^2 = t_{01} + c_1 (2 c_0 - c_1) x \).

**F\(_p^6\) Inversion**: Performing again a direct inversion as in \( \mathbb{F}_p^3 \), we find

\[
d_0 + d_1 x = (c_0 + c_1 x)^{-1} = \frac{1}{c_0^2 - c_0 c_1 + c_1^2} \begin{pmatrix} c_1 - c_0 \\ -c_1 \end{pmatrix},
\]

so that we still only require one \( \mathbb{F}_p \) inversion. Precomputing \( t_0 = (c_1 - c_0) \), \( t_{01} = c_0 c_1 \), and \( \Delta = t_0^2 + t_{01} \), the coefficients of the inverse \( d \) are given by \( d_0 = \Delta^{-1} t_0 \), \( d_1 = -\Delta^{-1} c_1 \).

If we are working in \( G_{p^6} \), then as in \( F_1 \) inversions are essentially free thanks to the cheap Frobenius endomorphism.

\( \sigma : F_1 \to F_2 \): In addition to the individual arithmetic of \( F_1 \) and \( F_2 \) we need to specify an efficiently computable isomorphism between them. Writing \( x \) and \( y \) in terms of \( z \) we find that \( x = z^3 \), and
In the current notation, the group operation in CEILIDH as originally described is performed

5.3.4.iii The Representation

For a cubic extension, all bases have determinant divisible by three. We can reduce this to just
one division by three however, (or a multiplication by its precomputed inverse) by writing

\[
\sigma = \begin{cases}
0 & 1 & -1 & 0 & 0 & 1 \\
0 & -1 & 1 & 0 & 1 & 0 \\
-1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & -1 & 0 & 1 & 0 \\
0 & -1 & 0 & 0 & 0 & 1 \\
-1 & 0 & 0 & 0 & 0 & 0
\end{cases}.
\]

Since \(\sigma^{-1}\) has determinant three, a naive evaluation of \(\sigma\) necessitates four divisions by three.
It is not possible to eliminate all of these since for our \(F_1\), writing \(\mathbb{F}_{p^6}\) as a quadratic extension
of a cubic extension, all bases have determinant divisible by three. We can reduce this to just
one division by three however, (or a multiplication by its precomputed inverse) by writing

\[
\sigma^{-1} = \begin{cases}
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & -1 & 1 \\
0 & 0 & 1 & 0 & 1 & -1 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0
\end{cases},
\]

5.3.4.iii The Representation \(F_3\)

In the current notation, the group operation in CEILIDH as originally described is performed
in \(F_2\) [RuSi03], and the inverse birational maps \(\psi : \mathcal{A}^2(\mathbb{F}_p) \setminus V(f) \simarrow T_6(\mathbb{F}_p) \setminus \{1, x^2\}\), and
\(\rho : T_6(\mathbb{F}_p) \setminus \{1, x^2\} \simarrow \mathcal{A}^2(\mathbb{F}_p) \setminus V(f)\), are given by

\[
\psi(a_1, a_2) = \frac{1 + a_1 y + a_2 (y^2 - 2) + (1 - a_1^2 - a_2^2 + a_1 a_2) x}{1 + a_1 y + a_2 (y^2 - 2) + (1 - a_1^2 - a_2^2 + a_1 a_2) x^2},
\]

where \(V(f)\) is the set of zeros of \(f(a_1, a_2) = 1 - a_1^2 - a_2^2 + a_1 a_2 = 0\) in \(\mathcal{A}^2(\mathbb{F}_p)\); and for
\(\beta = \beta_1 + \beta_2 x\), with \(\beta_1, \beta_2 \in \mathbb{F}_p^3\), let \((1 + \beta_1)/\beta_2 = u_1 + u_2 y + u_3 (y^2 - 2)\). Then \(\rho(\beta) =
(u_2/u_1, u_3/u_1)\).

The following lemma is needed for the description of \(F_3\), and is implicit in the CEILIDH
construction already given.

**Lemma 5.3.4** There is an isomorphism

\[
\tau : \text{Ker}[N_{\mathbb{F}_p^6/\mathbb{F}_p^3}] \simarrow \left\{ \frac{b + x}{b + x^2} : b \in \mathbb{F}_p^3 \right\} \cup \{1\},
\]

where for \(a = a_0 + a_1 x \in \text{Ker}[N_{\mathbb{F}_p^6/\mathbb{F}_p^3}] \setminus \{1\},

\[
\tau(a) = \left( \frac{b + x}{b + x^2} \right),
\]

with \(b = (1 + a_0)/a_1\) if \(a_1 \neq 0\) and \(b = a_0/(a_0 - 1)\) otherwise, and

\[
\tau^{-1}(b) = \frac{b^2 - 1}{b^2 - b + 1} + \frac{2b - 1}{b^2 - b + 1} x.
\]
Therefore using the inclusion of $T_6(\mathbb{F}_p)$ in $T_2(\mathbb{F}_{p^3})$, we can introduce the following representation:

**Definition 5.3.5** $F_3$ is the set of elements

$$\left\{ \frac{a_0 + a_1 x}{a_0 + a_1 x^2}, a_i \in \mathbb{F}_{p^3} \right\}.$$

When the coefficient $a_1$ of this representation equals 1, we say the element is reduced.

Note that if we do not need the reduced form of an element in $F_3$, then evaluating $\tau$ simplifies to $(1 + a_0 + a_1 x)/(1 + a_0 + a_1 x^2)$, so no inversion is necessary. Mapping an unreduced element back to $F_2$, we obtain

$$\frac{a_0 + a_1 x}{a_0 + a_1 x^2} = \frac{a_0^p - a_1^p}{a_0^p - a_0 a_1 + a_1^p} + \frac{2a_0 a_1 - a_1^p}{a_0^p - a_0 a_1 + a_1^p} x.$$

Using this fractional form alone does not seem to offer any advantage over the $F_2$ representation. However, considering the exponentiation of a reduced element $(g + x)/(g + x^2)$ of $T_6$, we already save one $\mathbb{F}_{p^3}$ multiplication for every multiplication by this element, since

$$\left( \frac{g + x}{g + x^2} \right) \times \left( \frac{a_0 + a_1 x}{a_0 + a_1 x^2} \right) = \left( \frac{ga_0 - a_1}{ga_0 - a_1} + \frac{ga_1 + a_0 - a_1}{ga_0 - a_1} x \right),$$

so we only need to compute $ga_0$ and $ga_1$, and a few additions, when our multiplier is in this form.

The reason this all works is that the rational representation of elements of $T_2$ can be embedded efficiently as a fraction in the field extension. Noting that we need only work with the numerator, the group law can be performed directly on this compressed element.

**$F_3$ Frobenius**

Let $a \in F_3$. Then

$$\left( \frac{a_0 + a_1 x}{a_0 + a_1 x^2} \right)^p = \left( \frac{a_0^p + a_1^p x^2}{a_0^p + a_1^p x} \right) = \left( \frac{(a_0^p - a_0) + a_1^p x}{(a_0^p - a_1) - a_1^p x^2} \right) = \left( \frac{(a_0^p - a_0) + a_1^p x}{(a_1^p - a_0^p) + a_1^p x^2} \right).$$

**$F_3$ Multiplication**

Multiplication by a reduced element is performed as in (13), or if by a non-reduced element, exactly as in $F_2$.

**$F_3$ Squaring**

This is performed as in $F_2$.

**$F_3$ Inversion**

This is straightforward, since elements are represented as fractions.

$$\left( \frac{a_0 + a_1 x}{a_0 + a_1 x^2} \right)^{-1} = \left( \frac{a_0 + a_1 x^2}{a_0 + a_1 x} \right) = \left( \frac{(a_1 - a_0) + a_1 x}{(a_1 - a_0) + a_1 x^2} \right).$$
5.4. Higher Dimension Tori

Also, since we use the intermediate representation \( F_3 \) between \( \mathcal{A}^2(F_p) \) and \( F_2 \), we must adjust the map \( \rho : F_3 \setminus \{1, x^2\} \xrightarrow{\sim} \mathcal{A}^2(F_p) \setminus V(f) \). Let \( \beta = (\beta_1 + \beta_2x)/(\beta_1 + \beta_3x^2) \in F_3 \), with \( \beta_1/\beta_2 = u_1 + u_2y + u_3(y^2 - 2) \); then \( \rho(\beta) = (u_2/u_1, u_3/u_1) \). We summarise the results regarding arithmetic in \( F_1, F_2 \) and \( F_3 \) in the following:

**Lemma 5.3.6** The cost of arithmetical operations in \( F_1, F_2 \) and \( F_3 \) are:

<table>
<thead>
<tr>
<th>Operation</th>
<th>( F_1 )</th>
<th>( F_2 )</th>
<th>( F_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Multiply</strong></td>
<td>( 18M + 53A )</td>
<td>( 18M + 54A )</td>
<td>( 18M + 54A )</td>
</tr>
<tr>
<td><strong>Square</strong></td>
<td>( 6M + 21A )</td>
<td>( 12M + 33A )</td>
<td>( 12M + 33A )</td>
</tr>
<tr>
<td><strong>Inverse</strong></td>
<td>2A</td>
<td>6A</td>
<td>3A</td>
</tr>
<tr>
<td><strong>Frobenius</strong></td>
<td>1A</td>
<td>10A</td>
<td>10A</td>
</tr>
<tr>
<td><strong>Reduce</strong></td>
<td>n/a</td>
<td>n/a</td>
<td>19M + 35A + 1</td>
</tr>
<tr>
<td><strong>Mixed Mul.</strong></td>
<td>n/a</td>
<td>n/a</td>
<td>12M + 33A</td>
</tr>
</tbody>
</table>

Here the operation **Reduce** refers to obtaining the reduced form of an element of \( F_3 \), and a **Mixed Mul.** refers to multiplying a non-reduced element by a reduced one. The cost of the map \( \rho : F_3 \to \mathcal{A}_2 \) assumes the element being compressed is in non-reduced form, as this is the case after an exponentiation in both \( F_1 \), or \( F_3 \). Also, for the map \( \tau^{-1} : F_3 \to F_2 \) we assume that the \( x \)-coefficient is in \( F_p \) only as in (12), and not \( F_p^3 \), as in practice one would only perform this operation when decompressing from \( \mathcal{A}_2 \) to \( F_1 \), and not from a non-reduced element.

5.3.4.iv Exponentiation

For exponentiation one can either use the representation \( F_1 \) and the methods of Section 5.2, or \( F_3 \) with some precomputation, to reduce the cost of squaring. In large characteristic, \( F_1 \) is the faster, however for small characteristic (as used in some pairing-based systems), \( F_3 \) is more efficient due to the low multiplication cost [GPS05].

5.4 Higher Dimension Tori

Lacking the assurance of rationality of higher dimensional tori, such as the cryptographically interesting \( T_{30} \) and \( T_{210} \), the question of whether it is possible to exploit the properties of these tori remained open. The work of van Dijk and Woodruff at CRYPTO 2004 demonstrated that in the case where several torus elements are to be transmitted, an optimal compression factor of \( n = (n/\phi(n)) \) over the field representation can be achieved asymptotically with the number of elements transmitted.

The reason underpinning this possibility is that whilst algebraic tori are not known to be rational, they are known to be stably rational, i.e., for every \( n \) there is an \( m \) such that there is an “almost bijection” between \( T_n(F_p) \times F_p^m \) and \( F^{\phi(n)+m} \). This allows one to chain elements together by using the \( m \) element surplus on the right as input surplus to the left. Originally the surplus \( m \) was 32, however in the recent work of van Dijk et al., this was shown to be reducible to \( m = 2 \), thus providing an improvement over XTR and CEILIDH for as few as two torus elements.
In this section we explain their method, and present optimised arithmetic for its implementation for the torus $T_{30}$.

### 5.4.1 Asymptotically Optimal Torus-Based Cryptography

Since $T_n$ is known to be rational only for special values of $n$, the above ideas do not lead to an optimal compression factor of $n/\phi(n)$ in general. Van Dijk and Woodruff [vDiWo04] overcome this problem in the case where several elements of $T_n$ are to be compressed. They construct a bijection:

$$
\theta : T_n(F_q) \times \times_{d|n, \mu(n/d) = 1} F_{q^d}^\times \to \times_{d|n, \mu(n/d) = 1} F_{q^d}^\times.
$$

Specializing their map to the case $n = 30$ gives

$$
T_{30}(F_q) \times F_q^\times \times F_{6}^\times \times F_{10}^\times \times F_{15}^\times \to F_{2}^\times \times F_{3}^\times \times F_{5}^\times \times F_{10}^\times,
$$

which can be reinterpreted as an “almost bijection” (see [vDiWo04])

$$
T_{30}(F_q) \times A_{32}(F_q) \to A_{40}(F_q).
$$

One can use this map to achieve an asymptotic compression factor of $30/8$. Indeed, to compress $m$ elements of $T_{30}(F_q)$, one can compress an element $x$ and split its image into $y_1 \in A_8(F_q)$ and $y_2 \in A_{32}(F_q)$. Then $y_1$ forms the affine input of the next compression. In the end, $8m + 32$ elements of $F_q$ are used to represent $m$ elements of $T_{30}(F_q)$. Observe that their map comes from the equation

$$
\Phi_{30}(x)(x - 1)(x^6 - 1)(x^{10} - 1)(x^{15} - 1) = (x^2 - 1)(x^3 - 1)(x^5 - 1)(x^{30} - 1),
$$

relating the orders of all the different component groups of domain and range. Since these groups are cyclic, one can map to and from their products as long as the orders of the component groups are coprime. For the map above there are some small primes that occur in the order of several component groups, but van Dijk and Woodruff are able to isolate and handle them separately.

### 5.4.2 The New Construction

The bijection (14), while asymptotically optimal, leaves open the question of whether one can obtain better compression for a fixed number of elements. Our new compression map, given by Equation (17) below (see Theorems 5.4.2 and 5.4.4), has this property. Using the fact that

$$
\Phi_n(x) = \prod_{d|n}(x^d - 1)^{\mu(n/d)},
$$

we have

**Proposition 5.4.1** If $p$ is a prime, and $a$ is a positive integer not divisible by $p$, then

$$
\Phi_{ap}(x)\Phi_a(x) = \Phi_a(x^p).
$$

The following result can be deduced from Proposition 5.4.1 above, using Lemma 6 of [vDiWo04] (see also pp. 60–61 of [Vos98]). Here, Res denotes the Weil restriction of scalars (see for example [Vos98] or [RuSi04]).
5.4. Higher Dimension Tori

Theorem 5.4.2 If $p$ is a prime, $q$ is a prime power, $a$ is a positive integer, $qa$ is not divisible by $p$, and $\gcd(\Phi_{ap}(q), \Phi_a(q)) = 1$, then
\[ T_{ap}(\mathbb{F}_q) \times T_a(\mathbb{F}_q) \cong (\text{Res}_{q^p/q} T_a)(\mathbb{F}_q) \cong T_a(\mathbb{F}_{q^p}). \]

The next result follows from Proposition 5.4.1, by doing double induction on the number of prime divisors of $n$ and the number of prime divisors of $m$.

Theorem 5.4.3 If $n$ is square-free and $m$ is a divisor of $n$, then
\[ \prod_{d \mid n, \mu(\frac{n}{md})=1} \Phi_n(x) \Phi_m(x^d) = \prod_{d \mid n, \mu(\frac{n}{md})=-1} \Phi_m(x^d). \]

The next result follows from Theorem 5.4.3, using the ideas in the proof of Theorem 3 of [vDiWo04].

Theorem 5.4.4 If $n$ is square-free and $m$ is a divisor of $n$, then there is an efficiently computable bijection (with an efficiently computable inverse)
\[ T_n(\mathbb{F}_q) \times \prod_{d \mid n, \mu(\frac{n}{md})=-1} T_m(\mathbb{F}_{q^d}) \rightarrow \prod_{d \mid n, \mu(\frac{n}{md})=1} T_m(\mathbb{F}_{q^d}). \]

Note that [vDiWo04] is based on the case $m = 1$ of Theorem 5.4.4. Theorem 5.4.4 is most useful to us when $T_m$ is rational. If $T_m$ is rational, then Theorem 5.4.4 gives efficiently computable “almost bijections” between $T_m$ and $A^{\phi(m)}$, and we have
\[ T_n \times A^{D(m,n)} \sim A^{\phi(n)+D(m,n)} \tag{16} \]
where
\[ D(m,n) = \phi(m) \sum_{d \mid n, \mu(\frac{n}{md})=-1} d \]
and $\sim$ denotes efficient “almost bijections”. The smaller $D(m,n)$ is, the better for our applications. Given the current state of knowledge about the rationality of the tori $T_m$, we take $m$ with at most two prime factors. Ideally, $m = 6$. One could also take $m = 2$. When $m = 6$, then Equation (16) gives
\[ T_{30} \times A^2 \sim A^{10} \quad \text{and} \quad T_{210} \times A^{24} \sim A^{72}. \]
As a comparison with the original bijection (14) for $n = 30$ which requires $8m + 32$ elements of $\mathbb{F}_q$ to represent $m$ elements in $T_{30}(\mathbb{F}_q)$, we see that this provides a considerable improvement.

Even better, using Proposition 5.4.1 and induction on the number of prime divisors of $n$, we also obtain the following.

