On the Bit Security of NTRUEncrypt

Mats Näslund[1], Igor E. Shparlinski[2], and William Whyte[3]

[1] Ericsson Research
SE-16480 Stockholm, Sweden
mats.naslund@era.ericsson.se

[2] Department of Computing, Macquarie University
Sydney, NSW 2109, Australia
igor@ics.mq.edu.au

[3] NTRU Cryptosystems, Inc.
5 Burlington Woods, Burlington, MA 01803,USA
wwhyte@ntru.com

Abstract. We show that in certain natural computational models every bit of a message encrypted with the NTRUEncrypt cryptosystem is as secure as the whole message.

1 Introduction

Whereas in the past cryptography used to be performed by national agencies running mainframe computers, today we see a trend towards the use of cryptography in thinner and thinner clients such as cellular phones, PDAs, etc. For public key cryptography to be widely deployed on these devices, we require high performance algorithms to obtain service behavior that is acceptable to the end-users. When designing faster and faster encryption schemes, there is of course always a risk that security is sacrificed, and new, high-performance schemes need to be thoroughly analyzed. One promising recent high-performance scheme is the NTRUEncrypt cryptosystem [12].

It is quite natural that the performance, and more importantly, the security of NTRUEncrypt is often compared to other schemes such as RSA, elliptic curves, etc. Unfortunately, like many other practical schemes, NTRUEncrypt does not have a formal proof of security. On the other hand, the security appears to be closely based on some quite well-studied lattice problems, and to date, no serious attack on NTRUEncrypt is known.

Still, it would be worthwhile to investigate whether new schemes such as NTRUEncrypt enjoy security properties that are known to exist in the more "classical" schemes. For instance, it is known that RSA enjoys the property of *bit-security*: finding any single bit of the plaintext is (via polynomial time reductions) as hard as finding the whole plaintext, see [11]. Establishing the same property for NTRUEncrypt would indicate a certain robustness in the scheme, and this is the problem we address in this paper.

Adopting a new (stronger) computational model, we answer the question positively. While it cannot be debated that our model makes the proof not seeming too hard, it far from trivializes the problem. Also, it is interesting to compare our methods and those previously used for the RSA scheme. In particular, the above mentioned results for RSA are obtained exploiting a quite undesirable property of "raw" RSA: multiplicativity; $x^e a^e \equiv (ax)^e \pmod{N}$. To obtain our results on NtruEncrypt we similarly make use of the basic scheme's malleability; given the NTRU encryption of x and one or more bits of x, it is possible to compute an encryption of x', corresponding to x with said bit(s) flipped.

Finally, in same classical model as that used in [11] we also obtain a bit-security result, though much weaker.

2 Preliminaries

We recall that in the NtruEncrypt cryptosystem [12], one selects integer parameters (N, q) and four sets $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_\varphi, \mathcal{L}_m$ of polynomials f, g, φ, m in the ring $\mathcal{R} = \mathbb{Z}_q[X]/(X^N - 1)$. The coefficients of these polynomials are constrained by the choice of an additional parameter, p, a small integer or polynomial. In the original presentation of NtruEncrypt [12], p was taken to be the integer 3, and the polynomials f, g, φ, m were trinary. The authors of NtruEncrypt have since recommended [13] that p be taken to be the polynomial $2 + X$, and this is current practice in implementations [4]. This choice for p, along with other optimizations suggested in [13], leads to f, g, φ, m being constructed from binary polynomials. Of particular relevance to this paper is that it leads to the message representative polynomial m being binary.

We denote by $\odot$ the operation of multiplication in the ring $\mathcal{R}$.

We also recall the *key creation* procedure where the *receiver Alice* randomly selects polynomials $f \in \mathcal{L}_f$ and $g \in \mathcal{L}_g$ such that f has an inverse modulo q and modulo p (this issue has been discussed in [13,18] together with heuristic estimates for a polynomial $f \in \mathcal{L}_f$ to satisfy these conditions).

Then *Alice* computes inverses f_q^* and f_p^* that satisfy

$$f \odot f_q^* \equiv 1 \pmod{q}, \qquad f \odot f_p^* \equiv 1 \pmod{p},$$

and the product

$$h \equiv p \odot f_q^* \odot g \pmod{q}.$$

Alice publishes the polynomial h as her public key, retaining f as her private key. The polynomial f_p^* should also be stored for later use, and the polynomials f_q^*, g may be discarded. Note that the paper [13] recommends that f have the form $1 + p \odot F$ with $F \in \mathcal{R}$; in this case, of course, $f_p^* = 1$.

