NTRU Cryptosystems Technical Report

Report # 6, Version 1

Title: Implementation Notes for NTRU PKCS Multiple Transmissions

Author: Jeffrey Hoffstein and Joseph H. Silverman

Release Date: May 26, 1998

Abstract. Multiple NTRU encryptions of a single message using a single key may compromise the security of the message. In this report we analyze this situation and describe scrambling techniques which allows secure multiple transmissions of a single message.

As observed in [1, §3.3], if a single message is encrypted multiple times using a single NTRU public key, then the message may be susceptible to a multiple transmission attack. Earlier versions of [1] described ways to thwart such attacks, but space limitations forced us to omit this material in the final version. In this note we summarize and expand this material. We give a detailed analysis of multiple transmission attacks and indicate fast and simple procedures which will eliminate such attacks in all practical settings.

Analysis of Multiple Transmission Attacks

Suppose that a message m is encrypted and transmitted several times using a single public key h and different random choices $\phi_1, \phi_2, \dots, \phi_r \in \mathcal{L}(d,d)$. (We recall that $\mathcal{L}(d,d)$ is the set of polynomials of degree $N-1$ with d coefficients equal to each of 1 and -1, and the remaining coefficients equal to 0.) Thus the encrypted messages are

$$e_i \equiv \phi_i * h + m \pmod{q} \quad \text{for } i = 1, 2, \dots, r.$$

The attacker computes the quantities

$$c_i \equiv (e_i - e_1) * h^{-1} \pmod{q} \quad \text{for } i = 2, 3, \dots, r,$$

where h^{-1} is the inverse of h modulo q. Notice that

$$c_i \equiv \phi_i - \phi_1 \pmod{q}.$$

But the coefficients of $\phi_i - \phi_1$ range from -2 to 2, so the attacker recovers the exact value of $\phi_i - \phi_1$.

Now let's analyze the possibilities for some particular coefficient of $\phi_i - \phi_1$, say the j^{th} coefficient. If we let

$$\begin{aligned} \alpha &= \text{the } j^{\text{th}} \text{ coefficient of } \phi_i, \\ \beta &= \text{the } j^{\text{th}} \text{ coefficient of } \phi_1, \\ \gamma &= \text{the } j^{\text{th}} \text{ coefficient of } \phi_i - \phi_1, \end{aligned}$$

then the possibilities for $\gamma = \alpha - \beta$ are listed in the following table:

$\alpha \setminus \beta$	-1	0	1
-1	0	-1	-2
0	1	0	-1
1	2	1	0

Coefficient of $\phi_i - \phi_1$

Hence if $\gamma = 2$ (respectively $\gamma = -2$), then the attacker deduces that $\beta = -1$ (respectively $\beta = 1$). In this way, the attacker will generally be able to recover approximately 2/9 of the coefficients of ϕ_1. Further, if $\gamma = 1$ (respectively $\gamma = -1$), the attacker is able to narrow down the possible values of β to $\beta = -1, 0$ (respectively $\beta = 0, 1$). Thus the attacker gains valuable information about another 4/9 of the coefficients of ϕ_1.

Thus each of the additional transmissions $e_2, e_3, \ldots, e_r$ will enable the attacker to exactly determine approximately 2/9 of the coefficients of ϕ_1, and will also give information about another 4/9 of the coefficients. It thus takes very few transmissions for the attacker to determine almost every coefficient of ϕ_1, and then a brute force search on the remaining coefficients will allow the attacker to recover ϕ_1 and m.

Remark 1. If an attacker decrypts a single message in this fashion, the information gained will not assist in decrypting any future messages. In other words, the encryption-decryption pair (h, f) remains secure.

Remark 2. Note that an attacker can easily determine if some of the encrypted messages in a list $e_1, e_2, \ldots, e_r$ have the same underlying plaintext. This is true because the quantity

$$(e_i - e_j) * h^{-1} \equiv (\phi_i - \phi_i) + (m_i - m_j) * h^{-1} \pmod{q}$$

will have all small coefficients if and only if the underlying messages m_i and m_j are identical.

Prevention of Multiple Transmission Attacks

The key to secure multiple transmissions using the NTRU PKCS is to perform some simple scrambling and/or padding of the message for each transmission. There are obviously many ways in which this can be done. We will describe two possibilities.

Method 1. This method doubles the size of the message that is transmitted for each block, but has the advantage of reducing message expansion to just a trifle higher than 2-to-1. In this formulation, the message m is now a polynomial modulo q, rather than a polynomial modulo p. In order to encrypt the message m, the encryptor chooses a random polynomial $\phi \in \mathcal{L}(d,d)$ and a second random polynomial Φ mod 3 and computes

$$e \equiv \phi * h + \Phi \pmod{q} \qquad \text{and} \qquad E \equiv \Phi * e + m \pmod{q}.$$

He then transmits the pair (e, E).

The decryptor first recovers Φ by using the private key to decrypt e in the usual way. (Note that Φ consists of -1's, 0's, and 1's, so knowing Φ modulo p with $p = 3$ is enough to recover Φ exactly.) Then he recovers the message m by simply computing $E - \Phi * e$ modulo q.