Theorem 5.4.5 If $n = p_1 \cdots p_k$ is a product of $k \geq 2$ distinct primes, then
\[ \Phi_n(x) \prod_{i=2}^{k-1} \Phi_{p_1 \cdots p_i}(x^{p_i+2 \cdots p_k}) = \Phi_{p_1 p_2}(x^{p_3 \cdots p_k}). \]
Applying this to \( n = 210 = 2 \cdot 3 \cdot 5 \cdot 7 \), one can similarly show

\[
T_{210}(F_q) \times T_{30}(F_q) \times T_6(F_{q^7}) \sim T_6(F_{q^{30}}).
\]

Now since \( T_6 \sim A^2 \), we obtain \( T_{210} \times T_{30} \times A^{14} \sim A^{70} \). Using \( T_{30} \times A^2 \sim A^{10} \) now gives \( T_{210} \times A^{22} \sim T_{210} \times A^{10} \times A^{12} \sim T_{210} \times (T_{30} \times A^2) \times A^{12} \sim T_{210} \times T_{30} \times A^{14} \sim A^{70} \), so

\[
T_{210} \times A^{22} \sim A^{70}.
\]

More generally, the above reasoning shows that if \( n = p_1 \cdots p_k \) (square-free), then

\[
T_n \times A^{\phi(p_1p_2)\cdots p_n-\phi(n)} \sim A^{\phi(p_1p_2)\cdots p_n},
\]

which for \( 6 \mid n \) gives

\[
T_n \times A^{n/3-\phi(n)} \sim A^{n/3}.
\]

Using Equation (17), one can compress \( m \) elements of \( T_n(F_q) \) down to just \((m-1)\phi(n) + n/3\) elements of \( F_q \) by either sequential or tree-based chaining as explained in §5.4.3.

5.4.2.i Applying the Construction to \( T_{30} \) Henceforth we focus on \( n = 30 \) since this improves upon previous schemes, has a straightforward parameter generation (see §5), and will be computationally efficient. Note that \( \gcd(\Phi_{30}(q), \Phi_6(q)) = 1 \). Indeed, using the first paragraph of the proof of Lemma 6 of [vDiWo04], the only possible prime dividing \( \gcd(\Phi_{30}(q), \Phi_6(q)) \) is 5, but it is easy to see that regardless of \( q \) we have \( \Phi_6(q) \mod 5 \in \{1, 2, 3\} \), which proves our claim. By Theorem 5.4.2 we now have

\[
T_{30}(F_q) \times T_6(F_q) \cong T_6(F_{q^5}).
\]

The compression is based on a sequence of maps

\[
T_{30}(F_q) \times (A^2(F_q) \setminus V(f)) \rightarrow T_{30}(F_q) \times T_6(F_q) \rightarrow T_6(F_{q^5}) \rightarrow A^2(F_{q^5}) \setminus V(f_5),
\]

where \( V(f_5) \) denotes \( V(f) \) over \( F_{q^5} \). We denote by \( \theta \) the forward composition of the three maps above, and by \( \theta^{-1} \) the composition of the inverses. Note that if we have \( m \) elements in \( T_{30}(F_q) \), we compress them down to \( 8m + 2 \) elements of \( F_q \). Thus the compression outperforms CEILIDH and XTR when as few as two elements are compressed.

The first and last maps are based on CEILIDH decompression and compression, respectively. We discuss some possibilities for the map \( \sigma(\cdot, \cdot) \) between \( T_{30}(F_q) \times T_6(F_q) \) and \( T_6(F_{q^5}) \) in §5.4.4 below.

5.4.2.ii Missing points With regard to the functionality of \( \theta \), the only remaining issue is that the outer two maps based on CEILIDH do not give everywhere-defined injections.

We can slightly modify the CEILIDH maps, so that for compression we get an injection \( \psi': A^2(F_q) \rightarrow T_6(F_q) \times \{0, 1\} \) and for decompression an injection \( \rho': T_6(F_q) \times \{0, 1\} \rightarrow A^2(F_q) \). Note that \( \psi' \) and \( \rho' \) need not be inverses. The two missing points in \( \rho' \)'s domain can easily be added by using a table lookup into two arbitrarily chosen points in \( V(f) \). The resulting map is \( \rho' \).
Given the different cardinalities of $T_6(F_p)$ (namely $p^2 - p + 1$) and $A^2(F_p)$ (namely $p^2$), there are certain points in $A^2(F_p)$ that do not decompress. If we concentrate on the case $p \equiv 2 \mod 9$ or $p \equiv 5 \mod 9$, then the variety $V(f)$ is defined by $f(v_1, v_2) = 1 - v_1^2 - v_2^2 + v_1 v_2$. For fixed $v_2$ this has at most 2 roots, and if this is the case then their difference is $(4 - 3v_2^2)^{1/2}$. If this expression equals 2 then $v_2 = 0$, in which case $v_1 \in \{ -1, 1 \}$. Thus we have a map $\psi : A^2(F_q) \to T_6(F_q)$:

- If $f(v_1, v_2) \neq 0$, then $\psi'(v_1, v_2) = (\psi(v_1, v_2), 0)$,
- Else if $v_2 \neq 0$, then $\psi'(v_1, v_2) = (\psi(v_1 + 2, v_2), 1)$,
- Else $\psi'(v_1, v_2) = (\psi(v_1 + 1, v_2), 1),$

where the extra bit indicates whether the input landed in the variety.

### 5.4.3 Applications

Our new map saves a significant amount of communication in applications where many group elements are transmitted. For instance the compression can be used to agree on a sequence of keys using Diffie-Hellman as in §5.1 of [vDiWo04]. Other applications include verifiable secret sharing, electronic voting and mix-nets, and private information retrieval.

In our applications we compress many elements. This is done by using part of the output of the $i$-th element as the affine input for the compression of the $(i+1)$-st element. This sequential chaining is simple, but has the drawback of needing to decompress all elements in order to obtain the first element. Alternatively, one can use trees to allocate the output of previous compressions. For instance, the output of the first compression is split into five pieces, which are subsequently used as the affine input when compressing elements two through six. The output of the second compression is used to compress elements seven through twelve, etc. When compressing $m$ elements, decompressing a specific element now takes $O(\log m)$ atomic decompressions on average.

#### 5.4.3.1 ElGamal encryption

Our first application is ElGamal encryption with a small message domain, where we obtain an additional 10% compression over CELIDH even for the encryption of a single message. This contrasts starkly with the original mapping of [vDiWo04] that cannot be used to achieve any savings for single-message encryption.

Let $q$ and $l$ be primes such that $l \mid \Phi_{30}(q)$. Let $g \in F_{q^{30}}^\times$ have order $l$, so that $\langle g \rangle \subseteq T_{30}(F_q)$. For random $a$, $1 \leq a \leq l - 1$, let $a$ be Bob’s private key and $A = g^a$ his public key. Without loss of generality, let $M = \{0, 1, \ldots, m - 1\}$ be the set of possible messages. We assume that $m$, the cardinality of $M$, is small. We apply the mapping of §5.4.2 to the generalized ElGamal encryption scheme.

**Encryption ($M$):**

1. Alice represents the message $M$ as $g^M \in \langle g \rangle$.
2. Alice selects a random integer $k$, $1 \leq k \leq l$, and computes $d = g^k$. 

3. Alice sets $e = g^M \cdot (g^a)^k$.

4. Alice expresses $d \in T_6(F_{q^5})$ as $(d_1, d_2) \in A^2(F_q) \times A^1(F_q) \cong A^2(F_{q^5})$ by using CEILIDH. Alice compresses $e \in T_{30}(F_q)$ and $d_2 \in A^1(F_q)$ as $(e, d_2) = T$, and outputs $(d_1, T)$.

**Decryption $(d_1, T)$:**

1. Bob computes $\theta^{-1}(T) = (e, d_2)$ and uses CEILIDH to reconstruct $d$.

2. Bob uses his private key $a$ to recover $g^M = d^{-a} e$.

3. Bob recovers $M$ from $g^M$ using the fact that $M$ comes from a small domain (e.g., using Pollard lambda or a table lookup).

The ciphertext is represented as 18 symbols in $F_q$, which is a 10% improvement over a solution in which CEILIDH is used to compress both $d$ and $e$. Note that the mapping of [vDiWo04] in §5.4.1 cannot be used to achieve any savings in this case.

Our scheme preserves homomorphy, that is, without knowing the secret key $a$ one can compute the encryption of $M_1 + M_2$ given encryptions of $M_1$ and $M_2$ separately. This is useful in applications such as the efficient two-party computations proposed by Schoenmakers and Tuyls [ScTu04] which use homomorphic ElGamal encryption for a small number of messages.

Exactly as for XTR and CEILIDH (with 6 replaced by 30), the security of our schemes follows from the difficulty of the DDH problem in $F_{q^{30}}^\times$, the fact that $T_{30}(F_q)$ is the primitive subgroup of $F_{q^{30}}^\times$, and the fact that our map and its inverse are efficiently computable.

The representation of $M$ as $g^M$ in $\langle g \rangle$ is not efficient when $m$ is large. We leave it as an open problem to adapt our scheme to handle a larger message domain. We note that one solution is to use hybrid ElGamal encryption instead. Indeed, we may apply the mapping of §5.4.2 to hybrid ElGamal encryption, adapting a protocol in §5.3 of [vDiWo04]. In general, though, this solution does not preserve the homomorphic property.

### 5.4.3.ii ElGamal Signatures

We apply the mapping of §5.4.2 to the generalized ElGamal signature scheme, adapting a protocol in §5.2 of [vDiWo04]. Here the signature has the form $(d, e)$, where $d \in \langle g \rangle$ and $1 \leq e \leq l - 1$. The idea is to use part of $e$ in the affine component of $\theta$, which can be done without any loss since $\log_2 e \approx 160$ while $2 \log_2 q \approx 70$. Since the affine component of [vDiWo04] is much larger, this is not possible in their setting.

For a random $a$, $1 \leq a \leq l - 1$, let $a$ be Alice’s private key and $A = g^a$ her public key. Let $h : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ be a cryptographic hash function. We have the following generalized ElGamal signature scheme for input message $M$:

**Signature Generation $(M)$:**

1. Alice selects a random secret integer $k$, $1 \leq k \leq l$, and computes $d = g^k$.

2. Alice then computes $e = k^{-1}(h(M) - ah(d)) \mod l$. 
3. Alice expresses $e$ as $(e_1, e_2) \in F_2^2 \times \{0, 1\}^\ast$, computes $\theta(d, e_1) = T$, and outputs $(e_2, M, T)$ as her signature.

**Signature Verification** $(e_2, M, T)$:

1. Bob computes $\theta^{-1}(T) = (d, e_1)$ and recovers $e$.
2. Bob accepts the signature if and only if $A^{h(d)d^e} = g^{h(M)}$.

We note that in practice one has the alternative of using Schnorr’s signature scheme, which already achieves optimal compression.

5.4.3.iii Voting Schemes

We will discuss a recent voting scheme by Kiayias and Yung [KiYu02], which is based on the discrete logarithm problem and for which we propose to use $T_{30}$. We give a comparison with a cutting edge scheme based on Paillier encryption due to Damgård and Jurik [DaJu03].

Let $L$ denote the number of voters. Each voter has a secret key $a_i$, and a public key $g^{a_i}$. The $i$-th voter chooses $L$ random exponents $s_{i,j} \in Z_l$ which satisfy $\sum_j s_{i,j} = 0$, where $j$ ranges from 1 to $L$ (the scheme is self-tallying, which basically means that the voters themselves serve as the talliers). Voter $i$ computes and posts $g^{a_j s_{i,j}}$ for all $j$, and a zero-knowledge proof that his post is well-formed. Define $t_j = \sum_i s_{i,j}$ and observe that

(a) $\sum_j t_j = 0$,

(b) $t_j$ is a random element in $Z_l$ if at least one user is honest.

From the posts, the $j$-th voter can compute $g^{a_j t_j}$, and then by using $a_j$ can also compute $g^{t_j}$. If $f$ is another public generator of $\langle g \rangle$, the vote of the $j$-th voter is a bit $b_j$ from which he can calculate and post $g^{a_j s_{i,j}} f^{b_j}$. From all such posts, we have $\prod_j g^{a_j s_{i,j}} f^{b_j} = f^{\sum_j b_j}$. Since the tally $\sum_j b_j < L$, it can be found with Pollard’s lambda method in $O(\sqrt{L})$ multiplications (and one already needs $\Theta(L)$ multiplications to compute $\prod_j g^{a_j s_{i,j}}$).

Damgård and Jurik [DaJu03] propose a similar scheme as Kiayias and Yung, but use Paillier encryption [Pai99] as a starting point. Again, there are $L$ voters, $L$ secret keys $Sk_i$, and public keys $Pk_i$. The $i$-th voter chooses $L$ random integers $s_{i,j}$ in a predefined range with $\sum_j s_{i,j} = 0$. Voter $i$ posts $E_{Pk_i}(s_{i,j})$ for all $j$, and a zero-knowledge proof that these values are well-formed. Define $t_j$ as above, and observe properties (a) and (b) again hold (the latter statistically). From the posts, voter $j$ computes $E_{Pk_i}(t_j)$, and using $Sk_j$ gets $t_j$. If his vote is $b_j$, he then posts $p_j = t_j + b_j$. Observe that $\sum_j p_j = \sum_j b_j$. Hence, tallying is more efficient than in the scheme of Kiayias and Yung, using $L$ additions versus $O(L)$ multiplications.

Although the Paillier-based scheme can be expected to be faster, a scheme based on $T_{30}$ is considerably more compact. We give an analysis of the communication required for both schemes under the assumption that one wants $\log_2 n = 1024$, and that one achieves the same level of security with $30 \log_2 q = 1024$ and a subgroup size of 160 bits.

The communication of the Kiayias-Yung scheme is dominated by the sending of $g^{a_j s_{i,j}}$ for all $i, j$, together with their zero-knowledge proofs. When looking at the zero-knowledge proofs...
80  5. XTR, Subgroup- and Torus-based Cryptography

used, one sees that each voter transmits $4L$ group elements and $L$ exponents. Thus we can use $T_{30}$ compression in $\mathbb{F}_{q^{30}}$ here. This results in roughly $4L \cdot 8 \log_2 q + 160L = 32L \log_2 q + 160L \approx 1253L$ bits per voter.

The communication of the Damgård-Jurik scheme is similarly dominated by the sending of the $E_{PK_j}(s_{i,j})$ for all $i, j$ with their zero-knowledge proofs. We note that their encryption scheme $E$ is not exactly the same as that of Paillier, but a modification of it where the ciphertext size is at least $3 \log_2 n$ bits, $n$ being an RSA modulus. Moreover, each proof that $E_{PK_j}(s_{i,j})$ is well-formed costs at least $5 \log_2 n$ bits. Thus $8L \log_2 n = 8192L$ bits are transmitted per voter in this stage.

Hence our improvement is roughly a factor of 6.5. An additional optimization is for the bulletin board to compress the list of public keys $g_i$ when distributing this at the beginning. We note that although we improve in communication and bulletin board size, the computational workload has clearly increased.

5.4.3.iv Mix-nets A typical re-encrypting mix-net involves $M$ servers that each process $n$ ciphertexts. To process the batch, a server randomly permutes and rerandomises the ciphertexts. In the literature, both ElGamal and Paillier type schemes are used. We save on the communication for both sequential and parallel mix-nets [GoJu04]. In a sequential re-encrypting mix-net, server $i$ processes the $n$ ciphertexts, then passes them to server $i + 1$, and after the $M$-th server is done, they perform some kind of threshold decryption. Here $n \cdot M$ bits are communicated using either Paillier or ElGamal, but we save using ElGamal together with $T_{30}$ compression. Our savings is a factor of $30/8$. In parallel mixing, the servers process disjoint batches of input ciphertexts in parallel, and in between processing rounds they transmit $n/M$ ciphertexts to each other, and again we save using $T_{30}$ compression.

5.4.4 Representations and Algorithms for $T_{30}$

In this subsection we discuss implementation issues concerning field representation, key generation, and efficient exponentiation.

5.4.4.i Field Representations Since $T_{30}(\mathbb{F}_q) \subset \mathbb{F}_{q^{30}}$, we need a model of the latter that permits fast multiplication, squaring, inversion and a fast Frobenius automorphism. We also require arithmetic for $T_6$, over both $\mathbb{F}_q$ and $\mathbb{F}_{q^5}$. Since $T_{30}(\mathbb{F}_q) \subset T_6(\mathbb{F}_{q^5})$, we may model the arithmetic of $T_{30}(\mathbb{F}_q)$ by the latter, possibly at the risk of losing some optimizations.

5.4.4.i.a The base field $\mathbb{F}_q$ We base our implementation on high performance arithmetic in $\mathbb{F}_q$ using the representational method of Montgomery [Mon85]. For $T_{30}$ one is likely to use characteristics $q$ between 32 and 64 bits long, corresponding to a 2-word value on a 32-bit architecture. We are careful to distinguish between those small, 2-word values required by $T_6(\mathbb{F}_{q^5})$ and more general values of $q$ (which we need for comparison purposes). Essentially, we employ the trivial program specialisation techniques described by Avanzi [Ava04] to construct compact, straight line code sequences for the 2-word case. This affords a significant improvement over code for general sizes of $q$. Other than the length, we do not rely on
assumptions on the value of \( q \), although one could expect incremental improvements by doing so. Also, our choice of extension degree poses some congruence conditions on \( q \).

### 5.4.4.i.b The extension \( \mathbb{F}_{q^5} \)

We use a degree five subfield of the degree 10 extension \( \mathbb{F}_q[t]/(\Phi_{11}(t)) \), and use the Gaussian normal basis \([t + t^{10}, t^7 + t^4, t^5 + t^6, t^2 + t^9, t^3 + t^8]\). For this to work we require that \( q \neq \pm 1 \mod 11 \) [Noe01]. Since the extension degree is small, we perform inversions using the Itoh-Tsujii algorithm [ItTs88].

### 5.4.4.i.c The torus \( T_6 \)

For the torus \( T_6 \) we take \( q \equiv 2 \mod 9 \) or \( q \equiv 5 \mod 9 \) and use arithmetic based on the degree six extension field defined by adjoining a primitive ninth root of unity to the base field, as in [StLe02, RuSi03, GPS04]. Note that in \( T_6 \) we have virtually free inversion, as it is just the cube of the Frobenius automorphism.

### 5.4.4.ii Compression and Decompression

Our new compression and decompression algorithms require two components: CEILIDH and CRT. We use an implementation of CEILIDH as given in [GPS04].

