A MULTIPLE BIRTHDAY ATTACK ON NTRU

Raphael Overbeck*
EPFL, LASEC, Building INF - Station 14, 1015 Lausanne, Switzerland

Keywords: NTRU, Public key cryptography, birthday paradox.

Abstract: In this paper we view the possibilities to lance a multiple (iterative) birthday attack on NTRU. Recently Wagner's algorithm for the generalized birthday problem (Wagner, 2002) allowed to speed-up several combinatorial attacks. However, in the case of NTRU we can not hope to to apply Wagner's algorithm directly, as the search space does not behave nicely. In this paper we show that we can nevertheless draw profit from a multiple birthday approach. Our approach allows us to attack `ees251ep6` parameter set on a computer with only 2^{52} Bits of memory and about 2^9 times faster as with Odlyzko's combinatorial attack – this is an improvement factor about 2^{43} in space complexity. We thus contradict the common believe, that in comparison to computational requirements, the "storage requirement is by far the larger obstacle" (Howgrave-Graham, 2007) to attack NTRU by combinatorial attacks. Further, our attack is about 2^7 times faster than the space-reduced variant from (Howgrave-Graham, 2007) employing the same amount of memory.

1 INTRODUCTION

The asymmetric NTRU encryption system (NTRU) (Hoffstein et al., 1998) is a well known cryptosystem, which to our knowledge is secure for large parameter sets, e.g. with $N \approx 500$. For small parameter sets, however, NTRU was subject to several attacks connected to lattice theory like attacks on alternative keys (Coppersmith and Shamir, 1997), exploitation of decryption errors (Howgrave-Graham et al., 2003) and the ones using dimension reduced lattices or zero-forcing (May and Silverman, 2001; Silverman, 1999). These attack exploit the fact, that the private NTRU key is presumably the shortest vector in a lattice which can be deduced from the public NTRU key. Besides lattice-based attacks, there exists a combinatorial attack originally due to Odlyzko, see e.g. (Howgrave-Graham, 2007). After several iterations of attacks and countermeasures, NTRU was considered for standardization (P1363.1/D9, 2003). In that proposal, parameters of NTRU are chosen such that the combinatorial attack is (theoretically and practically) the fastest one. This is the case if the NTRU parameters have $p = 2$ with a small d_f.

1.1 Our Contribution

We concentrate on adapting the attacks on NTRU to machines with limited storage capacity and thus allowing distributed attacks on NTRU. Evaluating the possible applications of an multiple birthday attack on NTRU, we reduce the memory requirements of a combinatorial-only attack. This is an important issue: N. Howgrave-Graham states in (Howgrave-Graham, 2007) that the large storage requirements of Odlyzko's attack is "by far the larger obstacle" than the runtime for attacking NTRU with today's hardware.

A direct application of the standard solution for generalized birthday paradox to NTRU is not possible: The probability, that the secret NTRU vector remains in the search space during the iterations of Wagner's algorithm is too small to allow an efficient attack. We highlight this problem and present a workaround which keeps track of the success probability during the attack.

To perform a multiple birthday attack, we split the secret NTRU vector $\mathbf{f}$ into eight parts instead of two in Odlyzko's attack. Further, we guess a permutation of the positions of the secret $\mathbf{g}$, such that the first positions of $\mathbf{g} = \mathbf{f}\mathsf{H}$ are zero, where H is the public NTRU key. Since there are many ways to split up the secret vector $\mathbf{f}$ into eight parts, we can search the space of possible solutions by an iterative birthday approach:

*This work was funded by DFG grant OV 102/1-1

Overbeck R. (2008).
A MULTIPLE BIRTHDAY ATTACK ON NTRU.
In *Proceedings of the International Conference on Security and Cryptography*, pages 237-244
DOI: 10.5220/0001916602370244

We search for those splits of $\mathbf{f}$, which lead to a $\mathbf{g} = \mathbf{f}\mathsf{H}$ with the first positions zero. This can be done by generating first a list of vectors of weight one fourth of the weight of $\mathbf{f}$ with the first $\ell^{[1]}$ positions zero via the birthday approach. Then, we can search the sum of pairs of such vectors, which are zero on the first $\ell > \ell^{[1]}$ positions and have the half weight of $\mathbf{f}$. In the last part of the attack, we can relax the "birthday" property, searching for those pairs of the latter vectors, which sum to a vector with the first $\ell + \mu$ positions binary. Balancing $\ell^{[1]}, \ell$ and μ, we can be sure, that the correct $\mathbf{f}$ is among the generated vectors. This way we obtain sets of almost the same size at each iteration and thus an attack, which requires much less memory than Odlyzko's attack. Further, such an attack is competitive with the fastest known NTRU attack (Howgrave-Graham, 2007) in terms of product of time and space and even better than the space reduced variant presented in the same paper.

