

# An Algebraic Approach to NTRU ( $q = 2^n$ ) via Witt Vectors and Overdetermined Systems of Nonlinear Equations

J.H. Silverman<sup>1</sup>, N.P. Smart<sup>2</sup>, and F. Vercauteren<sup>2</sup>

<sup>1</sup> Mathematics Department,  
Box 1917, Brown University,  
Providence, RI 02912,  
U.S.A.

`joseph_silverman@brown.edu`

<sup>2</sup> Dept. Computer Science,  
University of Bristol,  
Merchant Venturers Building,  
Woodland Road,  
Bristol, BS8 1UB,  
United Kingdom.

`{nigel,frederik}@cs.bris.ac.uk`

**Abstract.** We use the theory of Witt vectors to develop an algebraic approach for studying the NTRU primitive with  $q$  parameter equal to a power of two. This results in a system of nonlinear algebraic equations over  $\mathbb{F}_2$  having many symmetries, which is reminiscent of the approach of Courtois, Murphy, Pieprzyk, Robshaw and others for studying the structure of block ciphers such as the AES. We study whether this approach to NTRU provides any immediate security threat and conclude that under the most favourable assumptions, the method is of asymptotic interest but is completely impractical at current or likely future parameter sizes.

## 1 Introduction

NTRU [7] is a public key encryption scheme whose security is based on a polynomial factorisation problem in the ring  $\mathbb{Z}_q[X](X^N - 1)$ . It is an interesting system to study for a number of reasons. Firstly, it does not depend on the traditional hard problems, such as factoring or discrete logarithms, on which other practical public key schemes are based. Indeed the best known heuristic attack is that of finding a short vector in a lattice, which appears to be a very hard problem. Furthermore, schemes based on factoring or discrete logarithms can be broken in the quantum setting using Shor's algorithm [15]. Currently, there is no quantum algorithm which significantly improves the classical approach to breaking NTRU. Secondly, the basic arithmetic operations in NTRU are particularly simple making it suitable for use in constrained environments where traditional public key schemes have difficulty.

Algebraic attacks on block and stream ciphers have been of considerable interest in the research literature over the last few years. Starting with the XL and FXL algorithms of Courtois, Klimov, Patarin and Shamir [3], a number of papers has appeared studying applications of the XL algorithm to the security of block and stream ciphers, see for example [4, 5, 11]. These algebraic attacks work by reducing the determination of the secret key in a cryptographic system to the solution of an overdetermined system of quadratic equations over a small finite field. Then using the technique of linearisation, this nonlinear system is mapped into a large system of linear equations. The resulting linear system is then solved and the key is recovered. There is some controversy over the precise practicality of these algebraic attacks in various situations. For example, it is not known whether the XL algorithm runs in subexponential time for systems that are only slightly overdetermined.

In this paper we reduce the problem of recovering an NTRU private key to an overdetermined system of quadratic equations over  $\mathbb{F}_2$ . Experimentally we find that these systems of equations tend to have a single solution, which is encouraging since from the prior literature on the XL algorithm, it appears that XL has a better chance of success if there is a unique solution. This leads us to perform numerical experiments (in small dimension) testing the applicability of the XL algorithm to the system of nonlinear equations derived from NTRU problem instances.

The paper is organised as follows: In Section 2 we briefly describe the NTRU encryption algorithm so as to fix notation. Then in Section 3 we recall the basic theory of Witt vectors and use it to reduce the problem of determining the NTRU private key to that of solving an overdetermined system of quadratic equations. In Section 4 we report on our numerical experiments applying the XL algorithm to the resulting system of equations. Finally in Section 5 we give some conclusions of our work.

We end this introduction by noting that the technique based on Witt vectors introduced in this paper applies to many cryptosystems other than the NTRU cryptosystem. The isomorphism  $\mathbb{Z}_{2^m} \cong W_m(\mathbb{F}_2)$  implies that virtually any cryptosystem based on arithmetic modulo  $2^m$  (together with logical operations and rotations) can be analysed using Witt vectors. This remark applies to symmetric key cryptosystems that could not previously be algebraically analysed due to their use of arithmetic modulo  $2^m$ , for example to RC5 [12], RC6 [13] and IDEA [9].

## 2 The NTRU Cryptosystem

### 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 interval  $(-\lceil q/2 \rceil, \lfloor q/2 \rfloor]$ . For  $N$  a positive integer, we identify the set  $\mathbb{Z}^N$  (respectively  $\mathbb{Z}_q^N$ ) with the ring of

polynomials

$$P(N) = \frac{\mathbb{Z}[X]}{(X^N - 1)}, \quad \text{respectively} \quad P_q(N) = \frac{\mathbb{Z}_q[X]}{(X^N - 1)},$$

via the natural association  $f = (f_0, f_1, \dots, f_{N-1}) = \sum_{i=0}^{N-1} f_i X^i$ . Note that the modulus  $q$  is not necessarily 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 that we denote by  $\star$  to distinguish it from ordinary multiplication  $\cdot$  in  $\mathbb{Z}$  or  $\mathbb{Z}[X]$ . Let  $h = f \star g$ . Then for each  $0 \leq k < N$ , the  $k^{\text{th}}$ -coefficient  $h_k$  of  $h$  is given by

$$h_k = (f \star g)_k = \sum_{i=0}^k f_i g_{k-i} + \sum_{i=k+1}^{N-1} f_i g_{n+k-i} = \sum_{i+j \equiv k \pmod{N}} f_i \cdot g_j.$$

This is the ordinary polynomial product reduced modulo  $X^N - 1$ , so it is both commutative and associative. The symmetric representation of  $\mathbb{Z}_q$ , i.e. reducing in the interval  $(-\lceil q/2 \rceil, \lfloor q/2 \rfloor]$ , implies that the product of two polynomials with coefficients of small absolute value will again be a polynomial with small coefficients. We write  $P_q^*(N)$  for the multiplicative group of units in  $P_q(N)$  and we denote the inverse polynomial of  $f \in P_q^*(N)$  by  $f_q^{-1}$ .

