Biuletyn WAT
Vol. LXVI, Nr 4, 2017

Realizacja koprocesora wspierającego poszukiwanie logarytmu dyskretnego na krzywych eliptycznych z wykorzystaniem częściowej wiedzy

MICHAŁ KĘDZIERSKI, MICHAŁ MISZTAL, MICHAŁ WROŃSKI

Wojskowa Akademia Techniczna, Wydział Cybernetyki, Instytut Matematyki i Kryptologii,
ul. Gen. W. Urbanowicza 2, 00-908 Warszawa,
kedzierski.michael@gmail.com, michal.misztal@wat.edu.pl, michal.wronski@wat.edu.pl

Streszczenie. W niniejszej pracy podjęto się analizy realizacji w strukturach programowalnych koprocesora wspierającego poszukiwanie logarytmu dyskretnego na krzywych eliptycznych nad ciałem $GF(p)$, gdzie p oznacza dużą liczbę pierwszą. Główna idea realizacji koprocesora polega na zastosowaniu wielu podukładów zdolnych do dodawania punktów, ale o stosunkowo niewielkiej złożoności. Przedstawiono przypadek uproszczony, zakładając, że znamy l najbardziej znaczących bitów parametru klucza k i wykorzystujemy jednowymiarową metodę Gaudry'ego-Schosta. Następnie zaprezentowano uogólnienie i analizę przypadku, gdy nieznane bity znajdują się w wielu rozłącznych przedziałach. W tym celu zaproponowano wykorzystanie wielowymiarowej metody Gaudry'ego-Schosta. Na końcu przedstawiono rozwiązanie, które zapewnia najlepszy stosunek całkowitej przepustowości do ceny urządzenia.
Słowa kluczowe: krzywe eliptyczne, logarytm dyskretny na krzywych eliptycznych (ECDLP), ataki z częściową informacją, wielowymiarowy algorytm Gaudry'ego-Schosta
DOI: 10.5604/01.3001.0010.8185

1. Wstęp

W niniejszej pracy podjęto się analizy możliwości realizacji w strukturach programowalnych FPGA koprocesora wspierającego poszukiwanie logarytmu dyskretnego na krzywych eliptycznych nad ciałem $GF(p)$, przy założeniu, że mamy częściową informację o poszukiwanej krotności.

Idea algorytmów opartych na problemie logarytmu dyskretnego na krzywych eliptycznych jest stosunkowo prosta. Mianowicie, jeżeli mamy dany generator grupy

punktów P o rzędzie $ord(P) = r$, gdzie r jest liczbą pierwszą, i chcemy obliczyć jego k-tą krotność $Q = [k]P$, możemy to zrobić bardzo szybko. Niestety, jeżeli znamy tylko punkty P oraz Q, natomiast nie znamy wartości k, to wyznaczyć ją jest (przy odpowiednio dobranym r) bardzo trudno.

Ponieważ poszukiwanie logarytmu dyskretnego na krzywych eliptycznych jest problemem trudnym obliczeniowo, przy rzędach grupy punktów długości od 192 do 521 bitów znalezienie takiego logarytmu jest praktycznie niemożliwe przy wykorzystaniu obecnie dostępnych mocy obliczeniowych.

Tym samym jedynym praktycznym sposobem znalezienia logarytmu dyskretnego na krzywej eliptycznej jest przeprowadzenie poszukiwań z wykorzystaniem pewnych informacji o poszukiwanej krotności. Informacje takie możemy zdobyć, przeprowadzając jeden z ataków typu *side channel*, na przykład atak poboru mocy czy kryptoanalizę akustyczną.

Ataki te wykorzystują pewne wady wykonanych implementacji algorytmów. Jako przykład może tutaj posłużyć implementacja obliczania krotności punktu na krzywej eliptycznej metodą binarną z rozróżnianiem operacji dodawania i podwojenia punktów na krzywej eliptycznej. Jeżeli nie zabezpieczymy się dodatkowo przed takim atakiem, istnieje wówczas duże prawdopodobieństwo zdobycia za pomocą ataku *side channel* pewnych informacji na temat krotności punktu, której poszukujemy.

Celem jest zaproponowanie rozwiązania, które można by zaimplementować na platformie wykorzystującej wiele struktur FPGA w ten sposób, aby łączna przepustowość uzyskanego rozwiązania była jak największa, przy jednocześnie jak najmniejszej cenie wykorzystywanych struktur.

Badania zostały przeprowadzone z wykorzystaniem krzywej NIST P-192, której rząd grupy punktów jest 192-bitową liczbą pierwszą.

Przetestowano zajętość i przepustowość różnych modułów przeznaczonych do obliczania krotności punktu, a następnie przeprowadzono poszukiwania struktury programowalnej, która zapewnia największą całkowitą przepustowość przy jak najmniejszym koszcie wykorzystywanego w tym celu sprzętu.

Pokazano, że dysponując budżetem 100 000 dol. (jest to kwota stosunkowo niewielka na przykład dla instytucji rządowych), można znaleźć w rozsądnym czasie (ok. 30 dni) logarytm dyskretny na krzywej NIST P-192 przy założeniu, że udało nam się poznać (np. przy wykorzystaniu ataku *side channel*) trochę ponad połowę bitów (102 ze 192 bitów) poszukiwanej krotności.