1.2 Related Work

At CRYPTO 2007, N. Howgrave-Graham showed that the security level for the NTRU parameters proposed in (P1363.1/D9, 2003) is lower than intended (Howgrave-Graham, 2007). For his attack, Howgrave-Graham used a hybrid lattice-reduction and combinatorial attack against NTRU. By heuristic arguments he concludes, that he can attack the `ees251ep6` parameter set in $2^{76.2}$ modular additions on a machine with $2^{65.6}$ bits of memory or in $2^{89.2}$ modular additions on a machine with $2^{53.6}$ bits of memory. Unfortunately, so far we are not able to combine both approaches since Howgrave-Graham uses the concept of **s**-admissible vectors, which prevents an iterative birthday approach in the search part of his attack.

1.3 Organization

In the next section we recall NTRU and the basic notations. Then, we revise the generalized birthday paradox and view it's application to NTRU. To conclude, we give numbers and a comparison to the other attacks on NTRU.

2 PRELIMINARIES

In this paper we view only integer lattices, i.e. subvector spaces of $\mathbb{Z}^N$. We will call $\mathrm{wt}(\mathbf{f})$ the *(Hamming) weight* of a vector $\mathbf{f} \in \mathbb{Z}^N$, which corresponds to the number of non-zero entries in $\mathbf{f}$. If J is a subset of the positions of $\mathbf{f}$, we write $\mathbf{f}_J = (\mathbf{f}_i)_{i \in J}$. For an introduction into lattice theory see (Micciancio and Goldwasser, 2002).

NTRU according to (P1363.1/D9, 2003) works as follows: System parameters are three primes N, q and $p = 2$. NTRU uses the ring $\mathcal{R} = \mathbb{Z}[X]/(X^N - 1)$. The elements of that ring are identified with their unique representations in $\mathbb{Z}[X]$ of degree less than N. We will denote as weight $\mathrm{wt}(f)$ of a polynomial $f \in \mathcal{R}$ the number of non-zero coefficients. The NTRU secret key are two binary polynomials $f, g \in \mathcal{R}$ of weight d_f, d_g respectively. There are various variants of NTRU. In this paper we concentrate on the one, where the public key is given as $h = (f^{-1}g \mod q)$, where "mod q" means reduction of the coefficients modulo q. All attacks on one variant of NTRU may usually be adapted for other variants.

A description how NTRU en- and decryption work can be found, e.g., in (Hoffstein et al., 1998; Howgrave-Graham et al., 2003). However, since attacks on NTRU ciphertexts usually can be adapted to attack the secret keys and vice versa, this paper deals with attacks on the secret NTRU keys, only. We thus omit giving details on en- and decryption.

The **NTRU lattice** is obtained from a matrix representation of multiplications in $\mathcal{R}$. We can easily deduce a (cyclic) matrix $\mathsf{H} \in \mathbb{F}_q^{N \times N}$ representing the multiplication of polynomials with h in $\mathcal{R}$. With the N-dimensional identity matrix Id_N we obtain:

$$\mathsf{C}_{\mathrm{NTRU}} := \mathbf{f} \left[\begin{array}{c|c} \mathrm{Id}_N & \mathsf{H} \end{array} \right] \equiv (\mathbf{f}, \mathbf{g}) \mod q \qquad (1)$$

for the coefficient vectors $\mathbf{f}, \mathbf{g}$ of f and g. Note that $\left[\begin{array}{c|c} \mathrm{Id}_N & \mathsf{H} \end{array} \right]$ defines a double-cyclic code over $\mathbb{F}_q$. To obtain the NTRU lattice out of the matrix $\left[\begin{array}{c|c} \mathrm{Id}_N & \mathsf{H} \end{array} \right]$, vectors allowing the reduction of $\mathbf{f}\mathsf{H}$ modulo q are added and in some cases, $\mathbf{f}$ is scaled by an α:

$$\mathsf{L}_{\mathrm{NTRU}} := \left[\begin{array}{c|c} \alpha \mathrm{Id}_N & \mathsf{H} \\ \hline 0 & q\mathrm{Id}_N \end{array} \right] \qquad (2)$$

According to the Gauss-heuristic, $(\alpha\mathbf{f}, \mathbf{g})$ can be assumed to be the shortest vector in the NTRU lattice if α is properly chosen (up to double-cyclic shifts). Most attacks aim to find this vector either by lattice reduction or by a combinatorial approach. In the following we will take $\alpha = 1$, as this is a suitable value.

3 THE GENERALIZED BIRTHDAY PARADOX

Many combinatorial attacks could be sped-up by Wagner's solution for the generalized Birthday para-

dox. Wagner's main theorem (Wagner, 2002) can be summarized as follows:

Theorem 3.1 (q-Generalized Birthday Problem). *Let $r,a \in \mathbb{N}$ with $(a+1)|r$ and the sets $\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_{2^a} \subseteq \mathbb{F}_{q^r}$ be of cardinality $q^{\frac{r}{a+1}}$, then, a solution of the equation*

$$\sum_{i=1}^{2^a} x_i = \mathbf{0} \text{ where } x_i \in \mathcal{L}_i, \tag{3}$$

can be found in $\mathcal{O}(2^a q^{\frac{r}{a+1}})$ operations (over $\mathbb{F}_{q^r}$).

