Kryptografia Stosowana

Projekt: Atak/Problem MinRank na schemat podpisu Rainbow

Jakub Gardziński
Mikołaj Kłos
Marcin Kruszewski
Mateusz Pietrzak

Prowadzący: dr Adam Komorowski

Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych
Telekomunikacja - Semestr 24Z

27 stycznia 2025

Spis treści

1 Opis teoretyczny ataku

Ataki kryptograficzne to działania mające na celu złamanie algorytmów kryptograficznych lub ich implementacji w celu złamania zabezpieczeń takich jak poufność, integralność, uwierzytelnienie i niezaprzeczalność. Mogą one być pasywne, gdzie atakujący tylko analizuje dane (np. szyfrogramy), lub aktywne, gdzie wchodzi w interakcję z systemem (np. manipulując przesyłanymi danymi).

Podstawowymi klasami ataków są:

- Atak pasywny: Atakujący monitoruje dane (np. szyfrogramy) bez ich modyfikacji. Przykładem jest analiza szyfrogramów w celu identyfikacji schematu szyfrowania.
- Atak aktywny: Atakujący manipuluje danymi lub wchodzi w interakcję z systemem, np. w celu podszycia się pod inną osobę.

Do typowych celów ataków kryptograficznych zaliczamy:

- Złamanie poufności: Uzyskanie informacji chronionych.
- Złamanie integralności: Modyfikacja danych bez wykrycia.
- Złamanie uwierzytelnienia: Podszycie się pod inną osobę lub system.
- Złamanie niezaprzeczalności: Wykazanie, że ktoś zaprzecza swojemu udziałowi w akcji kryptograficznej.

1.1 Matematyczne podstawy

Problem MinRank można zdefiniować jako zadanie znalezienia macierzy X o minimalnym rzędzie, która spełnia następujące równanie:

$$\text{rank}(X) = r \quad \text{dla } AXB = C,$$

gdzie:

- $A \in \mathbb{F}^{m \times n}$, $B \in \mathbb{F}^{p \times q}$, $C \in \mathbb{F}^{m \times q}$ to zadane macierze,
- $X \in \mathbb{F}^{n \times p}$ to macierz niewiadoma,
- r to minimalny rząd macierzy X.

W kryptosystemach opartych na wielomianach wielowymiarowych (MQ) problem MinRank jest używany do analizy strukturalnych słabości algorytmów. W szczególności, w kryptosystemach bazujących na układach równań kwadratowych, atak MinRank pozwala na redukcję złożoności problemu przez wykorzystanie struktur macierzy związanych z kluczami prywatnymi [4].

Ataki MinRank opierają się na następujących elementach matematycznych:

- **Macierze symetryczne:** W przypadku wielomianowych równań kwadratowych, współczynniki kwadratowe można reprezentować jako macierze symetryczne, co pozwala na manipulację strukturą układu.
- **Łańcuchy jądrowe:** Jądra równań centralnych w kryptosystemach MQ mogą tworzyć malejący łańcuch przestrzeni podprzestrzeni, co jest kluczowe w redukcji złożoności problemu.

Złożoność obliczeniowa ataku MinRank zależy od parametrów kryptosystemu:

- n — liczba zmiennych,
- m — liczba równań,
- L — liczba warstw w schemacie,
- r — rząd macierzy,
- q — rozmiar pola skończonego $\mathbb{F}_q$.

Dla schematów takich jak STS (*Step-Wise Triangular Systems*) atak MinRank ma złożoność:

$$O(mn^3Lq^r + n^2Lrq^r).$$

1.2 Przykłady zastosowania

Ataki MinRank znajdują szerokie zastosowanie w kryptoanalizie wielu kryptosystemów, szczególnie tych opartych na układach równań wielomianowych. W przypadku schematów kryptograficznych opartych na wielomianach kwadratowych, takich jak schematy C* (Matsumoto-Imai) oraz HFE (ang. *Hidden Field Equations*), ataki MinRank są stosowane do analizy algebraicznej równań publicznego klucza. Na przykład w HFE, wielomiany definiowane nad rozszerzeniami ciał skończonych można zredukować do problemu minimalnego rzędu macierzy, co pozwala na skuteczną analizę strukturalną i potencjalne osłabienie systemu. W ten sposób ataki MinRank umożliwiają odkrycie ukrytych zależności w macierzach powiązanych z kluczem prywatnym [1].

