AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Metody kryptografii w analizie danych

Kryptografia postkwantowa – Algorytm CFPKM

Autorzy:

Gabriela Bocheńska, Roksana Cieśla, Aleksandra Stachniak

1. Wprowadzenie

Algorytm CPFKM (Cryptographic Polynomial-based Function Key Management) jest kryptograficznym algorytmem opartym na wielomianach, który jest przeznaczony do bezpiecznego zarządzania kluczami, został zaprojektowany z myślą o odporności na ataki kwantowe. Algorytm ten wykorzystuje kryptografię opartą na kodach, co sprawia, że jest odporny na ataki zarówno klasyczne, jak i kwantowe.

Cechy Algorytmu CPFKM:

1. **Asymetryczny**: CPFKM jest algorytmem kryptografii asymetrycznej, co oznacza, że wykorzystuje parę kluczy - klucz publiczny do szyfrowania i klucz prywatny do deszyfrowania.
2. **Bezpieczeństwo opiera się na problemach wielomianowych**: Algorytm wykorzystuje trudność problemów związanych z wielomianami, aby zapewnić bezpieczeństwo kryptograficzne. Specyficzne właściwości wielomianów i ich wartości są wykorzystywane do generowania kluczy i operacji kryptograficznych.
3. **Zarządzanie kluczami**: CPFKM jest przeznaczony do zarządzania kluczami, co obejmuje generowanie kluczy, enkapsulację kluczy (szyfrowanie) oraz dekapsulację kluczy (deszyfrowanie).
4. **Efektywność**: Algorytm jest zaprojektowany tak, aby być efektywnym pod względem obliczeniowym, co oznacza, że może być stosowany w praktycznych aplikacjach wymagających wysokiej wydajności.

Kluczowe komponenty:

- Generowanie kluczy: Proces tworzenia pary kluczy publicznych i prywatnych przy użyciu losowych kodów liniowych.
- Szyfrowanie: Użycie klucza publicznego do bezpiecznego szyfrowania wiadomości.
- Deszyfrowanie: Użycie klucza prywatnego do odszyfrowania wiadomości.

2. Ogólna specyfikacja algorytmu CFPKM

2.1 Przestrzeń parametrów

Algorytm wykorzystuje następujące parametry:

- q: Duża liczba całkowita, która definiuje skończone pole F_q dla pierścienia wielomianów P. Zazwyczaj jest to liczba w postaci 2^k, gdzie k jest dodatnią liczbą całkowitą.
- n: Liczba zmiennych określająca pierścień wielomianów P.
- m: Liczba równań w systemie równań.
- s: Całkowita liczba definiująca zakres wartości, z którego są losowo wybierane sekrety i błędy.

- B: Liczba najbardziej znaczących bitów używanych do utworzenia klucza sesji.

2.2 Klucz prywatny i klucz publiczny

Klucz prywatny składa się z losowej wartości początkowej (seed) oraz wektora tajnych wartości, $sa\,\epsilon[0,s]^n$ losowo wybranego z rozkładu jednostajnego U_n^s. Wartość początkowa służy do wygenerowania układu wielomianów$f'_1,\cdots,\ f'_n\epsilon F_q[x_1,\dots,x_n]$, które są później używane do tworzenia klucza publicznego. Struktura klucza prywatnego to:

$$SK = (seed||sa)$$

Klucz publiczny zawiera tę samą losową wartość nasiona oraz wektor $b_1\epsilon F_q^m$. Wektor ten jest wynikiem rozwiązania układu wielomianów kwadratowych (lub wyższego stopnia) z dodanym szumem $f_1,\cdots,\ f_m\epsilon F_q[x_1,\dots,x_n]$, dla losowej wartości tajnej sa. Każdy i-ty element wektora jest określany jako: $b_{1i} = f_i(sa)$, gdzie każdy f_i to wielomian z szumem o postaci: $f_i(x_1,\dots,x_n) = f'_i(x_1,\dots,x_n) + e_i$, gdzie e_i jest szumem wybranym losowo z tego samego zakresu $\langle 0,s\rangle$ co sa. Struktura klucza publicznego to:

$$PK = (seed||b_1)$$

Algorytmy wymiany kluczy

Algorytm wymiany kluczy obejmuje trzy główne procedury:

- Generowanie klucza prywatnego i publicznego.
- Enkapsulacja klucza sesji za pomocą klucza publicznego.
- Dekapsulacja klucza sesji za pomocą klucza prywatnego.