Suppose the *sender Bob* wants to *encrypt* and send a secret message to *Alice*. *Bob* selects a message m from the set of plaintexts $\mathcal{L}_m$. Next, *Bob* selects a random "blinding" polynomial $\varphi \in \mathcal{L}_\varphi$ and uses the public key h to compute

$$e \equiv \varphi \odot h + m \pmod{q}. \qquad (1)$$

Bob then transmits e to *Alice*.

We do not discuss the decryption procedure and thus the structure on the sets $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_\varphi, \mathcal{L}_m$ which maximizes the chance of unambiguous decryption (rather we refer to [12,13] for details). However the set $\mathcal{L}_m$ plays a crucial role and we recall that $\mathcal{L}_m$ consists of 2^N polynomials with $0, 1$-coefficients, that is

$$\mathcal{L}_m = \left\{ a_0 + a_1 X + \ldots + a_{N-1} X^{N-1} \mid a_0, a_1 \ldots, a_{N-1} \in \{0, 1\} \right\}.$$

Thus each coefficient of $m \in \mathcal{L}_m$ carries one bit of information. We also use the fact that the set $\mathcal{L}_\varphi$ is closed under multiplication by X^i for $i = 0, 1, \ldots, N-1$.

Here we prove that NtruEncrypt encryption has a very attractive bit security property. Namely, each coefficient of the encrypted message m is as secure as the whole message. Detailed surveys of bit security results are given in [6,8]; several more recent results can be found in [2,3,9,10,11,14,15,17]. Obviously any such results admit opposite points of view. One can express it as "a part is as hard as the whole" and say that this is an advantage of the corresponding cryptosystem. On the other hand, one can express the same property as "the whole is as weak as a part" and say that it is a weakness. We do not here favor either of the above points of view, but we note that the former one is more traditional and has been the main emphasis of prior work.

Our method is based on the following observation: Given an encryption e of a message m, we derive from (1) that

$$e_i = X^i e \equiv X^i \odot \varphi \odot h + X^i \odot m \pmod{q} \tag{2}$$

is a valid encryption of the message $m_i = X^i \odot m$ corresponding to the choice of the "blinding" polynomial $\varphi_i = X^i \odot \varphi \in \mathcal{L}_\varphi$. It is easy to see that m_i is just a cyclic shift of the message m and that since X^i is invertible modulo $X^N - 1$, φ_i is uniformly distributed in $\mathcal{L}_\varphi$. Thus given an oracle returning a prescribed coefficient of the message from a given encryption, one queries this oracle with the values of e_i, $i = 0, 1, \ldots, N-1$, to recover all bits of m one by one. This is a normally undesirable property since it means that the scheme is so-called malleable. We again note that similarly undesirable properties of RSA is the central ingredient in obtaining bit-security results for RSA, so this is not in any way strange. In fact, here we also use one other malleability aspect, see below.

Here we show that the same idea can be used for much weaker oracles which return the value of some coefficient of the message only for a very small proportion of the messages.

We remark that our arguments rely on the fact that the sets $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_\varphi, \mathcal{L}_m$ which guarantee unambiguous decryption are public. In particular, this means that any "guess" for a message m can be efficiently verified by an attacker by checking whether $(e - m) \odot h^{-1} = \varphi$ with some $\varphi \in \mathcal{L}_\varphi$. We note that for the parameter sets given in [12,13] (see also [4]), the polynomial h (equivalently, g) is not invertible. However, because of the structure of φ, it is possible to use a "pseudo-inverse" of h to perform the check [16].

We say that an algorithm is polynomial time if its running time is bounded by $(N \log q)^{O(1)}$. In fact, typically q is of the same order as N, so this last expression can be replaced by simply $N^{O(1)}$. We do not evaluate the exact complexities (which depend on the accepted model of computation) but this is a very simple, although somewhat tedious task.

Finally, we use $\log z$ to denote the binary logarithm of $z > 0$.

3 Main Results

Earlier bit-security results, such as those referenced above, use the model of having access to an oracle that returns the ith bit of the message with probability $1/2 + \varepsilon$, for some small $\varepsilon > 0$ (which could be a function of some other parameters), where the probability is taken uniformly over the sample space of messages and auxiliary encryption parameters (and also, over the internal coinflips of the oracle). Here we use a somewhat different model.