The algorithm proposed by Wagner is iterative: Fist, he searches for partial collisions of the sets $\mathcal{L}_i$ and $\mathcal{L}_{i+2^{a-1}}$, $i = 1, \cdots, 2^{a-1}$, that is, such pairs $(x_i, x_{i+2^{a-1}})$ that $LSB_{\frac{r}{a+1}}(x_i + x_{i+2^{a-1}}) = 0$. This way, one obtains 2^{a-1} Lists with approximately $2^{\frac{r}{a+1}}$ pairs where the last $\frac{r}{a+1}$ entries are zero and can be omitted in the next step. A recursive application of this step leads to a solution of Equation (3).

Proof. (Theorem 3.1). Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two lists of n-vectors with q^ℓ elements, then $\mathcal{L}_1 \times \mathcal{L}_2$ contains about $q^{2\ell}$ elements $(\mathbf{x}_1, \mathbf{x}_2)$ and thus about q^ℓ elements with $(\mathbf{x}_1 + \mathbf{x}_2)_{\{1,\cdots,\ell\}} = \mathbf{0}$. We can generate the latter elements as follows: We sort the elements of $\mathcal{L}_1$ and $\mathcal{L}_2$ in lexicographic order, which takes $\mathcal{O}(q^\ell \log(q^\ell))$ operations.[2] Now, we can for each element $\mathbf{x}_1 \in \mathcal{L}_1$ look up the elements in $\mathcal{L}_2$ with the same value, which takes $\mathcal{O}(|\mathcal{L}_1| \cdot \log|\mathcal{L}_2|) = \mathcal{O}(q^\ell \log(q^\ell))$ operations again. We can apply the argument iteratively, which concludes the proof. ■

In general, the q-generalized birthday paradox allows to find one solution among many possible in quite efficient time: The set of elements is $\mathcal{L}_1 \times \mathcal{L}_2 \times \cdots \times \mathcal{L}_{2^a}$ and thus of size $q^{2^a r/(a+1)}$ with about $q^{2^a/(a+1)}$ solutions of Equation 3. By Wagner's algorithm, we can find one of these solutions in $\mathcal{O}(2^a q^{r/(a+1)})$ operations instead of $\mathcal{O}(q^{r/2})$ operations with the standard birthday attack.

A direct application to NTRU is not possible. Let us view for example the `ees251ep6` parameter set $(N = 251, q = 197, d_f = 48, d_g \approx N/2)$. Here we could try to set $a = 3$ and

$$\mathcal{L}_i = \left\{ \mathbf{x}\mathsf{H} \mid \mathbf{x} \in \{0,1\}^N, \mathrm{wt}(\mathbf{x}) = df/8 = 6 \right\}.$$

[2] Here and in the following, any criteria for sorting is valid. One could, e.g., take lexicographic ordering or maybe the evaluation of a non-cryptographic hash. We will assume that sorting a list $\mathcal{M}$, e.g. by "Smoothsort", costs the same time as computing the sorting criteria for each element. However, in the worst case, sorting costs $\mathcal{O}(|\mathcal{M}| \cdot \log|\mathcal{M}|)$ operations.

We get that $|\mathcal{L}_i| = \binom{N}{6} \approx q^5 \approx 2^{41}$ and could thus hope to generate binary vectors $\mathbf{f}$, such that $\mathbf{f} \cdot \mathsf{H}$ is zero at some $20 = 5 \cdot (a+1)$ positions by the q-generalized birthday paradox. In a random lattice, we would expect, that there are about 2^{20} such vectors, so that we could assume, that we can find the secret NTRU vector with probability 2^{-20}, if the chosen 20 positions of $\mathbf{g}$ are zero. However, this is not true: There are $\binom{df}{df/2}$ ways to split the secret NTRU vector $\mathbf{f}$ into two parts $\mathbf{f} = \mathbf{x}_1 + \mathbf{x}_2$ of weight $df/2$. Thus, the probability that there is a $\mathbf{x}_1$, such that $\mathbf{x}_1 \cdot \mathsf{L}_{\mathrm{NTRU}}$ is zero at the last 10 position is only $\binom{df}{df/2} \cdot q^{-10} < 2^{-31}$. The probability, that such a $\mathbf{x}_1$ splits nicely again is $\binom{df/2}{df/4} \cdot q^{-5} < 2^{-16}$, turning an attack impossible. In the next section we will explain how to work around this problem.

4 A MULTIPLE BIRTHDAY ATTACK FOR NTRU

To apply a multiple birthday attack to NTRU, we have to ensure that at each iteration there is at least one element in the search space, which leads to the secret vector $\mathbf{g}$. By multiplying $\mathsf{L}_{\mathrm{NTRU}}$ with a permutation matrix P we can assume without loss of generality that the first positions of $\mathbf{g}$ are zero. Our goal is to generate a list $\mathcal{L}$ of N-vectors $\mathbf{f}$ of weight d_f with the first $\ell \in \mathbb{N}$ positions of $\mathbf{g} = \mathbf{f}\mathsf{H}$ zero and the next $\mu \in \mathbb{N}$ positions binary by applying the birthday paradox iteratively like in Wagner's algorithm. However, in the NTRU case, numbers and probabilities do not behave nicely, so that we can not apply the generalized birthday paradox directly – as explained in the previous section. Nevertheless, we can apply the principle.

