On adapting NTRU for Post-Quantum Public-Key Encryption

On adapting NTRU for Post-Quantum Public-Key Encryption

Simone Dutto[1], Guglielmo Morgari[2], and Edoardo Signorini[1,2]

[1] Department of Mathematical Sciences L. Lagrange, Politecnico of Torino, Italy
[2] Telsy Spa, Italy
simone.dutto@polito.it, guglielmo.morgari@telsy.it,
edoardo.signorini@polito.it

1 Introduction

The main research project on Post-Quantum Cryptography (PQC) is the NIST PQC Standardization Process [1], which focuses on selecting post-quantum Key Encapsulation Mechanisms (KEMs) and Digital Signature Schemes.

Public-Key Encryption (PKE) schemes will not be standardized since, in general, the submitted KEMs are obtained from PKE schemes and the inverse process is simple. However, there are cases for which this is not straightforward, like the NTRU submission [2].

This work focuses on solving this problem by introducing a PKE scheme obtained from the KEM proposed in the NTRU submission, while maintaining its IND-CCA2 security (indistinguishability under adaptive chosen ciphertext attack).

2 The NTRU submission

NTRU [2] is a finalist in the NIST PQC Standardization Process that presents a KEM in which the security is based on the Shortest Vector Problem (SVP) on the NTRU lattice [6] (several variations and generalization exist, see e.g. [8, 4, 5]).

The proposed KEM achieves IND-CCA2 security by exploiting an OW-CPA (one-wayness under chosen plaintext attack) PKE scheme with either of two sets of parameters: NTRU-HPS and NTRU-HRSS-KEM. Because of its larger range of addressed security levels, we focused on NTRU-HPS.
More precisely, let $\mathbb{Z}_q = \{-\frac{q}{2}, \ldots, 0, \ldots, \frac{q}{2} - 1\}$ and $\mathbb{Z}_3 = \{-1, 0, 1\}$, given

$$(n, q) \in \{(509, 2048), (677, 2048), (821, 4096)\}$$

corresponding to three different security levels, with $p = 3$ and $d = \frac{q}{8} - 2$, and considering $\phi_n(x) = 1 + x + \ldots + x^{n-1} \in \mathbb{Z}_3[x]$ or $\phi_n(x) \in \mathbb{Z}_q[x]$, all polynomials

De Cifris Koine – DE CIFRIS SEMINARS –https://doi.org/10.69091/koine/vol-2-T04

are represented as arrays in the three following rings:
- $R_q = \mathbb{Z}_q[x]/(x^n - 1)$,
- $S_q = \mathbb{Z}_q[x]/(\phi_n)$,
- $T = \mathbb{Z}_3[x]/(\phi_n)$.

Moreover, $T(d) = \left\{ \sum_{i=1}^{n-2} t_i x^i \in T \,|\, \#\{t_i = 1\} = \#\{t_i = -1\} = d/2 \right\}$.
The OW-CPA PKE scheme consists of the following algorithms.

$\underline{\mathsf{OWCPA.keygen}(seed)}$	$\underline{\mathsf{OWCPA.encrypt}(\mathbf{h}, (\mathbf{r}, \mathbf{m}))}$
1. $seed \to \mathbf{f} \in T$	1. return $\mathbf{c} = \mathbf{r} \cdot \mathbf{h} + \mathbf{m} \in R_q$
2. $seed \to \mathbf{g} \in T(d)$	$\underline{\mathsf{OWCPA.decrypt}((\mathbf{f}, \mathbf{f}_3, \mathbf{h}_q), \mathbf{c})}$
3. $\mathbf{f}_q = \mathbf{f}^{-1} \in S_q$	1. $\mathbf{a} = \mathbf{c} \cdot \mathbf{f} \in R_q$
4. $\mathbf{h} = 3\mathbf{g} \cdot \mathbf{f}_q \in R_q$	2. $\mathbf{m} = \mathbf{a} \cdot \mathbf{f}_3 \in T$
5. $\mathbf{h}_q = \mathbf{h}^{-1} \in S_q$	3. $\mathbf{r} = (\mathbf{c} - \mathbf{m}) \cdot \mathbf{h}_q \in S_q$
6. $\mathbf{f}_3 = \mathbf{f}^{-1} \in T$	4. if $(\mathbf{r}, \mathbf{m}) \in T \times T(d)$ return $(\mathbf{r}, \mathbf{m}, 0)$
7. return $\mathbf{h}$, $(\mathbf{f}, \mathbf{f}_3, \mathbf{h}_q)$	5. else return (0, 0, 1)

Since the message $m \in T(d)$ is ternary and constrained, and the security is only OW-CPA, this is not directly suitable for PKE.

3 Obtaining a PKE scheme from NTRU

In order to obtain a PKE scheme using the OW-CPA PKE scheme from NTRU [2], another work is considered: NTRUEncrypt [3], a first round submission that inspired NTRU-HPS in NTRU [2].

NTRUEncrypt exploits a padding function to encode the message in a ternary polynomial with at least 256 bits of entropy and a message masking to achieve IND-CPA security (indistinguishability under chosen plaintext attack). Then, the NAEP transformation [7] is adopted to obtain an IND-CCA2 PKE scheme.

3.1 Message padding function

The padding is an invertible map $\mathrm{Pad} : (\mathbb{Z}_{2^8})^L \times \{0,1\}^* \to T(d)$ with $\mathrm{Pad}(msg, seed) = m$, where the *seed* allows to add bits of entropy. Since in NTRU-HPS $m \in T(d)$, while in NTRUEncrypt it has no constraints, a new padding is required. Our definition uses an encoding function to obtain a ternary polynomial, and then exploits the *seed* to add bits of entropy and achieve the constraint.