Procedury te zapewniają bezpieczne zarządzanie i wymianę kluczy w systemie kryptograficznym, co przedstawiono na załączonym schemacie.

Alice		Bob
KeyGen() :		
$\mathrm{seed} \xleftarrow{\$} \{0,1\}^8$		
$\mathrm{f} \leftarrow \mathrm{PolGen(seed)}$		
$\mathbf{sa} \xleftarrow{\$} \mathcal{U}_s^n$		
$\mathbf{e_1} \xleftarrow{\$} \mathcal{U}_s^m$		
$\mathbf{b_1} \leftarrow f(\mathbf{sa}) + \mathbf{e_1}$		
$\mathrm{PK} \leftarrow \mathrm{seed} \Vert \mathbf{b_1}, \mathrm{SK} \leftarrow \mathrm{seed} \Vert \mathbf{sa}$		
	$\xrightarrow[\in \mathbb{Z}_q^{m \times 1} \times \{0,1\}^{\mathrm{SEEDSIZE}}]{\mathrm{PK}}$	
		Encaps():
		$\mathrm{seed}, \mathbf{b1} \leftarrow unpack\_pk(PK)$
		$\mathrm{f} \leftarrow \mathrm{PolGen(seed)}$
		$\mathbf{sb} \xleftarrow{\$} \mathcal{U}_s^n$
		$\mathbf{e_2} \xleftarrow{\$} \mathcal{U}_s^m$
		$\mathbf{b_2} \leftarrow f(\mathbf{sb}) + \mathbf{e_2}$
		$\mathbf{e_3} \xleftarrow{\$} \mathcal{U}_s^m$
		$\mathbf{b_3} \leftarrow f(\mathbf{sb}) \odot \mathbf{b_1} + \mathbf{e_3}$
		$\mathbf{c} \leftarrow \mathrm{CrossRound}(\mathbf{b_3}, B)$
		$\mathrm{Key}_B \leftarrow \mathrm{Rounding}(\mathbf{b_3}, B)$
		$\mathrm{ct} \leftarrow \mathbf{c} \Vert \mathbf{b2}$
	$\xleftarrow[\in \mathbb{Z}_q^{m \times 1} \times \mathbb{Z}_2^{m \times 1}]{\mathrm{ct}}$	
Decaps():		
$\mathbf{b_2}, \mathbf{c} \leftarrow unpack_c t(ct)$		
$\mathrm{seed}, \mathbf{sa} \leftarrow unpack_s k(SK)$		
$\mathrm{f} \leftarrow \mathrm{PolGen(seed)}$		
$\mathrm{Key}_A \leftarrow \mathrm{Red}(f(\mathbf{sa}) \odot \mathbf{b_2}, \mathbf{c}, B)$		

2.3. Funkcje algorytmu KEM (Key Encapsulation Mechanism)

2.3.1 Generowanie Pary Kluczy

Funkcja: **crypto_kem_keypair()**

W tej funkcji generowane są klucz publiczny (PK) i klucz prywatny (SK). Proces ten przebiega następująco:

- **Generowanie nasiona:** Tworzona jest losowa wartość początkowa, zwana nasionem. Nasiono to będzie używane do generowania wielomianów.
- **Generowanie wielomianów:** Funkcja **polgen** wykorzystuje nasiono do wygenerowania układu m wielomianów kwadratowych o n zmiennych. Każdy wielomian $f_i(x)$ ma współczynniki wybrane z zakresu [0, q]. Ustawienie ziarna dla generatora liczb losowych zapewnia deterministyczność, co jest przydatne podczas testowania i debugowania. Wielomiany są generowane z losowymi współczynnikami, co zapewnia, że każdy wygenerowany wielomian jest unikalny i trudny do przewidzenia.
- **Struktura wielomianów:** Każdy wielomian f_i jest reprezentowany przez trzy wektory: QD (współczynniki dla kwadratowych jednomianów), L (współczynniki dla liniowych

jednomianów) i C (wyraz wolny). Te wektory są wypełniane przy użyciu funkcji losowej i nasiona.