Assume that we are given an oracle $\mathcal{NTRU}_\gamma$ which for any sequence of k *distinct* and *valid* encryptions $e_1, \dots, e_k$, returns the leading coefficient of the corresponding messages $m_1, \dots, m_k$ for at least $\lfloor \gamma k \rfloor$ queries and returns the error message symbol $*$ for the rest of the queries. We do not impose any restrictions on the behavior of this oracle on *invalid* inputs.

It is important to remark that repeatedly querying this oracle with the same message cannot help to extract any new information because it maintains the required rate of correct outputs only for distinct messages. In the same way, we do not specify the behavior of this oracle on invalid queries e which do not correspond to any valid message as we only require it to be correct on valid encryptions. For example, for such queries it can output 0 and 1 with probability $\gamma/2$ each, and output $*$ with probability $1 - \gamma$. Thus the oracle cannot be used to distinguish between valid and invalid encryptions.

On the other hand, we assume that it is a *Las Vegas* type oracle which never gives a wrong answer, although sometimes may give no answer at all. One can clearly argue that this model is qualitatively much stronger than the "classical" model, though on the other hand, it is quantitatively quite comparable as it only returns correct answers for some (non-negligible) γ-fraction of all queries. Moreover, the results obtained are stronger, since the reconstruction algorithm for the message m can be made fully deterministic.

It is known [5] that when N is composite then the whole scheme is not secure, thus we consider only the case of prime N.

For this and other algorithms our general strategy is to create several valid encryptions, by "flipping" already known (or "guessed") bits of m. In other words, we again exploit the malleability of basic NTRU. If all of them are distinct, then each time the oracle provides new information to us. On the other hand, if not all of them are distinct then, as the following statement shows, we can recover m without even using the oracle.

Lemma 1. *Let e be any* NtruEncrypt *encryption of a message $m \in \mathcal{L}_m$ and let $\{r_1, \dots, r_k\} \subset \mathcal{R}$ be such that for all $i = 1, \dots, k$, the polynomial $e + r_i$ is a valid encryption of $m + r_i \in \mathcal{L}_m$. Then, either*

∘ *the polynomials* $\{X^j \odot (m+r_i),\ j=0,\ldots,N-1,\ i=1,\ldots,k\}$ *are all distinct,*

or,

∘ *given the polynomials* $r_1,\ldots,r_k$ *one can recover* m *from* e *in polynomial time.*

Proof. We remark that if

$$X^{j_1} \odot (m+r_1) = X^{j_2} \odot (m+r_2), \tag{3}$$

for two distinct pairs $(j_1,r_1) \neq (j_2,r_2)$, then

$$\left(X^{j_1} - X^{j_2}\right) \odot m = X^{j_2} \odot r_2 - X^{j_1} \odot r_1.$$

If $j_1 = j_2$ then,

$$X^{j_1} \odot (r_2 - r_1) = 0$$

and because X and $X^N - 1$ are relatively prime, we obtain $r_1 = r_2$ which contradicts our assumption. For $j_1 \neq j_2$ we remark that the greatest common divisor of $X^{j_1} - X^{j_2}$ and $X^N - 1$ is $X - 1$. Therefore, each two distinct pairs $(j_1,r_1) \neq (j_2,r_2)$ uniquely define m modulo $(X^N-1)/(X-1)$. Because m has $0,1$ coefficients there could be at most 2 possible values for m which can be found and verified by checking whether $(e-m) \odot h^{-1} \odot p^{-1} \in \mathcal{L}_\varphi$. □

Theorem 1. *Let* N *be a prime. For any constant* $A > 0$*, given an oracle* $\mathcal{NTRU}_\gamma$ *with any*

$$\gamma \geq N^{-A},$$

there exists a deterministic polynomial time algorithm which, given a valid encryption e *of a message* $m \in \mathcal{L}_m$*, makes* $O(\gamma^{-2}N)$ *calls to the oracle* $\mathcal{NTRU}_\gamma$ *and finds the message* m*.*

Proof. The algorithm below calls the oracle $\mathcal{NTRU}_\gamma$ with valid encryptions of the form $X^j \odot (e+r)$ which correspond to some unknown messages of the form $X^j \odot (m+r)$ for some polynomial $r \in \mathcal{R}$ (chosen in such a way that $X^j \odot (m+r) \in \mathcal{L}_m$, and thus $X^j \odot (e+r)$ is its valid encryption).