Podobną podatność można zaobserwować w schematach STS (ang. *Step-Wise Triangular Systems*), które opierają się na równaniach trójkątnych o wielowarstwowej strukturze. Przykładem są kryptosystemy RSE(2)PKC i RSSE(2)PKC, które zostały zaprojektowane z myślą o wydajnej (de)szyfracji. Ataki MinRank wykorzystują fakt, że centralne równania w tych schematach tworzą hierarchię podprzestrzeni jądrowych. Analiza tej hierarchii pozwala na redukcję problemu do układów o minimalnym rzędzie, co umożliwia efektywne odzyskanie klucza prywatnego lub wygenerowanie fałszywego podpisu cyfrowego.

Innym interesującym przykładem zastosowania MinRank jest schemat podpisu cyfrowego Rainbow, który opiera się na wielowarstwowym rozszerzeniu konstrukcji Oil-Vinegar (OV). Konstrukcja Rainbow charakteryzuje się układami równań, gdzie zmienne w kolejnych warstwach są rozdzielone na tzw. zmienne „Oil" i „Vinegar". Atak MinRank pozwala analizować te warstwy iteracyjnie, redukując problem do macierzy o minimalnym rzędzie. To prowadzi do skutecznego odzyskania klucza prywatnego, ponieważ każda warstwa może być badana oddzielnie, wykorzystując jej specyficzną strukturę.

Schematy Oil-Vinegar, zarówno zbalansowane, jak i niezbalansowane, również są podatne na ataki MinRank. Konstrukcja tych schematów zakłada częściowe oddzielenie zmiennych „Oil" i „Vinegar", co pozwala na uproszczenie równań do macierzy powiązanych z kluczem prywatnym. Atak MinRank wykorzystuje różnicę między liczbą zmiennych „Oil" a „Vinegar", co prowadzi do obniżenia rzędu macierzy i łamania schematu.

Wreszcie, uogólnione schematy kryptograficzne, takie jak TPM (ang. *Triangle Plus Minus*) i konstrukcje oparte na biracjonalnych permutacjach, również mogą być analizowane przy użyciu MinRank. W tych przypadkach ataki MinRank pozwalają na uproszczenie równań z dodatkowymi modyfikacjami (np. równania typu „Plus" w TPM) do układów o minimalnym rzędzie, co znacząco osłabia bezpieczeństwo tych systemów.

2 Specyfikacja problemu bazowego kryptosystemu Rainbow

Kryptosystem Rainbow bazuje na problemach matematycznych związanych z wielomianami nad ciałami skończonymi, a jego fundamentem jest trudność rozwiązywania układów równań wielomianowych o wielu zmiennych. Problematyka ta jest uważana za jeden z trudniejszych problemów w matematyce obliczeniowej, co czyni ją atrakcyjną bazą dla kryptografii. Rainbow należy do rodziny kryptosystemów opartych na problemach wielomianowych z wieloma zmiennymi (ang. *Multivariate Quadratic Problems* (MQ-problems)), które polegają na znajdowaniu rozwiązań dla układów równań kwadratowych nad ciałem skończonym.

