Atak Rho Pollarda

Paweł Murdzek, Jakub Włodarz, Patryk Średniawa, Piotr Szewczyk,
Bui Giang Nam, Piotr Gutowski, Zuzanna Popławska
Instytut Telekomunikacji, Politechnika Warszawska

***Abstract*—Niniejszy artykuł przedstawia analizę ataku Rho Pollarda, generycznego algorytmu służącego do rozwiązywania problemu logarytmu dyskretnego (DLP). Praca koncentruje się na zastosowaniu tej metody w kontekście kryptografii krzywych eliptycznych (ECC), w szczególności dla krzywych zdefiniowanych nad ciałami binarnymi (F_{2^m}).**

W pierwszej części artykułu szczegółowo omówiono niezbędne podstawy matematyczne, obejmujące zasady arytmetyki modularnej w ciałach binarnych oraz kluczowe właściwości krzywych eliptycznych. Następnie praca przechodzi do analizy źródła podatności, wyjaśniając, w jaki sposób cykliczność podgrupy punktów na krzywej eliptycznej jest wykorzystywana przez algorytmy atakujące.

Główna część artykułu poświęcona jest problemowi logarytmu dyskretnego na krzywych eliptycznych (ECDLP), który stanowi fundament bezpieczeństwa systemów ECC. Przedstawiono dokładną specyfikację ataku Rho Pollarda, omawiając jego zasady działania, złożoność obliczeniową oraz praktyczne ograniczenia.

W końcowej sekcji zaprezentowano praktyczną implementację ataku. Obejmuje ona zarówno ogólny schemat algorytmu w formie pseudokodu, jak i jego konkretną realizację w języku C++. Pracę zamyka prezentacja i dyskusja wyników uzyskanych podczas testowania zaimplementowanego algorytmu.

I. Teoria matematyczna

A. Opis arytmetyki modularnej dla ciał binarnych

Arytmetyka modularna odgrywa kluczową rolę w informatyce i kryptografii. Zastosowanie ma w ciałach skończonych. Artykuł będzie skupiał się na szczególnym ich rodzaju - ciałach binarnych (oznaczane jako $\mathbf{GF}(2^n)$ lub $\mathbf{F}_{2^n}$). Są one ciekawe ze względu na łatwość implementacji w rozwiązaniach sprzętowych.

Dodawanie jest zgodne z operacją XOR:

$$0+0=0 \pmod 2 \qquad 0+1=1 \pmod 2$$
$$1+0=1 \pmod 2 \qquad 1+1=0 \pmod 2$$

Warto wspomnieć, że dodawanie jest odwracalne ($a+a=0$).

Mnożenie jest zgodne z operacją AND:

$$0\cdot 0=0 \pmod 2 \qquad 0\cdot 1=0 \pmod 2$$
$$1\cdot 0=0 \pmod 2 \qquad 1\cdot 1=1 \pmod 2$$

Elementem odwrotnym do 1 jest 1.

1) Rozszerzone ciała binarne $\mathbf{GF}(2^n)$: W kryptografii, dla potrzeb obliczeniowych, często operuje się na większych zbiorach elementów. Elementy te są reprezentowane jako wielomiany o współczynnikach z $\mathbf{GF}(2)$ i stopniu mniejszym niż n. Zbiór tych wielomianów stanowi 2^n elementów ciała $\mathbf{GF}(2^n)$. Działania arytmetyczne wykonywane są na wielomianach, a następnie wynik redukuje się modulo pewien nierozkładalny wielomian stopnia n. Dodawanie jest wykonywane z użyciem XOR bit po bicie na binarnych reprezentacjach elementów.

Mnożenie za to wykonuje się w dwóch krokach. Zaczyna się od standardowego mnożenia $A(x)\cdot B(x)$. Następnie wynik mnożenia jest dzielony przez ustalony nieredukowalny wielomian $P(x)$ stopnia n o współczynnikach z $\mathbf{GF}(2^n)$. Reszta z tego dzielenia jest wynikiem mnożenia modularnego:

$$A(x)\cdot B(x)\equiv R(x) \pmod{P(x)}$$

Warto wspomnieć, że wielomian $P(x)$ pełni rolę modułu i zapewnia, że wynik pozostaje w $\mathbf{GF}(2^n)$. Dla każdego niezerowego elementu $A(x)\in\mathbf{GF}(2^n)$ istnieje element odwrotny $A^{-1}(x)$, taki że:

$$A(x)\cdot A^{-1}(x)\equiv 1 \pmod{P(x)}$$

W celu znalezienia tej odwrotności korzysta się z rozszerzonego algorytmu Euklidesa dla wielomianów.