Although it seems that Chinese remaindering is straightforward, there is some flexibility in choosing the map \( \sigma : T_{30}(\mathbb{F}_q) \times T_6(\mathbb{F}_q) \rightarrow T_6(\mathbb{F}_{q^5}) \). Following [vDiWo04] we have \( \sigma(x, y) = x^\beta y^\gamma \), where \( \alpha \Phi_{30}(q) + \beta \Phi_6(q) = 1 \). The inverse is given by \( \sigma^{-1}(z) = (z^{\Phi_6(q)}, z^{\Phi_{30}(q)}) \). The cost of the forward computation (i.e., \( \sigma \)) is an exponentiation in \( T_{30}(\mathbb{F}_q) \), an exponentiation in \( T_6(\mathbb{F}_q) \), and a multiplication. Depending on the context, the exponentiation in \( T_{30}(\mathbb{F}_q) \) may be combined with an exponentiation performed as part of a particular protocol. The inverse is a double exponentiation.

Also attractive is the simple \( \sigma'(x, y) = xy \) with inverse \( \sigma'^{-1}(z) = (zy^{-1}, y) \) where \( y = z^{\alpha \Phi_{30}(q)} \). Clearly the forward map only costs a multiplication. For the inverse we first compute \( y \) using a single exponentiation. Note that the exponent here is larger than in the case of \( \sigma \), but the total amount of exponent is similar in both cases (although asymptotically it is not the total amount that counts, it is what is relevant in practice). Moreover, we are typically concerned with the case where the preimage \( x \in T_{30}(\mathbb{F}_q) \) has an order \( l \) dividing \( \Phi_{30}(q) \) so we know that \( z \) has order dividing \( l(q^2 - q + 1) \), which we can use to reduce the exponent \( \alpha \Phi_{30}(q) \). As noted before, the computation of \( y^{-1} \) is virtually free, so this method is clearly preferable to the first suggestion.

### 5.4.4.iii Arithmetic Costs

Let \( M, A, S \) and \( I \) represent the cost of multiplication, addition, squaring and inversion in \( \mathbb{F}_q \), respectively. In Table 1 (cf. [GPS04, Lemma 3]) we detail the relative costs for arithmetic in \( \mathbb{F}_{q^5} \), and for both \( T_6(\mathbb{F}_q) \) and \( T_6(\mathbb{F}_{q^5}) \). Compression and decompression are based on CEILIDH.

### 5.4.4.iv Exponentiation in \( T_{30} \)

In protocols, one is required to perform one of three operations involving exponentiation: a single exponentiation in \( T_{30}(\mathbb{F}_q) \), a double exponentiation in \( T_{30}(\mathbb{F}_q) \), or a single exponentiation in \( T_6(\mathbb{F}_{q^5}) \) (for the map \( \sigma'^{-1} \) described above). Since \( T_{30}(\mathbb{F}_q) \subset T_6(\mathbb{F}_{q^5}) \), we can perform all three of these in \( T_6(\mathbb{F}_{q^5}) \) using the methods developed.
### 5. XTR, Subgroup- and Torus-based Cryptography

<table>
<thead>
<tr>
<th></th>
<th>$\mathbb{F}_{q^5}$</th>
<th>$T_6(\mathbb{F}_q)$</th>
<th>$T_6(\mathbb{F}_{q^5})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiply</td>
<td>$15M + 75A$</td>
<td>$18M + 53A$</td>
<td>$270M + 1615A$</td>
</tr>
<tr>
<td>Square</td>
<td>$6M + 21A$</td>
<td>$2A$</td>
<td>$10A$</td>
</tr>
<tr>
<td>Inverse</td>
<td>$65M + 300A + I$</td>
<td>$1A$</td>
<td>$5A$</td>
</tr>
<tr>
<td>Frobenius</td>
<td>$0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compress</td>
<td>$15M + 31A + I$</td>
<td>$290M + 1580A + I$</td>
<td></td>
</tr>
<tr>
<td>Decompress</td>
<td>$27M + 52A + I$</td>
<td></td>
<td>$470M + 2585A + I$</td>
</tr>
</tbody>
</table>

Table 3: Arithmetic costs.

in [StLe02]; the main properties one can exploit are the degree two Frobenius automorphism and fast squaring.

In a subgroup of order $l$ where $l|(q^{10} - q^5 + 1)$, we write an exponent $m$ as $m \equiv m_1 + m_2q^5 \mod l$, where $m_1$ and $m_2$ are approximately half the bit-length of $l$ (as in [StLe02]). One can find $m_1$ and $m_2$ very quickly having performed a one-time Gaussian two dimensional lattice basis reduction, and using this basis to find the closest vector to $(m, 0)^T$. Having split the exponent, to compute $a^m$ for random $a$ and $m$, we perform a double exponentiation $a^{m_1}(a^{q^5})^{m_2}$ using the Joint Sparse Form (JSF) of the integers $m_1$ and $m_2$ [Sol01], which on average halves the number of pairs of non-zero bits in their paired binary expansion. The use of the JSF in the above is possible since we have virtually free inversion.

To perform a double exponentiation, we split both exponents as with the single exponentiation, and perform the necessary four-fold multi-exponentiation as a product of two double exponentiations, combining the required squarings.

In general one may also be able to exploit the additional structure of $T_{30}$, which possesses an automorphism of degree eight, namely, the Frobenius automorphism. One can in principle employ exactly the same method as above and perform an eight-fold multi-exponentiation. However for the parameter sizes we consider in this paper, we use a much simpler method based on the $q$-ary expansion of an exponent $m$. Specifically, since our value of $q$ will be small we can write an exponent $m$ as $m = \sum m_i q^i$.

For our implementation where $q^{30}$ has size approximately 1024 bits, exponentiating by a 160 bit exponent consists of five terms in the $q$-ary expansion, and hence we perform a five-fold multi-exponentiation. To perform this one can use ideas of Proos [Pro03], which extend the JSF to more than two exponents. However due to the amount of precomputation required, for exponents of cryptographic interest we use a naive combination of the JSF and the non-adjacent form (NAF). This results in an effective exponent length of around the same size as $q$, significantly reducing the number of squarings needed for exponentiation.

With regard to decompression, the exponent in this case is slightly longer than for a single exponentiation. Again we use a $q$-ary expansion, consisting of seven terms in this instance, and apply the JSF to three pairs of them and the NAF to the remaining one. This is a generalization of an idea for trace zero varieties described in [AvLa05].

Note that for larger parameter choices, one can clearly construct more efficient multi-exponentiation methods than those we have optimised for 1024 bit fields. We omit the details.
5.4.5 Security

Granger and Vercauteren have recently described an attack on algebraic tori that belongs to the index calculus type of attacks [GrVe05], and directly exploits the rational representations of the torus elements. Their results do not affect the security of the cryptosystems LUC, XTR, or CEILIDH over prime fields. However, the practical efficiency of our method against other methods needs further examining, for certain choices of p and m in regions of cryptographic interest. In particular, the security of the methods based on extensions of degree 30 must be newly assessed.

5.5 Conclusions

The methods presented in this Chapter allow efficient computation in subgroups of finite fields, by exploiting suitable representations. Therefore, the corresponding cryptographic primitives are not new, and their security has been studied extensively already. Moreover, this line of research prompts also new investigations of security which lead to interesting developments also for other special cases of the discrete logarithm primitive, such as Avanzi and Mihăilescu’s PAFFs [AvMi03].

On the whole, the methods here make implementing cryptographic primitives designed around the discrete logarithm problem in finite fields quite interesting, by reducing the performance gap with other systems like those designed around the discrete logarithm in Jacobians of algebraic curves. Yet, they represent more a middle term solution than a long term one, because of the disparity of the complexity of solving the problems underlying the two primitives.

References – XTR, Subgroup- and Torus-based Cryptography


5. XTR, Subgroup- and Torus-based Cryptography


5.5. Conclusions


5. XTR, Subgroup- and Torus-based Cryptography
6 Multivariate Systems

Contributors: An Braeken, Christopher Wolf, and Bart Preneel

6.1 Hidden Field Equations in Asymmetric Cryptography

General picture Public key cryptography is used in e-commerce systems for authentication (electronic signatures) and secure communication (encryption). In contrast to secret key cryptography, public key cryptography has advantages in terms of key distribution. Moreover, signature schemes cannot be obtained by secret key schemes. The security of using current public key cryptography for encryption centres on the difficulty of solving certain classes of problem. The RSA scheme relies on the difficulty of factoring very large numbers, while the difficulty of solving discrete logarithms provide the basis for the ElGamal and Elliptic Curve schemes [MvOV96]. Given that the security of these public key schemes rely on such a small number of problems that are currently considered hard, research on new schemes that are based on other classes of problems is worthwhile. Such work provides a greater diversity that forces cryptanalysts to expend additional effort by concentrating on a completely new type of problem.

In addition, important results on the potential weaknesses of existing public key schemes are emerging. Techniques for factorisation and solving discrete logarithm continually improve. Polynomial time quantum algorithms can be used to solve both problems [Sho97]; fortunately, quantum computers with more than 7 bits are not yet available and it seems unlikely that quantum computers with 100 bits will be available within the next 10–15 years. Nevertheless, this stresses the importance of research into new algorithms for asymmetric encryption and signature schemes.

Alternative Schemes In 1996, Patarin proposed the use of a special class of polynomials over finite fields for public key cryptography called “Hidden Field Equations” (HFE) [Pat96]. The scheme supports both encryption and digital signatures; its security is related to the difficulty of solving a random system of multivariate quadratic equations over finite fields. This problem is known to be \( \mathcal{NP} \)-complete (cf [GJ79, p. 251] and [PG97b, App.] for a detailed proof). The HFE scheme generalises and improves the Matsumoto-Imai-system [MI88] which was broken by Patarin [Pat95].

In particular, Hidden Field Equations have been used to construct digital signature schemes, e.g., Quartz [CGP01] and Sflash [CGP03]. Sflash has been developed to suit the smart-card environment; it is a modified and secured version of the Matsumoto-Imai-system. In the present section, we concentrate on its sister-scheme Quartz, which is based on Hidden Field Equations. In contrast to other schemes, Quartz allows very short signatures, namely only 128 bit (RSA: 1024–2048 bit). The main drawback of Quartz is the size of its public key: 71 kB, which is rather high (RSA: 1024–2048 bit). But due to its very short signature size, it is still of high interest for applications with only limited bandwidth for signature transfer. By the construction of Quartz, its signatures are secure against birthday attacks. Moreover,
6. Multivariate Systems

checking the validity of a given signature is very fast. In particular, no crypto-coprocessor is required, even for 8-bit CPUs.

Until recently, signature schemes based on HFE were believed to be secure. The attack of Faugère and Joux — using Gröbner bases (cf [FJ03] and § 6.1.3) — raised serious doubts. On the other hand, HFE offer many variations which make it possible to counter attacks (cf § 6.1.1.ii). In particular, at present it seems possible to vary Quartz in a way that the attack of Faugère and Joux no longer applies. However, more research is needed in this area to obtain secure schemes based on Hidden Field Equations.

Outline This section on Hidden Field Equations is organised as follows: first, we give an introduction to Hidden Field Equations (§ 6.1.1). In § 6.1.2 we outline the signature scheme Quartz. An overview of recent attacks against HFE and possible countermeasures is presented in § 6.1.3. We then draw conclusions.

6.1.1 Hidden Field Equations

HFE is based on polynomials over finite fields and extension fields. The general idea is to use a polynomial over an extension field as a private key and a vector of polynomials over the underlying finite field as public key. HFE also uses private affine transformations to hide the extension field and the private polynomial. This way HFE provides a trapdoor for an MQ-problem (system of Multivariate Quadratic equations).

6.1.1.i Mathematical Background Fig. 1 gives an outline of the structure of HFE. $S$ and $T$ represent two affine transformations and $P$ is the private polynomial. Hence, the private key is represented by the triple $(S, P, T)$.

$$\text{input } x$$

$$x = (x_1, \ldots, x_n)$$

$$\text{private: } S$$

$$x'$$

$$\text{private: } P$$

$$y'$$

$$\text{private: } T$$

$$\text{output } y$$

$$\text{public: } (p_1, \ldots, p_n)$$

Fig. 1: MQ-trapdoor $(S, P, T)$ in HFE

The polynomials $(p_1, \ldots, p_n)$ are the public key. These public polynomials as well as the private affine transformations $S$ and $T$ are over $\mathbb{F}$, a finite field with cardinality $q := |\mathbb{F}|$. The
private polynomial $P$ is defined over $E$, an extension field of $F$ generated by the irreducible polynomial $i(x)$ of degree $n$.

### 6.1.1.i.a Encryption and Decryption of Messages using the Private Key

The private polynomial $P$ (with degree $d$) over $E$ is an element of $E[x]$. To keep the public polynomials small, the private polynomial $P$ must have the property that its terms are at most quadratic over $F$. In the case of $E = GF(2^n)$ this means that the powers have Hamming weight at most 2. In symbols:

$$
P : E \rightarrow E \quad P(x) = \sum c_i x^{h_i}, \text{ where}
\begin{align*}
c_i &\in E, h_i \leq d, h_i \neq h_j \forall i \neq j, \\
h_i &= \begin{cases} 0, & \text{constant term} \\ q^a, & a \in \mathbb{N}_0 \\ q^b + q^c, & b, c \in \mathbb{N}_0 \end{cases} \text{ (linear and quadratic terms)}
\end{align*}
$$

Since the affine transformations $S$ and $T$ are over $F$ it is necessary to transfer the message $M$ from $E$ to $F^n$ in order to encrypt it (cf Fig. 2). This is done by regarding $M$ as a vector

$$\text{plaintext } M \downarrow \quad \text{side computation: redundancy } r
\begin{array}{l}
x = (x_1, \ldots, x_n) \\
\quad \text{private: } S \\
\quad \text{private: } P \\
\quad \text{private: } T \\
\quad \text{public: } (p_1, \ldots, p_n)
\end{array}
\downarrow
\begin{array}{l}
x' \\
y' \\
y
\end{array}
$$

Fig. 2: HFE for encryption of the message $M$ with ciphertext $(y, r)$

$(x_1, \ldots, x_n) \in F^n$. Thus we no longer think about the extension field as a field but as an $n$-dimensional vector-space over $F$ with the rows of the identity matrix $I$ as basis of $F^n$. To encrypt $(x_1, \ldots, x_n)$ we first apply $S$, resulting in $x'$. At this point $x'$ is transferred from $F^n$ to $E$ so we can apply the private polynomial $P$ which is over $E$. The result is denoted as $y' \in E$. Once again, $y'$ is transferred to the vector $(y'_1, \ldots, y'_n)$, the transformation $T$ is applied and the final output $y \in E$ is produced from $(y_1, \ldots, y_n) \in F^n$.

To decrypt $y$, the above steps are done in reverse order. This is possible if the private key $(S, P, T)$ is known. The crucial step in the deciphering is not the inversion of $S$ and $T$, but
rather the computation of the solutions of \( P(x') = y' \). As \( P \) has degree \( d \) there are up to \( d \) different solutions \( X' := \{x'_1, \ldots, x'_d\} \in \mathbb{E} \) for this equation. Addition of redundancy to the message \( M \) provides an error-correcting effect that makes it possible to select the right \( M \) from the set of solutions \( X' \). This redundancy is added at the first step (see Fig. 2). Daum provides an estimation of the length of this redundancy [Dau01].

Another way of circumventing this problem would be to take the polynomial \( P \) bijective. Unfortunately, Patarin showed that all possible bijections (e.g., Dickson polynomials, Dobbertin polynomials, power functions) lead to very insecure schemes. We refer to [Pat96, § 7.3] for a detailed discussion of this point.

6.1.1.i.b Message Signature  In addition to encryption/decryption, HFE can also be used for signing a message \( M \). As for decryption, we assume that without the trapdoor \((S, P, T)\) it is computationally not feasible to obtain a solution \((x_1, \ldots, x_n)\) for the system of equations

\[
\begin{align*}
  y_1 &= p_1(x_1, \ldots, x_n) \\
  y_2 &= p_2(x_1, \ldots, x_n) \\
  \vdots \\
  y_n &= p_n(x_1, \ldots, x_n),
\end{align*}
\]

where \((p_1, \ldots, p_n)\) are quadratic polynomials in the variables \( x_1, \ldots, x_n \). In Fig. 3, we follow this notation, so the input for signature generation is denoted with \( y \), while the output is called \( x \). In addition, the message \( M \) consists of \( t \) elements from \( \mathbb{F} \), i.e., \( M = (M_1, \ldots, M_t) \in \mathbb{F}^t \). The vector \( r = (r_1, \ldots, r_f) \in \mathbb{F}^f \) is randomly chosen (see below).

If one knows the private key \( k = (S, P, T) \), the problem of finding a solution \( x \) for given \( y \), reduces to find a solution to the equation \( P(x') = y' \) where the polynomial \( P \in \mathbb{E}[x] \) has degree \( d \). This is feasible. Unfortunately for HFE, \( P(x') \) is usually not a surjection and therefore \( \exists y' : P(x') \neq y' \ \forall x' \in \mathbb{E} \). Keeping this in mind, we cannot find a solution \((x_1, \ldots, x_n)\) for each MQ-problem with a HFE trapdoor. So from a practical point of view, if we do not succeed in finding a solution \( x' \) for a certain \( y' \) in \( P(x') = y' \), we have to try another \( y' \) until we obtain a result \( x' \). In HFE, the number of \( y' \)-values we have to try is small [Pat96]. For a special system such as Quartz (see § 6.1.2), we expect to find a solution for one given \( y' \) with probability \( 1 - \frac{1}{e} \), i.e., approx. 60%. However, as Quartz tries up to 128 different values for
y' for a given message, the overall probability for not finding any solution drops to approx. $2^{-185}$ and is therefore negligible [CGP01, p. 9].