By Lemma 1, in the algorithm below we always assume that each message (of the form $X^j \odot (m+r)$ where j and $r \in \mathcal{R}$ are known) it produces is distinct from all the previously generated messages, otherwise the above procedure finds m immediately. That is, each time a new message is produced we apply the above procedure (for this message and all previously used messages) and either find m or make sure this new message has never occurred before. Thus the corresponding encryptions are pairwise distinct as well.

We put $L = \lceil \log \gamma^{-1} \rceil$ and "guess" the L highest coefficients of m. For each guess we execute the below procedure. We remark that the total number of guesses $2^L = O(\gamma^{-1})$ and also it is enough to consider only the case when our guess is correct. Indeed, as we have already remarked, for each guess, the candidate for the message m produced by the below algorithm can easily be verified. Thus it is enough to find the right message m at least once (that is, for the right guess).

So we now assume that L highest coefficients $a_{N-1}, \dots, a_{N-L+1}$ of m are known.

To find the coefficient a_{N-i}, $i = L+1, \dots, N$, we consider the following 2^L valid messages (which are flips and cyclic shifts of m)

$$m_i^{(\nu)} = X^{i-1} \odot \left(m - \sum_{j=N-L}^{N-1} (a_j - \nu_j) X^j \right)$$

where $\nu = (\nu_{N-L}, \dots, \nu_{N-1})$ runs over all L dimensional binary vectors, and compute the corresponding encryptions

$$e_i^{(\nu)} = X^{i-1} \odot \left(e - \sum_{j=N-L}^{N-1} (a_j - \nu_j) X^j \right).$$

For each $i = L+1, \dots, N$, the algorithm makes 2^L calls of the oracle $\mathcal{NTRU}_\gamma$ with $e_i^{(\nu)}$. By the conditions on the oracle $\mathcal{NTRU}_\gamma$, because $2^L\gamma \geq 1$, it either returns correctly a_i at least once or we obtain an equation of the form (3).

To estimate the number of calls to the oracle $\mathcal{NTRU}_\gamma$, we remark that for each of the 2^L guesses the algorithm makes $O\left(2^L N\right)$ calls. Therefore the total number of calls is $O\left(2^{2L} N\right) = O\left(\gamma^{-2} N\right)$. □

We now consider the more classical *Monte Carlo* type oracles, which always return a result which however is correct only with with probability bounded away from 1/2. However, we make one additional assumption.

Namely, assume that we are given an oracle $\widetilde{\mathcal{NTRU}}_\varepsilon$ which for any sequence of k *distinct* and *valid* encryptions $e_1, \dots, e_k$, returns the leading coefficient of the corresponding messages $m_1, \dots, m_k$ for at least $(0.5 + \varepsilon)k + O(1)$ queries and returns an arbitrary value for other messages. Thus, the added assumption is that the advantage of the oracle is in a sense uniformly distributed over the sample space.

As before we assume that the oracle $\widetilde{\mathcal{NTRU}}_\varepsilon$ is consistent in its replies, it maintains the required rate of correct outputs only for distinct messages. Thus, asked several times with the same value of e, it returns the same value (correct or not) for the leading coefficient of the corresponding message m. Again, as before, we do not specify the behavior of this oracle on invalid queries e which do not correspond to any valid message. For example, for such queries it can output 1 with probability $0.5+\varepsilon$ and 0 with probability $0.5-\varepsilon$. As in the case of the previous oracle, this oracle cannot be used to distinguish valid from invalid encryptions either.

Theorem 2. *Let N be a prime. For any constant $A > 0$, given an oracle $\widetilde{\mathcal{NTRU}}_\varepsilon$ with any*

$$\varepsilon \geq N^{-A},$$

there exists a deterministic polynomial time algorithm which, given a valid encryption e of a message $m \in \mathcal{L}_m$, makes $O(\varepsilon^{-2}N)$ calls of the oracle $\widetilde{\mathcal{NTRU}}_\varepsilon$ and finds the message m.

Proof. Let C be a constant such that the oracle $\widetilde{\mathcal{NTRU}}_\varepsilon$ returns the correct answer for at least $(0.5+\varepsilon)k - C$ pairwise distinct and correct queries (that is, C is the implicit constant in $O(1)$ in the definition of $\widetilde{\mathcal{NTRU}}_\varepsilon$).