We also need a “small” element  $p$  of  $P(N)$  that is relatively prime to  $q$ , by which we mean that  $p$  and  $q$  generate the unit ideal in  $P(N)$ . Typically one chooses  $p$  to be 2, 3 or  $2 + X$ . In this paper we concentrate on the case  $p = 2 + X$ , since that is the choice made in the NTRU challenges [2]. However, our methodology applies to other values of  $p$ . Reduction of a polynomial in  $P(N)$  modulo  $2+X$  proceeds by rewriting each integer coefficient  $n = 2a+b$  as  $(-X)a+b$  and then iterating. It is easily seen that reduction modulo  $2 + X$  always leads to a polynomial with coefficients in  $\{0, 1\}$ .

We denote the quotient ring  $P(N)/(p)$  by  $P_p(N)$ , we let  $P_p^*(N)$  be the multiplicative group of units in  $P_p(N)$ , and we write  $f_p^{-1}$  for the inverse polynomial of  $f \in P_p^*(N)$ .

## 2.2 The NTRU Primitive

We sketch the NTRU cryptosystem as developed in [7]. The public parameters consist of values for  $(N, p, q)$  as above. The value of  $q$  is chosen to lie between  $N/2$  and  $N$  and may 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 integers  $d$  which are used to define several families of binary polynomials of  $P_q(N)$  as follows: The notation  $\mathcal{L}(d)$  is used to denote the set of polynomials in  $P_q(N)$ , with  $d$  coefficients equal to 1 and all other coefficients set to zero. These sets are used to define three sets of polynomials  $\mathcal{L}_f$ ,  $\mathcal{L}_g$  and  $\mathcal{L}_r$ . The standard [1] contains two common choices for

these sets. In this paper we focus on the following choice; the other case can be dealt with in a similar (but slightly more complicated) manner.

We first define the sets  $\mathcal{L}_f = \{1 + p \star F : F \in \mathcal{L}(d_f)\}$ ,  $\mathcal{L}_g = \mathcal{L}(d_g)$  and  $\mathcal{L}_r = \mathcal{L}(d_r)$ , for certain integer parameters  $d_f, d_g$  and  $d_r$ . Notice that with the above choice of  $f \in \mathcal{L}_f$ , we have  $f_p^{-1} = 1$ .

A public key encryption algorithm consists of three subprocedures: A key generation algorithm, an encryption algorithm and a decryption algorithm. We do not describe these procedures for NTRU here since, for the purposes of this article, it suffices to know that a private key is generated by choosing random  $f \in \mathcal{L}_f$  and  $g \in \mathcal{L}_g$  and computing  $f_q^{-1} \in P_q^*(N)$  and  $f_p^{-1} \in P_p^*(N)$ , which for our choice of  $\mathcal{L}_f$  equals 1. Then the private key consists of the pair  $(f, f_p^{-1})$  and the public key is the polynomial  $h$  defined by the formula

$$h \equiv p \star f_q^{-1} \star g \pmod{q}.$$

Breaking an NTRU key is the problem of recovering  $f$  from a given  $h$ .

The traditional way to try to break an NTRU public key, in the sense of recovering the private key, is to embed the NTRU problem into a lattice as a shortest (or closest) vector problem and use lattice reduction techniques to try and recover the private key. Whilst lattice reduction algorithms themselves run in polynomial time, this method is unlikely to work for large key sizes since the resulting reduced lattice basis only approximates the shortest lattice vector to within an exponential factor.

Lattice based attacks are currently the best known heuristic attacks against NTRU. The best known deterministic attack is based on a meet-in-the-middle strategy, see [8, 16]. It has complexity roughly

$$\frac{1}{\sqrt{N}} \binom{N/2}{d_f/2}. \tag{1}$$

It is highly advisable to choose  $N$  to be a prime. Firstly, this tends to maximise the probability that the private key has an inverse modulo  $p$  and  $q$ . More importantly, the use of composite  $N$  leads to so called composite attacks on NTRU [6], which can significantly reduce the time needed to recover the private key. It is also important to choose  $N$  sufficiently large. Parameter sets with  $N = 107$  have been broken via lattice techniques in a few hours of computer time [10]. Depending on the desired level of security, recommended choices for  $N$  (see [1]) include 167, 251, 347 or 503.

### 3 Reduction to Algebraic Equations over $\mathbb{F}_2$

In this section we show how the NTRU problem for  $q = 2^n$  leads in a natural way to a system of nonlinear equations over  $\mathbb{F}_2$ . Since  $q = 2^n$ , we can apply a lifting strategy by considering the equality  $f \star h \equiv p \star g \pmod{2^m}$  for intermediate values  $0 < m \leq n$  and equate the bits of the coefficients of both sides. In particular, for  $m = 2$  (i.e. working modulo 4) we obtain a highly structured system of  $N$  quadratic equations in  $N$  binary variables. The most elegant approach to this analysis is based on the theory of Witt vectors over  $\mathbb{F}_2$ .

