NTRU Cryptosystems Technical Report

Report # 016, Version 1
Title: Protecting NTRU Against Chosen Ciphertext and Reaction Attacks
Author: Jeffrey Hoffstein and Joseph H. Silverman
Release Date: June 9, 2000

Abstract. This report describes how the Fujisaki-Okamoto Self-Referential Technique (FOSRT) can be used to make the NTRU Public Key Cryptosystem resistant to adaptive chosen ciphertext attacks and to reaction attacks.

Many asymmetric ciphers are susceptible to (adaptive) chosen ciphertext attacks. An attacker sends a series of purported ciphertexts $e_1, e_2, \ldots$ and uses the decryptions to deduce information about either the secret key or about an intercepted ciphertext e that was used to create $e_1, e_2, \ldots$. The user Alice may try to guard against such attacks by padding her plaintext so that Bob can detect valid plaintexts from invalid plaintexts, but then the attacker may be able to gain useful information by simply observing which ciphertexts are accepted and which ciphertexts are rejected. An example of such an attack against RSA and a suggested defense can be found in [2] and [3]. Adaptive chosen ciphertext attacks against NTRU have also been formulated and various countermeasures described, see [9] and [10].

Another type of attack called a reaction attack [6] can be used against some cryptosystems, including NTRU [8]. In a reaction attack, one can takes a ciphertext e and creates ciphertexts $e_1, e_2, \ldots$ such that for each ciphertext e_i, there is a significant positive probability it will decrypt to the same plaintext as e and a significant positive probability it will decrypt to a different plaintext than e. Some specific reaction attacks against NTRU, again with assorted countermeasures, are given in [8] and [9].

In this report we describe two methods of Fujisaki and Okamoto [5] that can be used to defend NTRU against both adaptive chosen ciphertext attacks and reaction attacks. The basic idea is to use a hash of the plaintext (suitably padded) as the random component required in the encryption process. The decrypted plaintext is then checked by redoing the encryption. Since the plaintext reinserts itself into the encryption process, we have dubbed this the

Fujisaki-Okamoto Self-Referential Technique (FOSRT).

Although FOSRT, as applied to NTRU, has a small drawback in that it requires computation of one extra convolution product, NTRU remains extremely fast even with this extra computation. We also note that an alternative defense against chosen ciphertext and reaction attacks takes advantage of NTRU's fast key creation to create transient (e.g., one-per-session) keys.

Section 1. NTRU and FOSRT

We assume that the reader is familiar with NTRU encryption and decryption, as described for example in [7]. The public key is a polynomial $h(X)$ with coefficients modulo q and the plaintext is a polynomial $m(X)$ with coefficients modulo p. In order to encrypt raw plaintext with no padding, Alice chooses a random polynomial $r(X)$ with coefficients modulo p (generally with a further constraint on the coefficients) and computes the ciphertext $e(X)$ as the polynomial

$$e(X) \equiv r(X) * h(X) + m(X) \pmod{q}.$$

(All polynomial multiplications are done using the rule $X^N = 1$, so are really convolution products on the vectors of coefficients.)

In order to encrypt a plaintext message M using the Fujisaki-Okamoto Self-Referential Technique (FOSRT), Alice first chooses a random string R of (say) 40 to 80 bits. She uses M and R to form the plaintext polynomial $m(X)$ in some standard way so as to introduce sufficient randomness into the message (see Remark 1 below). She also applies a pre-selected hash function $\mathcal{H}$ that converts a bit string $M\|R$ into an unpredictable valid polynomial $r(X)$. Alice then computes the NTRU ciphertext as usual,

$$e(X) \equiv r(X) * h(X) + m(X) \pmod{q},$$

and sends it to Bob.

Bob begins the decryption process as usual, recovering (after some computation) the plaintext polynomial $m(X)$. He converts $m(X)$ back into the bit string $M\|R$ and pulls off the ostensible message M. He next computes

$$\mathcal{H}(M\|R) * h(X) + m(X) \pmod{q}.$$

If this is equal to the ciphertext $e(X)$ that he received from Alice, he accepts the message. Otherwise, he rejects the message as indecipherable.

Notice that a valid NTRU ciphertext using FOSRT necessarily has the form

$$(\text{Valid } r) * h + (\text{Valid } m) \pmod{q},$$

so it is not possible to create valid FOSRT-NTRU ciphertexts that have a significant positive probability of deciphering to two different plaintexts.† This means

† In this context we should mention that for NTRU, there is a very small probability that a ciphertext formed with a valid r and a valid m will decipher to something other than m. However, for suggested NTRU parameters, the probability of these wrap/gap failures is extremely small, so it would require millions (or billions) of messages to generate a few examples. See [7] and [1] for details.

that NTRU with FOSRT should be secure against reaction attacks such as those described in [6] and [8].

