

Metody kryptografii w analizie danych

Kryptografia postkwantowa – Algorytm CFPKM

Autorzy:

Gabriela Bocheńska, Roksana Cieśla, Aleksandra Stachniak

1. **Wprowadzenie**

CFPKM, czyli algorytm do Zarządzania Kluczami Publicznymi z Ochroną Przyszłościową (Code-Based Forward Secure Public Key Management), 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.

Główne cechy CFPKM:

* Ochrona przyszłościowa: Gwarantuje, że nawet jeśli klucze długoterminowe zostaną ujawnione, wcześniejsze klucze sesji pozostaną bezpieczne.
* Bezpieczeństwo oparte na kodach: Wykorzystuje trudność dekodowania losowych kodów liniowych, co zapewnia wysoką odporność na ataki.
* Zarządzanie kluczami publicznymi: Zapewnia bezpieczne zarządzanie kluczami publicznymi w systemach kryptograficznych.

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 ​ dla pierścienia wielomianów 𝑃. Zazwyczaj jest to liczba w postaci , 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, losowo wybranego z rozkładu jednostajnego . Wartość początkowa służy do wygenerowania układu wielomianów,⋯ , *,* które są później używane do tworzenia klucza publicznego. Struktura klucza prywatnego to:

**Klucz publiczny** zawiera tę samą losową wartość nasiona oraz wektor ​. Wektor ten jest wynikiem rozwiązania układu wielomianów kwadratowych (lub wyższego stopnia) z dodanym szumem,⋯ , *,* dla losowej wartości tajnej 𝑠𝑎. Każdy i-ty element wektora jest określany jako: , gdzie każdy to wielomian z szumem o postaci:  *,* gdzie ​ jest szumem wybranym losowo z tego samego zakresu ⟨0,𝑠⟩ co 𝑠𝑎. Struktura klucza publicznego to:

**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.

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

**2.3. Funkcje algorytmu KEM (Key Encapsulation Mechanism)**

**2.3.1 Generowanie Pary Kluczy**

Funkcja: **generate\_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 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 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 o wymiarze 𝑚.
* **Obliczanie wektora :** Wartości wielomianów są obliczane dla tajnego wektora 𝑠a i dodawany jest do nich szum, tworząc wektor :
* **Tworzenie kluczy:** Klucz publiczny (PK) jest tworzony przez połączenie nasiona i wektora . Klucz prywatny (SK) jest tworzony przez połączenie nasiona i tajnego wektora 𝑠𝑎.

**2.3.2 Enkapsulacja klucza**

Funkcja: **encrypt(pk, plaintext)**

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), (szum) i (szum).
* **Obliczanie wektorów:** Obliczane są wartości wielomianów dla wektora 𝑠b z dodanym szumem, tworząc wektor : . Obliczany jest wektor jako iloczyn wektorów i z dodanym szumem ​.
* **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 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 Boba z najważniejszych bitów wektora . 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 i 𝑐. Wspólny sekret (SS) to wynik działania funkcji **rounding** na .

**2.3.3 Dekapsulacja Klucza**

Funkcja: **decrypt(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 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

Funkcja **Red(kem\_rec(key, w, c, B\_BAR**)): Funkcja **Red** sprawdza, czy wartości 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.

Początek formularza