Considering a bijection $\zeta : (\mathbb{Z}_2)^5 \to (\mathbb{Z}_3)^4$ with outputs among the permutations of the elements of the arrays $(0, 0, 1, -1)$, $(0, 1, 1, -1)$, $(0, 1, -1, -1)$, our encoding function is $\underline{\zeta} : (\mathbb{Z}_{2^8})^L \to (\mathbb{Z}_3)^{32\,L/5}$, with

$$\underline{\zeta}(m_1, \ldots, m_L) = \zeta\big(m_1[1{:}5]\big) \| \zeta\big(m_1[6{:}8] \| m_2[1{:}2]\big) \| \ldots \| \zeta\big(m_L[4{:}8]\big) .$$

In the encoded message

$$8 \cdot L/5 \leqslant \#\{1's\}, \#\{-1's\} \leqslant 16 \cdot L/5\,,$$

so that the maximum length of msg is $L \in \mathbb{N}$ such that 5 divides L and $16\,L/5 \leqslant d/2$, i.e. $L = 5\lfloor d/32 \rfloor$.

The last $r = n - 1 - 32\,L/5$ coefficients are generated through the *seed*, while reaching the constraint and adding at least 256 bits of entropy. In the worst cases, the missing 1's and -1's are $a = d/2 - 16\,L/5$ and $b = d/2 - 8\,L/5$ (or viceversa). Thus, the possible completions are $\binom{r}{a}\binom{r-a}{b}$ and the minimum entropy is

$$H_{\min} = \log_2\left(\binom{r}{a}\binom{r-a}{b}\right).$$

The obtained results are:
- for $n = 509, q = 2048, L = 35, H_{\min} = 301$;
- for $n = 677, q = 2048, L = 35, H_{\min} = 367$;
- for $n = 821, q = 4096, L = 75, H_{\min} = 399$.

Finally, Pad^{-1} takes the first $32\,L/5$ entries and applies the inverse of $\underline{\zeta}$. Its output is always a byte array of length L.

3.2 Message masking

As in NTRUEncrypt, the IND-CPA security is achieved by masking the message. However, the only way to mask $m = \mathrm{Pad}(msg, seed) \in T(d)$ while maintaining the constraint is to apply a permutation, which is not secure. Thus, the message is masked before the padding function using the digest of the required random polynomial $r \in T$, resulting in

$$m = \mathrm{Pad}(msg \oplus \mathrm{Hash}(r), seed) \in T(d)\,.$$

3.3 The PKE scheme

The algorithm for key generation is OW-CPA.keygen from NTRU.

To encrypt $msg \in (\mathbb{Z}_{2^8})^L$ using the public key h, the steps are:

- to sample $r \in T$, obtain $m = \mathrm{Pad}(msg \oplus \mathrm{Hash}(r), seed) \in T(d)$;
- to return $c = \text{OW-CPA.encrypt}(h, r, m)$.

To decrypt c with the secret key (f, f_3, h_q), the algorithm procceds as follows:

- $(r, m, fail) = \text{OW-CPA.decrypt}(f, f_3, h_q, c)$;
- if $fail = 0$ then return $msg = \mathrm{Pad}^{-1}(m) \oplus \mathrm{Hash}(r)$, else return $\perp$.

4 Conclusions

In this work a IND-CCA2 PKE scheme is obtained from the KEM in the NTRU [2] submission to the NIST PQC Standardization Process. Inspired by NTRUEncrypt, the NAEP transformation is used and two new functions are introduced: i) a padding function that adds more than 256 bits of entropy and encodes messages of 35 or 75 bytes depending on the security level; ii) a message masking that gives IND-CPA security to the resulting scheme. Performance and data-size of the obtained PKE scheme are analogous to those of the KEM in NTRU (benchmarks available in [9]), making the PKE scheme a valid post-quantum alternative.

References

1. G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kesley, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, and D. Smith-Tone. NIST Internal Report 8309: Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process. Technical report, NIST, 2020.
2. C. Chen, O. Danba, J. Hoffstein, A. Hülsing, J. Rijneveld, J. Schanck, P. Schwabe, W. Whyte, and Z. Zhang. NIST PQC Standardization Round 2 Submission: NTRU, Algorithm Specifications And Supporting Documentation. Technical report, NIST, 2019.
3. C. Chen, J. Hoffstein, W. Whyte, and Z. Zhang. NIST PQC Standardization Round 1 Submission - NTRUEncrypt, a lattice based encryption algorithm. Technical report, NIST, 2017.
4. Y. Doröz and B. Sunar. Flattening NTRU for evaluation key free homomorphic encryption. *Journal of Mathematical Cryptology*, 14(1):66–83, 2020.
5. J. Hoffstein, J. H. Silverman, W. Whyte, and Z. Zhang. A signature scheme from the finite field isomorphism problem. *Journal of Mathematical Cryptology*, 14(1):39–54, 2020.
6. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *LNCS*, pages 267–288. Springer, Heidelberg, June 1998.
7. Nick Howgrave-Graham, Joseph H. Silverman, Ari Singer, and William Whyte. NAEP: Provable security in the presence of decryption failures. Cryptology ePrint Archive, Report 2003/172, 2003. `https://eprint.iacr.org/2003/172`.
8. G. De Micheli, N. Heninger, and B. Shani. Characterizing overstretched NTRU attacks. *Journal of Mathematical Cryptology*, 14(1):110–119, 2020.
9. VAMPIRE working groups. eBACS: ECRYPT Benchmarking of Cryptographic Systems - Measurements of key-encapsulation mechanisms, indexed by machine. Technical report, VAMPIRE: Virtual Applications and Implementations Research Lab, 2020.