Notice that even if a single message m is transmitted many times, the attacker will not be able to use a multiple transmission attack, since eliminating m from two transmissions (e_1, E_1) and (e_2, E_2) only gives the value of the quantity

$$E_2 - E_1 \equiv \Phi_2 * e_2 - \Phi_1 * e_1 \pmod{q}.$$

This is sufficiently intertwined so that the known values of e_1 and e_2 do not allow the attacker to directly recover Φ_1 and Φ_2. Of course, there does exist a lattice attack based on the fact that Φ_1 and Φ_2 are comparatively short, but the associated lattice is even less amenable to lattice reduction methods than the lattices which appear in the usual lattice attacks.

Remark 3. For longer messages, the above idea can be extended in the following way to give "turbo-NTRU," a version of the NTRU PKCS having essentially no message expansion. First fix some fast deterministic (preferably non-linear) method which takes as input a polynomial modulo q and outputs a binary polynomial. For example, given a polynomial m modulo q, let $B(m)$ be the binary polynomial obtained by reducing the coefficients of m modulo 2 and then squaring the polynomial modulo 2. Now a list of message polynomials $m_1, m_2, m_3, \ldots$ modulo q is transmitted as a sequence of encrypted polynomials $e, e_1, e_2, e_3, \ldots$ computed as follows. Choose random ϕ and Φ as above, and let

$$\begin{aligned} e &= \phi * h + \Phi \pmod{q} \\ e_1 &= \Phi * h + m_1 \pmod{q} \\ e_2 &= B(m_1) * h + m_2 \pmod{q} \\ e_3 &= B(m_2) * h + m_3 \pmod{q} \\ &\vdots \qquad\qquad \vdots \end{aligned}$$

Decryption is easy, since the usual NTRU decryption procedure yields Φ, and then each successive message block can be computed from the previous one. Notice the importance of starting with a random Φ, rather than with m_1, so as to thwart multiple transmission attacks. We also note the usual drawback of this sort of streaming, namely if some block contains a single tranmission error, then all succeeding blocks will be indeciperable.

Method 2. This method is useful for small messages when one wants to keep the size of the transmitted block as small as possible. The idea is to reserve a certain number of initial bits of the message for random padding, and to use these bits to modify the rest of the message in some easily invertible manner. For concreteness, we illustrate using the moderate security parameters $(N, p, q) = (107, 3, 64)$ described in [1, §4.1].

In this case a message block consists of 107 numbers modulo 3. We will assume that the message to be transmitted consists of 82 numbers modulo 3, say given by a polynomial $n(X)$ of degree 81, leaving 25 mod 3 numbers unassigned. So we first choose a random mod 3 polynomial $z(X)$ of degree 24. Next we use some fast deterministic (preferably non-linear) method which takes as input a mod 3 polynomial $z(X)$ of degree 24 and gives as output a mod 3 polynomial $B(z(X))$ of degree (approximately) 81. For example, we could let $B(z(X)) = z(X)^3 \pmod{3}$, or $B(z(X)) = z(X)^4 \pmod{3, X^{82}}$. Then we set

$$m(X) = z(X)X^{82} + n(X) + B(z(X)) \pmod{3}$$

and encrypt $m(X)$ as usual (i.e, as $e = \phi * h + m \pmod{q}$).

In order to decrypt, one first uses standard NTRU decryption to recover $m(X)$, next $z(X)$ is pulled off from the top 25 coefficients of $m(X)$, and finally the actual message $n(X)$ is recovered as the bottom 82 coefficients of $m(X) - B(z(X)) \pmod{3}$.

Since each message is sent with a randomly chosen z, the messages will be safe from a multiple transmission attack unless the encryptor is unlucky enough to choose the same z for two (or more) transmissions. We can compute the probability of this occurring. (Note we do not want to assume that the encryptor keeps track of which z's have already been used.) This is a "matching birthday" type of problem, since we are asking for the probability of choosing some value twice when making choices from a pool of 3^{25} items. We find that if the message is sent 100 times, then the probability of a repeat is less than 2^{-27}, and if the message is sent 10000 times, the probability is still less than 2^{-14}. Indeed, the message would need to be sent 2^{17} times in order to have a 1% chance of a repeated transmission. As these numbers indicate, using 25 random coefficients will suffice to thwart multiple transmission attacks in any practical situation.

Remark 4. If N is larger than 107, then the masking procedure B may need to be modified, but strictly from a security viewpoint, it should not be necessary to increase the size of the padding polynomial z. For example, if $N = 167$ and $p = 3$, then we could take $B(z(X)) = z(X)^5 \pmod{3}$, and if $N = 503$ and $p = 3$, then we could take $B(z(X)) = z(X)^{17} \pmod{3}$. On the other hand, if the actual message is small, say no larger than half a message block, then it would be more efficient to choose $\deg z(X) = (N-3)/2$ and $\deg n(X) = (N-1)/2$ and simply conceal the message as the sum

$$m(X) = z(X)X^{(N+1)/2} + n(X) + \tilde{z}(X) \pmod 3,$$

where $\tilde{z}(X)$ is obtained from $z(X)$ by reversing the order of the coefficients. [In principle, one could just add $z(X)$, rather than $\tilde{z}(X)$. However, in that case $m(X)$ would have the form $z(X)(X^{(N+1)/2} + 1) + n(X)$.]

References

[1] NTRU: A Ring-Based Public Key Cryptosystem, a paper presented at ANTS 3 (Reed College,Portland, Oregon, June 21–25, 1998), to appear in Lecture Notes in Computer Science, Springer-Verlag.