The key issue of our attack is to balance parameters in a way, such that with sufficient probability, the secret NTRU vector stays in the search space at each stage of the multiple birthday attack. For the ease of presentation we will assume that $8 \mid d_f$.

4.1 An Approach with Symmetric Sets

We will denote with $\mathbf{x}_1, \mathbf{x}_2$ a split of $\mathbf{f}$ into two vectors of weight $d_f/2$ and $\mathbf{x}_1 + \mathbf{x}_2 = \mathbf{f}$, then we will split up these vectors into smaller parts until we have split $\mathbf{f}$ into 8 parts, see Figure 4.1.

Each part $\mathbf{x}_i^{[2]}$ is in the set

$$\mathcal{L}_i^{[2]} := \left\{ \mathbf{x}^{[2]} \in \{0,1\}^N \mid \mathrm{wt}(\mathbf{x}^{[2]}) = d_f/8 \right\},$$

$i = 1, \cdots, 8$. As all sets $\mathcal{L}_i^{[2]}$ look the same, we use the term "symmetric" sets. The principle how to generate $\mathcal{L}$ out of the $\mathcal{L}_i^{[2]}$ is given in Figure 4.2.

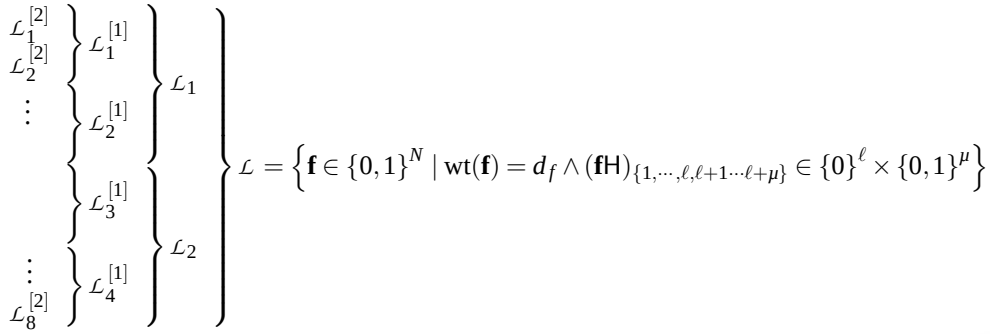

Figure 4.2: Scheme of the multiple birthday attack.

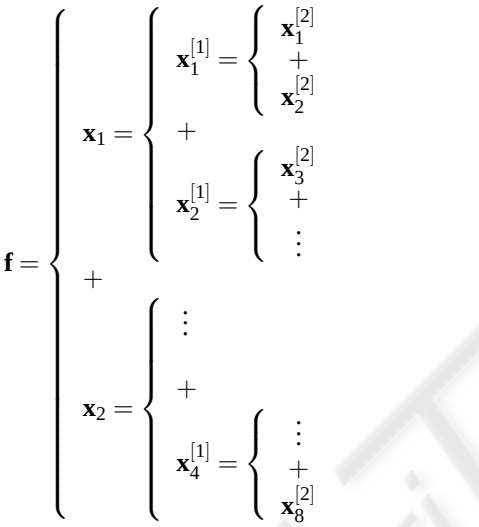

Figure 4.1: Splitting the secret NTRU vector.

We have to assure, that at each step the probability that the secret NTRU vector remains in the search space is not too low. It is obvious, that there are $\mathbf{x}_i^{[2]} \in \mathcal{L}_i^{[2]}$, $i = 1,\cdots,8$ such that $\mathbf{f} = \sum_{i=1}^{8} \mathbf{x}_i^{[2]}$.

In the first iteration we want to generate the sets $\mathcal{L}_i^{[1]}$, $i = 1,\cdots,4$ with $|\mathcal{L}_i^{[1]}| \approx |\mathcal{L}_i^{[2]}|$ such that there are $\mathbf{x}_i^{[1]} \in \mathcal{L}_i^{[1]}$ of weight $d_f/4$ with $\sum_{i=1}^{4} \mathbf{x}_i^{[1]} = \mathbf{f}$. To do so, we choose $0 \le \ell^{[1]} \le \ell$ and set

$$\mathcal{L}_i^{[1]} = \left\{ \mathbf{x}^{[1]} \in \{0,1\}^N \;\middle|\; \begin{array}{l} \mathrm{wt}(\mathbf{x}^{[1]}) = d_f/4 \\ \wedge (\mathbf{x}^{[1]}\mathsf{H})_{\{1,\cdots,\ell^{[1]}\}} = \mathbf{0} \end{array} \right\},$$