Put $L = \lceil \log \left(C\varepsilon^{-1}+2\right)\rceil$. We start with "guessing" the L highest coefficients of m. In fact, we repeat the below procedure with all $2^L = O(\varepsilon^{-1})$ possible "guesses" and remark that if our guess is correct, then the algorithm below finds the encrypted message correctly. Thus we just verify the obtained $2^L = O(\varepsilon^{-1})$ results in order to find which one corresponds to the encrypted message.

As in the proof of Theorem 1 we remark that by Lemma 1, all messages which appear in our algorithm are of the form $X^j \odot (m+r)$ where j and $r \in \mathcal{R}$ are known (and pairwise distinct). Thus we by checking polynomially many "suspects" we can assume that all valid encryptions used in our algorithm correspond to distinct messages.

Now, for each "guess" $(a_{N-L}, \ldots, a_{N-1})$ of the L highest coefficients of the message m we consider $2^L - 1$ "flips" of the L highest coefficients

$$m^{(\nu)} = m - \sum_{j=N-L}^{N-1} (a_j - \nu_j)X^j$$

and compute the corresponding encryptions

$$e^{(\nu)} = e - \sum_{j=N-L}^{N-1} (a_j - \nu_j)X^j$$

defined by an L-dimensional non-zero binary vector $\nu = (\nu_{N-L}, \ldots, \nu_{N-1})$.

Then for each $i = L+1, \ldots, N$ for which the corresponding coefficient a_{N-i} is still undefined, we query the oracle with $2^L - 1$ distinct encryptions $e_i^{(\nu)} = X^{i-1} \odot e^{(\nu)}$, getting (assuming our original guess of the L highest coefficients is correct) the value of a_{N-i} at least

$$\begin{aligned}(0.5+\varepsilon)\left(2^L-1\right) - C &= 0.5\left(2^L-1\right) + \varepsilon\left(2^L-1\right) - C \\ &\geq 0.5\left(2^L-1\right) + \varepsilon\left(C\varepsilon^{-1}+1\right) - C > 0.5\left(2^L-1\right)\end{aligned}$$

times. Thus we make the majority decision on the value of a_i. We remark that we make an odd number of queries thus the majority decision procedure is always correctly defined (even when our original guess of the coefficients is not correct). □

Finally, we can indeed consider the classical model where the oracle gives us the leading coefficient of m with probability $0.5+\delta$, where the probability is over random choices of m, φ and the (possible) internal random choices of the oracle. This complicates things since there could for example, be certain messages for which the oracle has no advantage at all. Still, we can obtain a (weak) non-trivial result as follows. Denote such an oracle $\widehat{\mathcal{NTRU}}_\delta$.

Theorem 3. *Let $0 < \alpha < 1$ be a constant. If there is an oracle $\widehat{\mathcal{NTRU}}_\delta$ with*

$$\delta \geq \frac{1}{2} - \frac{\alpha}{N},$$

there exists a deterministic polynomial time algorithm which, given a valid, random, encryption e of a random message $m \in \mathcal{L}_m$, makes N calls of the oracle $\widehat{\mathcal{NTRU}}_\delta$ and finds the message m with probability $1 - \alpha$.

Proof. Querying $\widehat{\mathcal{NTRU}}_\delta$ on $X^i \odot e$, gives us a_{N-i-1} except with probability at most α/N. Proceeding with cyclic shifts we can extract all bits, but unfortunately, the queried inputs are highly dependent. Still, for randomly chosen m and φ, the probability that any of the N calls is wrong is at most α. $\square$

4 Concluding Remarks and Open Questions

It is obvious that similar algorithms work for an oracle producing any other fixed coefficient of the message m rather than just the leading coefficient.

Finally, one can probably consider an oracle which is a combination of the oracles $\mathcal{NTRU}_\gamma$ and $\widetilde{\mathcal{NTRU}}_\varepsilon$, namely the oracle which for any sequence of k distinct and valid encryptions $e_1, \ldots, e_k$, returns $0, 1$ for at least $\gamma k + O(1)$ queries which is the correct value of the leading coefficient of the corresponding message for at least $(0.5 + \varepsilon)\gamma k + O(1)$ of them, while for the remaining queries it returns the error message symbol $*$.

It is easy to see that our approach applies to any choice of p, for example, to the original case $p = 3$, implying that finding any coefficient of the message m is as hard as finding the whole message.