For signature generation, we assume that the message $M \in \mathbb{F}^t$ and $n = t + f$. Here, $f \in \mathbb{N}$ is the number of free input variables for the MQ-problem. So $y = (M_1, \ldots, M_t)\|(r_1, \ldots, r_f)$ where $\|\cdot$ denotes the concatenation function and $(r_1, \ldots, r_f) \in R \mathbb{F}^f$ is chosen uniformly at random. The parameter $f$ has to be selected according to the field size of $\mathbb{F}$. As the parameters in the Quartz scheme are $\mathbb{F} = GF(2)$, and $f = 7$, there are $2^7 = 128$ different $y$-values for each given message $M$. In general, we have $q_f$ different $y$-values for a given message $M$. If we can solve the corresponding $P(x') = y'$ for one of these $q_f$ different $y$-values, we publish the corresponding $x = S^{-1}(x')$ as the signature of $M$. See Fig. 3 for the overall structure of a signature scheme.

Anybody who wants to verify that the message $m = (m_1, \ldots, m_t)$ was signed by the owner of the private key $K = (S, P, T)$ with $x = (x_1, \ldots, x_n)$, uses the public key, that is, $k = (p_1, \ldots, p_t)$ and compares (denoted $\equiv$):

$$
M_1 \equiv p_1(x_1, \ldots, x_n)
$$

$$
M_2 \equiv p_2(x_1, \ldots, x_n)
$$

$$
\vdots
$$

$$
M_t \equiv p_t(x_1, \ldots, x_n).
$$

If all $t$ equations are satisfied, the signature is valid. Otherwise, it is not. Note that only $t$ of the $m = t + f$ public equations are necessary to verify a signature, the equations $p_{t+1}, \ldots, p_{t+f}$ are not used. Therefore in a signature scheme, only $t$ equations will be published and $f$ equations can be discarded. First, this will lead to a shorter public key. Second, as we will see in § 6.1.1.ii, this is also expected to improve the security of HFE.

6.1.1.i.c Public Key: Generation and Encryption After explaining the overall structure of HFE we move on to public key generation. We describe the technique from Matsumoto-Imai [MI88] called “polynomial-interpolation”. In [Wol03], Wolf describes a faster way of generating the public key called “base-transformation”. Due to space limitations in this section, we only describe “polynomial-interpolation”.

We begin with a description of polynomial interpolation for fields $\mathbb{F} \neq GF(2)$. The key generation for $\mathbb{F} = GF(2)$ is slightly different, we deal with this case later in this section. For HFE, we want to obtain polynomials over $\mathbb{F}$ as the public key which have the form

$$
p_i(x_1, \ldots, x_n) = \sum_{1 \leq j \leq n} \gamma_{i,j,k}x_jx_k + \sum_{1 \leq j \leq n} \beta_{i,j}x_j + \alpha_i,
$$

for $1 \leq i \leq m$ and $\alpha_i, \beta_{i,j}, \gamma_{i,j,k} \in \mathbb{F}$ (constant, linear, and quadratic terms). To compute these polynomials $p_i$, we use polynomial interpolation, i.e., we need the output of these polynomials for several inputs. To do so, we exploit that the private key $K = (S, P, T)$ yields the same values as the public key. Therefore, we evaluate the function $T(P(S(x)))$ for several values of $x$:

- $\eta_0 \in \mathbb{F}^n$ is the 0 vector;
• \( \eta_j \in \mathbb{F}^n : 1 \leq j \leq n \), is a vector with its \( j \)th coefficient 1, the others 0;
• \( \eta_{j,k} \in \mathbb{F}^n : 1 \leq j < k \leq n \), is a vector with its \( j \)th and \( k \)th coefficient 1, the others 0.

These \( 1 + n + n(n - 1)/2 = n(n + 1)/2 + 1 \) vectors yield the required coefficients, as we see below:

\[
\begin{align*}
T(P(S(\eta_0)))_i &= \alpha_i \\
T(P(S(\eta_j)))_i &= \alpha_i + \beta_{i,j} + \gamma_{i,j,j} \\
T(P(S(a\eta_j)))_i &= \alpha_i + a\beta_{i,j} + a^2\gamma_{i,j,j} \text{ where } a \in \mathbb{F}, a \neq 0, 1 \\
T(P(S(\eta_{j,k})))_i &= \alpha_i + \beta_{i,j} + \beta_{i,k} + \gamma_{i,j,j} + \gamma_{i,k,k} + \gamma_{i,j,k}.
\end{align*}
\]

The values for \( \alpha_i, \beta_{i,j}, \gamma_{i,j,k} \) are obtained by

\[
\begin{align*}
\alpha_i &= T(P(S(\eta_0)))_i \\
\gamma_{i,j,j} &= \frac{1}{a(a-1)}[T(P(S(a\eta_j)))_i - aT(P(S(\eta_j)))_i + (1-a)\alpha_i] \\
\beta_{i,j} &= T(P(S(\eta_j)))_i - \gamma_{i,j,j} - \alpha_i \\
\gamma_{i,j,k} &= T(P(S(\eta_{j,k})))_i - \gamma_{i,j,j} - \gamma_{i,k,k} - \beta_{i,j} - \beta_{i,k} + \alpha_i.
\end{align*}
\]

This yields the public polynomials \( p_i(x_1, \ldots, x_n) \) for \( 1 \leq i \leq m \) in the case \( \mathbb{F} \neq \text{GF}(2) \).

To adapt the algorithm to \( \mathbb{F} = \text{GF}(2) \), we observe that \( x^2 = x \) over \( \text{GF}(2) \), i.e., all squares in only one variable become linear factors instead. Therefore, we can skip all terms with \( \gamma_{i,j,j} \), i.e., all quadratic terms in \( x_j^2 \) for \( 1 \leq j \leq n \). We can also take another point of view: as there is no element \( a \in \text{GF}(2) : a \neq 0, 1 \), we could not evaluate \( T(P(S(a\eta_j)))_i \) for such an \( a \) anyway.

### 6.1.1.ii Variations

In the previous subsection, we noted how HFE can be used for the encryption of messages and for signature generation. We now move on to the description of two important variations of HFE, namely HFE- and HFEv.

#### 6.1.1.ii.a HFE-: Hiding Public Equations

Especially for signature schemes, an obvious change is to keep \( 1 < f < n \) polynomials \( p_{n-1}, \ldots, p_n \) of the public key secret. As we discussed in § 6.1.1.i.b, this is a necessity if the private polynomial \( P \) is not a surjection. However, even if the private polynomial \( P \) is a bijection, this might be a good idea as it keeps parts of the structure of the private key secret.

As for a signature scheme, keeping some polynomials \( p_{n-1}, \ldots, p_n \) secret, is also expected to enhance the overall security of an encryption scheme. However, the number of equations removed can not be too high in this case. Indeed: keeping one equation \( p_n \) secret effectively means to take \( \log_2 q \) bits of information out of \( M' := \text{HFE}(M) \). To restore these missing \( \log_2 q \) bits, it is necessary to try all \( q \) possibilities for the encrypted messages \( M'_1, \ldots, M'_q \) until the correct one is found. As the equation \( P(x) = y \) has up to \( d \) solutions (see § 6.1.1.i.a), we need to transmit some redundancy \( r \) anyway. However, to additionally compensate the loss of \( \log_2 q \) bits of information, we will need to transmit more redundancy \( r \). In addition, we need to solve up to \( q \) times the equation \( P(x) = y \) for different values of \( y \).
6.1. Hidden Field Equations in Asymmetric Cryptography

As this is the most time consuming operation in HFE, it slows down the decryption process significantly. In general, by keeping \( f \) equations secret, we lose \( \log_2 q^f \) bits of information and have to try up to \( q^f \) different possible encrypted messages \( M_1, \ldots, M_{q^f} \), so we expect decryption to be \( O(q^f) \) times slower. However, in terms of an attack, this modification is very difficult to overcome as we will see in \( \S \) 6.1.3.

6.1.1.ii.b HFEv: Adding Vinegar Variables

While HFE- changes the public key, adding vinegar variables, i.e., HFEv, changes the structure of the private polynomial \( P \). Instead of using one private polynomial \( P \), this modification allows \( q^v \) many private polynomials \( P_1, \ldots, P_{q^v} \), where \( v \in \mathbb{N} \) denotes the number of vinegar variables \( z_1, \ldots, z_v \). As the private key should still be expressible in terms of at most quadratic polynomials \( p_1, \ldots, p_n \), there is a restriction on the way these \( q^v \) many private polynomials are obtained. In essence, the quadratic coefficients (i.e., coefficients with a power of the form \( q^a + q^b \) for some \( a, b \in \mathbb{N} \)) have to be the same for all these polynomials, while the linear coefficients depend on these vinegar variables \( z_1, \ldots, z_v \) in an affine way, and the constant term depends on them in an at most quadratic way. In symbols:

\[
P_{(z_1, \ldots, z_v)}(x) := \sum_{0 \leq i,j \leq d} a_{i,j}x^{q^i+q^j} + \sum_{0 \leq k \leq d} b_k(z_1, \ldots, z_v)x^{q^k} + c(z_1, \ldots, z_v)
\]

For a signature scheme, HFEv can be implemented very easily. The vinegar variables \( (z_1, \ldots, z_v) \in R^{F^v} \) are initialised with random values. After this step, there is only one private polynomial \( P \), so the rest of the algorithm keeps unchanged. For an encryption scheme, it is not so easy to introduce this \( v \) modification as a priori any of the \( q^v \) possible settings for the vinegar variables is equally likely. As in HFE-, it is necessary to check up to \( q^v \) different equations \( P(x) = y \) — but in this case, not the value of \( y \) but the polynomial itself changes. So for a signature scheme, it is possible to have many vinegar variables, while in an encryption scheme, their number must be small.

6.1.1.ii.c Working with a Subfield \( \bar{F} \) of \( F \)

While the aim in the last section was to enhance security, we now want to concentrate on the size of the public key. To obtain a smaller public key, we can choose both the coefficients in the two affine transformations \( S \) and \( T \) and also the coefficients in the private polynomial \( P \) in a way that it is possible to express the public key \( k = (p_1, \ldots, p_n) \) in terms of a proper subfield of \( F \). For example, for \( F = \text{GF}(128) \) and the subfield \( \bar{F} = \text{GF}(2) \), we save \( \frac{6}{7} \) of the size of the public key as each coefficient in the public polynomials \( (p_1, \ldots, p_n) \) does no longer require 7 bits, but only 1 bit. However, the message space is still \( \bar{F}^n = \text{GF}(128^n) \) over the original finite field \( \text{GF}(128) \).

Unfortunately, this modification has two drawbacks. First of all, it leads to a big degree \( d \), as \( d = q^a + q^b \) for some \( a, b \in \mathbb{N} \) grows exponentially with \( q = |\bar{F}| \). Secondly, it reduces dramatically the key space for the private key. As [Pat96] points out, it is possible that an attacker could make use of this special structure of the public key. In fact, this was the case...
Gilbert and Minier [GM02] observed that we are able to express the whole Sflash system over $\mathbb{F} = \text{GF}(2)$. By the field properties, all vectors in the message space with coefficients from $\mathbb{F}$ will be mapped to vectors over $\mathbb{F}$ in the signature space. This way, it is possible to perform a brute-force search on a total of $2^{37}$ points — which is certainly much less than the claimed security level of $2^{80}$. Further exploiting the fact that Sflash is a bijection, they are able to recover the private key. The attack complexity is about $|\mathbb{F}|^n$, i.e., $2^{37}$ here.

A key point in their attack is the fact that Sflash has a bijective structure. As we saw in sections 6.1.1.i.a and 6.1.1.i.b, this is not the case for HFE. Therefore, using coefficients from a subfield may not be dangerous for HFE. However, there are two concerns. First, having a ground field $\mathbb{F} \neq \text{GF}(2)$ leads to a rather large degree as $d = O(|\mathbb{F}|^a)$ for $a \in \mathbb{N}$. This leads to rather slow decryption / signature generation (see above). Second, we will obtain much less coefficients on the private polynomial $P$ if we use a ground field $\mathbb{F} \neq \text{GF}(2)$. In the light of the recent attacks of Faugère and Joux (cf § 6.1.3.ii), this is a dangerous situation. Therefore we conclude that this modification may be save for HFE, but it is certainly not of practical interest.

6.1.1.ii.d Modifications Revisited It follows from this section that HFE is a very flexible scheme which allows many modifications. Due to space limitations in this section, we were not able to describe all modifications for HFE known so far, e.g., HFE+ or more than one branch. They can be found in [Pat96] (cf [Wol02] for a more detailed explanation). Thus HFE can be adapted to different application domains and also react on different attacks by slightly changing its overall structure. In the remainder of this section we go briefly through these different variations on HFE and determine their use for encryption and signature.

For a signature scheme, HFE- and HFEv are certainly very useful as they do not slow down signature generation and enhance the overall security of HFE. In contrast, both HFE with more branches and HFE using a subfield of $\mathbb{F}$ for the public key, seem to lead to a less secure scheme. On the other hand, both modifications change HFE in a very desirable way: the first leads to a speed-up for signature generation while the second yields a smaller public key. All in all, combining the $v$ modification with HFE- can be used to construct both fast and secure digital signature schemes (see § 6.1.2).

For encryption, the situation seems to be worse. Both HFE- and HFEv will lead to a rather slow decryption process so neither too many equations can be removed (HFE-) nor too many variables added (HFEv). For other modifications, such as HFE+, the situation is better. In this case, the decryption step takes the same amount of time, however, the public key has more equations than variables. As solving over-defined equations in many variables looks sub-exponential (cf [Cou01]), it seems to be a good idea to have nearly the same number of equations as variables. A special choice for the private polynomial $P$, e.g., a bijection, is not secure [Pat96, § 7.3]. And combining this with HFE- does not yield results either, as we cannot remove too many equations from an encryption scheme. Having two or more branches for the private polynomial $P$ or using a subfield of $\mathbb{F}$ seems to have no special effect for an encryption scheme, so the same caution is necessary as for a signature scheme.
6.1.2 Quartz

In § 6.1.1.ii, we looked at two important modifications of HFE, namely HFE- and HFEv. In this section, we will see how they can be combined to obtain a practical signature scheme, namely Quartz. It was submitted to NESSIE [NES] but was not accepted. The purpose of this section is to describe why it failed.

6.1.2.1 Historical Note

The design goal of Quartz was not only to withstand all known attacks but also to have good chances to withstand future attacks as well. So the parameters in Quartz have been chosen rather conservatively, which results in a rather long signature time, namely 10s on average on a Pentium II 500 MHz [CGP01]. As we know now (March 2004), the choice of parameters was not conservative enough. We will discuss this point in more detail in § 6.1.3. When not stated otherwise, this section is based on [CGP01] and describes the second, reversed version of Quartz. The changes made from the first version (cf [CGP00a]) to the second version of Quartz are not due to security problems. On the one hand, they were made to speed up the whole algorithm without jeopardising its security. On the other hand, they allow a security proof for Quartz.

Table 4: Parameter for Quartz

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Quartz</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q = \lceil \sqrt{\mathbb{F}} \rceil$</td>
<td>2</td>
</tr>
<tr>
<td>$n = \delta_i(t)$</td>
<td>103</td>
</tr>
<tr>
<td>transformation $S$</td>
<td>$\mathbb{F}^{103} \rightarrow \mathbb{F}^{103}$</td>
</tr>
<tr>
<td>transformation $T$</td>
<td>$\mathbb{F}^{103} \rightarrow \mathbb{F}^{103}$</td>
</tr>
<tr>
<td>$l$ (equations removed)</td>
<td>3</td>
</tr>
<tr>
<td>$v$ (vinegar variables)</td>
<td>4</td>
</tr>
<tr>
<td>$m$ (equations)</td>
<td>100</td>
</tr>
<tr>
<td>$n$ (variables)</td>
<td>107</td>
</tr>
<tr>
<td>$d$ (degree)</td>
<td>129</td>
</tr>
<tr>
<td>Signature Length</td>
<td>128 bits</td>
</tr>
<tr>
<td>Private Key Size</td>
<td>3 kB</td>
</tr>
<tr>
<td>Public Key Size</td>
<td>71 kB</td>
</tr>
</tbody>
</table>

6.1.2.ii System Parameters

As we see in Table 4, the signature length (128 bits) is 21 bits larger than expected: as the extension field $\mathbb{F}$ has dimension 103 while 4 vinegar variables are added, we would expect a signature length of 107 bits. The reason for this difference lies in the fact that Quartz uses a so-called “Feistel-Patarin-Network” (sometimes also denoted “Patarin-Chained-Construction”) to compute the signature. Within this network, the HFE algorithm is called four times to compute a signature, i.e., this involves solving the underlying HFE problem four times (cf Table 6 and § 6.1.2.iii.b). This way, we need to add 4 times 7 bits to the number of public equations.
6. Multivariate Systems

6.1.2.iii System Description

To deal with the different security features of Quartz, we have to look at them and try to deduce if they enhance or jeopardise the security of Quartz. First of all, the private polynomial $P$ has full coefficients, i.e., it has non-trivial coefficients from $E$ and also all possible coefficients, i.e., every power which has Hamming weight two or lower. Together with the vinegar variables (denoted $z_1, \ldots, z_4$), the private polynomial $P$ of Quartz can be expressed as:

$$P(z_1, \ldots, z_4)(x) := \sum_{0 \leq i, j \leq 7} a_{i,j} x^{q^i+q^j} + \sum_{0 \leq k \leq 7} b_k(z_1, \ldots, z_v) x^{q^k} + c(z_1, \ldots, z_v)$$

for $a_{i,j} \in E$, $b_k(z_1, \ldots, z_v)$ are affine in $(z_1, \ldots, z_v)$, and $c(z_1, \ldots, z_v)$ is at most quadratic in $(z_1, \ldots, z_v)$.

As the polynomial has all coefficients and the degree is rather high, Quartz withstood at design time all known attacks up to a complexity level of $2^{80}$ — this level was requested for signature algorithms in NESSIE. This is also true if there is no $v$ modification. In fact, the degree is very high as in 2000 a degree of 25–33 was estimated to be high enough. In addition, Quartz is a HFEv rather than a “basic” HFE scheme. This modification is expected to further enhance the security of Quartz. Moreover, Quartz is also a HFE-scheme with 3 equations kept secret. As Quartz uses a very general polynomial $P$ and also the $v$ modification, the attacks known against basic HFE do not apply against Quartz. So removing only three equations from the public key seems to be sufficient for Quartz and is expected to enhance its overall security. We call the parts of Quartz discussed above the “HFE”-step, i.e., HFE($x$) := $T(P(S(x)))$ and its inverse HFE$^{-1}(y)$ := $S^{-1}(P^{-1}(T^{-1}(y)))$.

