AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Metody kryptografii w analizie danych

Kryptografia postkwantowa – Algorytm NTRUEncrypt

Autorzy:

Gabriela Bocheńska, Roksana Cieśla, Aleksandra Stachniak

1. Wprowadzenie

NTRU to zaawansowany kryptosystem oparty na kratowych problemach trudnych obliczeniowo, stworzony z myślą o zapewnieniu bezpiecznej komunikacji w erze post-kwantowej. Algorytm jest przeznaczony głównie do szyfrowania wiadomości oraz wymiany kluczy, zapewniając jednocześnie wysoką efektywność obliczeniową i niskie wymagania pamięciowe.

Cechy Algorytmu NTRU:

1. **Asymetryczny**: NTRU jest asymetrycznym algorytmem kryptograficznym, co oznacza, że używa pary kluczy: klucza publicznego do szyfrowania oraz klucza prywatnego do deszyfrowania.
2. **Oparty na kratach**: Wykorzystuje struktury matematyczne znane jako kraty (ang. lattice). Kraty to regularne, powtarzalne siatki punktów w wielowymiarowej przestrzeni. Problemy związane z kratami, takie jak Approximate Shortest Vector Problem (γ-SVP) i Unique Shortest Vector Problem (uSVP), są podstawą jego bezpieczeństwa.
3. **Odporny na ataki kwantowe:** W przeciwieństwie do tradycyjnych algorytmów, takich jak RSA czy ECC, które mogą być złamane za pomocą komputerów kwantowych, NTRU jest oparty na problemach, które są trudne do rozwiązania nawet dla komputerów kwantowych, co zapewnia dodatkową warstwę bezpieczeństwa.
4. **Efektywny obliczeniowo**: NTRU charakteryzuje się wysoką efektywnością obliczeniową. Operacje szyfrowania i odszyfrowywania są zoptymalizowane pod kątem szybkości, co sprawia, że algorytm ten jest wyjątkowo wydajny. Niskie wymagania pamięciowe sprawiają, że NTRU jest idealnym rozwiązaniem dla urządzeń o ograniczonych zasobach, takich jak urządzenia IoT, gdzie zarówno moc obliczeniowa, jak i pamięć są ograniczone.

Kluczowe komponenty:

- **Generowanie Kluczy (Key Generation)**: Proces generowania kluczy jest fundamentalnym krokiem w algorytmie NTRU. Obejmuje tworzenie klucza publicznego i klucza prywatnego. Klucz publiczny jest używany do szyfrowania wiadomości, podczas gdy klucz prywatny jest używany do ich odszyfrowywania.
- **Szyfrowanie (Encryption):** Proces szyfrowania jest używany do zabezpieczania wiadomości przy użyciu klucza publicznego. Obejmuje konwersję wiadomości do formy binarnej, generowanie losowego wielomianu r, a następnie obliczanie ciphertextu.
- **Deszyfrowanie (Decryption)**: Proces deszyfrowania jest używany do odzyskiwania oryginalnej wiadomości z ciphertextu przy użyciu klucza prywatnego. Obejmuje operacje na wielomianach i redukcje modulo, aby odzyskać oryginalną wiadomość.

- **Operacje na wielomianach**: Operacje na wielomianach są kluczowym aspektem algorytmu NTRU. Obejmują dodawanie, mnożenie, oraz znajdowanie odwrotności wielomianów. Te operacje są wykonywane w kontekście pierścienia wielomianów modulo, co zapewnia efektywność obliczeniową i bezpieczeństwo algorytmu.

2. Ogólna specyfikacja algorytmu

2.1. Parametry systemu

Inicjalizacja Parametrów: Na początku, przy inicjalizacji klasy `NTRUdecrypt`, ustawiane są parametry N (rozmiar pierścienia wielomianów), p (moduł odwrotności wielomianu f dla fp), q (moduł odwrotności wielomianu f dla fq), df (liczba współczynników równych 1 w wielomianie f), dg (liczba współczynników równych 1 w wielomianie g), i dr (liczba współczynników równych 1 w losowym wielomianie używanym do szyfrowania).