Efektywność obliczeniowa arytmetyki modularnej wynika z tego, że operacje można realizować bezpośrednio na bitach procesora. Z tego też powodu ciała $\mathbf{GF}(2^n)$ są szeroko stosowane w implementacjach sprzętowych oraz kryptografii.

Algorytmy implementujące powyższe operacje zostaną opisane w rozdziale V.

B. Właściwości krzywych eliptycznych

W ujęciu algebraicznym, krzywa eliptyczna E zdefiniowana nad ciałem K jest gładką (nieosobliwą) sześcienną krzywą algebraiczną w płaszczyźnie rzutowej $P^2(K)$, która posiada co najmniej jeden punkt K-wymierny. Punkt ten zazwyczaj wyznacza się jako element neutralny grupy.

Dla celów praktycznych, o ile charakterystyka ciała K jest różna od 2 i 3 (co ma miejsce np. dla $K=\mathbb{R}$, $K=\mathbb{Q}$ lub ciał skończonych $\mathbb{F}_p$ dla $p>3$), każdą krzywą eliptyczną można sprowadzić, przez odpowiednią transformację współrzędnych, do uogólnionej postaci Weierstrassa:

$$E: y^2=x^3+Ax+B$$

dla współczynników $A,B\in K$.

Kluczowym warunkiem jest nieosobliwość krzywej, która gwarantuje, że krzywa nie posiada "ostrzy" ani samoprzecięć. Warunek ten jest równoważny nieznikaniu wyróżnika (dyskryminanty) krzywej:

$$\Delta=-16(4A^3+27B^2)\neq 0$$

Gdy $\Delta=0$, krzywa jest osobliwa i nie posiada interesującej nas struktury grupowej [8].

Zbiór punktów K-wymiernych na krzywej E definiuje się jako:

$$E(K)=\{(x,y)\in KxK \mid y^2=x^3+Ax+B\}\cup\{O\}$$

Element O jest specjalnym, wyróżnionym punktem, nazywanym punktem w nieskończoności. Jego włączenie jest niezbędne do algebraicznego domknięcia operacji grupowych [8].

Najważniejszą własnością zbioru punktów $E(K)$ jest to, że tworzy on grupę przemienną względem operacji "dodawania" punktów. Operację tę definiuje się w sposób geometryczny:

1) **Element Neutralny:** Jest nim punkt w nieskończoności O. Dla dowolnego punktu P na krzywej:
$$P + O = P$$
2) **Element Odwrotny:** Dla punktu $P = (x, y)$, jego elementem odwrotnym jest punkt $-P = (x, -y)$, czyli jego geometryczne odbicie względem osi OX.
3) **Dodawanie $P+Q$ (gdy $P \neq Q$):** Aby dodać dwa różne punkty P i Q, prowadzimy przez nie prostą. Zgodnie z twierdzeniem Bézouta, prosta ta przetnie krzywą sześcienną w dokładnie trzech punktach [8]. Niech trzecim punktem przecięcia będzie R. Sumę $P+Q$ definiujemy jako $-R$ (czyli odbicie R względem osi OX).
$$P + Q = -R$$
4) **Podwajanie $P+P$:** Zamiast siecznej przez dwa punkty, prowadzimy prostą styczną do krzywej w punkcie P. Styczna ta przetnie krzywą w jeszcze jednym punkcie R. Sumę definiujemy jako $-R$.
$$P + P = -R$$
5) **Suma $P+(-P)$:** Prosta przechodząca przez $P = (x, y)$ i $-P = (x, -y)$ jest prostą pionową. Taka prosta nie przecina krzywej w trzecim punkcie w płaszczyźnie afinicznej – "przecina" ją natomiast w punkcie w nieskończoności O. Zatem $R = O$. Zgodnie z definicją, $P + (-P) = -O$. Ponieważ O leży "na osi symetrii" (można myśleć o nim jako o punkcie na pionowej linii w nieskończoności), jego odbiciem jest on sam, czyli $-O = O$.
$$P + (-P) = O$$

To geometryczne prawo składania spełnia wszystkie aksjomaty grupy (łączność, przemienność, element neutralny, element odwrotny), co czyni z $E(K)$ jedną z fundamentalnych struktur w teorii liczb.