### 3.1 Witt Vectors over $\mathbb{F}_2$

We briefly set the Witt vector notation needed to analyse NTRU. For the convenience of readers who are not familiar with the construction of the ring of Witt vectors, we have sketched the basic theory in Appendix A. For the general theory of Witt vectors, we refer the reader to [14, Section II.6].

The ring of Witt vectors of length  $m$ , denoted  $W_m(\mathbb{F}_2)$ , is the set  $\mathbb{F}_2^m$  of  $m$ -tuples, but with special addition and multiplication laws so that the map

$$\begin{aligned} W_* : W_m(\mathbb{F}_2) &\longrightarrow \mathbb{Z}_{2^m} \\ [a_0, \dots, a_{m-1}] &\longmapsto \sum_{i=0}^{m-1} a_i 2^i \pmod{2^m} \end{aligned} \quad (2)$$

is an isomorphism of rings. The precise construction of the map  $W_*$  and the ring operations on  $W_m(\mathbb{F}_2)$  are described in Appendix A, but as an example, we write them down for  $m = 2$ :

$$\begin{aligned} [a_0, a_1] + [b_0, b_1] &= [a_0 + b_0, a_0 b_0 + a_1 + b_1] \\ [a_0, a_1] \cdot [b_0, b_1] &= [a_0 b_0, a_0 b_1 + b_0 a_1] \end{aligned}$$

### 3.2 Generating Algebraic Equations over $\mathbb{F}_2$ for NTRU

The isomorphism between the ring of Witt vectors  $W_m(\mathbb{F}_2)$  of length  $m$  and the ring  $\mathbb{Z}_{2^m}$  transforms NTRU private key recovery into the problem of solving a system of nonlinear equations over  $\mathbb{F}_2$ . To see why this is true, consider the fundamental relation

$$f \star h \equiv p \star g \pmod{2^m} \quad (3)$$

for some  $m \leq n$ . Applying the inverse of the map  $W_*$  to the coefficients of the polynomials, we obtain the same equation in the ring  $W_m(\mathbb{F}_2)[X]/(X^N - 1)$ , i.e. in the ring of polynomials modulo  $X^N - 1$  whose coefficients are in the ring of Witt vectors of length  $m$ .

To ease notation, for any element  $e \in W_m(\mathbb{F}_2)[X]/(X^N - 1)$ , we will denote the coefficients of  $e$  in  $W_m(\mathbb{F}_2)$  by  $\mathbf{e}_0, \dots, \mathbf{e}_{N-1}$ , i.e.  $e = \sum \mathbf{e}_i X^i$ . Comparing the coefficients in equation (3) gives rise to  $mN$  equations over  $\mathbb{F}_2$ , since each coefficient of each  $X^k$  is a vector of length  $m$  over  $\mathbb{F}_2$ . Explicitly, equating the coefficients of  $X^k$  on the two sides of (3) leads to the formula

$$\sum_{i+j \equiv k \pmod{N}} \mathbf{f}_i \mathbf{h}_j = \sum_{i+j \equiv k \pmod{N}} \mathbf{p}_i \mathbf{g}_j \quad \text{in the ring } W_m(\mathbb{F}_2). \quad (4)$$

To illustrate this approach, we explicitly compute these equations for our choice of parameter sets  $\mathcal{L}_f$  and  $\mathcal{L}_g$  and modulus  $p = 2 + X$ . Recall that we have set  $\mathcal{L}_f = \{1 + p \star F : F \in \mathcal{L}(d_f)\}$  and  $\mathcal{L}_g = \mathcal{L}(d_g)$ . Since  $p = 2 + X$  and since  $g = \sum g_i X^i$  has *binary* coefficients, we conclude that

$$p \star g \equiv \sum_{k=0}^{N-1} \mathbf{R}_k X^k \pmod{2^m} \quad \text{with } \mathbf{R}_k = [g_{k-1}, g_k, 0, \dots, 0] \in W_m(\mathbb{F}_2). \quad (5)$$

(As usual, indices are taken modulo  $N$ .) Similarly, writing  $F = \sum F_i X^i \in \mathbb{F}_2[X]$ , we have

$$f \equiv 1 + (2 + X) \star F \equiv \sum_{i=0}^{N-1} \mathbf{f}_i X^i \pmod{2^m}$$

with  $\mathbf{f}_i \in W_m(\mathbb{F}_2)$  given by

$$\mathbf{f}_i = \begin{cases} [1 + F_{N-1}, F_0 + F_{N-1}, F_0 F_{N-1}, 0, \dots, 0] & \text{if } i = 0, \\ [F_{i-1}, F_i, 0, \dots, 0] & \text{if } 1 \leq i < N. \end{cases}$$

Write  $h \equiv \sum \mathbf{h}_i X^i$  with  $\mathbf{h}_i = [h_{i,0}, h_{i,1}, \dots, h_{i,m-1}]$ . Then

$$f \star h \equiv \sum_{k=0}^{N-1} \mathbf{L}_k X^k \quad \text{with} \quad \mathbf{L}_k = \sum_{i+j \equiv k \pmod{N}} \mathbf{f}_i \mathbf{h}_j. \quad (6)$$

Note that with the notation in (5) and (6), the fundamental relation (4) says that  $\mathbf{L} = \mathbf{R}$ . We will exploit this fact by equating the first two coordinates of  $\mathbf{L}$  and  $\mathbf{R}$ .