Although the HFE-step itself looks quite secure, there is an obvious attack using the birthday paradox: by computing $2^{50}$ different versions of the message $M$ and by applying the public key to $2^{50}$ different values for $x_1, \ldots, x_n$ we expect to obtain a valid signature for one version of the message $M$ as the HFE step alone uses only 100 public equations. This is far less than the complexity level of $2^{80}$ required in NESSIE. To overcome this problem, Quartz combines four invocations of the HFE-step in a so-called “Feistel-Patarin network”. The key idea of this network is not to store four times a full signature (i.e., a signature of 400 bits in total) but to save only the last signature.

![Fig. 4: Overall Structure of Quartz for Signature Generation](image-url)
completely. In addition, it stores 7 bits for each of the 4 signatures computed. The reason for this lies in the fact that the HFE-step of Quartz has only 100 bits of input but a 107 bit output. These additional 7 bits compensate for this expansion. The overall structure of Quartz is shown in Fig. 4. As we see there, signature generation with Quartz requires a message $M$.

![Fig. 5: Precomputation in Quartz](image)

precomputation step (see Fig. 5) before applying the network itself (see Fig. 6). The key idea of the precomputation step is to use three calls of a 160-bit hash function (SHA-1 in Quartz, cf [FIP] for SHA-1) to “expand” a 160-bit hash (denoted $m_0$ in Fig. 5) to four 100-bit values $h_1, h_2, h_3$ and $h_4$. During this process, the original hash $m_0$ is concatenated (operator $\|\|$) with the 8 bit values $0x00$, $0x01$ and $0x02$ (C notation for the numbers 0, 1 and 2) to obtain three 168 bit values. Each of them is hashed individually using a 160 bit hash function and then concatenated. The resulting 480 bit number is truncated to 400 bits and yields four 100-bit strings. If Quartz used a hash function with a 512 bit output rather than 160 bit, the precomputation step would be obsolete. Such functions have been accepted in the NESSIE project (e.g., algorithm “Whirlpool”) and also in NIST (only the algorithm in [FIP01]). But for the complexity level of $2^{80}$, it is sufficient to use a 160 bit hash function and to expand its output to 400 bits as done in the precomputation step of Quartz.

6.1.2.iii.b Feistel-Patarin network We will now concentrate on the “Feistel-Patarin network” itself as outlined in Fig. 6. It uses the output of the precomputation step as input. We describe the first step of the network. After initialising the counter $i$ with 0, it “xors” $h_0$ with 0 to obtain the intermediate value $y$. This is certainly obsolete as $h_0 \xor 0 = h_0$. However, during the run of the algorithm, $h_1, h_2, h_3$ are “xored” with the output of the previous step, so this “xor” operation is required for symmetry of the four steps. After this initialisation, the 100-bit value $y$ is hashed together with a secret 80 bit parameter $\Delta$ to obtain the random variables $r$ (3 bit) and the vinegar variables $z$ (4 bit). Both are fed into the HFE step to obtain a valid signature of 107 bits. According to [CGP01, Sec. 5.3] and as previously noted in § 6.1.1.i.b of this section, the probability to obtain a valid signature at one attempt
is ≈ 60%. If there is no valid signature, the hash of $y \parallel \Delta$ is rehashed. This is repeated until a valid signature is obtained. The probability that there is no valid signature at all for a given message $M$ is estimated to be $\leq 2^{-183}$ and hence negligible [CGP01, Sec. 5.3]. If a valid 107 bit signature is found, the least significant 100 bits of $x$ are fed back into the network while the most significant 7 bits are stored as output $g_1$. The other three steps are similar but $h_i$ is not “xored” with 0 but with the least significant 100 bits of $x$. In the final step, these 100 bits are not fed into the network but yield output $\tilde{s}$. In each step, it is possible that there is not only one, but up to $d = 129$ different solutions for the equation $x = HFE(y \parallel r)$. The Quartz-specification states that only one is chosen, namely the one with the least hash value (bit-wise comparison without sign bit).

6.1.2.iii.c Signature Verification To verify the validity of a signature, this network is reversed. As the public key consists of 100 polynomials $p_1, \ldots, p_{100}$ in 107 input variables $x_1, \ldots, x_{107}$, the 7 bit values $g_1, \ldots, g_4$ are used to obtain 107 bits input for the public key during each run. In addition, as the four 100 bit values $h_1, \ldots, h_4$ are “xored” each time, a signature is only valid if the overall output of this scheme is 0. In this case the signature is accepted.
6.1.4 Conclusions The Feistel-Patarin network is certainly a rather complicated security feature. However, as each signature depends on a 400 bit input (which is obtained from a 160 bit hash value), it seemed to be a rather strong signature system. Moreover, as Quartz uses the 160-bit hash function as a kind of cryptographically secure random number generator, it is deterministic, so each message has always the same signature (for the same private key $K$). In the original specification of HFE as a signature scheme it seemed to be necessary to use “real” randomness to obtain valid signatures. As real randomness often is a problem (e.g., in a stand-alone server without user interaction), this deterministic version makes it possible to use Quartz in more application domains.

As we will see in the attack section, Quartz as proposed in [CGP01] is no longer considered to be secure. Consequently, it has not been recommended by NESSIE [PBO+03].

6.1.3 Attacks

In this section, we give a brief overview of recent attacks against HFE. Due to space limitations, we can only sketch the corresponding articles. For a more detailed but partly outdated analysis, we refer to [Pat96].

6.1.3.i Kipnis-Shamir: Recover the Private Key In [KS99], Kipnis and Shamir show how to recover the private key of HFE from the system of public equations. The key point of this attack is to express the private key (i.e., polynomials over the finite field $F$) as sparse univariate polynomials over the extension field $E$. In addition, they observe that the special choice of the private polynomial $P$ in HFE gives rise to a matrix equation with very small rank (e.g., rank 13 for a $100 \times 100$ matrix). In [Cou01, § 8], their attack is improved and has now a workload of $\left(\frac{n}{\operatorname{Rank}_P}\right)^\omega = \mathcal{O}\left(n^{\log_q d + O(1)}\right)$. In this formula, $\operatorname{Rank}_P$ is the rank of the private polynomial $P$ in matrix form over the extension field $E$, $d$ its degree as a polynomial over $E$, and $\omega \approx 2.7$ the workload of solving linear equations. The attack is only applicable against basic HFE, i.e., fails for all its variations. On the other hand, it is the only attack known so far which can recover the private key of HFE.

In their paper, Kipnis and Shamir also introduce the “relinarization” technique which can solve quadratic equations with about $0.1 n^2$ linearly independent equations in $n$ variables. For the traditional linearization technique, we need about $0.5 n^2$ many equations. This technique has been improved in [CKPS00].

6.1.3.ii Faugère: Fast Gröbner Bases In 2002, Faugère reported to have broken the HFE-Challenge I in 96 hours [Fau02a]. Since then, his attacks have been improved and in 2003, Faugère and Joux published joint work on the security of HFE [FJ03] (cf [Fau03] for a more technical version). In a nutshell, their attack uses a fast algorithm to compute the Gröbner basis of a system of polynomial equations. By theoretical and empirical studies they show that basic HFE is polynomial for a fixed degree. The attack-complexity for different degrees is shown in Table 5.

For HFEv, Faugère and Joux outline in [FJ03, § 4.1] that the cryptanalysis is not more difficult in this case. But for HFE-, they get a higher workload. For the original Quartz-
Table 5: Attack complexity against basic HFE for different degrees $d$

<table>
<thead>
<tr>
<th>Degree $d$</th>
<th>$16 &lt; d \leq 128$</th>
<th>$128 &lt; d \leq 512$</th>
<th>$512 &gt; d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack (asymptotical)</td>
<td>$O(n^8)$</td>
<td>$O(n^{10})$</td>
<td>$\geq O(n^{12})$</td>
</tr>
<tr>
<td>Attack (for $n = 103$)</td>
<td>$\approx 2^{54}$</td>
<td>$\approx 2^{66}$</td>
<td>$\approx 2^{80}$</td>
</tr>
</tbody>
</table>

scheme, they establish a workload of $\approx 2^{62}$ — exploiting some further properties of their attack. However, these additional improvements are not within the scope of this section.

Using the estimations of [FJ03, § 4.1, 5.2–5.4] on Quartz, we establish that a degree of 129 and 7 equations removed (thus, without the modification HFEv) has an attack complexity of $\approx 2^{86}$. The corresponding “Quartz-7m”-scheme is therefore secure again. In fact, a similar result has been achieved 2002 in [CDF03] by increasing the degree $d$ of the private polynomial to 257. However, this estimation was only based on [Fau02a]. In the light of the article [FJ03] it turns out to be inaccurate.

6.1.3.iii Secure Versions of Quartz At present, we see two possibilities to obtain a secure version of the Quartz signature algorithm which is able to withstand the attack from [FJ03]. The first uses a degree of 513 for the public polynomial and keeps the other parameters unchanged. We call this version Quartz-513d and expect an attack complexity of $\approx 2^{82}$. However, due to the very large degree of the private polynomial, we do not expect this version to be of practical interest.

Therefore, we concentrate on the version already outlined: replace the 4 vinegar variables by removing 4 equations. The corresponding system has still the same signature size as Quartz but an estimated attack complexity of $\approx 2^{86}$. It therefore meets the NESSIE-requirements of $2^{80}$ TDES-computations.

Although these versions (cf Table 6) are secure against the recent attack from Faugère and Joux, we argue to be cautious as they have not been independently studied by other researchers. It is therefore well possible that they carry unnoticed weaknesses.

6.1.4 Conclusions

In this section, we outlined the structure of the Hidden Field Equations system (HFE) from Patarin and described two important variations, namely HFE- and HFEv. Using these variations, we described the original Quartz signature scheme. In the light of recent attacks, this scheme can no longer be considered to be secure. Therefore, we outlined how the internal structure of Quartz (i.e., its private key) can be changed in order to counter these attacks. In particular, neither the rather short signature size of 128 bits nor the signature-verification process of Quartz is affected by our proposed changes (cf Table 6).

This shows that Hidden Field Equations are still an interesting research topic which allow exceptional short signature sizes. However, before using HFE in practice, further research is needed to fully understand its security.
6.2. Enhanced TTM Multivariate Signature Schemes

In the last two decades, several public key schemes using multivariate quadratic equations (MQ) have been proposed, e.g., [MI88, Pat96, KPG99]. All of them use the fact that the MQ-problem, i.e., finding a solution \( x \in \mathbb{F}^n \) for a given system of \( m \) polynomial equations in \( n \) variables each

\[
\begin{align*}
y_1 &= p_1(x_1, \ldots, x_n) \\
y_2 &= p_2(x_1, \ldots, x_n) \\
&\vdots \\
y_m &= p_m(x_1, \ldots, x_n),
\end{align*}
\]

for given \( y_1, \ldots, y_m \in \mathbb{F} \) and unknown \( x_1, \ldots, x_n \) is difficult, namely \( \mathcal{NP} \)-complete (cf [GJ79, p. 251] and [PG97b, App.] for a detailed proof)). In the above system of equations, the polynomials \( p_i \) are of the form

\[
p_i(x_1, \ldots, x_n) = \gamma_{i,j,k} x_j x_k + \beta_{i,j} x_j + \alpha_i,
\]

for \( 1 \leq i \leq m; 1 \leq j \leq k \leq n \) and \( \alpha_i, \beta_{i,j}, \gamma_{i,j,k} \in \mathbb{F} \) (constant, linear, and quadratic terms).

### Tame-like schemes

One class of schemes using the MQ-problem with an embedded trapdoor uses Moh’s “Tame-transformation” method [Moh99]. As this construction only uses operations over small finite fields \( \mathbb{F} = \text{GF}(q) \) — typically, \( q=256 \) — they are very fast both for signature verification and generation. In addition, they are suitable for a smartcard environment as they do not need a cryptographic co-processor. Since the attack from Goubin et al. [GC00], the design of such schemes has drastically been improved.

In this section, we concentrate on the latest variation on Tame-signature schemes, due to Yang and Cheng [YC04]. We call them Enhanced TTM or enTTM for short. In a nutshell,
their construction makes use of two affine transformations $S, T$ and some central equations $\mathcal{P}'$. We have $S \in \text{AGL}_n(\mathbb{F})$ and $T \in \text{AGL}_m(\mathbb{F})$. Over finite fields, affine transformations can be decomposed in an invertible matrix $M_S \in \mathbb{F}^{n \times n}$ (resp. $M_T \in \mathbb{F}^{m \times m}$) and a vector $v_s \in \mathbb{F}^n$ (resp. $v_t \in \mathbb{F}^m$). This way, the transformation $S : \mathbb{F}^n \to \mathbb{F}^n$ can be written as $S(x) = M_S x + v_s$. Moreover, let

$$\mathcal{P}' := \begin{pmatrix} p'_1(x_1', \ldots, x_n') \\ p'_2(x_1', \ldots, x_n') \\ \vdots \\ p'_m(x_1', \ldots, x_n') \end{pmatrix}$$

be the vector of “central polynomials”. The public key is then computed as $\mathcal{P} := T \circ \mathcal{P}' \circ S$. To withstand all known attacks, Yang and Chen use special constructions for the central equations $\mathcal{P}'$ (cf sections 6.2.1.ii and 6.2.1.iii). In the remainder of the present section, we describe this construction.

### 6.2.1 Enhanced TTM

#### 6.2.1.i Design Criteria

In [YC04, § 4.1], the authors develop design criteria which have to be met by Tame-like systems in order to withstand all known attacks for a security level $\mathcal{C}$. Here, we denote $q = |\mathbb{F}|$, i.e., the number of elements of the finite field $\mathbb{F}$.

1. Each equation should have as many cross-terms with no repeated indices as possible.

2. Almost all linear combinations of central equations should result in quadratic forms of higher rank, only a relatively small number can have equal rank. In $k$ linear combinations of central equations share a minimal rank $r = 2l$, then we need

   $$q^r (m^2(n/2 - m/6) + mn^2)/k \leq \mathcal{C}$$

   The parameter $l$ indicates the number of cross-terms per equation (see below).

3. If the minimum number of appearances is $u$ in the central equations for any variable $x_i$, then

   $$q^u (un^2 + n^3/6) \leq \mathcal{C}$$

4. Let $A$ be a set of $h$ indices $0 \leq i < n$ such that every cross-term in the central equation has at least one index in $A$. To counter attacks which arise against UOV [KPG99], we need $h > n/2$. To defeat XL [CKPS00] and Gröbner attacks (e.g., [Fau02b]), we need $h < m$ and derive $\hat{h} := m - h - 1$ as resistance against these attacks.

The reasoning behind these criteria can be found in [YC04].
6.2.1.ii Enhanced TTS scheme  In [YC04, § 4.2], the authors introduce a Tame-like signature system “Enhanced TTS” using the following construction for the central equations:

\[ p_1 = x_1^* + \sum_{j=1}^{7} \gamma_{i,j} x_j^* (x_{8+i+j} \mod 9), \text{ for } i = 8 \ldots 16; \]

\[ p_{17} = \gamma_{17,1} x_1^* x_6^* + \gamma_{17,2} x_2^* x_5^* + \gamma_{17,3} x_3^* x_4^* + \gamma_{17,4} x_9^* x_{16}^* \]
\[ + \gamma_{17,5} x_{10}^* x_{15}^* + \gamma_{17,6} x_{11}^* x_{14}^* + \gamma_{17,7} x_{12}^* x_{13}^*; \]

\[ p_{18} = \gamma_{18,1} x_1^* x_7^* + \gamma_{18,2} x_2^* x_6^* + \gamma_{18,3} x_3^* x_5^* + \gamma_{18,4} x_{10}^* x_{17}^* \]
\[ + \gamma_{18,5} x_{11}^* x_{16}^* + \gamma_{18,6} x_{12}^* x_{15}^* + \gamma_{18,7} x_{13}^* x_{14}^*; \]

\[ p_i = x_i^* + \gamma_{i,0} x_{i-11}^* x_{i-9}^* + \sum_{j=19}^{27} \gamma_{i,j-18} x_{2(i-j)}^* x_j^* \]
\[ + \sum_{j=i+1}^{27} \gamma_{i,j-18} x_{i-j+19} x_j^*, \text{ for } i = 19 \ldots 27; \]

for \( \gamma_{i,j} \in \mathbb{F} \). The system is designed for a security level \( \mathcal{C} = 2^{80} \).

6.2.1.ii.a Signature Generation  Starting with a vector \( y = (y_8, \ldots, y_{27}) \in \mathbb{F}^{20} \), i.e., 160 bit for \( \mathbb{F} = \text{GF}(256) \), we first apply the inverse transformation of \( T \) to obtain \( y' = (y_8', \ldots, y_{27}') \). In symbols:

\[ y' = T^{-1}(y) \]

where \( T(y) = M_T y + v_t \) for an invertible matrix \( M_T \in \mathbb{F}^{20 \times 20} \) and a vector \( v_t \in \mathbb{F}^{20} \).

In the central equations \( P' \), the user assigns the variables \( x_1, \ldots, x_7 \in \mathbb{F} \) with random values. This leads to a linear system of equations in the unknowns \( x_8', \ldots, x_{16}' \) — which can be solved, e.g., by Gaussian elimination. In case these is no solution (probability 1/256), the user tries with another random-vector \( (x_1', \ldots, x_7') \in \mathbb{F}^7 \).

For \( x_17' \) and \( x_18' \), the system is only linear in the equations \( p_{17}' \) and \( p_{18}' \), respectively, i.e., the values of these two variables can be computed by evaluating the remainder of \( p_{17}' \) and \( p_{18}' \). Finally, a random value for \( x_0' \in \mathbb{F} \) is assigned. At most 9 possibles values do not allow a solution for \( x_{19}', \ldots, x_{27}' \in \mathbb{F} \). If the first attempt to solve this system fails, use another value for \( x_0' \) and start again.