II. Źródło ataku - cykliczność podgrupy

Bezpieczeństwo kryptografii krzywych eliptycznych (ECC) opiera się na trudności Problemu Dyskretnego Logarytmu (DLP) w ramach skończonej podgrupy cyklicznej. Mnożenie punktu P na krzywej w ciele skończonym nieuchronnie prowadzi do powtarzania się wyników, czyli "zapętlania" (np. $5P = 0$, a następnie $6P = 1P$). Te powtarzające się punkty tworzą właśnie podgrupę cykliczną, która jest określona jako "podstawa kryptografii krzywych eliptycznych i ważny aspekt kryptoanalizy" [4].

Atak Rho Pollarda jest fundamentalnym przykładem tego, jak ta właściwość jest wykorzystywana w kryptoanalizie. Jest to *atak generyczny*, co oznacza, że działa on na każdą skończoną grupę cykliczną, niezależnie od jej konstrukcji (czy są to punkty na krzywej, czy liczby w grupie multiplikatywnej) [7].

Związek między cyklicznością a atakiem można wyjaśnić następująco [4]:

1) **Skończona przestrzeń:** Podgrupa cykliczna, z definicji, ma skończony rząd (liczbę elementów), który oznaczymy jako n. Oznacza to, że istnieje tylko n unikalnych punktów, które można wygenerować przez mnożenie generatora P.
2) **Pseudolosowy spacer:** Atak Rho Pollarda polega na wykonywaniu "pseudolosowego spaceru" po punktach tej podgrupy. Rozpoczynając od pewnego punktu, generuje się sekwencję kolejnych, pozornie losowych punktów, stosując funkcję iterującą.
3) **Nieunikniona kolizja (Paradoks Urodzinowy):** Ponieważ liczba punktów w podgrupie jest skończona (n), każda sekwencja generowanych punktów musi w końcu zacząć się powtarzać. Jest to bezpośredni wynik zasady szufladkowej Dirichleta. Co więcej, zgodnie z tzw. **paradoksem urodzinowym**, statystycznie oczekuje się znalezienia kolizji (czyli punktu, do którego można dotrzeć na dwa różne sposoby) znacznie szybciej, niż wynosi rząd całej grupy. Zamiast n kroków, kolizji oczekuje się już po około $\mathcal{O}(\sqrt{n})$ krokach [2]. Sam paradoks urodzinowy ilustruje, że w zbiorze n elementów (np. dni w roku), prawdopodobieństwo znalezienia kolizji (dwóch osób z tą samą datą urodzin) przekracza 50% już przy wyborze około $\sqrt{n}$ elementów (w przypadku 365 dni, wystarczą tylko 23 osoby). W ataku Rho Pollarda "dniami roku" są punkty w podgrupie, a "osobami" są kroki algorytmu [3]
4) **Kształt "Rho" (ρ):** Nazwa algorytmu bierze się stąd, że sekwencja generowanych punktów przypomina grecką literę Rho (ρ) — składa się z "ogona" (punktów odwiedzonych przed pierwszą kolizją) i "pętli" (cyklu, w który wpada sekwencja).

Podsumowując, "cykliczność podgrupy" jest fundamentalną właściwością, która gwarantuje, że atak Rho Pollarda w ogóle zadziała. To właśnie skończoność i cykliczna natura grupy zapewniają, że "losowy spacer" musi ostatecznie znaleźć cykl, a paradoks urodzinowy dyktuje, że znalezienie tej kolizji jest obliczeniowo wykonalne (w złożoności pierwiastkowej) dla grup o niewystarczająco dużym rzędzie.

III. Problem logarytmu dyskretnego

Klasyczny problem logarytmu dyskretnego opiera się na wyznaczeniu wartości *x*, określanej również jako logarytm dyskretny *h* o podstawie *g* i spełniającej równość:

$$a^x \equiv h \pmod p$$

gdzie a - podstawa, x - logarytm dyskretny, h - reszta z dzielenia, p - modulo.