Podstawowym wyzwaniem jest to, że dla dużej liczby zmiennych i odpowiednio skonstruowanego układu, zadanie rozwiązania tych równań jest obliczeniowo trudne. Złożoność wynika z faktu, że układy te są silnie nieliniowe, co uniemożliwia zastosowanie prostych metod rozwiązywania, takich jak eliminacja Gaussa. Tradycyjne algorytmy rozwiązywania równań wielomianowych w dużej liczbie zmiennych mają wykładniczą złożoność czasową, co sprawia, że znalezienie rozwiązania jest praktycznie niemożliwe w akceptowalnym czasie dla dobrze skonstruowanego systemu. Rainbow rozwija tę koncepcję, wprowadzając układy równań o strukturze warstwowej, co zwiększa elastyczność i bezpieczeństwo systemu. Klucz prywatny kryptosystemu składa się z tzw. map warstwowych, które pozwalają na generowanie podpisów poprzez iteracyjne rozwiązywanie uproszczonych układów równań. Klucz publiczny jest natomiast wynikiem zamaskowania tych równań, co dodatkowo utrudnia ich analizę przez potencjalnego atakującego. Bezpieczeństwo kryptosystemu Rainbow wynika również z faktu, że nie istnieją obecnie efektywne algorytmy rozwiązujące ogólne układy równań kwadratowych w przestrzeni wielowymiarowej w czasie wielomianowym. Choć istnieją algorytmy atakujące ten problem, takie jak metoda brute force, algebraiczne redukcje czy metody baz Gröbnera, ich efektywność jest zbyt niska w stosunku do wymiarów

i parametrów stosowanych w kryptosystemie Rainbow.

Dodatkowym aspektem, który warto podkreślić, jest konstrukcja kryptosystemu odporna na komputery kwantowe. Wraz z rozwojem technologii kwantowej, wiele obecnych kryptosystemów staje się podatnych na ataki wykorzystujące algorytmy kwantowe, takie jak algorytm Shora. Rainbow, jako część kryptografii postkwantowej, opiera się na problemach, które są trudne zarówno dla klasycznych, jak i kwantowych komputerów, co zwiększa jego atrakcyjność jako narzędzia kryptograficznego w przyszłości.

2.1 Specyfikacja matematyczna problemu MQ

1. **Definicja**: Dany jest układ m wielomianów kwadratowych $f_1, f_2, \ldots, f_m$ z n zmiennymi:

$$f_i(x_1, x_2, \ldots, x_n) = \sum_{1 \leqslant j \leqslant k \leqslant n} q_{j,k}^{(i)} x_j x_k + \sum_{1 \leqslant j \leqslant n} l_j^{(i)} x_j + c^{(i)} \quad \text{dla } i = 1, 2, \ldots, m,$$

   gdzie:

   - $q_{j,k}^{(i)}, l_j^{(i)}, c^{(i)} \in \mathbb{F}_q$,
   - x_j to zmienne, a $\mathbb{F}_q$ to ciało skończone o q elementach.

2. **Cel**: Znalezienie wektora $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_q^n$, który spełnia wszystkie równania w układzie.

3. **Złożoność**: Rozwiązywanie problemu MQ jest NP-trudne, co oznacza, że nie istnieją znane efektywne algorytmy rozwiązywania tego problemu w czasie wielomianowym.

3 Praktyczne metody ataku

Atak MinRank na systemy podpisów cyfrowych Rainbow opiera się na wykorzystaniu słabości algebraicznych wynikających z budowy tych systemów. Rainbow, będący rozszerzeniem schematów opartych na wielomianach kwadratowych (MQ), zakłada trudność rozwiązywania układów równań wielomianowych nad ciałami skończonymi. Atak MinRank sprowadza się do redukcji problemu do wyznaczenia macierzy o minimalnym rzędzie, co pozwala osłabić lub złamać bezpieczeństwo algorytmu. W praktyce polega to na przedstawieniu problemu podpisu jako zadania znajdowania układów równań, które mają rozwiązanie o niższym rzędzie niż oczekiwany.

Wykorzystując techniki takie jak eliminacja Gaussa, optymalizacja nad strukturami macierzowymi czy redukcja stopnia wielomianów, atakujący może skonstruować algorytm, który w efekcie redukuje złożoność problemu. Zastosowanie nowoczesnych narzędzi komputerowych, jak również znajomość specyficznych parametrów użytych w implementacji Rainbow, znacząco zwiększa skuteczność tego podejścia, czyniąc MinRank praktycznym narzędziem w ocenie bezpieczeństwa kryptograficznego systemów opartych na MQ.