$i = 1,\cdots,4$. Note, that $\mathbf{x}_1^{[1]} + \mathbf{x}_2^{[1]}$ is a split of $\mathbf{x}_1$, where $\mathbf{f} = \mathbf{x}_1 + \mathbf{x}_2$ is a split of $\mathbf{f}$. There are $\binom{d_f/2}{d_f/4}$ possible $\mathbf{x}_1^{[1]}$. Thus, the expected number of valid $\mathbf{x}_i^{[1]}$ contained in $\mathcal{L}_i^{[1]}$ is

$$\binom{d_f/2}{d_f/4} / q^{\ell^{[1]}} \tag{4}$$

and should be larger than one. Like in Wagner's algorithm, $\mathcal{L}_i^{[1]}$ can be generated from $\mathcal{L}_{2i}^{[2]}$ and $\mathcal{L}_{2i-1}^{[2]}$ in time $O(|\mathcal{L}_{2i}^{[2]}| \log_2 |\mathcal{L}_{2i}^{[2]}|)$ by sorting techniques. In the second stage, we generate the sets

$$\mathcal{L}_i = \left\{ \mathbf{x} \in \{0,1\}^N \mid \mathrm{wt}(\mathbf{x}) = \frac{d_f}{2} \wedge (\mathbf{x}\mathsf{H})_{\{1,\cdots,\ell\}} = \mathbf{0} \right\},$$

$i = 1,2$, which should contain at least one of the vectors which sum up to $\mathbf{f}$. A element $(\mathbf{x}_1, \mathbf{x}_2)$ with $\mathbf{x}_1 + \mathbf{x}_2 = \mathbf{f}$ will be in $\mathcal{L}_1 \times \mathcal{L}_2$ with sufficient probability if we can expect that there is an element $\mathbf{x}_1 \in \mathcal{L}_1$, which matches $\mathbf{f}$ in $d_f/2$ positions. This is the case if the expected number

$$\binom{d_f}{d_f/2} / q^{\ell}. \tag{5}$$

is larger than one. Thus, we have to choose $\ell^{[1]}$ and ℓ in respective to the Equations (5) and (4) – however, there is no need to choose $2\ell^{[1]} = \ell$ like in Wagner's algorithm. To identify $\mathbf{f}$ by the last application of the birthday paradox, we do not longer search for exact collisions on a subset of the positions of $\mathbf{f} \cdot \mathsf{H}$ but for a binary collision in the first $\ell + \mu$ positions, i.e. those elements $(\mathbf{x}_1, \mathbf{x}_2) \in \mathcal{L}_1 \times \mathcal{L}_2$, such that $(\mathbf{x}_1 + \mathbf{x}_2) \cdot \mathsf{H} \in \mathcal{L}$, as defined in Figure 4.2 (otherwise $\mathbf{f} \neq \mathbf{x}_1 + \mathbf{x}_2$).

The cost of the presented multiple birthday attack is summarized in the theorem below, which we proof in the appendix. In the theorem, $\mathcal{W}_1$ represents the costs to sort the sets $\mathcal{L}_i^{[2]}$, $\mathcal{W}_2$ the cost, to generate the $\mathcal{L}_i^{[1]}$ and $\mathcal{W}_3$ the number of vector operations needed to generate the sorted sets $\mathcal{L}_i$. Finally, $\mathcal{W}_4$ gives the cost to generate $\mathcal{L}$ and search it for the secret NTRU vector.

Theorem 4.1. *Assume that $\ell^{[1]}, \ell$ and μ are such that $|\mathcal{L}_2|/q^{\mu}$ and the terms of the Equations (5) and (4) are larger than* 1*. Then, the iterative birthday attack on*

NTRU can be performed in

$$\mathcal{O}\Big(\underbrace{|\mathcal{L}_1^{[2]}| \cdot \log_2 |\mathcal{L}_1^{[2]}| \cdot d_f/8}_{W_1} + \underbrace{(1+\log_2\left(\frac{|\mathcal{L}_1^{[2]}|^2}{q^{\ell^{[1]}}}\right)) \cdot \frac{|\mathcal{L}_1^{[2]}|^2}{q^{\ell^{[1]}}}}_{W_2} + \underbrace{\left(1+\log_2\left(\frac{|\mathcal{L}_1^{[1]}|^2}{q^{\ell-\ell^{[1]}}}\right)\right)\frac{|\mathcal{L}_1^{[1]}|^2}{q^{\ell-\ell^{[1]}}}}_{W_3} + \underbrace{(\log_2|\mathcal{L}_1|+d_f)\,\frac{2^{NGN}}{q^{\mu}}|\mathcal{L}_1|^2}_{W_4} \Big) \tag{6}$$

vector operations, requires a memory of bit size $\mathcal{O}((N+(\ell+\mu)\log_2(q)) \cdot (|\mathcal{L}_i^{[2]}|+|\mathcal{L}_i^{[1]}|+|\mathcal{L}_i|))$ *and succeeds with probability* $N \cdot 2^{-\ell}$.