W zastosowaniach kryptograficznych powyższy problem znajduje zastosowanie ze względu na zachodzącą asymetrię obliczeniową. Potęgowanie modularne $g^x \pmod p$ jest

znacznie łatwiejsze od poszukiwania wartości x. Dzięki tej właściwości możliwe jest skuteczne zabezpieczanie m.in. kluczy publicznych.

Dla rozważanego ataku konieczne jest przeniesienie niniejszego problemu na krzywe eliptyczne, co zmienia cały proces. Głównym elementem algorytmu jest mnożenie danego punktu przez skalar określone odpowiednim równaniem:

$$Q = l * P$$

gdzie Q - klucz publiczny, l - klucz prywatny, P - punkt bazowy.

W tym przypadku otrzymanie klucza publicznego jest stosunkowo proste i polega na mnożeniu klucza prywatnego oraz punktu. Próba określenia klucza publicznego w ten sam sposób jest jednak wyjątkowo kosztowna obliczeniowo.

Powyższe rozwiązanie ma na celu uniknięcie wykorzystywania dużych liczb poprzez wykonywanie wszelkich operacji na punktach krzywej eliptycznej. Taka zmiana pozwala na zaoferowanie podobnego poziomu bezpieczeństwa przy użyciu krótszego klucza.

Dotychczas metody z rodziny algorytmów kryptografii krzywych eliptycznych uznawane były za niemożliwe do złamania przez obecne komputery. Atak Rho Pollarda określany jest jako pierwsze rzeczywiste zagrożenie dla tej dziedziny kryptografii [6].

IV. Specyfikacja ataku i jego ograniczenia

Bezpieczeństwo kryptografii opartej na krzywych eliptycznych wynika z trudności rozwiązania problemu logarytmu dyskretnego w podgrupie cyklicznej krzywej. Atakujący dąży do znalezienia klucza prywatnego l, znając punkt P oraz odpowiadający mu klucz publiczny $Q = lP$. Bezpieczeństwo wynika z ograniczenia efektywności wyznaczania l.

1) Specyfikacja ataku: Atak Rho Pollarda wykorzystuje właściwości skończonych grup cyklicznych w taki sposób, który umożliwia radykalne skrócenie czasu potrzebnego na znalezienie rozwiązania. [7] Kluczowym elementem ataku jest funkcja iteracyjna (lub funkcja pseudolosowa), umożliwiająca przemierzanie punktów krzywej po elementach grupy $E(K)$ w sposób deterministyczny.

Dla każdego punktu X w sekwencji, następny punkt X' jest obliczany jako:

$$X' = f(X)$$

Funkcja $f(X)$ jest zaprojektowana tak, by każdy punkt X w sekwencji był reprezentowany jako pewna kombinacja liniowa generatora P i klucza publicznego Q, gdzie współczynniki a i b są aktualizowane w każdym kroku:

$$X = a \cdot P + b \cdot Q$$

Atak nie szuka bezpośrednio klucza l. W pewnym momencie dochodzi do **kolizji** w sekwencji, czyli dwa różne ciągi iteracyjne i i j prowadzą do tego samego punktu $X_i = X_j$.

Ponieważ $Q = l \cdot P$, kolizja oznacza, że:

$$a_i \cdot P + b_i \cdot Q = a_j \cdot P + b_j \cdot Q$$

Podstawiając $Q = l \cdot P$ i grupując względem P, otrzymujemy:

$$(a_i + b_i \cdot l) \cdot P = (a_j + b_j \cdot l) \cdot P$$

Ponieważ rzędem grupy jest n, równość punktów oznacza równość skalarów modulo n:

$$a_i + b_i \cdot l \equiv a_j + b_j \cdot l \pmod{n}$$

Umożliwia to obliczenie wartości l poprzez analizę różnicy między sekwencjami. Przekształcając powyższe równanie, można **wyznaczyć klucz prywatny** l:

$$l \cdot (b_i - b_j) \equiv a_j - a_i \pmod{n}$$

Wartość l wylicza się za pomocą *Extended Euclidean Algorithm* w celu znalezienia odwrotności $(b_i - b_j)$ modulo n.