W praktyce wyniki tych analiz są wykorzystywane przez projektantów schematów kryptograficznych do:

1. **Dostosowania parametrów** – na przykład zwiększenia liczby równań wielomianowych w systemie lub powiększenia przestrzeni kluczy w celu zwiększenia odporności na ataki MinRank.

2. **Projektowania nowych wariantów algorytmów** – identyfikacja słabości może prowadzić do tworzenia zmodyfikowanych wersji schematu podpisu, które są odporniejsze na tego typu ataki.

3. **Oceny realnej odporności na obliczenia kwantowe** – badania takie pomagają w ustaleniu, czy schematy takie jak Rainbow spełniają wymagania bezpieczeństwa na poziomie post-kwantowym i czy mogą być rekomendowane jako standardy kryptograficzne.

Te praktyczne analizy są szczególnie istotne w kontekście trwającego konkursu NIST na standardy kryptografii post-kwantowej, w którym schemat Rainbow był jednym z finalistów, zanim został wyeliminowany właśnie z powodu problemów z bezpieczeństwem związanych z atakami, w tym MinRank.

4 Specyfika ataku MinRank na schemat Rainbow

Atak MinRank na schemat podpisu Rainbow opiera się na sprowadzeniu problemu łamania klucza prywatnego do problemu minimalnego rzędu macierzy. Specyfika tego ataku, w tym warunki jego przeprowadzenia i ograniczenia, przedstawia się następująco:

4.1 Kiedy atak może zostać przeprowadzony?

1. **Dostęp do klucza publicznego**: Aby przeprowadzić atak MinRank na schemat Rainbow, atakujący musi mieć dostęp do klucza publicznego. Klucz publiczny w tym kryptosystemie zawiera układ równań wielomianowych, które są wynikiem "mieszania" równań ukrytych (wewnętrznych) poprzez zastosowanie macierzy liniowych oraz zamaskowanie ich warstwową strukturą. Analiza tych równań umożliwia atakującemu sprowadzenie problemu kryptograficznego do problemu minimalnej rangi macierzy. Bez dostępu do klucza publicznego analiza schematu i podjęcie prób ataku nie są możliwe.

2. **Parametry systemu**: Skuteczność ataku MinRank w dużym stopniu zależy od parametrów używanych w konstrukcji systemu Rainbow. Gdy liczba równań, liczba zmiennych lub liczba warstw w schemacie Rainbow są źle dobrane, system staje się bardziej podatny na atak. Na przykład, jeśli liczba zmiennych jest zbyt mała, układ równań może stać się prostszy do analizy, co prowadzi do obniżenia bezpieczeństwa systemu. Odpowiednie dobranie parametrów jest kluczowe, aby utrudnić redukcję problemu do struktury macierzy minimalnej rangi.

3. **Wystarczająca moc obliczeniowa**: Przeprowadzenie ataku MinRank wymaga dużych zasobów obliczeniowych, ponieważ wiąże się z intensywnymi obliczeniami algebraicznymi. Atakujący musi być w stanie rozwiązywać duże układy równań wielomianowych oraz redukować macierze do minimalnego rzędu. W przypadku ataków kwantowych dodatkowym wymogiem jest dostęp do komputerów kwantowych o odpowiedniej liczbie kubitów, co umożliwia wykorzystanie algorytmów takich jak algorytm Grovera do przyspieszenia obliczeń.

4. **Znalezienie odpowiedniego modelu**: Kluczowym elementem ataku MinRank jest umiejętność przekształcenia problemu kryptograficznego w zadanie algebraiczne polegające na znalezieniu macierzy minimalnej rangi. Proces ten wymaga głębokiej znajomości technik redukcji algebraicznych, takich jak metoda Kipnisa-Shamira, a także zaawansowanych algorytmów stosowanych w kryptografii opartej na wielomianach. Umiejętność modelowania układu równań w odpowiednich przestrzeniach algebraicznych jest kluczowa dla skuteczności ataku.