Finally, the user applies the inverse of the affine transformation \( S \) to obtain a valid signature \( x = S^{-1}(x') \).

6.2.1.ii.b Signature Verification  To verify if the vector \( x \in \mathbb{F}^{28} \) is a valid signature for the vector \( y \in \mathbb{F}^{20} \), the user checks if the system of equations

\[
\begin{cases}
  y_1 = p_1(x_1,\ldots,x_{28}) \\
  y_2 = p_2(x_1,\ldots,x_{28}) \\
  \vdots \\
  y_m = p_m(x_1,\ldots,x_{28})
\end{cases}
\]

is satisfied for the public key polynomials \( p_1, \ldots, p_{20} \). If not, he rejects the signature.
6.2.1.ii.c Key Generation  

To generate the public key from a given private key, we use polynomial interpolation for multivariate polynomials.

We want to obtain polynomials over $\mathbb{F}$ as the public key which have the form

$$p_i(x_1, \ldots, x_n) = \gamma_{i,j,k}x_jx_k + \beta_{i,j}x_j + \alpha_i,$$

for $1 \leq i, j, k \leq n$ and some $\alpha_i, \beta_{i,j}, \gamma_{i,j,k} \in \mathbb{F}$. To compute these polynomials $p_i$, we use polynomial interpolation, i.e., we need the output of these polynomials for several inputs. To do so, we exploit that the private key $K = (S, \mathcal{P}', T)$ yields the same values as the public key. Therefore, we evaluate the function $T(\mathcal{P}'(S(x)))$ for several values of $x$: 

- $\eta_0 \in \mathbb{F}^n$ is the 0 vector
- $\eta_j \in \mathbb{F}^n : 1 \leq j \leq n$, is a vector with its $j^{th}$ coefficient 1, the others 0
- $\eta_{j,k} \in \mathbb{F}^n : 1 \leq j < k \leq n$, is a vector with its $j^{th}$ and $k^{th}$ coefficient 1, the others 0

These $1 + n + \frac{1}{2}n(n + 1) = \frac{1}{2}(n + 1)(n + 2)$ vectors yield the required coefficients, as we see below:

$$
T(\mathcal{P}'(S(\eta_0)))_i = \alpha_i \\
T(\mathcal{P}'(S(\eta_j)))_i = \alpha_i + \beta_{i,j} + \gamma_{i,j,j} \\
T(\mathcal{P}'(S(\eta_{j,k})))_i = \alpha_i + \beta_{i,j} + \beta_{i,k} + \gamma_{i,j,j} + \gamma_{i,k,k} + \gamma_{i,j,k}
$$

The values for $\alpha_i, \beta_{i,j}, \gamma_{i,j,k}$ are obtained by

$$
\alpha_i := T(\mathcal{P}'(S(\eta_0)))_i \\
\gamma_{i,j,j} := \frac{1}{a(a-1)} [T(\mathcal{P}'(S(\eta_{j,k})))_i - aT(\mathcal{P}'(S(\eta_j)))_i + (1-a)\alpha_i] \\
\beta_{i,j} := (T(\mathcal{P}'(S(\eta_j)))_i - \gamma_{i,j,j} - \alpha_i \\
\gamma_{i,j,k} := (T(\mathcal{P}'(S(\eta_{j,k})))_i - \gamma_{i,j,j} - \gamma_{i,k,k} - \beta_{i,j} - \beta_{i,k} - \alpha_i
$$

This yields the public polynomials for $\mathbb{F} \neq \text{GF}(2)$.

A broader discussion of key generation for multivariate quadratic schemes — also including the case $\mathbb{F} = \text{GF}(2)$ — can be found in [Wol03].

6.2.1.iii Scaled-Up Schemes  

In [YC04, § 4.3], Yang and Chen give several constructions for “Scaled-Up” Enhanced TTS schemes. For all these schemes, only the central equations $\mathcal{P}'$ change.

6.2.1.iii.a Odd Construction  

We have a parameter $l \geq 4$ and $(m, n) = (4l, 6l - 2)$ which translates to security parameters $(u, r, h) = (2l - 1, 4l - 6, l - 1)$. The name “odd
construction” is due to the fact that the parameter $u$ is odd.

\[
\begin{align*}
p'_i & := x'_i + \sum_{j=1}^{2l-3} \gamma'_{i,j} x'_j x'_{2l-2+(i+j+1 \mod 2l-1)}, \text{ for } 2l - 2 \leq i \leq 4l - 4; \\
p'_i & := x'_i + \sum_{j=1}^{l-2} \gamma'_{i,j} x'_{i+j-(4l-3)} x'_{i-j-2l} \\
& \quad + \sum_{j=l-1}^{2l-3} \gamma'_{i,j} x'_{i+j-3l+6} x'_{i+l-5-j}, \text{ for } 4l - 3 \leq i \leq 4l - 2; \\
p'_i & := x'_i + \sum_{j=1}^{i} \gamma'_{i,j} x'_{i-(4l-2)} x'_{2(i-j)} x'_j \\
& \quad + \sum_{j=i+1}^{6l-3} \gamma'_{i,j} x'_{4l-1+i-j} x'_j, \text{ for } 4l - 1 \leq i \leq 6l - 3,
\end{align*}
\]

for $\gamma'_{i,j} \in \mathbb{F}$. The operations key generation, signature generation, and signature verification follow the ideas outlined above.

6.2.1.i iii.b Even Construction Here, we have a parameter $l \geq 5$ and $(m, n) = (4l, 6l - 4)$ which translates to security parameters $(u, r, h) = (2l - 2, 4l - 10, l - 1)$. The name “even construction” is due to the fact that the parameter $u$ is even.

\[
\begin{align*}
p'_i & := x'_i + \sum_{j=1}^{2l-5} \gamma'_{i,j} x'_j x'_{2l-4+(i+j+1 \mod 2l-2)}, \text{ for } 2l - 4 \leq i \leq 4l - 7; \\
p'_i & := x'_i + \sum_{j=1}^{l-4} \gamma'_{i,j} x'_{i+j-(4l-6)} x'_{i-j-(2l+1)} \\
& \quad + \sum_{j=l-3}^{2l-5} \gamma'_{i,j} x'_{i+j-3l+5} x'_{i+l-4-j}, \text{ for } 4l - 6 \leq i \leq 4l - 3; \\
p'_i & := x'_i + \gamma'_{i,0} x'_{-2(l+1)} x'_{2(l-1)} + \sum_{j=4l-2}^{i} \gamma'_{i,j} x'_{2(i-j)} x'_j \\
& \quad + \sum_{j=i+1}^{6l-5} \gamma'_{i,j} x'_{4l-2+i-j} x'_j, \text{ for } 4l - 2 \leq i \leq 6l - 5.
\end{align*}
\]

for $\gamma'_{i,j} \in \mathbb{F}$. The operations key generation, signature generation, and signature verification follow the ideas outlined above.

6.2.1.iv Attacks The main attacks against enTTM schemes are rank attacks. These attacks have been explained in great detail in [YC04], we therefore refer to this article.
In addition, it is possible to attack any kind of multivariate scheme using Grobner bases or the XL algorithm. enTTM has been designed to withstand these attacks.

The authors of [YC04] point out that there may be a vulnerability in enTTM which is similar to the kernel attack in UOV. However, this point needs some more research as this connection is not clear yet.

6.2.2 Conclusion

In a nutshell, Enhanced TTM-signatures are a very interesting approach solving signature problems using multivariate quadratic equations. Due to their construction, they are suitable for the smartcard environment.

The big disadvantage of enTTM is the fact that it is rather new — it dates from this year. However, as it is a descendent of the TTM scheme of Moh from 1999, it can build on the cryptanalysis already done against this proposal.

In addition, they allow rather short key sizes — if compared to other multivariate quadratic systems. For example, Sflash\textsuperscript{v3} is already well beyond the 100 kB limit for its public key [CGP03]. In comparison, we have 8.6 kB here for enTTM with 20 equations and 28 variables over GF(256). The corresponding private key is in the range of 1.4 kB.

On a Pentium III, the authors of [YC04] obtain a signing time of 51\textmu s, a verification time of 110\textmu s and a key generation time of 15.1ms.

6.3 Unbalanced Oil and Vinegar Multivariate Signature Scheme

The UOV scheme is another example of schemes using multivariate quadratic equations (MQ) as their public key. All of them use the fact that the MQ-problem, i.e., finding a solution $x \in \mathbb{F}^n$ for a given system of $m$ polynomial equations in $n$ variables each

\[
\begin{align*}
  y_1 &= p_1(x_1, \ldots, x_n) \\
  y_2 &= p_2(x_1, \ldots, x_n) \\
  &\vdots \\
  y_m &= p_m(x_1, \ldots, x_n),
\end{align*}
\]

for given $y_1, \ldots, y_m \in \mathbb{F}$ and unknown $x_1, \ldots, x_n$ is difficult, namely \textup{NP}-complete (cf [GJ79, p. 251] and [PG97b, App.] for a detailed proof)). In the above system of equations, the polynomials $p_i$ are of the form

\[
p_i(x_1, \ldots, x_n) = \gamma_i,j,k x_j x_k + \beta_i,j x_j + \alpha_i,
\]

for $1 \leq i \leq m; 1 \leq j \leq k \leq n$ and $\alpha_i, \beta_i,j, \gamma_i,j,k \in \mathbb{F}$ (constant, linear, and quadratic terms).

6.3.1 The Unbalanced Oil and Vinegar Schemes

Here we give a description of the unbalanced oil and vinegar scheme as proposed by Patarin et al. [KPG99]. It is an extension of the Oil and Vinegar scheme of Patarin from 1998. The scheme consists of the following building blocks:
• An \((m, n + v)\) system of \(m\) quadratic polynomials \(p_i'\) in \(n + v\) variables: \(\mathcal{P}' = (p'_1(x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+v}), \ldots, p'_m(x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+v}))\).
  
  Moreover, each polynomial \(p_i'\) for \(1 \leq i \leq m\) can be represented as

\[
p_i' = x_{n+i}Lin_{i,n+1}(x_1, \ldots, x_{n+v}) + \cdots + x_{n+v}Lin_{i,n+1}(x_1, \ldots, x_{n+v}) + Af_i((x_1, \ldots, x_{n+v}),
\]

where \(Lin_{i,j}\) are different randomly chosen linear transformations in the variables \(x_1, \ldots, x_{n+v}\) for \(1 \leq i \leq m, n+1 \leq j \leq n + v,\) and \(Af_i\) are different affine transformations in the variables \(x_1, \ldots, x_{n+v}\) for \(1 \leq i \leq m.\)

Remark that each polynomial is affine when restricted to the subspace \(V = \langle e_{n+1}, \ldots, e_{n+v} \rangle,\) where \(e_i\) is the all zero vector with a one on position \(i,\) for \(i \in \{1, \ldots, n + v\},\) as the terms \(\sum_{1 \leq j, k \leq n} \alpha_{ijk} x_j x_k\) does not appear in the equation of \(p_i'.\)

This explains why the variables \(x_1, \ldots, x_n\) are called the oil variables and the variables \(x_{n+1}, \ldots, x_{n+v}\) the vinegar variables. In fact, this property is the "heart" of the algorithm.

• An affine transformations \(S \in \text{GL}_{n+v}(\mathbb{F})\) which is represented by an \((n + v) \times (n + v)\) invertible matrix \(M_S \in \mathbb{F}^{(n+v) \times (n+v)}\) and a constant \(m_S \in \mathbb{F}^{n+v}.

• An \((m, n + v)\) system of \(m\) quadratic polynomials \(p_i\) in \(n + v\) variables:

\[
\mathcal{P} = (p_1(x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+v}), \ldots, p_m(x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+v})).
\]

The system \(\mathcal{P}\) represents the composition of the linear transformation \(S\) and the system \(\mathcal{P}', i.e., \mathcal{P} = \mathcal{P}' \circ S.\)

We assume that the ground-field \(\mathbb{F}, a finite field of order q = |\mathbb{F}|,\) together with the parameters \(n\) and \(v\) are publicly known. The public key \(K\) is the system \(\mathcal{P}.\) The secret or private key \(K'\) is the pair \((S, \mathcal{P}')\). The system was originally defined for parameters \(v = n\) and called the oil and vinegar scheme. After a successful attack of Kipnis and Shamir [KS98], the scheme was extended to the unbalanced oil and vinegar scheme where \(v > n\) [KPG99].

6.3.1 Security Parameters

There are three security parameters:

• The ground field \(\mathbb{F}\) with \(q = |\mathbb{F}|\) and characteristic \(\text{char}(\mathbb{F}) = p\) a prime. Typical values are \(q=16\) or \(q=256.\)

• The number \(n\) of variables. It should be high enough to avoid brute-force search. In addition, it should be prime in order to avoid the existence of subspaces.

• The dimension \(v\) of the subspace \(V.\)

6.3.1 Key-size

6.3.1.ii.a Public Key Size

The public key size depends on the number of coefficients for the public polynomials. Counting the constant, linear and quadratic terms, we obtain the
following formula \( \tau(\tilde{n}) \) for the total number of terms of a public polynomial in \( \tilde{n} = n + v \) variables:

\[
\tau(\tilde{n}) = \begin{cases} 
1 + \tilde{n} + \frac{\tilde{n}(\tilde{n}-1)}{2} = 1 + \frac{\tilde{n}(\tilde{n}+1)}{2} & \text{if } \mathbb{F} = GF(2) \\
1 + \tilde{n} + \frac{\tilde{n}(\tilde{n}+1)}{2} = 1 + \frac{(\tilde{n}+1)(\tilde{n}+2)}{2} & \text{otherwise}
\end{cases}
\]

Thus, the public key size is equal to \( m \cdot \tau(\tilde{n}) \), which can be approximated by \( O((n + v)^2 m) \).

6.3.1.ii. Private Key Size  The private key size depends on the number of coefficients in the affine transformation and the private polynomial \( P' \). The number of coefficients in the affine transformation \( S \) is given by \((n + v)^2 + (n + v + 1)\). The number \( \tau(n + v, v) \) of coefficients of a secret polynomial \( p'_i \) is determined by the dimension \( v \) of the subspace and is given by:

\[
\tau(n + v, v) = \begin{cases} 
1 + \frac{(2n+v+1)v}{4} & \text{if } \mathbb{F} = GF(2) \\
\frac{(2n+v+2)(1+v)}{2} & \text{otherwise}
\end{cases}
\]

This number is obtained by computing the sum of the affine terms, which is equal to \( n + v + 1 \) and the sum of the quadratic terms, which is equal to \( \sum_{i=0}^{v-1} (n + v - i) \), because of the special structure of the polynomials as described above. If \( \mathbb{F} = GF(2) \), the sum starts with 1 and if \( \mathbb{F} \neq GF(2) \) the sum starts with 0. This difference is due to the fact that in \( \mathbb{F} = GF(2) \), we have \( x^2 = x \) for any \( x \in \mathbb{F} \).

The total size of the private key size is therefore \((n + v)^2 + (n + v + 1) + m^2 + m\tau(n + v, v)\), which can be approximated by \( O((n + v)^2 + mv(n + v)) \).

6.3.1.iii Signatures  The (unbalanced) oil and vinegar scheme is designed for a signature scheme. It is not suitable for encryption because of the parameter \( v \), which should be chosen too high to receive enough security. We perform the following steps for signing a message \( m \in \mathbb{F}^n \):

1. Assign random variables \( a_{n+1}, \ldots, a_{n+v} \) to all the vinegar variables.

2. After substituting the random values, the system \( m = P'(a) \) becomes linear. Solve this linear system for the remaining \( n \) variables \( a_1, \ldots, a_n \) of \( a \) by Gaussian elimination. If the linear system is singular, return to the first step and try with new random vinegar variables.

3. Map the solution \( a \) to the signature \( x \) by \( x = S^{-1}(a) \).

Verifying the signature \( x \in \mathbb{F}^{n+v} \) is just the evaluation of \( x \) by the public system \( P \). An attacker that wants to sign a given message \( m = (m_1, \ldots, m_m) \), needs to solve the system:

\[
m_1 = p_1(x_1, \ldots, x_n + v) \\
\vdots \\
m_m = p_m(x_1, \ldots, x_n + v)
\]

In general, this is an \( \mathcal{MQ} \) problem and therefore difficult to solve.
6.3.1.iv UOV/: Restricting to a Subfield If we choose the coefficients of the two linear transformations \( S, T \), and the private key \( P_0 \) in a way such that it is possible to express the public key \( K = (p_1, \ldots, p_n) \) in terms of a proper subfield of \( \mathbb{F} \), we obtain a smaller public key. For example, for the field \( \mathbb{F} = GF(256) \) and the subfield \( \mathbb{F} = GF(2) \), the size of the public key decreases by a factor of 8.

The drawback is that the key space for the private key reduces enormously. In [GSB01], a successful attack is applied to an earlier version of Sflash [CGP03], which used this trick to decrease its public key size.

6.3.2 Cryptanalysis

In this section, we first recall three main classes of cryptographic attacks on the (unbalanced) oil and vinegar scheme.

6.3.2.i Kipnis and Shamir Attack The main idea in the attack is to separate the oil and the vinegar variables, which enables an attacker to forge arbitrary signatures. The attack is very efficient for all \( v \leq n \). We describe the attack here for \( v = n \).