Ważne jest prawidłowe wykrycie kolizji. Do tego celu wykorzystuje się algorytm **Floyd's Cycle-Finding**, nazywany również metodą „żółwia i zająca". Algorytm ten polega na jednoczesnym śledzeniu dwóch kroków punktów — jeden porusza się o krok (X_{i+1}), podczas gdy drugi o dwa kroki (X_{i+2}). W momencie, gdy oba punkty się spotkają ($X_i = X_{2i}$), oznacza to, że znaleziono kolizję.

Istnieją różne warianty ataku:

- **Pollard Rho z punktami wyróżnionymi** — metoda usprawniająca detekcję kolizji,
- **Pohlig–Hellman** — wykorzystuje faktoryzację rzędu podgrupy,
- **Baby-step Giant-step** — charakteryzuje się podobną złożonością, lecz większym zużyciem pamięci.

2) Ograniczenia ataku: Ograniczenia nie wynikają z braku efektywności, lecz z konieczności minimalizowania ryzyka poprzez właściwy dobór parametrów ECC. Największe ograniczenia obejmują:

- Złożoność czasowa $O(\sqrt{n})$, gdzie n jest rzędem podgrupy cyklicznej. [1] Jeśli rząd podgrupy jest dużą liczbą pierwszą, żadna z metod nie oferuje praktycznego ataku. Jest to górna granica efektywności generycznych ataków DLP.
- Krzywe o niewłaściwych parametrach (np. anomalne lub o małym współczynniku cofaktora) mogą być podatne na specjalistyczne ataki redukujące ECDLP do prostszych problemów.
- Większość skutecznych ataków wymaga jednocześnie słabości matematycznych oraz błędów implementacyjnych.

V. Pseudokod ataku Rho Pollard

Algorytm Rho Pollarda został zaimplementowany z wykorzystaniem języka programowania C++. Realizacja ataku w postaci programu uruchamianego na procesorze nie stanowiła rozwiązania optymalnego pod względem wydajności, jednak zapewniła dobre wprowadzenie do zagadnień związanych z algorytmami kryptoanalizy. Dodatkowo oprogramowanie w C++ można przyśpieszyć akcelaratorami sprzętowymi np. za pomocy platformy OpenCL.

A. Algorytmy wykorzystane w implementacji obliczeń

Implementacja projektu zakłada 32-bitową architekturę, czyli każdy wektor reprezentujący wielomiany jest uporządkowany w 32-bitowe bloki. Pole F_{2^m} posiadające elementy rzędu maksymalnie $m-1$ będzie przedstawione jako tablica 32-bitowych wektorów binarnych. Wielkość tablicy oznaczona jako t wynosi $\lceil m/32 \rceil$. Wykorzystane algorytmy zostały zaczerpane z podręcznika Guide to Elliptic Curve Cryptography. [4]

1) Dodawanie: Dodawanie jest trywialne, należy dodać wszystkie wektory do siebie za pomocy operacji XOR. Algorytm 1 przedstawia kroki potrzebne przy dodawaniu wielomianów.

Algorithm 1 Algorytm dodawania wielomianów

Input: wielomiany a(z) i b(z) o rzędach m-1
Output: c(z) = a(z) + b(z)

for i from 0 to t-1 **do**
  $C[i] \leftarrow A[i] \oplus B[i]$
end for
Return(C)

Algorithm 2 Algorytm mnożenia wielomianów

Input: wielomiany a(z) i b(z) o rzędach m-1
Output: c(z) = a(z) · b(z)

$C \leftarrow 0$
for k from 0 to 31 **do**
  for j from 0 to t-1 **do**
    if a[j][k] == 1 **then**
      C[j] ← C[j] ⊕ B[j]
    end if
  end for
  if k $\neq$ 31 **then**
    B ← B · z
  end if
end for

2) Mnożenie: W algorytmie 2 mnożenie wykonywane na wielomianie b przez wielomian z oznacza przesunięcie bitowe o jeden bit w prawo.

Algorithm 3 Algorytm podnoszenia wielomianów do kwadratu dla architektury 32-bitowej

Input: wielomian a(z) rzędu m-1
Output: c(z) = a(z)^2

Dla każdego bajtu $d = (d_7, ..., d_2, d_1)$ oblicz 16 bitową wartość $T(d) = (0, d_7, ..., 0, d_1, 0, d_0)$
for i from 0 to t-1 **do**
  A[i] = (u_3, u_2, u_1, u_0) gdzie u_j jest bajtem
  C[2i] ← (T(u_1), T(u_0))
  C[2i+1] ← (T(u_3), T(u_4))
end for

3) Podniesienie do potęgi 2: Algorytm podniesienia do potęgi drugiej wielomianu (przedstawiony na algorytmie 3), polega na umieszczenia zer między bitami każdego wektora tablicy reprezentującej wielomian.

