NTRU Cryptosystems Technical Report

Report # 001, Version 2 (8/97)
Title: Commutative NTRU: Pseudo-code implementation
Author: Joseph H. Silverman
Release Date: March 13, 1997

Abstract. Pseudo-code is given for the commutative polynomial implementation of the NTRU Public Key Cryptosystem.

The commutative polynomial implementation of the NTRU PKCS uses polynomials of degree N. For the purposes of computer implementation, it is most convenient to represent polynomials as vectors

$$\mathbf{a} = [a_1, a_2, \ldots, a_N] \quad \text{corresponding to} \quad a(X) = \sum_{i=1}^{N} a_i X^{N-i}.$$

Pseudo-code will be given for the following high-level routines implementing the NTRU PKCS.

`CREATEKEY`(N, K, q, p, r)	Create an NTRU public/private key
`ENCODE`$(N, K, q, p, d, m, h_1, \ldots, h_K)$	Encode the message m
`DECODE`(N, q, p, e, f)	Decode the message e

We also give pseudo-code for the following low-level polynomial routines:

`STARMULTIPLY`(a, b, N, M)	Multiply $a * b$ (reduce modulo M if $M > 0$)
`INVERSEPOLY`(a, N, M)	Compute the inverse of a modulo $(M, X^N - 1)$

We leave for the reader to supply their own routine for performing division with remainder in the polynomial ring $(\mathbf{Z}/P\mathbf{Z})[x]$ when P is prime. We will further assume that the user is able to generate cryptographically secure random bits.

```
CREATEKEY(N, K, q, p, r)
[Comments. Create a private key (f, F_p) and public key (h_1, ..., h_K) corresponding
to the integers parameters (q, p, r, s). ]
    Choose a vector f with coordinates randomly selected
        between -r/2 and r/2.
    F_q = INVERSEPOLY(f, N, q)
    F_p = INVERSEPOLY(f, N, p)
    IF EITHER INVERSION FAILS, SELECT A NEW f AND REPEAT
    Choose vectors g_1, ..., g_K with coordinates randomly
        selected between -r/2 and r/2.
    DO i = 1 TO K
        h_i = STARMULTIPLY(F_q, g_i, N, q)
    END i LOOP
    RETURN Private Key (f, F_p) and Public Key (h_1, ..., h_K)
```

```
ENCODE(N, K, q, d, m, h_1, ..., h_K)
[Comments. Encode the message m using the public key (h_1, ..., h_K). The message
m should be a vector with coordinates between -s/2 and s/2 for an appropriate
value of s. ]
    e = m
    DO i = 1 TO K
        Randomly choose a vector φ having d coordinates equal to
            each of 1 and -1 and the rest 0.
        e = e + p · STARMULTIPLY(φ, h_i)
    END i LOOP
    e = e (mod q) with coefficients between -q/2 and q/2
    RETURN e
```

```
DECODE(N, q, p, f, F_p, e)
[Comments. Use the private key (f, F_p) to decode the message e. ]
    a = STARMULTIPLY(f, e, N, q)
    m = STARMULTIPLY(a, F_p, N, p)
    RETURN m
```

`STARMULTIPLY`$(\mathbf{a}, \mathbf{b}, N, M)$
[*Comments.* Compute $a(X) * b(X)$ modulo $X^N - 1$, and reduce modulo M if $M > 0$. This implementation requires N^2 multiplications. If either **a** or **b** has small coeffficients (e.g., is a binary polynomial), there are faster implementations. If N is large enough, it may be faster to compute the (convolution) product $a(X) * b(X)$ using fast Fourier transforms.]

```
    DO k = 1 TO N
        c_k = 0
        j = k - 1
        DO i = 1 TO N
            IF j = 0
                j = N
            ENDIF
            c_k = c_k + a_i * b_j
            j = j - 1
        END i LOOP
        IF M > 0
            c_k = c_k  (mod M) with |c_k| <= M/2
        ENDIF
    END k LOOP
    RETURN c = [c_1, c_2, ..., c_N]
```

$\mathtt{INVERSEPOLY}(\mathbf{a}, N, M)$

[*Comments.* Compute the inverse of $a(X)$ modulo the ideal $(M, X^N - 1)$. We will assume that $M = P^r$ is a power of a prime, since that suffices for most applications. To compute the inverse of $a(X)$ modulo $(M, X^N - 1)$ for general M, one first factors M as $M = \prod P_i^{r_i}$, next computes the inverse $a_i(X)^{-1}$ modulo $(P_i^{r_i}, X^N - 1)$, and finally combines the $a_i(X)^{-1}$'s using the Chinese remainder theorem to obtain $a(X)^{-1}$. The first part of our routine is adapted from [1, §1.3.2]. The second part uses the fact that if c is close to a^{-1}, then $2c - ac^2$ is much closer to a^{-1}.]

(Compute the inverse of $\mathbf{a}$ modulo P)

$\quad \mathbf{b} = X^N - 1;\ \ \mathbf{d} = \mathbf{a};\ \ \mathbf{u} = 1;\ \ \mathbf{v}_1 = 0;\ \ \mathbf{v}_3 = \mathbf{b};$

$\quad \mathtt{WHILE}\ \mathbf{v}_3 \neq 0$

$\quad\quad \mathtt{WRITE}\ \mathbf{d} = \mathbf{v}_3 * \mathbf{q} + \mathbf{t}_3 \pmod{P}\ \mathtt{WITH}\ \deg(\mathbf{t}_3) < \deg(\mathbf{v}_3)$

$\quad\quad\quad$ (This is long division of polynomials in $(\mathbf{Z}/P\mathbf{Z})[X]$.)

$\quad\quad \mathbf{t}_1 = \mathbf{u} - \mathbf{q} * \mathbf{v}_1;\ \ \mathbf{u} = \mathbf{v}_1;\ \ \mathbf{d} = \mathbf{v}_3;\ \ \mathbf{v}_1 = \mathbf{t}_1;\ \ \mathbf{v}_3 = \mathbf{t}_3$

$\quad \mathtt{END\ WHILE}$

$\quad \mathtt{IF}\ \deg(\mathbf{d}) > 0$

$\quad\quad$ `RETURN 'Inversion Fails'`

$\quad \mathtt{ENDIF}$

$\quad \mathbf{c} = d_0^{-1}\mathbf{u} \pmod{P}$

(If $r > 1$, refine to get an inverse modulo P^r)

$\quad Q = P$

$\quad \mathtt{WHILE}\ Q < P^r$

$\quad\quad \mathbf{c}' = \mathtt{STARMULTIPLY}(\mathbf{c}, \mathbf{c}, N, P^r)$

$\quad\quad \mathbf{c}' = \mathtt{STARMULTIPLY}(\mathbf{a}, \mathbf{c}', N, P^r)$

$\quad\quad \mathbf{c} = 2\mathbf{c} - \mathbf{c}' \pmod{P^r}$

$\quad\quad Q = Q^2$

$\quad \mathtt{END\ WHILE}$

$\quad \mathtt{RETURN}\ \mathbf{c} \pmod{P^r}$

References

[1] H. Cohen, *A course in computational algebraic number theory*, Graduate Texts in Math., vol. 138, Springer Verlag, Berlin, 1993

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.tiac.net/~ntru`