Similarly, adaptive chosen ciphertext attacks cannot be used against NTRU with FOSRT, since the only ciphertexts that are valid are those for which the random component (i.e., the $r(X)$ polynomial) matches in an unpredictable way the plaintext component (i.e., the $m(X)$ polynomial). This intuition is made precise in the the paper of Fujisaki and Okamoto [5], where they develop FOSRT for a general probabilistic asymmetric cipher. See Remark 2 for further discussion.

Remark 1. Combining the Plaintext M and Random String R
There are many ways to combine the plaintext M and random string R into a valid NTRU message polynomial $m(X)$. The simplest method, which probably suffices for most purposes, is to convert the concatenated bit string $M\|R$ directly into a list of mod p coefficients. For added security with very low overhead, one can use R to modify M in an unpredictable, but reproducible, way. For example, let

$$M' = \text{Hash}(R) \quad XOR \quad M,$$

and then translate the bit string $M'\|R$ directly into the coefficients of $m(X)$.

Remark 2. NTRU and IND-CPA
In technical terms, the Fujisaki-Okamoto construction FOSRT in [5] takes a public key cryptosystem Π that is Indistinguishable against Chosen Plaintext Attacks (IND-CPA) and uses it to create a new public key cryptosystem Π' that is Indistinguishable against (Adaptive) Chosen Ciphertext Attacks (IND-CCA). By work of Bellare, Desai, and Pointcheval [1], this also implies that the new system Π' is Non-Malleable against (Adaptive) Chosen Ciphertext Attacks (NM-CCA). In order to use FOSRT with NTRU, we need to discuss the extent to which NTRU is IND-CPA.

If NTRU is used with raw unpadded message polynomials $m(X)$, then it is not IND-CPA. Thus if m_1 and m_2 are known plaintexts and if e is the ciphertext for one of them, then an adversary can determine which is the correct plaintext by computing

$$h^{-1} * (e - m_1) \pmod q \qquad \text{and} \qquad h^{-1} * (e - m_2) \pmod q$$

and checking which one is a valid $r(X)$ polynomial.[†] However, as soon as a reasonable amount of random padding is added to the plaintext, NTRU appears to be indistinguishable against chosen plaintext attacks, allowing the use of FOSRT. For example, one can concatenate the random component R with a suitably modified version of M as described in Remark 1 . (We note that even if $m(X)$ is formed by

[†] In practice, an NTRU public key polynomial $h(X)$ does not have an inverse; but one can find a pseudo-inverse which will almost fulfill the same purpose.

simply concatenating the plaintext M with the random string R, an adversary will still need to distinguish betweeen $r + h^{-1} * u_1$ and $r + h^{-1} * u_2$, where one of u_1 and u_2 has a smaller number of nonzero coefficients than the other. This appears to be a difficult problem.)

Remark 3. Using Transient Keys to Avoid Loss of Efficiency
As mentioned above, a possible drawback to using FOSRT with NTRU is that the decryption process requires one additional polynomial multiplication (convolution product), as well as a few other less time-consuming operations. The effect of this is to reduce decryption speed about 40%. However, we note that the speed reduction can be avoided while still protecting against chosen ciphertext and reaction attacks if one takes advantage of NTRU's superfast key generation and uses transient (e.g., one-per-session) keys, since then no single key is used for long enough to accumulate the data needed for such an attack.

Section 2. The Fujisaki-Okamoto Hybrid Method

In technical language, the Fujisaki-Okamoto Self-Referential Technique described in the previous section creates a public key cryptosystem that is indidistinguishable against adaptive chosen ciphertext attacks (IND-CCA) in the random oracle mode, assuming only that the original system is indistinguishable against chosen plaintext attacks (IND-CPA) and contains sufficient variability (sufficiently γ-uniform, in the terminology of [4] and [5]).

Fujisaki and Okamoto have described another construction [4] in which the public key cryptosystem is only assumed to satisfy a One-Way Encryption (OWE) assumption, which is weaker than IND-CPA. Intuitively, their construction in [4] assumes only that an adversary is not able to find an algorithm that has a non-negligible chance of decrypting messages. Their construction is an example of a hybrid cryptosystem, because it combines a public key cryptosystem, such as NTRU, with a private key cryptosystem, such as 3DES or AES. NTRU fits very easily into the Fujisaki-Okamoto hybrid construction. In this section we give a brief description.