- **Generowanie tajnego wektora i szumu:** Losowo generowany jest tajny wektor sa o wymiarze n oraz wektor błędu e_1 o wymiarze m.
- **Obliczanie wektora $\boldsymbol{b_1}$:** Wartości wielomianów f_i są obliczane dla tajnego wektora sa i dodawany jest do nich szum, tworząc wektor $\boldsymbol{b_1}$: $b_{1i} = (f_i(sa) + e_{1i}) \, mod \, q$
- **Tworzenie kluczy:** Klucz publiczny (PK) jest tworzony przez połączenie nasiona i wektora b_1. Klucz prywatny (SK) jest tworzony przez połączenie nasiona i tajnego wektora sa.

2.3.2 Enkapsulacja klucza

Funkcja: **krypto_kem_enc(pk)**

Proces enkapsulacji klucza polega na zakodowaniu wspólnego sekretu przy użyciu klucza publicznego (PK):

- **Pobranie składników klucza publicznego:** Klucz publiczny jest rozpakowywany, aby uzyskać wektor $b1$ i nasiono.
- **Generowanie wielomianów:** Za pomocą funkcji **polgen** i nasiona generowany jest ten sam układ wielomianów kwadratowych, co przy generowaniu pary kluczy.
- **Losowe wektory:** Generowane są losowe wektory sb(tajny), e_2 (szum) i e_3 (szum).
- **Obliczanie wektorów:** Obliczane są wartości wielomianów dla wektora sb z dodanym szumem, tworząc wektor b_2: $b_{2i} = (f_i(sb) + e_{2i}) \, mod \, q$. Obliczany jest wektor b_3 jako iloczyn wektorów $f_i(sb)$ i b_1 z dodanym szumem e_3.
- **Pakowanie i rozpakowywanie kluczy:** Funkcje te służą do pakowania i rozpakowywania klucza tajnego (secret key, sk). Funkcja **pack_sk** łączy ziarno (**seed**) i inne wartości (**sa**) w jedną strukturę danych. **Rozpakowywanie**: Funkcja **unpack_sk** dzieli składowe na ich oryginalne części, co umożliwia ich późniejsze wykorzystanie.
- **Pakowanie i rozpakowywanie szyfrogramu:** Funkcja **pack_ct** łączy listę c i tablicę b2 w jeden ciąg bajtów. Funkcja **unpack_ct** dzieli ciąg bajtów na oryginalne składniki.
- **Funkcje pomocnicze:** Funkcja **kem_crossround1(in_val)** przekształca wektor b_3 na wektor wskazówek c. Funkcja wykonuje zaokrąglenie na każdym elemencie wektora. Każdy element wektora jest zaokrąglany przy użyciu funkcji rounding. Funkcja **rounding(in_val)** generuje klucz dla odbiorcy z najważniejszych bitów wektora b_3. Funkcja rounding jest używana do zaokrąglania wartości do najbliższego dopuszczalnego poziomu, co jest istotne w wielu algorytmach kryptograficznych. Dodanie 2**(B_BAR - 1) do wartości przesuwa ją w górę, umożliwiając zaokrąglenie w późniejszym kroku. Operacja modulo zapewnia, że wartość mieści się w zakresie [0, Q-1]. Przesunięcie w prawo (>> B_BAR) realizuje zaokrąglenie, zmniejszając precyzję wartości.
- **Tworzenie szyfrogramu:** Szyfrogram (ct) jest tworzony przez połączenie wektorów b_2 i c. Wspólny sekret (SS) to wynik działania funkcji **rounding** na b_3.