We get the complexity of the multiple birthday attack in binary operations from Equation (6) by multiplying W_1, W_2 and W_3 with $N+\log_2(q)\cdot(\ell+\mu)$ and by replacing $(\log_2|\mathcal{L}_1|+d_f)$ in W_4 with $\log_2|\mathcal{L}_1| \cdot (\mu\log_2(q)+N)+d_f \cdot N\log_2(q)$. The difference of the factors results from the different sizes of the vectors used in each set. So far, we treated only the case, where the term in (5) was larger than 1. However, we can permit smaller numbers as well, which lowers the success probability of the algorithm. Please note, that if $|\mathcal{L}_2|/q^{\mu} < 1$, the number of lookups in $\mathcal{L}_2$ remains the same, while the number of vector additions in the last step decreases with the expected number of $\mathbf{x}_2$ matching the $\mathbf{x}_1$.

Table 4.1 gives intuition of some parameter choices and the expected sizes of the sets generated during the attack.

4.2 Experimental Results

We fully implemented the attack and made various experiments for small parameter sets. Our experiments corroborate the numbers from Theorem 4.1. For the toy example $N=53, q=37, d_f=16, d_g \approx N/2$ (compare (Howgrave-Graham, 2007)) our attack generated lists of maximal 2^{14} vectors, needed 2^{20} vector operations and had a success probability of $1/4$.

4.3 Comparison with Other Attacks

The hybrid lattice reduction/combinatorial attack presented at CRYPTO 2007(Howgrave-Graham, 2007) by Howgrave-Graham performs a BKZ lattice reduction first and then tries to find the secret key in the reduced lattice. Howgrave-Graham presents as well a space-reduced variant of his attack which consists in additional guessing of some structure in the secret key. Table 4.2 gives an overview of the performance of the hybrid attack (where we give the binary complexity rather than the complexity in operations over $\mathbb{Z}_q$ like in (Howgrave-Graham, 2007)). We can see, that a multiple birthday attack is competitive in measure of product of time and space. In comparison with the space reduced variant of the hybrid attack, the multiple birthday attack is even slightly better – requiring about half the memory and $\approx 2^7$ times less binary operations.

Table 4.3 gives an overview over Odlyzko's attack and shows that a multiple birthday attack clearly outperforms the standard combinatorial attack in time and space complexity.

For a better comparison, we highlight the different attacks for the `ees251ep6` Parameter set in Table 4.4.

5 CONCLUSIONS

The iterative application of the birthday paradox to the NTRU problem allows performing distributed combinatorial attacks on machines with a smaller storage capacity than previously. We achieve an attack, which is about 2^7 times faster than the space-reduced attack from (Howgrave-Graham, 2007) on machines of about the same size (2^{52} bits of memory). However, even with the achieved reduction of the memory size we are still not able to perform combinatorial attacks on a Desktop PC. Therefore and because we are not able to reduce the runtime of combinatorial attacks, our results do not affect NTRU parameters from (P1363.1/D9, 2003) in a practical or asymptotic sense.

5.1 Open Questions

So far were not able to combine the multiple birthday attack presented in this paper with the hybrid attack by Howgrave-Graham. We thus leave this question for further research.

To better "tune" a multiple birthday attack, i.e. to get $|\mathcal{L}_i^{[2]}| \approx |\mathcal{L}_i^{[1]}| \approx |\mathcal{L}_i| \approx |\mathcal{L}|$ we propose to use relaxed "birthday" conditions. So far, we considered only binary or zero birthdays, that is, we say that two vectors $\mathbf{x}, \mathbf{y}$ have a "birthday" on a position i, if $(\mathbf{x}-\mathbf{y})\mathsf{H}_i$ is binary or zero. Likewise one could define birthdays as $(\mathbf{x}-\mathbf{y})\mathsf{H}_i \in \{-a, \cdots, a\}$ for some $a < q/2$. We did not have time to check this, but one

Table 4.1: The multiple birthday attack on NTRU – Time in binary operations.

NTRU Parameters	Multiple Birthday attack								
(N, q, d_f, d_g)	$\ell^{[1]}, \ell, \mu$	$\lvert\mathcal{L}_i^{[2]}\rvert$	$\lvert\mathcal{L}_i^{[1]}\rvert$	$\lvert\mathcal{L}_i\rvert$	Eq. (4)	Eq. (5)	Eq. (6)	Time	Space
Toy example $(53, 37, 16, N/2)$	$1, 3, 3$	$2^{10.4}$	2^{13}	2^{14}	2	2^{-2}	$2^{21.3}$	2^{28}	$2^{20.5}$ bits
Toy example $(107, 67, 32, N/2)$	$2, 5, 5$	2^{22}	2^{26}	$2^{31.5}$	$2^{1.5}$	$2^{-1.1}$	$2^{43.3}$	2^{52}	2^{39} bits
`ees251ep6` $(251, 197, 48, N/2)$	$2, 8, 7$	2^{38}	2^{51}	2^{50}	2^{6}	2^{-16}	$2^{83.5}$	2^{87}	$2^{59.7}$ bits
`ees251ep6` $(251, 197, 48, N/2)$	$3, 9, 7$	2^{38}	$2^{43.5}$	2^{42}	$2^{-1.5}$	2^{-23}	$2^{84.2}$	2^{90}	2^{52} bits
`ees397ep1` $(397, 307, 74, N/2$	$4, 9, 14$	$2^{69.5}$	$2^{80.6}$	2^{83}	2	2^{-20}	2^{124}	$2^{138.4}$	2^{92} bits
`ees491ep1` $(491, 367, 91, N/2)$	$5, 14, 15$	2^{75}	$2^{86.7}$	$2^{95.4}$	$2^{-0.18}$	2^{-32}	2^{154}	2^{158}	2^{167} bits

Table 4.2: Performance of Howgrave-Graham's Attack – Time in binary operations.