The initial Witt ring operation functions are  $S_0(\mathbf{a}, \mathbf{b}) = a_0 + b_0$  and  $P_0(\mathbf{a}, \mathbf{b}) = a_0 b_0$  (see Table 4 in Appendix A), so we find that

$$g_{k-1} = R_{k,0} = L_{k,0} = \sum_{i+j \equiv k \pmod{N}} f_{i,0} h_{j,0} = h_{k,0} + \sum_{i+j \equiv k \pmod{N}} F_{i-1} h_{j,0}.$$

Replacing  $k$  by  $k+1$ , this gives a formula

$$g_k = h_{k+1,0} + \sum_{i+j \equiv k+1 \pmod{N}} F_{i-1} h_{j,0} \quad (7)$$

expressing  $g_k$  as an  $\mathbb{F}_2$ -linear combination of the  $F_i$ . Similarly, using the four Witt ring operation functions from Appendix A Table 4,

$$S_0 = a_0 + b_0, \quad S_1 = a_0 b_0 + a_1 + b_1, \quad P_0 = a_0 b_0, \quad P_1 = a_0 b_1 + b_0 a_1,$$

we compute

$$\begin{aligned} g_k &= R_{k,1} = L_{k,1} \\ &= \sum_{\substack{i < s \\ i+j \equiv k \pmod{N} \\ s+t \equiv k \pmod{N}}} f_{i,0} h_{j,0} f_{s,0} h_{t,0} + \sum_{i+j \equiv k \pmod{N}} (f_{i,0} h_{j,1} + f_{i,1} h_{j,0}) \\ &= \sum_{\substack{i < s \\ i+j \equiv k \pmod{N} \\ s+t \equiv k \pmod{N}}} F_{i-1} F_{s-1} h_{j,0} h_{t,0} + h_{k,0} \sum_{s+t \equiv k \pmod{N}} F_{s-1} h_{t,0} \\ &\quad + \sum_{i+j \equiv k \pmod{N}} (F_{i-1} h_{j,1} + F_i h_{j,0}) + h_{k,1}. \quad (8) \end{aligned}$$

Notice that (8) expresses  $g_k$  as an  $\mathbb{F}_2$ -quadratic combination of the  $F_i$ .

Equating the two formulae (7) and (8) for  $g_k$ , we arrive at a system of  $N$  quadratic equations over  $\mathbb{F}_2$  in the  $N$  variables  $F_0, F_1, \dots, F_{N-1}$ . We further observe that the resulting system of quadratic equations has a high degree of symmetry. Indeed, there are two sets of indices  $S$  and  $T$  such that each equation  $L_{k,1} - g_k = 0$  can be written as

$$\sum_{\substack{i < j \\ i, j \in S_k}} F_i F_j + \sum_{i \in T_k} F_i + h_{k,0} \sum_{i \in S_k} F_i = h_{k+1,0} + h_{k,1}, \quad (9)$$

with

$$S_k = \{i + k \pmod{N} \mid i \in S\} \quad \text{and} \quad T_k = \{i + k \pmod{N} \mid i \in T\}.$$

*Example 1.* We give an example with  $m = 2$  illustrating the nice structure of the system of quadratic equations. We let  $N = 17$ ,  $q = 2^7$ ,  $d_f = 6$  and  $d_g = 5$ , and for the private key we take

$$\begin{aligned} f &= 1 + p \star F \\ &= 1 + (2 + X) \star (X^{14} + X^{11} + X^6 + X^5 + X^4 + X) \\ &= X^{15} + 2X^{14} + X^{12} + 2X^{11} + X^7 + 3X^6 + 3X^5 + 2X^4 + X^2 + 2X + 1, \\ g &= X^{15} + X^{13} + X^{12} + X^7 + 1. \end{aligned}$$

The corresponding public key is

$$\begin{aligned} h &= 105X^{16} + 45X^{15} + 32X^{14} + 119X^{13} + 53X^{12} + 89X^{11} + 67X^{10} + 3X^9 \\ &\quad + 86X^8 + 52X^7 + 56X^6 + 69X^5 + 101X^4 + 81X^3 + 52X^2 + 6X + 29. \end{aligned}$$

Given  $h$ , we can easily compute the index sets  $S$  and  $T$  and the constant terms of the quadratic equations (9); we obtain

$$\begin{aligned} S &= \{0, 1, 3, 4, 5, 6, 7, 11, 12, 13, 16\}, \\ T &= \{3, 6, 7, 8, 15\}, \\ [h_{k+1,0} + h_{k,1}]_{0 \leq k < N} &= [0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1]. \end{aligned}$$

In this case, the system of equations (9) has only one solution, namely that given by the coefficients of  $F = X^{14} + X^{11} + X^6 + X^5 + X^4 + X$ .

### 3.3 Additional Equations

As we have seen, the results of Section 3.2 with  $m = 2$  can be used to derive  $N$  inhomogeneous quadratic equations over  $\mathbb{F}_2$  in the  $N$  variables  $F_0, \dots, F_{N-1}$ . However, due to our choice of parameters, we know that exactly  $d_f$  of the values of the  $F_i$  are equal to one and the rest are zero. This immediately gives us an extra linear equation

$$\sum_{i=0}^{N-1} F_i \equiv d_f \pmod{2}.$$

We can similarly make use of the second least significant bit of  $d_f$  to obtain a quadratic relation on the  $F_i$ . Precisely, if we write  $d_f \equiv d_{f,0} + 2d_{f,1} \pmod{4}$  with  $d_{f,0}, d_{f,1} \in \{0, 1\}$ , then

$$\sum_{i < j} F_i F_j \equiv d_{f,1} \pmod{2}.$$

Using other bits of  $d_f$  will result in similar equations, but unfortunately they are no longer quadratic, but instead will have higher degree.