2.3.3 Dekapsulacja klucza

Funkcja: **crypto_kem_dec(sk, ct)**

Proces dekapsulacji klucza polega na odzyskaniu wspólnego sekretu przy użyciu szyfrogramu (ct) i klucza prywatnego (sk):

- **Pobranie składników klucza prywatnego**: Klucz prywatny jest rozpakowywany, aby uzyskać tajny wektor sa i nasiono.
- **Rozpakowanie szyfrogramu:** Szyfrogram jest rozpakowywany, aby uzyskać wektory b_2 i c.
- **Generowanie wielomianów:** Za pomocą nasiona generowany jest ten sam układ wielomianów kwadratowych, co wcześniej.
- **Obliczanie wektora:** Obliczane są wartości wielomianów dla wektora sa i mnożone przez b_2.

Funkcja **Red(kem_rec(key, w, c, B_BAR)**): Funkcja **Red** sprawdza, czy wartości w_i spełniają warunki i zwraca odpowiednie zaokrąglone wartości, tworząc wspólny sekret (SS).

- **Wynik dekapsulacji:** Wspólny sekret (SS) jest odzyskiwany i zwracany jako wynik funkcji.

3. Uwagi

CFPKM opiera się na niewielkich tajemnicach i błędach, co stanowi jedną z wad proponowanego schematu. Jednakże, istnieje wiele zalet tego podejścia.

Przede wszystkim, mechanizm kapsułkowania klucza został zaprojektowany w sposób, który wykorzystuje dane obu użytkowników do wygenerowania wspólnego klucza, co odróżnia go od tradycyjnych metod. Ta elastyczność pozwala na łatwe dostosowanie CFPKM do różnych protokołów wymiany i uzgadniania klucza.

Co więcej, CFPKM oferuje korzyści pod względem kosztów komunikacji i rozmiaru klucza. W porównaniu z innymi podobnymi mechanizmami, CFPKM zapewnia podobny poziom bezpieczeństwa przy znacznie mniejszych wartościach parametrów. Ta oszczędność wynika z wykorzystania nowoczesnych problemów kryptograficznych, które są trudne do rozwiązania nawet dla potencjalnych atakujących.

W związku z tym, chociaż istnieją pewne wady, to korzyści płynące z zastosowania CFPKM są znaczące i sprawiają, że jest to atrakcyjna opcja dla wielu aplikacji kryptograficznych.

- **Zastosowanie w kryptografii postkwantowej:** Algorytm może być częścią większego schematu kryptograficznego mającego na celu zapewnienie bezpieczeństwa w obliczu przyszłych komputerów kwantowych. Może korzystać z technik opartych na problemach trudnych do rozwiązania nawet dla komputerów kwantowych, takich jak lattice-based cryptography.
- **Deterministyczność i losowość:** Wykorzystanie losowych ziaren do generowania kluczy i wartości jest kluczowe dla zapewnienia bezpieczeństwa. Stałe ziarna mogą prowadzić do przewidywalnych wyników, co osłabia bezpieczeństwo.

- **Efektywność:** Operacje takie jak zaokrąglanie, pakowanie i rozpakowywanie są zoptymalizowane pod kątem wydajności. Użycie operacji bitowych i funkcji numpy zapewnia szybkie wykonanie tych operacji.
- **Bezpieczeństwo:** Kluczowe operacje, takie jak generowanie wielomianów i zaokrąglanie, są projektowane w taki sposób, aby minimalizować możliwość odwrócenia lub przewidzenia wyników, co jest istotne dla zachowania bezpieczeństwa kluczy i szyfrogramów.

Przedstawiony algorytm jest złożonym schematem kryptograficznym, który łączy wiele fundamentalnych operacji, aby zapewnić bezpieczną komunikację. Każdy komponent pełni istotną rolę w całym procesie, od generowania losowych wartości po pakowanie i rozpakowywanie danych, co umożliwia bezpieczne przechowywanie i przesyłanie informacji. W kontekście kryptografii postkwantowej, takie algorytmy mogą odgrywać kluczową rolę w przyszłości, zapewniając bezpieczeństwo w świecie, gdzie komputery kwantowe mogą złamać obecne metody kryptograficzne.