We take only the quadratic terms of the private \( P' \) and the public \( P \) equations into account. So, we can uniquely represent the private key equations (resp. public key equations) by \( x^t P_i' x \) (resp. \( x^t P_i x' \)) for \( 0 \leq i \leq m \), where \( P_i' \) and \( P_i \) are upper-triangular matrices belonging to \( \mathbb{F}^{2n \times 2n} \). In the following analysis, we use the unique symmetric matrices \( P_i' + P_i'^T \) and \( P_i + P_i^T \), which we will recall for simplicity \( P_i' \) and \( P_i \). Note that because of the special structure of the equation \( P' \), the matrices \( P_i' \) for \( 1 \leq i \leq m \) have the form:

\[
P_i' = \begin{pmatrix} 0 & A_i \\ B_i & C_i \end{pmatrix}
\]

where \( 0, A_i, B_i, C_i \) are submatrices of dimension \( n \times n \). Because \( P = P' \circ S \), we get that

\[
P_i = M_S \begin{pmatrix} 0 & A_i \\ B_i & C_i \end{pmatrix} M_S^T.
\]

It is clear that each \( P_i' \) maps the subspace \( x_{n+1} = \cdots = x_{2n} \) (oil subspace) to the subspace \( x_1 = \cdots = x_n = 0 \) (vinegar subspace). If \( P_i' \) is invertible, we can then conclude that each \( P_i' P_j'^{-1} \) maps the oil subspace to itself. Consequently the image of the oil subspace under \( S \), called the subspace \( O \), is a common eigenspace for each \( P_i P_j^{-1} \) with \( 1 \leq i < j \leq m \). In [KS98], two very efficient algorithms are explained for computing the common eigenspace \( O \) of a set of transformations. Picking a subspace \( V \) for which \( O + V = \mathbb{F}^{2m} \) enables us to separate the oil and the vinegar variables.

In [KPG99], an extension based on a probabilistic approach of the previous attack is described which works for \( v > n \) and \( v \approx n \) with complexity \( q^{v-n-1}n^4 \).

6.3.2.ii Improved Relinearization Attacks The attack of Meier and Tacier as described in [CGMT02] works only for fields of characteristic 2 and on unbalanced oil and vinegar schemes with \( n = (k+1)m \). The main idea is to fix \( k \cdot m \) linear relations between the
unknowns. After eliminating these linear relations, a simpler system of \( m \) equations and \( m \) unknowns is obtained, where a few quadratic equations have become linear in the remaining \( m \) variables. This system is then solved with relinearization techniques as proposed in [CKPS00]. To obtain the simpler system, the following theorem is used on one or more quadratic public polynomials \( P_1 \).

**Theorem 6.3.1** A nondegenerate form \( Q \) over \( \mathbb{F} \) with \( n \) odd can be transformed in

\[
x_1x_2 + x_3x_4 + \cdots + x_{n-2}x_{n-1} + x_n^2.
\]

If \( n \) is even, then \( Q \) can be transformed in one of the two following forms:

\[
x_1x_2 + x_3x_4 + \cdots + x_{n-1}x_n
\]

\[
x_1x_2 + x_3x_4 + \cdots + x_{n-1}x_n + x_{n-1}^2 + ax_n^2,
\]

where \( a \) is an element whose trace over \( \mathbb{F} \) has value 1.

Inspired by this form, the linear relations are created. The attack improves if there exists any degeneration in the public public polynomials, i.e., if they can be rewritten with fewer variables after a linear change of the variables. Algorithms for computing the order of degeneration are described in [PG97a].

**6.3.2.ii.a Unbalanced Oil and Vinegar Schemes with \( v \geq n^2 \)** In [KPG99], an algorithm is given that solves unbalanced oil and vinegar schemes of \( n \) quadratic equations in at least \( n^2 + n \) variables or with parameters \( v \geq n^2 \). Therefore, a specific linear transformation \( x \mapsto y \) is applied on the input variables \( x \) of \( \mathcal{P}(x) \) such that the terms \( y_1y_2, \ldots, y_1y_n, y_2y_3, \ldots, y_2y_n, \ldots, y_{n-1}y_n \) do not appear anymore in the system \( \mathcal{P}(y) \). Introducing again suitable linear relations enables us to solve the system by Gaussian elimination with a very high probability.

**6.3.2.iii Solving IP** For UOV, we have a very special structure for the private system \( \mathcal{P}' \). If we write the private equations \( p_i' \) as \( p_i'(x_1, \ldots, x_n) = \gamma_{i,j,k}' x_j x_k + \beta_{i,j}' x_j + \alpha_{i}' \), then in any equation \( i \) the coefficients \( \gamma_{i,j,k}' \) for \( k, j > l \) have the value 0. Therefore we relay on the difficulty of finding the transformation \( S \) for given public system of equations \( \mathcal{P} \) and only partly given private system of equations \( \mathcal{P}' \).

This problem has already been studied under the name “Isomorphism of Polynomials” (IP). The results provided in [PGC98, GMS02] show that this problem is difficult, even for completely known private system \( \mathcal{P}' \).

As long as IP cannot be solved for given \( \mathcal{P} \) and \( \mathcal{P}' \), UOV is secure from this type of attack.

**6.3.3 Conclusions**

Unbalanced Oil and Vinegar schemes are known since 1999. They allow the construction of interesting signature schemes. Due to the short field size of 8-bit, they are very suitable for the smartcard environment.
The attacks known so far against UOV are basically exponentially. Therefore, they require a very high workload for reasonably small parameters, e.g., $q = 256$, $n = 48$, and $m = 16$. This leads to a public key size of 18 kB.

In the framework of multivariate quadratic schemes, UOV are an interesting topic for further research — especially as they are much different from other schemes such as C$^*$ or HFE and do therefore not fall to the same kind of attacks.

References – Multivariate Cryptosystems


6. Multivariate Systems


References – Multivariate Cryptosystems


6. Multivariate Systems
7

Code-based Cryptosystems

Contributor: Nicolas Sendrier

7.1 Introduction

Robert J. McEliece proposed the first system based on algebraic coding theory in 1978 [McE78]. At that time, the only other known systems were RSA [RSA78] and Merkle and Hellman’s knapsack. McEliece’s intent was to take advantage of the very efficient encoding and decoding algorithms for the binary Goppa codes to propose a very fast asymmetric encryption scheme. The security was related to the difficulty of decoding in a linear code [BMvT78]. Later, in 1986, Harald Niederreiter proposed a dual version [Nie86] of the system, with an equivalent security [LDW94]. A specialized version of the latter based on Reed-Muller code was also proposed in 1994 [Sid94]. The first digital signature scheme using codes was presented in 2001 [CFS01].

So far, these systems have successfully resisted to all cryptanalysis attempts. More, all known attacks have a fully exponential complexity compared with the block length, and there are proven reductions to clearly identified (and difficult) coding theory problems [Sen02b]. In addition to those security issues, these systems possess some very interesting features:

- for all encryption schemes, both encryption and decryption are very fast ("small" quadratic complexity, see §7.2),
- the block length for the Niederreiter variant can be kept small, typically 300 bits or even less (the price is a larger public key, see below),
- the signature scheme produces the shortest known signature, less than 100 bits.

Of course all of the above is true when we match nowadays security requirements, that is more than $2^{80}$ CPU operations for the best known attack. On the other hand, the code-based cryptosystems have some strong drawbacks:

- the public key is very large, typically 0.5 Mbits for the encryption schemes and 9 Mbits for the signature scheme,
- the cryptogram is larger than the message by 20 to 50% (100% in McEliece’s original version, but the choice was not optimal),
- the cost for producing one digital signature is high, a software implementation with secure parameters requires more than one minute.\(^7\)

\(^7\)can be reduced to one second with a FPGA implementation.
7. Code-base Cryptosystems

7.2 Description

Let \( \mathcal{F} \) denote a family of binary linear codes of length \( n \), dimension \( k \) (codimension \( r = n - k \)) for which a \( t \)-error correcting procedure is known. We denote by \( W_{n,t} \) the set of binary words of length \( n \) and Hamming weight \( t \).

**Encryption schemes:**

<table>
<thead>
<tr>
<th></th>
<th>McEliece</th>
<th>Niederreiter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key generation</strong></td>
<td>( C \in_R \mathcal{F} )</td>
<td>( C \in_R \mathcal{F} )</td>
</tr>
<tr>
<td></td>
<td>( G_0 ) generator matrix</td>
<td>( H_0 ) parity check matrix</td>
</tr>
<tr>
<td></td>
<td>( G = SG_0P )</td>
<td>( H = SH_0P )</td>
</tr>
<tr>
<td></td>
<td>( S ) non singular ( k \times k )</td>
<td>( S ) non singular ( r \times r )</td>
</tr>
<tr>
<td></td>
<td>( P ) permutation ( n \times n )</td>
<td>( P ) permutation ( n \times n )</td>
</tr>
<tr>
<td><strong>Public key</strong></td>
<td>( G )</td>
<td>( H )</td>
</tr>
<tr>
<td><strong>Cleartext</strong></td>
<td>( x \in \mathbb{F}_2^k )</td>
<td>( x \in W_{n,t} )</td>
</tr>
<tr>
<td><strong>Ciphertext</strong></td>
<td>( y = xG + e )</td>
<td>( y = Hx^T )</td>
</tr>
<tr>
<td>( e \in_R W_{n,t} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Secret key</strong></td>
<td>a ( t )-error correcting procedure for ( C )</td>
<td>applying the secret decoding procedure</td>
</tr>
<tr>
<td><strong>Decryption</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Digital signature scheme:** The public key and the key generation mechanism is the same as for Niederreiter’s scheme. Let \( M \) denote the message to be signed, \( h() \) a cryptographic hash function (producing binary strings of suitable length) and \( || \) the concatenation. The signature is best described with the following pseudo-code:

```plaintext
procedure sign (input: a message \( M \), output: a pair in \( W_{n,t} \times \mathbb{N} \))
    \( i \leftarrow 0 \)
    repeat
        \( y \leftarrow h(M || i) ; x \leftarrow \text{decode}(y) \)
        if \( x \neq \text{FAIL} \) then return \((x, i)\)
        \( i \leftarrow i + 1 \)
```

Remarks:

- The procedure \( \text{decode()} \) takes as input any word \( y \in \mathbb{F}_2^r \) and returns \( x \in W_{n,t} \) such that \( y = Hx^T \) or \( \text{FAIL} \) if such an \( x \) do not exist.
- The loop is repeated \( t! \) times in average.
- The signature is an element \( (x, i) \in W_{n,t} \times \mathbb{N} \) and the verifier checks \( h(M || i) = Hx^T \).

\(^{8}\) where \( \in_R \) stands for “picked at random in”
7.3 Security

The code $C$ in the family $\mathcal{F}$ is called the hidden code. The coordinates of this code are permuted randomly, and the public key is a random generator or parity check matrix of the permuted code. The two approach for the cryptanalysis are:

- **decoding attacks**: decode errors in an arbitrary linear code,
- **structural attacks**: recover information on the algebraic structure of the hidden code.

In this subsection we assume that binary Goppa codes are being used.

### 7.3.1 Theoretical security

We state here two NP problems, corresponding to the decoding and structural attacks respectively. The exact status of these problems is not known, but both are believed to be hard in the worst case (NP-complete?) and in the average case.

**Problem 1 (Goppa Bounded Decoding - GBD)**

**Instance:** An $r \times n$ binary matrix $H$ and a word $s$ of $\mathbb{F}_2^n$.

**Question:** Is there a word $e$ in $\mathbb{F}_2^n$ of weight $\leq r/\log_2 n$ such that $He^T = s$?

This problem is tightly related to the NP-complete syndrome decoding (SD) problem. In fact, in SD the weight $w$ of the word we are looking for is a parameter in the instance. Here we specialized SD by letting $w = r/\log_2 n$, the practice of coding theory tells us this is a difficult case, complexity theory was not able to conclude anything so far.

**Problem 2 (Goppa Code Distinguishing - GD)**

**Instance:** An $r \times n$ binary matrix $H$.

**Question:** Does $H$ belong to $G_{n,t}$ for some $t$?

In this last statement, we denote by $G_{n,t}$ the set of all parity check matrices of $t$-error correcting binary Goppa codes of length $n$. Solving this problem is connected with the existence of properties of linear codes which are both easy to compute (ideally polynomial time) and invariant by permutation of the coordinates. No known easy invariant is able to distinguish Goppa codes from random codes. Finding new easy invariants is an important issue in algebraic coding theory, and is considered very difficult.

The security of code based systems is provably reduced to the difficulty of those two problems [Sen02b].
7.3.2 Best known attacks

7.3.2.i Decoding attacks This approach has been thoroughly examined in the last 20 years [LB88, Ste89, Leo88, vT90, Can96, CC98]. The best known implementation is due to Canteaut and Chabaud, and is based, as most of the other works on information set decoding. The binary workfactor\(^9\) of this attack is equal to

\[ WF = 2^{(tm - t \log_2(tm))(1 + o(1))} = P(n)2^{tm - t \log_2(tm)} \]

where \(P(n)\) is bounded by a polynomial of small degree (possibly 0) in \(n\). A precise evaluation of this workfactor requires the computation of a Markov chain for each value of the code parameters. Result of those computations are given in Figure 7 (values of \(R\) close to 0 or 1 are not significant).

![Graph showing \(\log_2 WF\) versus the code rate for Goppa codes of various length with the Canteaut-Chabaud decoder, \(R = 1 - tm/n\)](image)

When the code rate \(R\) is close to one, as it is the case for the signature scheme, the best attack, asymptotically, is the split syndrome decoding (see [Bar98]). The complexity of this attack is \(2^{tm(1/2 + o(1))}\) in time and memory. For the parameters proposed in [CFS01] the Canteaut-Chabaud attack is still the best in practice.

7.3.2.ii Structural attacks The best known structural attack is basically an exhaustive search on the key space, using the support splitting algorithm as a distinguisher. The support splitting algorithm [Sen00] can detect whether or not two linear codes are equal up to a permutation of their coordinates. Though difficult in the worst case [PR97], this problem can

\(^9\)number of binary operations
be solved in most cases (in particular for Goppa codes). The cost of this attack is related to
the number of binary irreducible Goppa codes and is equal to \(O(2^{tm}/tm) = 2^{tm(1+o(1))}\). This
is always larger than for the decoding attack.

7.4 Choosing the parameters

The code is chosen among irreducible binary Goppa codes. A \(t\)-error correcting binary Goppa
code of length \(n = 2^m\) has dimension \(k = n - tm\). Other choices are possible, but we loose
the security reduction of §7.3.1.

Table 7 gives the main figures of the encryption schemes in terms of the code parameters.
The integer \(m\) is the degree of the extension of \(\mathbb{F}_2\) used in most calculations. Niederreiter’s
scheme is faster for encryption, and slower for decryption.

Table 8 instanciates Table 7 for two sets of parameters. The first, \((m, t) = (11, 30)\), is
secure with todays security requirements, and the second, \((m, t) = (10, 50)\), was initially
proposed by McEliece. It is remarquable that the encryption/decryption costs decreases
slightly with the most secure parameters. The cost are given in binary operations and assume
a table lookup implementation of the finite field operations. Note that these “number of binary
operations” are just to be understood here as a scale of complexity for the algorithms, not as
a reliable mean to derive computation time.

<table>
<thead>
<tr>
<th></th>
<th>McEliece</th>
<th>Niederreiter</th>
</tr>
</thead>
<tbody>
<tr>
<td>size of the cryptogram</td>
<td>(n)</td>
<td>(tm)</td>
</tr>
<tr>
<td>size of the cleartext</td>
<td>(n - tm)</td>
<td>(tm - t \log_2(t/e))</td>
</tr>
<tr>
<td>encryption cost (per cleartext bit)</td>
<td>(tm/2)</td>
<td>(t)</td>
</tr>
<tr>
<td>decryption cost (per cleartext bit) (^{10})</td>
<td>(c_1tm)</td>
<td>(c_2tm)</td>
</tr>
<tr>
<td>public key size in bits</td>
<td>(tm(n - tm))</td>
<td></td>
</tr>
<tr>
<td>security (decoding attack)</td>
<td>(2^{tm(1+o(1))})</td>
<td></td>
</tr>
</tbody>
</table>

\(^{10}\) where \(c_1 \in [1, 2]\) and \(c_2 \in [3, 5]\)

Table 7: Sizes and costs (in binary operations) for Goppa code based encryption schemes

<table>
<thead>
<tr>
<th>((m, t))</th>
<th>McEliece</th>
<th>Niederreiter</th>
</tr>
</thead>
<tbody>
<tr>
<td>((11, 30))</td>
<td>2048</td>
<td>1024</td>
</tr>
<tr>
<td>((10, 50))</td>
<td>1718</td>
<td>524</td>
</tr>
<tr>
<td>size of the cleartext</td>
<td>165</td>
<td>250</td>
</tr>
<tr>
<td>encryption cost (per cleartext bit)</td>
<td>456.8</td>
<td>1454.2</td>
</tr>
<tr>
<td>decryption cost (per cleartext bit)</td>
<td>566940</td>
<td>262000</td>
</tr>
<tr>
<td>public key size in bits</td>
<td>286.4</td>
<td>262.1</td>
</tr>
<tr>
<td>security (decoding attack)</td>
<td>262.1</td>
<td></td>
</tr>
</tbody>
</table>

Table 8: Instances of Table 7

An implementation in C of McEliece’s encryption for \((m, t) = (11, 30)\) runs at 30 Mbits/s
on a computer equipped with 2 Ghz Pentium 4 processor. Niederreiter’s scheme is much faster
for the linear algebra part, it is limited though by the encoding of information into words of $W_{n,t}$. Using a method derived from runlength source coding (see [Sen02a]) Niederreiter’s encryption for the same code and on the same machine runs at 60 Mbits/s. Both these implementation can certainly be improved with a little bit of hack. I am not aware of any optimized full implementation of decryption for any of these systems. Extrapolating from the tables, it seems that several Mbits/s is a reasonable guess.

**Digital signature scheme:** Again binary irreducible Goppa codes are used here. We summarize the main features of the signature scheme in Table 9. The signature includes a word of length $n = 2^m$ and weight $t$ whose syndrome relatively to the public key (a parity check matrix) has a prescribed value, giving $t - 1$ positions instead of $t$ doesn’t cost much; instead of checking that a vector is equal to zero, the verifier checks that it is a column of the public key (which is known in advance and can be stored properly). Other tradeoffs are possible, and if one is willing to spend 10 seconds to check the signature, its size can be reduced to 80 bits with the same security.