4) Redukcja: Algorytm redukcji wielomianu otrzymanego przez mnożenie lub potęgowanie znajduję w opisie algorytmu 4.

Algorithm 4 Algorytm redukcji wielomianu

Input: wielomian c(z) o rzędzie 2m-2, wielomian redukcyjny f(z) = z^m + r(z)
Output: c(z) mod f(z)

for k od 0 do 31 **do**
  Oblicz $u_k(z) = z^k r(z)$
end for
for i od 2m-2 do m **do**
  if c_i = 1 **then**
    j = $\lfloor (i-m)/32 \rfloor$, k=(i-m) - 32*j
    C[j] ← C[j] ⊕ u_k(z)
  end if
end for
Zwróć C

Algorithm 5 Algorytm inwersji wielomianu

Input: wielomian a(z)
Output: a^{-1} mod f

u ← a, v ← f
$g_1 \leftarrow 1$, $g_2 \leftarrow 0$
while u $\neq$ 1 **do**
  j ← rząd(u) - rząd(v)
  if j < 0 **then**
    u ↔ v, $g_1 \leftrightarrow g_2$, j ← -j
    u ← u + z^{j}gv
    $g_1 \leftarrow g_1 + z^j g_2$
  end if
end while
Zwróć(g_1)

5) Inwersja: Inwersja z wykorzystaniem rozszerzeniu algorytmu Euklidesa jest przedstawiona w algorytmie 5.

VI. Algorytm Rho Pollarda

Algorytm 6 opisuje pseudokod algorytmu faktoryzacji Rho Pollarda dla krzywych eliptycznych.

Algorithm 6 Algorytm Rho Pollarda

generator G i punkt P
liczba naturalna x spełniająca P = xG

Inicjalizacja:
$i \leftarrow 0$
$a_0, b_0 \leftarrow$ losowe liczby naturalne między 2 a rządem grupy n
$X_0 \leftarrow O$
while i $\leq$ n, gdzie n to rząd grupy **do**
    Oblicz a_i, b_i
    Oblicz a_{2i}, b_{2i}
    Oblicz $X_i = a_i G + b_i P$
    Oblicz $X_{2i} = a_{2i} G + b_{2i} P$
    if $X_i = X_{2i}$ **then**
        if $b_i = b_{2i}$ **then**
            Zacznij algorytm od nowa
        else
            Zwróć $x = (a_i - a_{2i})(b_{2i} - b_i)^{-1} \mod n$
        end if
    else
        i = i+1
    end if
end while

A. Funkcje generujące parametry w następnej iteracji algorytmu Rho Pollarda

Funkcje w algorytmie 6 generujące wartości a_i, b_i, a_{2i}, b_{2i} są wykonywane na podstawie współrzędnej na osi x obecnego obliczanego punktu X_i. Cała przestrzeń osi x jest podzielona na 3 grupy poprzez wykonaniu modulo 3 z współrzędnej x.

W zależności od wyniku działania modulo 3, X_i równa się:

$$X_{i+1} \begin{cases} X_i + Q & \text{dla } x \mod 3 = 0 \\ 2 \cdot X_i & \text{dla } x \mod 3 = 1 \\ X_i + P & \text{dla } x \mod 3 = 2 \end{cases} \quad (1)$$

Natomiast współczynniki a_{i+1} i b_{i+1} są generowane w następujący sposób

$$a_{i+1} \begin{cases} a & \text{dla } x \mod 3 = 0 \\ 2 \cdot a \mod f(x) & \text{dla } x \mod 3 = 1 \\ a + 1 \mod f(x) & \text{dla } x \mod 3 = 2 \end{cases} \quad (2)$$

$$b_{i+1} \begin{cases} b + 1 \mod f(x) & \text{dla } x \mod 3 = 0 \\ 2 \cdot b \mod f(x) & \text{dla } x \mod 3 = 1 \\ b & \text{dla } x \mod 3 = 2 \end{cases} \quad (3)$$

Gdzie n to rząd krzywej.