Parameter set	Hybrid attack		
(N, q, d_f, d_g)	Space reduced	Time	Space
Toy example $(107, 67, 32, N/2)$	no	$2^{50.6}$	$2^{36.2}$ bits
`ees251ep6` $(251, 197, 48, N/2)$	no	$2^{83.8}$	$2^{65.6}$ bits
`ees251ep6` $(251, 197, 48, N/2)$	yes	$2^{96.8}$	$2^{53.6}$ bits

Table 4.3: Odlyzko's attack on NTRU – Time in binary operations.

Parameter set	Odlyzko's Attack		
(N, q, d_f, d_g)	# Vectors	Time	Space
Toy example $(53, 37, 16, N/2)$	2^{20}	$2^{31.1}$	$2^{28.4}$ bits
Toy example $(107, 67, 32, N/2)$	2^{44}	$2^{57.3}$	$2^{53.5}$ bits
`ees251ep6` $(251, 197, 48, N/2)$	$2^{84.2}$	$2^{99.7}$	$2^{95.2}$ bits
`ees397ep1` $(397, 307, 74, N/2$	2^{134}	$2^{147.0}$	2^{151} bits
`ees491ep1` $(491, 367, 91, N/2)$	$2^{166.5}$	2^{184}	$2^{178.7}$ bits

Table 4.4: Summary of available attacks on NTRU `ees251ep6`.

Attack	Time	Space	Time · Space
Odlyzko	$2^{99.7}$	$2^{95.2}$ bits	$2^{194.9}$
Hybrid	$2^{83.8}$	$2^{65.6}$ bits	$2^{149.4}$
Hybrid space reduced	$2^{96.8}$	$2^{53.6}$ bits	$2^{150.4}$
Multiple birthday	2^{90}	$2^{52.0}$ bits	$2^{142.0}$

might well achieve to reduce the storage requirements even more by this approach.

REFERENCES

Coppersmith, D. and Shamir, A. (1997). Lattice attacks on NTRU. *Proc. of Eurocrypt '97, LNCS*. Springer-Verlag.

Hoffstein, J., Pipher, J., and Silverman, J. (1998). NTRU: a ring based public key cryptosystem. *Proc. of ANTS III*, 1423 of LNCS:267–288, Springer–Verlag.

Howgrave-Graham, N. (2007). A hybrid lattice-reduction and meet-in-the-middle attack against ntru. In *Proc. of CRYPT'07*, volume 4622 of *Lecture Notes in Computer Science*, pages 150–169. Springer.

Howgrave-Graham, N., Nguyen, P., Pointcheval, D., Proos, J., Silverman, J., Singer, A., and Whyte, W. (2003). The impact of decryption failures on the security of NTRU encryption. *To appear in Proc. of CRYPTO '03, LNCS*, 2729:226–246. Springer-Verlag.

May, A. and Silverman, J. (2001). Dimension reduction methods for convolution modular lattices. *Proc. of CaLC 2001, LNCS*, 2146:111–127. Springer-Verlag.

Micciancio, D. and Goldwasser, S. (2002). *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts.

P1363.1/D9, I. (2003). Draft standard for public-key cryptographic techniques based on hard problems over lattices. W. Whyte (editor).

Silverman, J. (1999). Dimension reduced lattices, zero-forced lattices, and the NTRU public key cryptosystem. *NTRU Technical Report*, 013. available at www.ntru.com.

Wagner, D. (2002). A generalized birthday problem. In Yung, M., editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer.

APPENDIX

Odlyzko's Combinatorial Attack

Odlyzko proposed to randomly split $\mathbf{f}$ in two binary parts of weight $d_f/2$, say $\mathbf{f}_1, \mathbf{f}_2 \in \mathbb{F}_q^N$, such that $\mathbf{f}\mathsf{H} = \mathbf{f}_1\mathsf{H} + \mathbf{f}_2\mathsf{H}$. Now, it is sufficient to list all the $\binom{\lfloor N/2 \rfloor}{d_f/2}$ possible $\mathbf{f}_1\mathsf{H}_I$ with $\mathbf{f}_1$ of weight $d_f/2$ and check for each possible $\mathbf{f}_2$ (of weight $d_f/2$), if $\mathbf{f}_1\mathsf{H} + \mathbf{f}_2\mathsf{H}$ is binary. Because of the possible rotations there are about $N\binom{d_f}{d_f/2}$ correct choices for $\mathbf{f}_1$. By the birthday paradox, one correct pair $\mathbf{f}_1, \mathbf{f}_2$ can be found after approximately

$$\sqrt{\frac{1}{N} \cdot \binom{N}{d_f/2}^2 / \binom{d_f}{d_f/2}} \tag{7}$$

samples, where each can be generated in $N \cdot d_f/2$ additions modulo q. Here, a "collision" for the birthday paradox is characterized by the binary sum of two samples, which can be easily checked if the list of samples is sorted[3]. For the `ees251ep6` parameter set from (P1363.1/D9, 2003) ($N = 251$, $q = 197$, $d_f = 48$, $d_g = N/2$), this attack requires storing a list of $2^{84.2}$ vectors, compare Table 4.3.