<table>
<thead>
<tr>
<th>Signature length in bits</th>
<th>$\approx (t - 1)m + \log_2 t$</th>
<th>132</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature cost</td>
<td>$tt^2m^2$</td>
<td></td>
</tr>
<tr>
<td>Verification cost</td>
<td>$t^2m$</td>
<td>&lt; 1μs</td>
</tr>
<tr>
<td>Public key size in bits</td>
<td>$tm(2^m - tm)$</td>
<td>9416448</td>
</tr>
<tr>
<td>Security (decoding attack)</td>
<td>$2^{tm(1/2+o(1))}$</td>
<td>$2^{83.7}$</td>
</tr>
</tbody>
</table>

Table 9: Signature scheme with Goppa codes

Some research on FPGA implementation of the scheme are in progress. Preliminary results indicates that for $(m, t) = (16, 9)$, a signature time of less than one second is achievable.

### 7.5 Other related systems

The use of Reed-Muller has also been proposed [Sid94] instead of Goppa codes. There is only one secret code in that system. But, because this code is weakly self-dual, the support splitting algorithm do not apply, and the system remains apparently secure.

Another possibility, initiated by Ernst Gabidulin [GPT91], consists in using the rank metric instead of the Hamming metric. The main difficulty, as it is shown in [Gib96], is to securely hide the code structure. The latest developments (see [GOHA03] for instance) appear promising, and offer the advantage of code-based cryptosystems while offering a much smaller public key size (only 16 kbits – of course this is small with respect to the other code-based cryptosystems currently known.)

### 7.6 Conclusions

These systems are not well suited to classical public key infrastructures. In SSL or PGP like systems, certified public keys are often transmitted along with messages or to initiate a session. Large public keys can thus be a problem when bandwidth is critical.
It must be noted however that with the increase of storage capacity, the public key, even for the signature scheme is not so large if it is simply to be stored. Signatures are very short and very easy to check. Encryption is fast and, with Niederreiter’s scheme, can have a small block length. Depending on the application, these advantages might compensate the drawbacks.

References – Code-based Cryptosystems


7. Code-base Cryptosystems


Cryptosystems based on Drinfeld Modules

Contributors: Carlos Cid, Gebhard Böckle, and Roberto Avanzi

8.1 Introduction

Around 1986 cryptosystems based on elliptic curves were first proposed by N. Koblitz and V. Miller in [Kob87] and [Mil86]. While originally they may have been viewed as rather complicated and impractical in comparison to the then known systems, namely the Diffie-Hellman key exchange and RSA, they have become increasingly popular since, due to the shorter key length they require. Indeed cryptosystems based on elliptic curves have now been implemented on a large scale in practice, and systems designed around the Jacobian varieties of hyperelliptic curves and other geometric structures are being proposed, considered and implemented efficiently.

In 1974 Drinfeld [Drin74] introduced what he called elliptic modules. They are now called Drinfeld-modules in the literature. Their importance (and creation) is derived from the fact that over function fields they have properties very similar to elliptic curves over number fields. Both, elliptic curves and Drinfeld-modules can also be considered over finite fields. Drinfeld modules can be defined as commutative subrings of the ring of regular endomorphisms of the additive group of a finite field. While the general theory of Drinfeld modules was developed (independently) by V. Drinfeld and D. Hayes in the 1970s, the first example of Drinfeld module was constructed by L. Carlitz back in the 1930s.

Bearing in mind the above analogy between elliptic curves and Drinfeld modules, it was clearly only a matter of time before some cryptosystems based on Drinfeld modules were suggested: Analogous versions of Diffie-Hellman, ElGamal and RSA cryptosystems can be easily constructed using Drinfeld modules. In fact, Drinfeld Modules are closely related to elliptic curves, and given the relation between both structures, one would hope that cryptosystems based on Drinfeld modules would share (at least some of) the properties of ECC.

However, Scanlon [Scan01] showed that these somehow natural ways of using Drinfeld modules in cryptology are insecure. More recently, Gillard et al [GLPR03] proposed a new cryptosystem using an one-way trapdoor function based on Drinfeld modules. Likewise, this construction has also been shown to be insecure in [BCG03].

We have no knowledge of a cryptosystem based on Drinfeld modules which has so far been officially proposed and is considered secure.

8.2 Drinfeld Modules

Below we will give a brief overview of the mathematics behind Drinfeld modules. A more in-depth insight can be found in [Hay92, Pan02].

The first example of a Drinfeld module is due to Carlitz and dates back to the 1930’s, long before Drinfeld introduced the concept of elliptic modules in the 1970’s. Drinfeld modules
are commutative algebraic structures in the ring of endomorphisms of the additive group of a finite field.

More specifically, let \( K = \mathbb{F}_q \) be a finite field of characteristic \( p > 0 \) and \( \tau : x \mapsto x^p \) be the Frobenius map. One can show that the ring \( \text{End}(G_{a,K}) \) of endomorphisms of the additive group of \( K \) is isomorphic to

\[
K\{\tau\} = \{ \sum a_i \tau^i \mid a_i \in K \},
\]

the ring of twisted polynomials in \( \tau \) over \( K \), with \( \tau a = a^p \).

If \( C \) is a smooth irreducible projective curve over \( K = \mathbb{F}_q \), \( \infty \) a closed point of \( C \), and \( \mathcal{A} \) the ring of regular functions on \( C \backslash \{ \infty \} \), let \( \gamma : A \to K \) be a \( \mathbb{F}_q \)-algebra homomorphism, and \( D : K\{\tau\} \to K \) the left inverse of the natural inclusion \( \epsilon : K \hookrightarrow K\{\tau\} \), with \( D(\sum a_i \tau^i) = a_0 \).

A Drinfeld module over \( K \) is a \( \mathbb{F}_q \)-ring homomorphism

\[
\phi : A \to K\{\tau\}
\]
such that \( D \circ \phi = \gamma \).

If we denote \( \phi(a) \) by \( \phi_a \in \text{End}(G_{a,K}) \), then \( \phi \) induces on the additive group of \( K \) a new structure as an \( A \)-module:

\[
a * x = \phi_a(x) \quad (a \in A, x \in K)
\]

Thus a Drinfeld module can be seen as a non-linear, commutative subring of the (non-commutative) ring of regular endomorphisms of the additive group of a finite field, with a given presentation.

### 8.3 Cryptosystems based on Drinfeld Modules

The most obvious ways of using Drinfeld modules in cryptology is by modifying the commonly used group based cryptosystems, such as Diffie-Hellman, ElGamal and RSA, and replacing the underlying group by the additive group of a finite field and multiplication by integers (or exponentiation) by the action of the Drinfeld module [Scan01].

#### 8.3.1 A realization and a simple cryptosystem

In its simplest realization, a Drinfeld module is given by a polynomial \( \sigma \in \mathbb{F}_p[x] \) of the form

\[
\sigma(x) = a_0 x + a_1 x^p + \ldots + a_r x^r
\]

for some \( a_i \in \mathbb{F}_p \). If we regard one can easily verify that the map \( x \mapsto \sigma(x) \) defines a linear endomorphism of \( \mathbb{F}_q \) as a vector space of dimension \( d \) over \( \mathbb{F}_p \). Therefore \( \sigma \) may be represented by an \( d \times d \)-matrix over \( \mathbb{F}_p \).

The vector space \( \mathbb{F}_q \) is now given the structure of a module over \( \mathbb{F}_p[t] \) via the assignment

\[
(\mathbb{F}_p[t] \times \mathbb{F}_q) \to \mathbb{F}_q:
\]

\[
(b_0 + b_1 t + \ldots + b_s t^s, a) \mapsto b_0 a + b_1 \sigma(a) + \ldots + b_s \sigma(\ldots(\sigma(a))) \quad (s \\
\downarrow \quad a)
\]
8.3. Cryptosystems based on Drinfeld Modules

In the definition of Drinfeld module given in the previous Subsection, \( A = \mathbb{F}_p[t] \), the ring of regular functions of a genus 0 variety, and \( \tau = \sigma \).

Let now \( \sigma \) and the \( \mathbb{F}_p[t] \)-module structure on \( \mathbb{F}_q \) just introduced. Then for \( \alpha \in \mathbb{F}_q \) the finite \( \mathbb{F}_p[t] \)-module \( \mathbb{F}[t] \cdot \alpha \) is cyclic by its very definition. Since \( \mathbb{F}_p[t] \) is a principal ideal domain, the ideal \( \text{Ann}(\alpha) := \{ h(t) \in \mathbb{F}_p[t] : h(t)\alpha = 0 \} \) is principal, say with generator \( g(t) \), and so \( \mathbb{F}[t] \cdot \alpha \cong \mathbb{F}_p[t]/(g(t)) \) as \( \mathbb{F}_p[t] \)-modules. The discrete logarithm problem in this context is the following: Suppose \( \beta \) is another element in the submodule \( \mathbb{F}[t] \cdot \alpha \) of \( \mathbb{F}_q \). Find a polynomial \( h(t) \in \mathbb{F}_p[t] \) such that \( \beta = h(t)\alpha \).

If this discrete logarithm problem was difficult, then one could design the following Diffie-Hellman key exchange: Suppose the parameters \( p, q, \sigma \), and \( \alpha \in \mathbb{F}_q \), as well as a generator \( g(t) \) of \( \text{Ann}(\alpha) \) are publicly known. A simple protocol for two protagonists \( A \) and \( B \) to exchange a secret key is the following: They both choose randomly polynomials \( g_A(t) \) and \( g_B(t) \) in \( \mathbb{F}_p[t] \) which have no common divisors with \( g(t) \). If then \( A \) sends \( g_A(t)\alpha \) to \( B \) and \( B \) sends \( g_B(t)\alpha \) to \( A \), they both share the secret key

\[
\beta := g_A(t)g_B(t)\alpha = g_B(t)g_A(t)\alpha.
\]

Since the discrete logarithm problem is assumed to be hard, it would be likely that for a listener \( C \) it is difficult to recover \( \beta \) from the public information \( \alpha, g_A(t)\alpha \) and \( g_B(t)\alpha \). (This should be compared to the ElGamal system based on elliptic curves: There one fixes, for instance, the parameters \( p, a \) prime, \( E \) and elliptic curve over \( \mathbb{F}_p \), \( P \) a point on \( E(\mathbb{F}_p) \) and \( N \) the order of \( P \). These data correspond to \( p, q, \sigma \) and \( g(t) \).)

8.3.2 The general case and cryptanalysis

A more general Drinfeld module version of the the Diffie-Hellman cryptosystem can be defined as follows:

- Fix \( p \) prime, \( K = \mathbb{F}_{p^d} \) a finite field, a Drinfeld module \( \phi : A \to K\{\tau\} \) and \( x \in K \) (these are publicly known).

- Alice and Bob choose \( a \) and \( b \in A \), respectively (the private keys).

- Alice sends \( \phi_a(x) \in K \) to Bob, while Bob sends \( \phi_b(x) \) to Alice.

The shared secret is then given by:

\[
\phi_{ab}(x) = \phi_a(\phi_b(x)) = \phi_b(\phi_a(x)) \in K
\]

The security of the scheme would be related to the supposed intractability of the Discrete Logarithm Problem on Drinfeld Modules: given a finite field \( K = \mathbb{F}_{p^d} \), a Drinfeld module \( \phi : A \to K\{\tau\} \), and elements \( x, y \in K \), find \( a \in A \) such that

\[
\phi_a(x) = y.
\]

A Drinfeld module version of the RSA cryptosystem can also be found in [Scan01].
Drinfeld Modules and Elliptic Curves are intimately related. In particular, Drinfeld modules of rank 2 look very much like elliptic curves over $\mathbb{C}$. Given the nonlinearity of Drinfeld modules and the connection between both structures, one would expect that such schemes should be secure. Additionally, the underlying group structure is additive, so implementation should also be quite efficient.

However T. Scanlon [Scan01] showed that there exist polynomial time algorithms which solve the discrete logarithm and the inversion problems for Drinfeld modules. This follows from the fact that, given a Drinfeld module $A = \phi(A) \subseteq \text{End}(G_0, K)$ over $K = \mathbb{F}_{p^d}$, even though the elements of $A$ are most likely not $K$-linear maps, they are clearly $\mathbb{F}_p$-linear. So one can defeat the schemes described above using quite straightforward linear algebra techniques.

Thus the conclusion found in [Scan01] that “any cryptosystem based on the supposed infeasibility of solving the Drinfeld module versions of the discrete logarithm and inversion problems is insecure”.

More recently, R. Gillard, F. Leprevost, A. Panchishkin and X.F. Roblot suggested a cryptosystem using an one-way trapdoor function based on Drinfeld modules as follows: given $A = \mathbb{F}_p[T]$ and $\phi$ a Drinfeld module over $K = \mathbb{F}_{p^d}$, one chooses $c_1, c_2 \in A$ and $\sigma$ bijection on $K$, and defines the function $\psi : \mathbb{F}_{p^d} \rightarrow \mathbb{F}_{p^d}$ as

$$\psi(z) = (\phi_{c_1} \circ \sigma \circ \phi_{c_2})(z).$$

One can now express $\psi(z)$ as a (sparse) polynomial in $K[z]$ with degree $< p^d$, which will be the public key. The trapdoor (i.e. private key) is given by $(c_1, c_2, \sigma)$. The bijection $\sigma$ should be simple to calculate (e.g. a polynomial, $z \mapsto z^e + \delta$, $\delta \in K$). The parameters suggested are: prime $p \approx 2^{32}$, $d = 5$ or 6 and $e$ a small integer, coprime with $p^d - 1$ (e.g. $e = 5$ or 7).

Thus the trapdoor one-way function $\psi : \mathbb{F}_{p^d} \rightarrow \mathbb{F}_{p^d}$ can be defined as

$$\psi(z) = \lambda_1((\lambda_2(z))^e + \delta)$$

where $\lambda_1, \lambda_2$ are two bijective $\mathbb{F}_p$-linear maps on the vector space $\mathbb{F}_{p^d}$. In fact, the linear maps $\lambda_1$ and $\lambda_2$ are of the form:

$$b_0 + b_1 \tau + \cdots + b_{d-1} \tau^{d-1}$$

where $\tau$ is the Frobenius map on $\mathbb{F}_{p^d}$ and $b_i \in \mathbb{F}_{p^d}$.

Note that this scheme is reminiscent of the HFE family of cryptosystems, though its security is supposed to be based on the underlying structure of Drinfeld modules rather than on the private polynomial $\sigma$.

In [BCG03], Blackburn, Cid and Galbraith showed how an adversary can directly recover the private key in the encryption method of [GLPR03] using only the public key. For parameters as those suggested by the authors of [GLPR03], (e.g. $d \leq 5$, $e = 5, 6$) the cryptosystem was broken within 1 second on a single 700 MHz Pentium III CPU. They used a linearization method (see also [KiSh99]).

One needs first to guess $e$, which is known to be small. Then, using the public key, one can generate pairs $(z, w)$, where $w = \psi(z)$. Each pair gives rise to a relation of type

$$\lambda_1^{-1}(w) = \lambda_2(z)^e + \delta.$$
8.4. Conclusions

which in turn can be represented as a multivariate polynomial over $\mathbb{F}_{p^d}$ in $2d + 1$ variables.

Now since this is a public-key cryptosystem, one can generate as many pairs as needed with the public key, and thus obtain a large multivariate polynomial system which can be solved using the well-known technique of linearisation: consider all monomials which appear in the system as independent variables and solve the system using linear algebra techniques. In this case, one needs at most $\binom{d+e-1}{e} + d + 1$ pairs. Experiments in [BCG03] showed that, given the function $\psi$ with the parameters suggested, one can recover a private key in few seconds.

The underlying problem with constructing cryptosystems based on Drinfeld modules seems to be related with the linearity of the construction. As noted above, given a Drinfeld module $A = \phi(A)$ over a finite field $K = \mathbb{F}_{p^d}$, the elements of $A$ are clearly $\mathbb{F}_p$-linear. Attacks on the proposed schemes have all exploited this structure, and the natural ways to mitigate such attacks are expected to affect its efficiency and applicability, and thus limit its use in practice.

8.4 Conclusions

Currently there are no known cryptosystems based on Drinfeld modules which are considered to be secure. While more research is needed to investigate applications of Drinfeld modules in cryptology, it is uncertain whether one can use Drinfeld modules to construct efficient, secure public-key cryptosystems.

Drinfeld Modules are algebraic structures which are closely related to elliptic curves. The main difference between elliptic curves and Drinfeld-modules which impacts in a deep way the security of the corresponding discrete logarithm problem is the complexity of the addition of points. For Drinfeld-modules this is linear in the input. For elliptic curves this operation is highly non-linear. Similarly the operation $P \mapsto nP$, $n \in \mathbb{N}$, is extremely non-linear for elliptic curves, while $\alpha \mapsto f(t)\alpha$ is a superposition of a finite number of linear operations, and so it is very close to being linear.

This discrepancy is the deeper reason behind the findings of Scanlon that the El Gamal system with Drinfeld-modules can be broken in polynomial time. The reason why the attack in [BCG03] works is more or less the same. The introduction of the exponent $e$ causes the complexity of an attack to be polynomial of degree $e$. Again its weakness is the linearity of the Drinfeldian structure. The linearity of the structure has also been used to show that other proposed schemes were likewise insecure.

Having a closer look at the system proposed in [GLPR03], one cannot refrain from having the impression that really the Drinfeld-module structure is only used to encode some linear algebra, while the part of the cryptosystem which is responsible for its ‘security’ is the exponent $e$. In this type of systems the use of Drinfeld-modules is rather inessential and could directly be replaced by linear algebra - but these systems are know to be weak. One has to wait for the announced publication [GLPR04] to see whether this is indeed the case, i.e., whether the security will indeed come from the Drinfeld-module structure.

In fact linearity is deeply engrained in the definition of a Drinfeld-module structure. This suggests that Drinfeld-modules are no more useful than linear algebra in cryptography. So one has to await the cryptosystem that will be described in [GLPR04], or any other cryptosystem
based on Drinfeld-modules, and thoroughly evaluate them.

References — Cryptosystems based on Drinfeld Modules