Thus using the results of Section 3.2 with  $m = 2$ , we obtain  $N + 2$  inhomogeneous quadratic equations over  $\mathbb{F}_2$  in the  $N$  variables  $F_0, \dots, F_{N-1}$ . The study of such systems of equations has already received much attention in cryptography, see for example [3–5, 11], via the XL algorithm. In the next section we consider how the XL algorithm may be applied to analyse our system of equations.

## 4 Experiments with the XL Algorithm

We have conducted extensive experiments with small values of  $N$  and found that the system of  $N + 2$  quadratic equations corresponding to  $m = 2$  almost always has a unique solution, namely the target private key  $F$ . Very rarely we found that the system of equations had more than one solution, but even then there was only one solution with the correct Hamming weight  $d_f$ .

Thus the formulae derived in Sections 3.2 and 3.3 generally give us a (slightly) overdetermined system of multivariate polynomial equations (of degree 2) whose solution is the desired private key. Such systems of equations might lend themselves to XL-type techniques as described in [3–5, 11]. We note that in general, solving such systems is NP-hard. However, this is a worst case analysis, and as a practical matter we are more interested in the average case difficulty of our more special problem.

In the paper [3], Courtois, Klimov, Patarin and Shamir propose an algorithm called XL to solve overdetermined systems of multivariate polynomial equations over a finite field. We restrict attention to equations of degree at most two. In this case, the equations defining the system are multiplied by monomials of degree less than  $D - 2$  to produce equations of degree at most  $D$ . This results in a much larger number of equations, albeit of much higher degree. Note since we are working in characteristic 2, the degree of the monomial equals the number of variables occurring in the monomial. The monomials of degree greater than one in this enlarged system are then replaced by “dummy variables”, a technique called linearisation. The resulting linear system is solved, and one hopes that the original system is solved at the same time. Note that the method only works if the resulting system of linear equations has rank approximately equal to the number of linearised variables.

The arguments of [4], where multivariate quadratic equations over  $\mathbb{F}_2$  are studied, imply that if the number of equations  $n_e$  is roughly equal to the number of variables  $n_v$ , then the XL algorithm runs in exponential time. The authors of [4] conjecture that the XL method runs in polynomial time when  $n_e = O(n_v^2)$ ,

and that it does not run in polynomial time when  $n_e/n_v^2 \rightarrow 0$ . However, for systems where the number of equations is slightly larger than the number of variables, the authors in [3] leave as an open question whether the XL algorithm might in fact run in subexponential time.

In our situation with  $m = 2$ , we derived  $N + 2$  quadratic equations in  $N$  variables. Since  $N + 2$  is only slightly larger than  $N$ , it is not clear that we obtain any advantage. This led us to conduct the following experiments and to analyse the XL algorithm as applied to the specific set of  $N + 2$  inhomogeneous equations in  $N$  unknowns described in Sections 3.2 and 3.3.

After linearisation, we obtain

$$\alpha = (N + 2) \sum_{j=0}^{D-2} \binom{N}{j}$$

linear equations in at most  $\beta = \sum_{k=1}^D \binom{N}{k}$  variables. We let  $\gamma$  denote the number of linearly independent equations, i.e.  $\gamma$  is the rank of the matrix associated to the system of linear equations, and we set

$$\mu = \frac{\gamma}{\alpha} = \text{proportion of the linear equations that are independent.}$$

The XL algorithm will succeed if  $\beta = \gamma$  (or at least if  $\beta \approx \gamma$ ), i.e. if  $\alpha \cdot \mu \approx \beta$ . A straightforward analysis (cf. [4]) implies that we need to choose

$$D \gtrsim \frac{N}{\sqrt{\mu(N+1)}}.$$

In such a situation, the complexity of actually solving the resulting linear system is  $O(\beta^{2.3})$  using sparse linear algebra techniques.

#### 4.1 (# of variables) $\approx$ (# of independent equations)

We are assuming that we have more linear equations than variables. In this section we assume that the number of *independent* equations  $\gamma$  is approximately equal to the number of variables  $\beta$ , since if this does not happen, then we are unlikely to be successful. In other words, we study the situation when

$$\lambda = \frac{\gamma}{\beta} \approx 1.$$

If  $\lambda = 1$ , then the system has a unique solution, but in general this will not be the case. If  $\lambda < 1$ , then the system of equations does not uniquely determine the value of the variables. Instead, it determines a vector subspace  $V_\lambda$  of solutions whose dimension is

$$\dim(V_\lambda) = (\# \text{ of variables}) - (\# \text{ of indep. equations}) = \beta - \gamma = \beta(1 - \lambda).$$