- **N**: Rząd pierścienia wielomianów.
- **p**: Moduł odwrotności wielomianu f dla fp.
- **q**: Moduł odwrotności wielomianu f dla fq.
- **d**: Liczba jedynek w losowym wielomianie r używanym do szyfrowania.
- **df**: Liczba jedynek w prywatnym wielomianie f.
- **dg**: Liczba jedynek w prywatnym wielomianie g.

2.2. Generowanie kluczy

Proces generowania kluczy obejmuje tworzenie losowych wielomianów f i g, a następnie obliczanie klucza publicznego h.

- **Generowanie Wielomianów f i g**: Następnie wywoływana jest metoda `genfg()`, która losowo generuje wielomiany f i g. Wielomiany te są wygenerowane w sposób, który zapewnia spełnienie wymagań dotyczących liczby jedynek w tych wielomianach.

- **Obliczanie Odwrotności Wielomianu f**: Po wygenerowaniu wielomianów f i g, następuje obliczenie odwrotności wielomianu f. Metoda `invf()` sprawdza, czy istnieją odwrotności wielomianu f względem wartości p i q. Jeśli tak, to odwrotności te są obliczane i przechowywane w klasie.

- **Obliczanie Klucza Publicznego h**: Po wygenerowaniu i obliczeniu odwrotności wielomianu f, klucz publiczny h jest obliczany za pomocą wzoru: Funkcja `genh()` wykonuje to obliczenie.

$$h = p * f_q * g \ (mod\ q)$$

- **Zapis Kluczy**: Ostatecznie, klucze publiczny i prywatny są zapisywane do plików.

.

Funkcje:

- *genPubPriv*: Generuje klucze publiczne i prywatne.
- *genfg*: Generuje losowe wielomiany f i g oraz ich odwrotności.
- *genh*: Oblicza klucz publiczny h.

Algorithm 1 ntru-pke.KEYGEN

Input: A parameter set PARAM $= \{N, p, q, d\}$ and a *seed*.
1: Instantiate Sampler with $\mathcal{T}(d+1, d)$ and *seed*;
2: $\mathbf{f} \leftarrow$ Sampler
3: **if** $\mathbf{f}$ is not invertible mod q **then** go to step 2 **end if**
4: $\mathbf{g} \leftarrow$ Sampler
5: $\mathbf{h} = \mathbf{g}/(p\mathbf{f} + 1) \bmod q$
Output: Public key $\mathbf{h}$ and secret key $(p\mathbf{f}, \mathbf{g})$

2.3. Szyfrowanie

Proces szyfrowania w NTRU polega na mnożeniu losowego wielomianu r z kluczem publicznym h oraz dodawaniu wiadomości m. W wyniku tych operacji powstaje zaszyfrowana wiadomość e.

- **Przygotowanie wiadomości:** Wiadomość do zaszyfrowania jest konwertowana na formę binarną. Jej długość nie może przekraczać N.
- **Generowanie losowego wielomianu r:** Generowany jest losowy wielomian r o współczynnikach 1, -1 i 0, który służy jako maska do wiadomości.
- **Obliczanie Zaszyfrowanej Wiadomości:** Zaszyfrowana wiadomość e e jest obliczana jako:

$$e = r * h + m \ (mod\ q),$$

gdzie m jest wiadomością, a h kluczem publicznym.

Funkcje:

- *encryptString*: Konwertuje wiadomość na bity, generuje losowy wielomian *r,* ustawia wiadomość *m* i oblicza zaszyfrowaną wiadomość *e*.
- *encrypt*: Przeprowadza faktyczne szyfrowanie wiadomości *m* przy użyciu klucza publicznego *h* i losowego wielomianu *r*.

Algorithm 2 ntru-pke.ENCRYPT

Input: Public key $\mathbf{h}$, message msg of length $mlen$, a parameter set PARAM and a $seed$.