4.2 Ograniczenia ataku MinRank

1. **Wysoka złożoność obliczeniowa**: Atak MinRank, choć teoretycznie możliwy, cechuje się wysoką złożonością obliczeniową, która szybko rośnie wraz ze wzrostem rozmiaru parametrów systemu Rainbow. Dobrze skonstruowane parametry schematu, takie jak duża liczba zmiennych i odpowiednia struktura warstw, znacznie zwiększają czas i zasoby potrzebne do przeprowadzenia ataku. Problem MinRank należy do klasy problemów NP-trudnych, co oznacza, że znalezienie jego rozwiązania wymaga dużej ilości czasu przy użyciu klasycznych metod.

2. **Skalowalność**: W schematach Rainbow, w których liczba zmiennych i równań jest znaczna, atak MinRank wymaga operowania na ogromnych macierzach. Takie zadania algebraiczne wymagają dużych ilości pamięci operacyjnej oraz czasu obliczeniowego, co ogranicza praktyczną wykonalność ataku w tradycyjnych środowiskach obliczeniowych. W rezultacie skalowanie parametrów kryptosystemu, takie jak zwiększanie liczby zmiennych czy dodawanie kolejnych warstw, znacząco utrudnia przeprowadzenie skutecznego ataku.

3. **Złożoność równań**: Kryptosystem Rainbow, w swoich bardziej zaawansowanych wersjach, opiera się na układach równań wielomianowych o wysokich stopniach i dużej liczbie zmiennych. Taka konstrukcja znacząco zwiększa złożoność redukcji algebraicznych, co skutecznie utrudnia atakującemu przeprowadzenie redukcji do problemu minimalnej rangi macierzy. Warianty systemu z wyższymi stopniami wielomianów mogą również wymagać zastosowania bardziej wyrafinowanych algorytmów, które są trudniejsze do zaimplementowania w praktyce.

4. **Odporność na obliczenia kwantowe**: Choć atak kwantowy, w szczególności wariant MinRank oparty na algorytmie Grovera, teoretycznie przyspiesza pewne aspekty analizy problemu, jego skuteczność wciąż jest ograniczona przez strukturę systemu Rainbow. Przy odpowiednio dużej liczbie zmiennych i warstw, wymagania dotyczące liczby kubitów i mocy obliczeniowej komputerów kwantowych stają się tak duże, że atak staje się praktycznie niemożliwy. Dzięki temu dobrze skonstruowane schematy Rainbow pozostają odporne nawet na potencjalne zagrożenia związane z komputerami kwantowymi.

5 Pseudokod i implementacja ataku

5.1 Pseudokod

```
FUNKCJA gf2_rank(macierz):
    Skopiuj macierz, aby nie modyfikować oryginału
    rows     liczba wierszy w macierzy
    cols     liczba kolumn w macierzy
    rank_val     0
    pivot_col     0

    DLA r OD 0 DO rows-1:
        JEŚLI pivot_col >= cols:
            PRZERWIJ

        Szukaj wiersza z jedynką w pivot_col (pivot_row)
        JEŚLI nie znaleziono (pivot_row == rows):
            pivot_col     pivot_col + 1
            KONTYNUUJ

        Zamień wiersz r z pivot_row
        DLA każdego innego wiersza rr:
            JEŚLI mat[rr][pivot_col] == 1:
                Dodaj wiersz r do rr (modulo 2)

        rank_val     rank_val + 1
        pivot_col     pivot_col + 1

    ZWRÓĆ rank_val

FUNKCJA add_matrices_gf2(A, B):
    rows     liczba wierszy w A
    cols     liczba kolumn w A
    C     pusta macierz
    DLA każdego wiersza r:
        row     pusty wiersz
        DLA każdej kolumny c:
            row[c]     A[r][c] XOR B[r][c]
        Dodaj row do C
    ZWRÓĆ C

FUNKCJA scalar_mult_matrix_gf2(scalar, A):
    JEŚLI scalar == 0:
        ZWRÓĆ macierz zerowa o wymiarach A
    W PRZECIWNYM RAZIE:
        ZWRÓĆ kopię macierzy A

FUNKCJA linear_combination_gf2(macierze, współczynniki):
    Utwórz macierz wynikową jako macierz zerową
    DLA każdej macierzy M i współczynnika a:
        Mnoż macierz M przez a
        Dodaj wynik do macierzy wynikowej
    ZWRÓĆ macierz wynikową

FUNKCJA generate_random_matrix_gf2(n, m):
    Utwórz losową macierz n x m, gdzie każdy element to 0 lub 1
    ZWRÓĆ macierz

FUNKCJA brute_force_min_rank(macierze, max_rank):
    k     liczba macierzy
```

```
    total_combinations     2^k

    DLA combo OD 1 DO total_combinations-1:
        Utwórz wektor współczynników na podstawie combo
        Oblicz kombinację liniową macierzy
        Oblicz rangę wynikowej macierzy
        JEŚLI ranga <= max_rank:
            ZWRÓĆ współczynniki i wynikową macierz

    ZWRÓĆ NULL

FUNKCJA example_usage():
    k, n, m     liczby określające liczbę macierzy i ich wymiary
    macierze     lista losowych macierzy n x m
    Wyświetl macierze
    Spróbuj znaleźć kombinację liniową z rangą <= 3
    JEŚLI znaleziono:
        Wyświetl współczynniki i wynikową macierz
    W PRZECIWNYM RAZIE:
        Wyświetl komunikat o braku rozwiązania

GŁÓWNY PROGRAM:
    example_usage()
```