The XL algorithm thus reduces the original problem to that of performing an exhaustive search through the vectors in the space  $V_\lambda$ , where

$$\#V_\lambda = 2^{\dim(V_\lambda)} = 2^{\beta(1-\lambda)}. \tag{10}$$

$N$	$d_f$	$D$	$\beta$	$\beta^{2.3}$	$\frac{1}{\sqrt{N}} \binom{N/2}{d_f/2}$	$\beta^{2.3} = 2^{\beta(1-\lambda)}$
200	68	14	$\approx 10^{22}$	$\approx 10^{49}$	$\approx 10^{26}$	$\lambda \approx 1 - 10^{-20}$
400	134	20	$\approx 10^{34}$	$\approx 10^{77}$	$\approx 10^{53}$	$\lambda \approx 1 - 10^{-32}$
600	200	25	$\approx 10^{45}$	$\approx 10^{102}$	$\approx 10^{81}$	$\lambda \approx 1 - 10^{-42}$
800	268	28	$\approx 10^{52}$	$\approx 10^{119}$	$\approx 10^{108}$	$\lambda \approx 1 - 10^{-49}$
1000	334	32	$\approx 10^{61}$	$\approx 10^{139}$	$\approx 10^{136}$	$\lambda \approx 1 - 10^{-58}$
1200	400	35	$\approx 10^{68}$	$\approx 10^{156}$	$\approx 10^{163}$	$\lambda \approx 1 - 10^{-65}$
1400	468	37	$\approx 10^{74}$	$\approx 10^{169}$	$\approx 10^{191}$	$\lambda \approx 1 - 10^{-71}$

**Table 1.** Complexity Comparison

*Remark 1.* If there is a meet-in-the-middle search, the exponent in (10) should be divided by 2. We also note that in small numerical examples, we often found that back substitution yielded parts of the desired solution even when the solution space was large. See Remark 2 below.

Table 1 gives, for various values of  $N$  and  $d_f$ , the smallest value of  $D$  such that  $\alpha > \beta$ . In other words, assuming that  $\lambda \approx 1.0$ , the table gives the smallest value of  $D$  such that the XL method has a chance of recovering the secret key. The table also gives the associated value of  $\beta$ , the resulting  $O(\beta^{2.3})$  linear system complexity, the comparable complexity value (1) for the meet-in-the-middle attack of [8, 16], and the value of  $\lambda$  necessary to make the search space complexity  $2^{\beta(1-\lambda)}$  from (10) equal to the linear system complexity  $O(\beta^{2.3})$ . In other words, the value of  $\lambda$  in the last column is the solution to  $\beta^{2.3} = 2^{\beta(1-\lambda)}$  and represents the proportion of linear equations that need to be independent in order to make the complexity of finding the solution space  $V_\lambda$  approximately the same as the complexity of searching through the solution space. (But see Remark 2 which suggests that somewhat smaller values of  $\lambda$  may still be useful.)

Assuming  $\lambda = 1$ , then Table 1 shows that eventually the algebraic XL-based approach is more efficient than the previously best known deterministic attack. This is still true if we only have  $\lambda \approx 1$ , but the last column of the table suggests that the difference  $1 - \lambda$  needs to be extremely small. In other words, we need an extremely high proportion of the equations to be independent in order to succeed. We further note that, even under the most favourable (and unlikely) assumption that  $\lambda = 1$ , the data in Table 1 shows that the advantage of the XL-based approach only occurs for  $N > 1000$ , which is considerably larger than any value used in practise [1, 2, 7].

## 4.2 Experimental Evaluation of $\lambda$

The validity of the runtime estimates derived in Section 4.1, especially as given in Table 1, depends on the assumption that  $\lambda$  is very close to 1 when we choose a value for  $D$  for which  $\alpha > \beta$ . In order to investigate this assumption, we carried out the following experiments.

For various small values of  $N$  and various values of  $d_f$  and  $d_g$ , we generated NTRU keys and formed the set of  $N+2$  inhomogeneous quadratic equations in  $N$

$N$	$D$	dim	$\lambda_{\min}$	$\lambda_{\max}$	$\bar{\lambda}$
11	3	561	0.96	1.00	0.99
13	3	1092	0.92	1.00	0.97
17	4	9401	0.91	1.00	0.99
19	4	16663	0.87	1.00	0.97
23	4	44551	0.95	0.96	0.95

**Table 2.** Experimental values of  $\lambda$

variables described in Section 3. We applied XL linearisation with increasing values of  $D$  until we obtained a value of  $D$  that made  $\alpha > \beta$ . We then computed the rank  $\gamma$  of the resulting matrix and the ratio  $\lambda = \gamma/\beta$  of independent equations to variables. Unfortunately, the size of the matrices grew very rapidly, so we were only able to perform experiments with small values of  $N$ .

For each value of  $N$ , we generated 10 keys. Table 2 gives the results of our experiments. The table lists the minimum, maximum, and mean values of  $\lambda$  over the 10 experiments, as well as the value of  $D$  required and the column dimension of the eventual linear system, i.e. the number of columns in the matrix.

*Remark 2.* Although often we were unable to obtain a system of full rank, it is an interesting experimental observation that in most cases we were able to recover a significant proportion of the underlying NTRU private key using back substitution. In other words, although we expect to have to search the entire space of solutions to find the NTRU private key, it happened that many of the coefficients of the private key were already determined. We do not know to what extent this is an artifact of our use of small values of  $N$  and to what extent it might continue to hold for cryptographically interesting values of  $N$ . In any case, it is a topic that merits further study.

We also considered a variant of the XL algorithm called FXL. In FXL, one fixes the value of some of the the variables and then applies the XL algorithm to the remaining variables. (This may be compared with the zero-forcing technique of May [10].) The effect of FXL is to reduce the number of variables whilst maintaining the number of equations. Thus not only does the resulting system of linear equations have smaller dimension, but one might also expect the value of  $\lambda$  to be larger. We experimented with this variant by fixing one or two variables. Note that there are two cases, since either the fixed variables have been guessed correctly, or they have been guessed incorrectly.