Complexity of the Multiple Birthday Attack with Symmetric Sets

Proof. (Theorem 4.1). We fist observe, that the sets $\mathcal{L}_i^{[2]}$ are the same for each i. The same holds for the $\mathcal{L}_i^{[1]}$ and $\mathcal{L}_i$. In the following we assume, that in each list, we store as well the value $(\mathbf{f}\cdot\mathsf{H})_{\{1,\cdots,\ell+\mu\}}$ together with $\mathbf{f}$. Computing and sorting $\mathcal{L}_i^{[2]}$ after the lexicographic order of $(\mathbf{f}, \mathbf{f}\cdot\mathsf{H})_{\{1,\cdots,\ell^{[1]}\}}$ (most significant bit on the right) takes

$$W_1 := \tbinom{N}{d_f/8} \cdot \log_2(\tbinom{N}{d_f/8}) \cdot d_f/8$$

operations on vectors in $\mathbb{F}_2^N \times \mathbb{F}_q^{\ell+\mu}$. Storing these sets requires

$$M_1 := \tbinom{N}{d_f/8} \cdot (N + (\ell+\mu)\log_2(q))$$

bits. Generating the set(s) $\mathcal{L}_i^{[1]}$ takes

$$W_2 := \left(1 + \log_2\left(\tbinom{N}{d_f/8}^2 / q^{\ell^{[1]}}\right)\right)\tbinom{N}{d_f/8}^2 / q^{\ell^{[1]}}$$

operations, since for each element of $\mathcal{L}_{2i-1}^{[2]}$ we have about $\binom{N}{d_f/8}/q^{\ell^{[1]}}$ matching elements of $\mathcal{L}_{2i}^{[2]}$. Storing $\mathcal{L}_i^{[1]}$ requires

$$M_2 := \tbinom{N}{d_f/4}/q^{\ell^{[1]}} \cdot (N + (\ell+\mu)\log_2(q))$$

bits. Generating the set(s) $\mathcal{L}_i$ takes

$$W_3 := \left(1 + \log_2\left(\tbinom{N}{d_f/4}^2 / q^{\ell-\ell^{[1]}}\right)\right) \cdot \tbinom{N}{d_f/4}^2 / q^{\ell-\ell^{[1]}}$$

operations. The resulting lists have a size of

$$M_3 := \tbinom{N}{d_f/2}/q^{\ell} \cdot (N + \mu\log_2(q))$$

[3] In (Howgrave-Graham, 2007) it is proposed to use an ordering according to the signs of the entries if $\mathbf{f}_1\mathsf{H}$ is represented as a vector with entries in the interval $[-q/2, q/2]$. However, other sorting criteria can be used as well.

bits. On the construction of $\mathcal{L}_i$ we sort the lists directly after the last $\ell+\mu$ Positions of $\mathbf{f}\mathsf{H}$. We choose μ such that $|\mathcal{L}_i|/q^\mu$ is relatively small. To find the secret NTRU vector, for each element $\mathbf{x}_1 \in \mathcal{L}_1$, we search $\mathcal{L}_2$ for the elements $\mathbf{x}_2$, with a binary difference from $\mathbf{x}_1 \cdot \mathsf{H}$ in the last μ Positions, which requires 2^μ lookups each requiring $\log_2 |\mathcal{L}_2|$ operations on vectors in $\mathbb{F}_2^N \times \mathbb{F}_q^{\ell+\mu}$ plus $|\mathcal{L}_1|/q^\mu$ comparisons. As soon as we have found a binary $((\mathbf{x}_1+\mathbf{x}_2)\mathsf{H})_{\{1,\cdots,\ell+\mu\}}$, we check that $\mathbf{f}=\mathbf{x}_1+\mathbf{x}_2$ is binary and compute $\mathbf{g}=\mathbf{f}\mathsf{H}$. If $\mathbf{g}$ is binary, we have found the secret NTRU vector. This last step requires

$$W_4 := \left(\log_2\left(\binom{N}{d_f/2}/q^\ell\right)+d_f\right) \cdot \frac{2^\mu}{q^{NGN}} \cdot \binom{N}{d_f/2}^2/q^{2\ell}$$

operations on vectors in $\mathbb{F}_2^N \times \mathbb{F}_q^N$. Thus, the total workfactor for a multiple birthday attack on NTRU starting with $\mathcal{L}_i^{[2]}$ is $\sum_{i=1}^4 W_i$ and requires about $\sum_{i=1}^4 M_i$ bits of memory. The success probability results from assuming $\mathbf{g}$ to be zero in the first ℓ coordinates and the fact, that there are about N different cyclic shifts of $\mathbf{f}$, which can serve as secret vector. ■