5.2 Dokumentacja kodu

5.2.1 Opis funkcjonalności kodu

Główne funkcjonalności kodu obejmują:

- Obliczanie rzędu macierzy nad ciałem GF(2) z użyciem algorytmu Gaussa-Jordana (`gf2_rank`).
- Tworzenie losowych macierzy nad GF(2) (`generate_random_matrix_gf2`).
- Dodawanie i mnożenie macierzy przez skalary nad GF(2) (`add_matrices_gf2`, `scalar_mult_matrix_gf2`).
- Obliczanie kombinacji liniowych macierzy nad GF(2) (`linear_combination_gf2`).
- Przeprowadzanie ataku typu MinRank przy użyciu metody brute force (`brute_force_min_rank`).

5.2.2 Parametryzacja kodu

Parametry kodu mogą być zmieniane w funkcji `example_usage`, która ilustruje sposób użycia głównych funkcji:

- `k` - liczba macierzy wejściowych.
- `n`, `m` - liczba wierszy i kolumn macierzy.
- `r` - maksymalny rząd, dla którego kod szuka kombinacji liniowych macierzy (`max_rank`).

Zmiana tych parametrów pozwala użytkownikowi kontrolować wielkość danych wejściowych oraz granicę poszukiwań rzędu w ataku MinRank.

5.2.3 Wynik działania kodu

Wynikiem działania kodu jest:

- Lista współczynników kombinacji liniowej, które prowadzą do macierzy o rzędzie mniejszym bądź równym `max_rank`.
- Wynikowa macierz będąca kombinacją liniową z obliczonym rzędem.

5.2.4 Wyniki

- Jeśli atak się powiedzie, zostaną wypisane współczynniki kombinacji liniowej oraz wynikowa macierz.
- Jeśli atak się nie powiedzie, program poinformuje, że nie znaleziono żadnej kombinacji spełniającej warunek rzędu.

5.3 Prezentacja przykładu działania

Uruchomiliśmy kod dla następujących parametrów:

- $k = 3$
- $n = 4$
- $m = 4$
- $r = 2$

Oznacza to, że wygenerowane zostaną 3 macierze o wymiarach 4x4, a program docelowo będzie chciał zredukować rząd wynikowej macierzy do $r <= 2$. W tej sytuacji możliwe są 2 wyniki zależne od tego, czy programowi uda się znaleźć macierz wynikową metodą brute-force, co przedstawiają rysunki 1 i 2:

```
Wygenerowane macierze (nad GF(2)):
M1:
[0, 0, 0, 1]
[1, 1, 1, 0]
[1, 0, 1, 0]
[1, 0, 1, 1]

M2:
[0, 0, 1, 0]
[1, 1, 0, 1]
[1, 0, 0, 1]
[1, 0, 0, 1]

M3:
[1, 0, 0, 1]
[0, 0, 0, 1]
[0, 0, 1, 0]
[0, 0, 1, 1]

Znaleziono współczynniki dające rząd <= 2
Współczynniki: [1, 1, 0]

Kombinacja liniowa (macierz wynikowa):
[0, 0, 1, 1]
[0, 0, 1, 1]
[0, 0, 1, 1]
[0, 0, 1, 0]
Rząd = 2

Process finished with exit code 0
```