Table 3 gives the results of our experiments fixing one of the private key variables. We give the values of  $\lambda_{\min}$ ,  $\lambda_{\max}$  and  $\bar{\lambda}$  for the cases that the guessed fixed value was correct. For the final column of Table 3, labelled  $\perp$ , we performed experiments in which the guessed fixed value was incorrect. The listed value gives the proportion of such experiments for which the resulting matrix was consistent, i.e. gave a solution. Thus the value in column  $\perp$  represents the probability that one would not be able to tell that an incorrect guess had been made for the fixed coordinate. In each row of Table 3, we performed 10 experiments with the correct guess and 10 experiments with the incorrect guess.

$N$	$D$	$c$	$\lambda_{min}$	$\lambda_{max}$	$\bar{\lambda}$	$\perp$
11	3	385	0.99	1.00	0.99	0.2
13	3	793	0.99	1.00	0.99	0.1
17	3	2516	0.80	0.95	0.93	0.9
19	4	12615	0.92	1.00	0.99	0.2
23	4	35442	1.00	1.00	1.00	0.1

**Table 3.** Experimental  $\lambda$  values using FXL

We conducted a similar set of experiments in which we fixed (guessed) the value of two of the variables. However, the results were worse than when fixing one variable. Hence it appears that fixing variables does not lead to an improvement for our problem, at least for the small size of  $N$  considered here.

## 5 Conclusion

We have shown how the NTRU primitive can be reformulated, using Witt vectors, into a system of overdetermined nonlinear (quadratic) equations over  $\mathbb{F}_2$ . We have thus introduced the tool of Witt vectors into the field of cryptographic analysis. We have studied how one can attempt to solve this nonlinear system by reducing it to a much larger system of linear equations using the XL algorithm. We note that the analysis of the XL algorithm is itself quite controversial and that its effectiveness is not well understood. We performed experiments using equations generated from the NTRU primitive with small parameters and found that the rank of the eventual XL linear system is a good, but not a perfect, indicator of success.

If the XL algorithm behaves as claimed by its inventors and if it turns out either that the resulting linear systems have very close to full rank or that almost full rank is not actually necessary for success, then the approach described in this paper provides the asymptotic best known deterministic attack against NTRU. However, even under the most favourable assumptions, our method is less efficient than known methods at the largest current recommended parameter values. We have found no evidence to support the assertion that XL performs as claimed by its inventors, nor evidence to refute their claims. Further research on the applicability and efficiency of the XL algorithm is thus warranted, as is further research on the other questions raised in this paper.

## References

1. Consortium for Efficient Embedded Security. *Efficient embedded security standards #1: Implementation aspects of NTRU and NSS, Version 1*, 2002.
2. NTRU CryptoLab. *Challenge Problems* Available from <http://www.ntru.com/cryptolab/challenges.htm>.
3. N. Courtois, A. Klimov, J. Patarin and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. *Advances in Cryptology – EUROCRYPT 2000*, Springer-Verlag LNCS 1807, 392–407, 2000.

4. N. Courtois and J. Patarin. About the XL algorithm over  $GF(2)$ . *CT-RSA 2003*, Springer-Verlag LNCS 2612, 141–157, 2003.
5. N. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. *Advances in Cryptology – ASIACRYPT 2002*, Springer-Verlag LNCS 2501, 267–287, 2002.
6. C. Gentry. Key recovery and message attacks on NTRU-composite. *Advances in Cryptology – EUROCRYPT 2001*. Springer-Verlag LNCS 2045, 182–194, 2001.
7. J. Hoffstein, J. Pipher and J.H. Silverman. NTRU: a ring-based public key cryptosystem. *Algorithmic number theory – ANTS III*, Springer-Verlag LNCS 1423, 267–288, 1998.
8. N. Howgrave-Graham, J.H. Silverman and W. Whyte. A meet-in-the-middle attack on an NTRU private key. NTRU Cryptosystems Technical Report #004, Version 2, June 2003.
9. X. Lai. On the design and security of block ciphers. ETH Series in Information Processing, 1992.
10. A. May and J.H. Silverman. Dimension reduction methods for convolution modular lattices. *Cryptography and Lattices*, Springer-Verlag LNCS 2146, 110–125, 2001.
11. S. Murphy and M. Robshaw. Comments on the security of the AES and the XL technique. Unpublished Manuscript, 2002.
12. R.L. Rivest. The RC5 encryption algorithm. *Fast Software Encryption, Second International Workshop*, Springer-Verlag LNCS 1008, 86–96, 1995.
13. R.L. Rivest, M. Robshaw, R. Sidney and L. Yin. The RC6 block cipher. Submission to AES process, 1998.
14. J.-P. Serre. *Local Fields*. Springer-Verlag GTM 67, 1979.
15. P.W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, **26**, 1484–1509, 1997.
16. J.H. Silverman and A. Odlyzko. A Meet-In-The-Middle Attack on an NTRU Private Key. *Technical Report #004*, NTRU Cryptosystems, 1997.

## A The Ring of Witt Vectors over $\mathbb{F}_2$

In this appendix we briefly recall the construction of the ring of Witt vectors  $W_m(\mathbb{F}_2)$  of length  $m$ . The ring  $W_m(\mathbb{F}_2)$  is isomorphic to the ring of integers modulo  $2^m$ , but it has the useful property that the ring operations are described purely in terms of polynomials. For the general theory of Witt vectors, we refer the reader to [14, Section II.6].

A Witt vector of length  $m$  is simply an element of  $\mathbb{F}_2^m$ , i.e. a vector consisting of  $m$  components in  $\mathbb{F}_2$ . To turn this set into a ring, we need to define an addition law and a multiplication law. These operations are defined by the requirement that the map

$$\begin{aligned}
 W_* : W_m(\mathbb{F}_2) &\longrightarrow \mathbb{Z}_{2^m} \\
 [a_0, \dots, a_{m-1}] &\longmapsto \sum_{i=0}^{m-1} a_i 2^i \pmod{2^m}
 \end{aligned} \tag{11}$$

is a ring isomorphism.