As above, Alice starts with her plaintext M. She randomly chooses a valid $m(X)$ polynomial (which is completely independent of her plaintext) and computes an $r(X)$ polynomial as a hash $\mathcal{H}(m, M)$ of m and M. She then uses NTRU to form the ciphertext

$$e(X) \equiv r(X) * h(X) + m(X) \pmod{q}.$$

Next she uses another hash function to compute $\mathcal{G}(m)$, where $\mathcal{G}(m)$ is a key for a symmetric cipher such as 3DES or AES. She uses this key and the symmetric cipher to encrypt the plaintext M and create the (symmetric) ciphertext E. To recapitulate in formulas,

$$\begin{aligned} e &= \mathrm{NTRU}_h(\mathcal{H}(m, M), m) \equiv \mathcal{H}(m, M) * h + m \pmod{q}, \\ E &= \mathrm{SymCiph}_{\mathcal{G}(m)}(M). \end{aligned}$$

Alice sends this pair (e, E) to Bob.

Bob's decryption process proceeds as follows:

- Bob uses NTRU decryption (and his private key) to recover $\hat{m}$ from e.
- Bob computes the hash $\mathcal{G}(\hat{m})$, which is a key for the symmetric cipher.
- Bob uses symmetric cipher decryption with the key $\mathcal{G}(\hat{m})$ to decrypt the ciphertext E and recover the plaintext $\hat{M}$.
- Bob computes the hash $\mathcal{H}(\hat{m}, \hat{M})$, which is an $r(X)$ polynomial for NTRU.
- Bob checks that the ciphertext e is equal to the quantity $\mathcal{H}(\hat{m}, \hat{M}) * h + \hat{m} \pmod{q}$, in which case he accepts the plaintext $\hat{M}$. Otherwise he rejects the message as indecipherable.

References

[1] M. Bellare, A. Desai, D. Pointcheval, P. Rogaway, Relations Among Notions of Security for Public-Key Encryption Schemes, *Advances in Cryptology—CRYPTO '98*, Springer-Verlag, 1992.

[2] D. Bleichenbacher, Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS#1, *Advances in Cryptology—Crypto '98*, Springer-Verlag, 1992.

[3] D. Bleichenbacher, B. Kaliski, J. Staddon, Recent results on PKCS#1: RSA encryption standard, RSA Laboratories' Bulletin, Number 7, June 26, 1998.

[4] E. Fujisaki, T. Okamoto, Secure integration of asymmetric and symmetric encryption schemes, *Advances in Cryptology—CRYPTO '99*, Lecture Notes in Computer Science 1666, Springer-Verlag, 1999, 537–554

[5] E. Fujisaki, T. Okamoto, How to Enhance the Security of Public-Key Encryption at Minimum Cost, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E83-A, No.1, Special Issue on Cryptography and Information Security (January 2000)

[6] C. Hall, I. Goldberg, B. Schneier, Reaction attacks against several public-key cryptosystems, preprint April 1999, available at `www.counterpane.com`

[7] J. Hoffstein, J. Pipher, J.H. Silverman, NTRU: A new high speed public key cryptosystem, Algorithmic Number Theory (ANTS III), Portland, OR, June 1998, Lecture Notes in Computer Science 1423, J.P. Buhler (ed.), Springer-Verlag, Berlin, 1998, 267–288

[8] J. Hoffstein, J.H. Silverman, Reaction Attacks Against the NTRU Public Key Cryptosystem, NTRU Technical Report #015, August 1999, available at `www.ntru.com`

[9] E. Jaulmes, A. Joux, A chosen-ciphertext attack against NTRU, in *Proceedings of CRYPTO 2000*, Lecture Notes in Computer Science, Springer-Verlag, to appear.

[10] J.H. Silverman, Plaintext Awareness and the NTRU PKCS, NTRU Technical Report #007, July 1998, available at `www.ntru.com`

[11] J.H. Silverman, Wraps, Gaps, and Lattice Constants, NTRU Technical Report #011, January 1999, available at `www.ntru.com`

Comments and questions concerning this technical report should be addressed to

`techsupport@ntru.com`

Additional information concerning NTRU Cryptosystems and the NTRU Public Key Cryptosystem are available at

`www.ntru.com`