Rys. 1: Wynik programu w przypadku znalezienia macierzy wynikowej o zadanym rzędzie

```
Wygenerowane macierze (nad GF(2)):
M1:
[1, 1, 1, 0]
[0, 0, 1, 0]
[0, 1, 0, 1]
[0, 1, 0, 1]

M2:
[1, 1, 0, 1]
[0, 1, 1, 0]
[0, 0, 0, 1]
[1, 0, 0, 1]

M3:
[1, 0, 0, 1]
[1, 1, 1, 0]
[0, 1, 1, 1]
[1, 0, 1, 1]

Nie znaleziono żadnej kombinacji o rzędzie <= 2

Process finished with exit code 0
```

Rys. 2: Wynik programu w przypadku braku znalezienia macierzy wynikowej o zadanym rzędzie

5.4 Implementacja ataku w Pythonie

```python
import random


def gf2_rank(matrix):
    """
    Funkcja oblicza rząd macierzy (nad GF(2))
    przy wykorzystaniu redukcji Gaussa-Jordana w GF(2).
    """
    # Kopiujemy macierz, aby nie modyfikować oryginału
    mat = [row[:] for row in matrix]
    rows = len(mat)
    cols = len(mat[0]) if rows > 0 else 0

    rank_val = 0
    pivot_col = 0

    for r in range(rows):
        if pivot_col >= cols:
            break

        # Szukamy wiersza z jedynką w pivot_col
        pivot_row = r
        while pivot_row < rows and mat[pivot_row][pivot_col] == 0:
            pivot_row += 1

        # Jeśli nie znaleziono wiersza z jedynką w tej kolumnie,
        # przechodzimy do następnej kolumny
        if pivot_row == rows:
            pivot_col += 1
            continue

        # Zamiana wiersza obecnego (r) z pivot_row
        mat[r], mat[pivot_row] = mat[pivot_row], mat[r]

        # Teraz mat[r][pivot_col] to pivot (1). Redukujemy pozostałe wiersze
        for rr in range(rows):
            if rr != r and mat[rr][pivot_col] == 1:
                # Dodajemy (modulo 2) wiersz r do rr, aby wyzerować pivot_col w rr
                for cc in range(cols):
                    mat[rr][cc] ^= mat[r][cc]

        rank_val += 1
        pivot_col += 1

    return rank_val


def add_matrices_gf2(A, B):
    """
    Dodaje dwie macierze A i B (elementy w GF(2)),
    zwraca nową macierz C = A + B (xor na każdym elemencie).
    Zakładamy, że A i B mają takie same wymiary.
    """
    rows = len(A)
    cols = len(A[0])
    C = []
    for r in range(rows):
        row = [(A[r][c] ^ B[r][c]) for c in range(cols)]
        C.append(row)
    return C


def scalar_mult_matrix_gf2(scalar, A):
    """
    Mnoży macierz A przez skalar w GF(2).
    Skoro scalar jest 0 lub 1, to:
    - 0 * A = macierz zerowa,
    - 1 * A = A.
```

```
    """
    if scalar == 0:
        # Zwracamy macierz zerową o tych samych wymiarach
        rows = len(A)
        cols = len(A[0])
        return [[0] * cols for _ in range(rows)]
    else:
        # Zwracamy kopię A (bo 1 * A = A w GF(2))
        return [row[:] for row in A]


def linear_combination_gf2(matrices, coeffs):
    """
    Oblicza liniową kombinację macierzy (over GF(2))
    z wykorzystaniem współczynników w coeffs (również w GF(2)).

    matrices: lista macierzy [M1, M2, ..., Mk]
    coeffs: współczynniki [a1, a2, ..., ak] (0 lub 1)
    """
    # Zakładamy, że matrices i coeffs mają zgodne wymiary
    rows = len(matrices[0])
    cols = len(matrices[0][0])

    # Inicjujemy macierz wynikową jako macierz zerową
    result = [[0] * cols for _ in range(rows)]

    for M, a in zip(matrices, coeffs):
        # Dodajemy a*M do result