It is hard to resist comparing our results to those known for other functions (in the classical model). While our results here are much weaker, the initial results for the RSA function, [7], were of the precisely the same strength as ours. Also, the best known results for Diffie-Hellman bits, [3,9], are (to date) not any stronger. In fact, they are obtained for much more powerful oracles (which return much longer bit strings and are always correct). Considering that it has taken quite some time before the "full" security of RSA bits was shown, [1], we hope that the methods developed in this paper in some form can eventually lead to an analogous result for NTRU. Unlike the RSA scheme, it is easy to see that NtruEncrypt has properties that guarantee that if any single bit is secure, then so are the others. Extending the results for the least significant bits of RSA, [1], to all bits, [11], on the other hand has taken almost twenty years.

Finally, it should be noted that whereas "a bit" is certainly a natural study-item for security, it is at the same time also a quite arbitrary concept. It would be of great interest to show security of *any* non-trivial predicate of an NTRU encrypted message.

References

1. W. Alexi, B. Chor, O. Goldreich and C. P. Schnorr, 'RSA and Rabin functions: Certain parts are as hard as the whole', *SIAM Journal on Computing*, **17** (1988), 194–209. 69
2. D. Boneh and I. E. Shparlinski, 'On the unpredictability of bits of the elliptic curve Diffie–Hellman scheme', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **2139** (2001), 201–212. 64
3. D. Boneh and R. Venkatesan, 'Hardness of computing the most significant bits of secret keys in Diffie–Hellman and related schemes', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **1109** (1996), 129–142. 64, 69
4. Consortium for Efficient Embedded Security, Efficient Embedded Security Standard #1, Draft 4. Available from `http://www.ceesstandards.org`. 63, 64
5. C. Gentry, 'Key recovery and message attack on NTRU-composite', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **2045** (2001), 182–194. 65
6. M. Goldman, M. Näslund and A. Russell, 'Complexity bounds on general hard-core predicates', *J. Cryptology*, **14** (2001), 177–195. 64
7. S. Goldwasser, S. Micali and P. Tong, 'Why and how to establish a private code on a public network (Extended abstract)', *Proc. 23rd Annual IEEE Symp. on Foundations of Comp. Sci.*, 1982, 134–144. 69
8. M. I. González Vasco and M. Näslund, 'A survey of hard core functions', *Proc. Workshop on Cryptography and Computational Number Theory*, Singapore 1999, Birkhäuser, 2001, 227–256. 64
9. M. I. González Vasco and I. E. Shparlinski, 'On the security of Diffie–Hellman bits', *Proc. Workshop on Cryptography and Computational Number Theory*, Singapore 1999, Birkhäuser, 2001, 257–268. 64, 69
10. M. I. González Vasco and I. E. Shparlinski, 'Security of the most significant bits of the Shamir message passing scheme', *Math. Comp.*, **71** (2002), 333–342. 64
11. J. Håstad and M. Näslund, 'The security of individual RSA and discrete log bits', *J. of the ACM*, (to appear). 62, 63, 64, 69
12. J. Hoffstein, J. Pipher and J. H. Silverman, 'NTRU: A ring based public key cryptosystem', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **1433** (1998), 267–288. 62, 63, 64
13. J. Hoffstein and J. H. Silverman, 'Optimizations for NTRU', *Proc. the Conf. on Public Key Cryptography and Computational Number Theory, Warsaw, 2000*, Walter de Gruyter, 2001, 77–88. 63, 64
14. N. A. Howgrave-Graham, P. Q. Nguyen and I. E. Shparlinski, 'Hidden number problem with hidden multipliers, timed-release crypto and noisy exponentiation', *Math. Comp.*, (to appear). 64
15. W.-C. W. Li, M. Näslund and I. E. Shparlinski, 'The hidden number problem with the trace and bit security of XTR and LUC', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **2442** (2002), 433–448. 64
16. P. Q. Nguyen and D. Pointcheval, 'Analysis and improvements of NTRU encryption paddings', *Lect. Notes in Comp. Sci.*, Springer-Verlag, Berlin, **2442** (2002, 210–225. 64
17. C. P. Schnorr, 'Security of almost all discrete log bits', *Electronic Colloq. on Comp. Compl.*, Univ. of Trier, **TR98-033** (1998), 1–13. 64
18. J. H. Silverman, 'Invertibility in truncated polynomial rings', *NTRU Cryptosystem Tech. Report 9*, 1998, 1–8. 63