$\mathbf{m} = \text{Pad}(msg, seed)$
$rseed = \text{HASH}(\mathbf{m}|\mathbf{h})$
Instantiate Sampler with $\mathcal{T}$ and $rseed$;
$\mathbf{r} \leftarrow$ Sampler
$\mathbf{t} = \mathbf{r} * \mathbf{h}$
$tseed = \text{HASH}(\mathbf{t})$
Instantiate Sampler with $\mathcal{T}$ and $tseed$;
$\mathbf{m}_{mask} \leftarrow$ Sampler
$\mathbf{m}' = \mathbf{m} - \mathbf{m}_{mask} \pmod{p}$
$\mathbf{c} = \mathbf{t} + \mathbf{m}'$

Output: Ciphertext $\mathbf{c}$

2.4. Deszyfrowanie

Zaszyfrowana wiadomość *e* jest odszyfrowywana przy użyciu klucza prywatnego *f*. Proces odszyfrowywania obejmuje operacje na wielomianach i redukcje modulo, aby odzyskać oryginalną wiadomość *m*.

- **Przygotowanie wielomianu a:** Odszyfrowywanie rozpoczyna się od obliczenia wielomianu a przy użyciu klucza prywatnego f**:**

$$a = f * e \ (mod\ q)$$

- **Redukcja modulo p:** Następnie wielomian aa jest redukowany modulo p, aby uzyskać:

$$b = a \ (mod\ p)$$

- **Odszyfrowanie wiadomości:** Ostatecznie wiadomość *m*m jest odzyskiwana poprzez pomnożenie *b* przez odwrotność *fp*:

$$m = f_p * b \ (mod\ p)$$

Funkcje:

- *decryptString*: Konwertuje zaszyfrowaną wiadomość na tablicę numpy, odszyfrowuje każdy blok i zwraca odszyfrowaną wiadomość jako ciąg znaków.
- *decrypt*: Przeprowadza odszyfrowywanie wiadomości *e* przy użyciu klucza prywatnego *f*, obliczając pośrednie wielomiany i odzyskując oryginalną wiadomość *m*.

Algorithm 3 ntru-pke.DECRYPT

Input: Secret key $\mathbf{f}$, public key $\mathbf{h}$, ciphertext $\mathbf{c}$, and a parameter set PARAM.

$\mathbf{m}' = \mathbf{f} * \mathbf{c} \pmod{p}$
$\mathbf{t} = \mathbf{c} - \mathbf{m}$
$tseed = \text{HASH}(\mathbf{t})$
Instantiate Sampler with $\mathcal{T}$ and $tseed$;
$\mathbf{m}_{mask} =$ Sampler
$\mathbf{m} = \mathbf{m}' + \mathbf{m}_{mask} \pmod{p}$
$rseed = \text{HASH}(\mathbf{m}|\mathbf{h})$
Instantiate Sampler with $\mathcal{T}$ and $rseed$;
$\mathbf{r} \leftarrow$ Sampler
$msg, mlen = \text{Extract}(\mathbf{m})$
if $p \cdot \mathbf{r} * \mathbf{h} = \mathbf{t}$ **then**
    $result = msg, mlen$
else
    $result = \perp$
end if

Output: $result$

2.5. Operacje na wielomianach

Kluczowym aspektem algorytmu NTRU są operacje na wielomianach, w tym dodawanie, mnożenie oraz znajdowanie odwrotności wielomianów. Te operacje są wykonywane w kontekście pierścienia wielomianów modulo.

Funkcje:

- *poly_inv*: Znajduje odwrotność wielomianu w ciele Galois GF(*poly_mod*).
- *padArr*: Dodaje wiodące zera do tablicy numpy, aby osiągnąć pożądany rozmiar tablicy.
- *genRand10*: Generuje losowy wielomian o określonej liczbie jedynek i minus jedynek, używany do generowania losowych wielomianów *f, g* i *r.*

3. Uwagi

NTRU ma najmniejsze rozmiary kluczy publicznych i zaszyfrowanych wiadomości w porównaniu do innych rozwiązań opartych na kratach, takich jak NewHope czy Kyber. Jest to szczególnie interesujące dla protokołów wymiany kluczy, ponieważ w tych protokołach kluczowe jest, aby dane wymiany mieściły się w pojedynczym Maksymalnym Rozmiarze Transmisji (MTU). W przeciwnym razie można oczekiwać, że pewne pakiety zostaną utracone, co potencjalnie spowolni protokół i wymaga dodatkowych funkcji na poziomie protokołu (takich jak zarządzanie fragmentacją), a nawet może spowodować niepowodzenie wymiany kluczy.