Clearly, the operations in  $W_m(\mathbb{F}_2)$  will be more complicated than component-wise addition and multiplication, since we need to take carry bits into account.

$S_0(\mathbf{a}, \mathbf{b}) = a_0 + b_0$ $S_1(\mathbf{a}, \mathbf{b}) = a_0b_0 + a_1 + b_1$ $S_2(\mathbf{a}, \mathbf{b}) = a_0b_0a_1 + a_0b_0b_1 + a_1b_1 + a_2 + b_2$ $S_3(\mathbf{a}, \mathbf{b}) = a_0b_0a_1a_2 + a_0b_0a_1b_2 + a_0b_0b_1a_2 + a_0b_0b_1b_2$ $\quad + a_1b_1a_2 + a_1b_1b_2 + a_2b_2 + a_3 + b_3$
$P_0(\mathbf{a}, \mathbf{b}) = a_0b_0$ $P_1(\mathbf{a}, \mathbf{b}) = a_0b_1 + b_0a_1$ $P_2(\mathbf{a}, \mathbf{b}) = a_0b_0a_1b_1 + a_0b_2 + b_0a_2 + a_1b_1$ $P_3(\mathbf{a}, \mathbf{b}) = a_0b_0a_1b_1a_2 + a_0b_0a_1b_1b_2 + a_0b_0a_1b_1 + a_0b_0a_2b_2$ $\quad + a_0a_1b_1b_2 + a_0b_3 + b_0a_1b_1a_2 + b_0a_3 + a_1b_2 + b_1a_2$

**Table 4.** The first few Witt addition and multiplication polynomials

We thus have to find functions (multivariate polynomials)

$$S_0, \dots, S_{m-1}, P_0, \dots, P_{m-1} \in \mathbb{F}_2[X_0, \dots, X_{m-1}]$$

such that for all Witt vectors  $\mathbf{a} = [a_0, \dots, a_{m-1}]$  and  $\mathbf{b} = [b_0, \dots, b_{m-1}]$ , we have

$$\begin{aligned} W_*([S_0(\mathbf{a}, \mathbf{b}), \dots, S_{m-1}(\mathbf{a}, \mathbf{b})]) &\equiv W_*(\mathbf{a}) + W_*(\mathbf{b}) \pmod{2^m}, \\ W_*([P_0(\mathbf{a}, \mathbf{b}), \dots, P_{m-1}(\mathbf{a}, \mathbf{b})]) &\equiv W_*(\mathbf{a}) \cdot W_*(\mathbf{b}) \pmod{2^m}. \end{aligned}$$

The easiest way to compute the functions  $S_0, \dots, S_{m-1}$  and  $P_0, \dots, P_{m-1}$  is to use Witt polynomials. The  $i^{\text{th}}$  Witt polynomial  $W_i \in \mathbb{Z}[X_0, \dots, X_i]$  is defined by

$$W_i(X_0, \dots, X_i) = X_0^{2^i} + 2X_1^{2^{i-1}} + \dots + 2^i X_i. \quad (12)$$

One then proves [14] that there are unique polynomials

$$\varphi_0, \dots, \varphi_{m-1}, \psi_0, \dots, \psi_{m-1} \in \mathbb{Z}[X_0, \dots, X_{m-1}, Y_0, \dots, Y_{m-1}]$$

with the property that for all  $0 \leq i < m$ ,

$$\begin{aligned} W_i(\varphi_0, \dots, \varphi_i) &= W_i(X_0, \dots, X_i) + W_i(Y_0, \dots, Y_i), \\ W_i(\psi_0, \dots, \psi_i) &= W_i(X_0, \dots, X_i) \cdot W_i(Y_0, \dots, Y_i). \end{aligned} \quad (13)$$

The polynomials  $S_i$  for addition and  $P_i$  for multiplication may then be recovered as  $S_i = \varphi_i \pmod{2}$  and  $P_i = \psi_i \pmod{2}$ . The first few addition and multiplication polynomials are listed in Table 4.

*Example 2.* Let  $m = 4$  and consider the Witt vectors  $\mathbf{a} = [1, 0, 1, 1]$  and  $\mathbf{b} = [1, 1, 1, 0]$ . Then we have  $W_*(\mathbf{a}) \equiv 13 \pmod{16}$  and  $W_*(\mathbf{b}) \equiv 7 \pmod{16}$ , so  $W_*(\mathbf{a}) + W_*(\mathbf{b}) \equiv 4 \pmod{16}$  and  $W_*(\mathbf{a}) \cdot W_*(\mathbf{b}) \equiv 11 \pmod{16}$ . Computing the functions  $S_0, \dots, S_3$  and  $P_0, \dots, P_3$ , we indeed obtain

$$\begin{aligned} [S_0(\mathbf{a}, \mathbf{b}), \dots, S_3(\mathbf{a}, \mathbf{b})] &= [0, 0, 1, 0] && \text{with } W_*([0, 0, 1, 0]) = 4, \quad \text{and} \\ [P_0(\mathbf{a}, \mathbf{b}), \dots, P_3(\mathbf{a}, \mathbf{b})] &= [1, 1, 0, 1] && \text{with } W_*([1, 1, 0, 1]) = 11. \end{aligned}$$