        mult = scalar_mult_matrix_gf2(a, M)
        result = add_matrices_gf2(result, mult)
    return result


def generate_random_matrix_gf2(n, m):
    """
    Generuje losową macierz n x m nad GF(2).
    """
    return [[random.randint(0, 1) for _ in range(m)] for _ in range(n)]


def brute_force_min_rank(matrices, max_rank=1):
    """
    Naiwny atak typu MinRank (brute force) w GF(2),
    z pominięciem trywialnego rozwiązania (wszystkie współczynniki == 0).

    matrices: lista macierzy (każda o wymiarach n x m)
    max_rank: szukana maksymalna ranga (np. 1, 2, ...)
    """
    k = len(matrices)
    if k == 0:
        return None

    total_combinations = 1 << k  # 2^k

    # Rozpoczynamy od 1, żeby pominąć combo == 0 (czyli [0,0,...,0])
    for combo in range(1, total_combinations):
        # Budujemy wektor współczynników w GF(2)
        coeffs = [(combo >> i) & 1 for i in range(k)]

        # Obliczamy kombinację liniową
        comb_matrix = linear_combination_gf2(matrices, coeffs)

        # Sprawdzamy rangę
        rank_c = gf2_rank(comb_matrix)

        if rank_c <= max_rank:
            return coeffs, comb_matrix

    return None
```

```
139
140 def example_usage():
141     # Przykład działania: mamy np. 3 macierze 4x4
142     k = 5
143     n = 6
144     m = 6
145     r = 3
146     matrices = [generate_random_matrix_gf2(n, m) for _ in range(k)]
147
148     # Wyświetlamy wygenerowane macierze
149     print("Wygenerowane macierze (nad GF(2)):")
150     for i, M in enumerate(matrices):
151         print(f"M{i + 1}:")
152         for row in M:
153             print(row)
154         print()
155
156     # Próbujemy znaleźć kombinację liniową o rzędzie <= r
157     result = brute_force_min_rank(matrices, max_rank=r)
158     if result is not None:
159         coeffs, comb_matrix = result
160         print("Znaleziono współczynniki dające rząd <= "+str(r))
161         print("Współczynniki:", coeffs,"\n")
162         print("Kombinacja liniowa (macierz wynikowa):")
163         for row in comb_matrix:
164             print(row)
165         print(f"Rząd = {gf2_rank(comb_matrix)}")
166     else:
167         print("Nie znaleziono żadnej kombinacji o rzędzie <= "+str(r))
168
169
170 if __name__ == "__main__":
171     example_usage()
```

Literatura

[1] Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, Jean-Pierre Tillich. Revisiting algebraic attacks on minrank and on the rank decoding problem, Jun 2023. https://arxiv.org/abs/2208.05471.

[2] Seong-Min Cho, Seung-Hyun Seo. Q-rminrank attack: The first quantum approach for key recovery attacks on rainbow, 2022.

[3] cordis.europa.eu CORDIS. Nowatorskie rozwiązania kryptograficzne eliminują zagrożenia związane z komputerami kwantowymi, Oct 2022. https://cordis.europa.eu/article/id/442409-novel-cryptography-eliminates-the-looming-threat-of-quantum-computers/pl.

[4] Antonio J. Di Scala, Carlo Sanna. Smaller public keys for minrank-based schemes, Aug 2023. https://arxiv.org/abs/2302.12447.

[5] Adam Komorowski. Krys - prezentacje wykładowe, 2025.

[6] Netalit. What is crypto ransomware?, Feb 2024. https://www.checkpoint.com/cyber-hub/ransomware/what-is-crypto-ransomware/.