VII. Wyniki

Implementacja w C++ nie została zrealizowana ze względu na trudności z stworzeniem struktury danych w postaci tablic bitów. Maksymalna wartość dla której udao się zaprogramować działającą implementację to 32 bity.

W celu porównania stworzony został odpowiednik w języku programowania Python, gdzie nie trzeba się martwić problemami z implementacjami odpowiednich struktur danych. Jednak taki stopień swobody jest uzyskany kosztem wydajności i ograniczeniem kontroli nad zasobami obliczeniowymi wykorzystanymi podczas obliczeń.

A. Wybrana krzywa

Za pomocy narzędzia ecgen [5] wygenerowano daną domenę krzywej eliptycznej:

- stopień m = 16
- współczynnik a = 0x2905
- współczynnik b = 0x886f
- nieredukowalny wielomian: $f(x) = x^(16) + x^(5) + x^3 x + 1$
- rząd krzywej n = 65920
- generator: G = (0xba04, 0x9b3b)

Warto zwrócić uwagę na rząd krzywej, gdyż nie jest to liczba pierwsza. Nie udało się wygenerować krzywej o rządzie w postaci liczby pierwszej mimo paru godzin działania programu ecgen. Rząd krzywej będzie miał znaczenie przy wynikach.

B. Przebieg ataku

Aby przyśpieszyć działanie algorytmu detekcji kolizji ograniczono liczbę iteracji do pierwiastka z rzędu krzywej (teoretyczne skomplikowanie obliczeniowe algorytmu Rho Pollarda). Zazwyczaj program potrzebował z 3-4 wykonań Rho Pollarda by znaleźć kolizję. Warto wspomnieć że algorytm Rho Pollarda nie musi zawsze zwrócić wynik ze względu na jego losową naturę.

W wielu wypadkach mimo znalezienia kolizji wynik był niepoprawny. Wzór wyniku opisany w algorytmie 6 wymaga obliczenia inwersji z $(b_{2i} - b_{2i}) \mod n$, więc $gcd((b_{2i} - b_{2i}), n) == 1$. Dla n które nie są liczbami pierwszymi trudno uzyskać wynik, które spełnia ten warunkek. Przy krzywych z rzędami, które nie są liczbą pierwszą Rho Pollard często się zakańcza bez powodzenia.

VIII. Podsumowanie

Artykuł omawia Atak Rho Pollarda jako algorytmu rozwiązania problemu logarytmu dyskretnego dla krzywych eliptycznych nad ciałami binarnymi F_{2^m}. W rozdziale I omówione zostały podstawy matematyczne: arytmetyki w ciałach binarnych oraz właściwości krzywych eliptycznych i wynikającą z nich strukturę grupową. Z właściwości krzywych eliptycznych, wynika że źródłem skuteczności ataku jest cykliczność podgrupy i nieuniknione kolizje wynikające z paradoksu urodzinowego.

Zaimplementowana została arytmetyka na ciałach binarnych oraz algorytm Rho Pollarda w jezyku programowania Python. Nie udało się przeprowadzić ataku na wygenerowanej w programie ecgen krzywej ze względu na rzędzie krzywej, który nie jest liczbą pierwszą.

REFERENCES

[1] S. Bai and R. P. Brent. On the efficiency of pollard's rho method for discrete logarithms. *Australian National University Technical Report*, 2011.

[2] Andrea Corbellini. Elliptic curve cryptography: breaking security (pollard's rho), June 2015.

[3] Anirban DasGupta. The matching, birthday and the strong birthday problem: a contemporary review. *Journal of Statistical Planning and Inference*, 130(1):377–389, 2005. Herman Chernoff: Eightieth Birthday Felicitation Volume.

[4] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer, New York, 2004.

[5] J08nY. ecgen. GitHub repository, 2026. Accessed: 2026-01-31.

[6] Neriman Gamze Orhon and Ayham Taleb. Parallel pollard's rho attack for elliptic curve cryptography, June 2014.

[7] J. M. Pollard. Monte carlo methods for index computation (mod p). *Mathematics of Computation*, 32(143):918–924, 1978.

[8] Joseph H Silverman. *The arithmetic of elliptic curves*, volume 106. Springer, 2009.