2. Teoria krzywych eliptycznych

Krzywą eliptyczną E zdefiniowaną nad ciałem $\mathbb{K}$, gdzie $char(\mathbb{K}) \neq 2,3$, nazywa się nieosobliwą krzywą zadaną w postaci normalnej (skróconej) Weierstrassa (patrz [1]), wyrażoną poniższym równaniem:

$$E : y^2 = x^3 + ax + b, \tag{1}$$

gdzie współczynniki $a, b \in \mathbb{K}$. Równanie (1) określa się również mianem skróconej postaci afinicznej równania Weierstrassa.

Często stosuje się zamiennie postaci afiniczną i rzutową krzywej eliptycznej. Nieosobliwa krzywa eliptyczna w postaci rzutowej zadana jest następującym równaniem:

$$E : Y^2Z = X^3 + aXZ^2 + bZ^3, \tag{2}$$

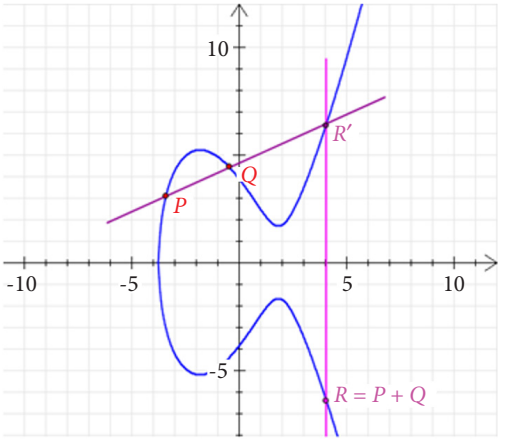
gdzie $a,\ b \in \mathbb{K}$. Nieosobliwość krzywej eliptycznej mówi nam, że jeśli:

$$F(X,Y,Z) = Y^2Z - X^3 - aXZ^2 - bZ^3, \tag{3}$$

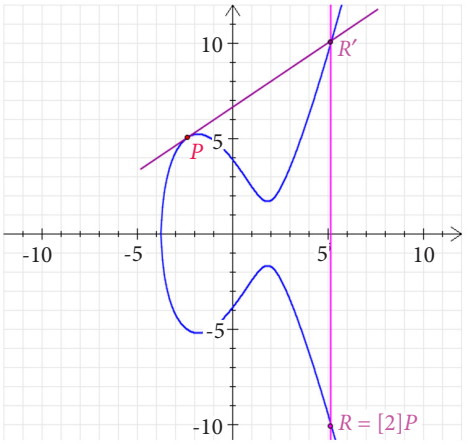
to pochodne cząstkowe $\partial F / \partial X$, $\partial F / \partial Y$ i $\partial F / \partial Z$ nie mogą być jednocześnie równe zeru w żadnym punkcie należącym do krzywej. Inaczej mówiąc, krzywa jest nieosobliwa, jeżeli nie posiada singularnych miejsc zerowych.

Każdą parę (x, y) spełniającą równanie (1) nazywamy punktem krzywej eliptycznej. Liczbę wszystkich różnych punktów na krzywej eliptycznej wraz ze specjalnym punktem, tzw. punktem w nieskończoności $(\mathcal{O})$, oznaczamy jako $\#E$ i nazywamy rzędem grupy punktów krzywej E.

Na krzywej eliptycznej E można zdefiniować działanie grupowe dodawania punktów. Pod pojęciem działania dodawania punktów na krzywej eliptycznej, oznaczanym jako +, rozumiemy działanie dwuargumentowe, łączne, przemienne, posiadające element neutralny $(\mathcal{O})$. Zasady dodawania i podwajania punktów zostały przedstawione na poniższych rysunkach.



Rys. 1. Dodawanie punktów P, Q na krzywej E w ciele liczb rzeczywistych



Rys. 2. Podwojenie punktu P na krzywej E w ciele liczb rzeczywistych

Wykorzystując dodawanie i podwojenie punktu, na krzywej eliptycznej można zdefiniować operację wyznaczania krotności punktu. Polega ona na iteracyjnym dodawaniu do siebie tego samego punktu, tzn.:

$$[k]P = \underbrace{P + P + \ldots + P}_{k}. \tag{4}$$

Jedną z efektywnych metod obliczania krotności punktu jest metoda binarna. Chcąc obliczyć k-tą krotność punktu P, należy dokonać konwersji liczby k na liczbę binarną $\left(k = \sum_{j=0}^{l-1} k_j \cdot 2^j\right)$, zainicjować punkt Q wartością $\mathcal{O}$, a następnie dla każdego bitu k_j podwoić punkt Q. Jeżeli $k_j = 1$, to dodatkowo należy wykonać operację $Q = Q + P$.

3. Problem logarytmu dyskretnego na krzywych eliptycznych

Problem logarytmu dyskretnego może zostać zdefiniowany w różnych strukturach algebraicznych. Powszechnie za takie struktury przyjmuje się: grupę addytywną z działaniem dodawania $(\mathbb{Z}/p\mathbb{Z}, +)$, grupę multiplikatywną z działaniem mnożenia modularnego $(\mathbb{Z}_p^*, \cdot)$ oraz grupę punktów krzywej eliptycznej z działaniem dodawania punktów $E(\mathbb{Z}_p, +)$. Trudność obliczenia logarytmu dyskretnego jest różna, w zależności od zastosowanej grupy.

W przypadku logarytmu dyskretnego na krzywych eliptycznych, jeżeli mamy dany generator grupy punktów P o rzędzie $ord(P) = r$, gdzie r jest liczbą pierwszą, i chcemy obliczyć jego k-tą krotność $Q = [k]P$ taką, że:

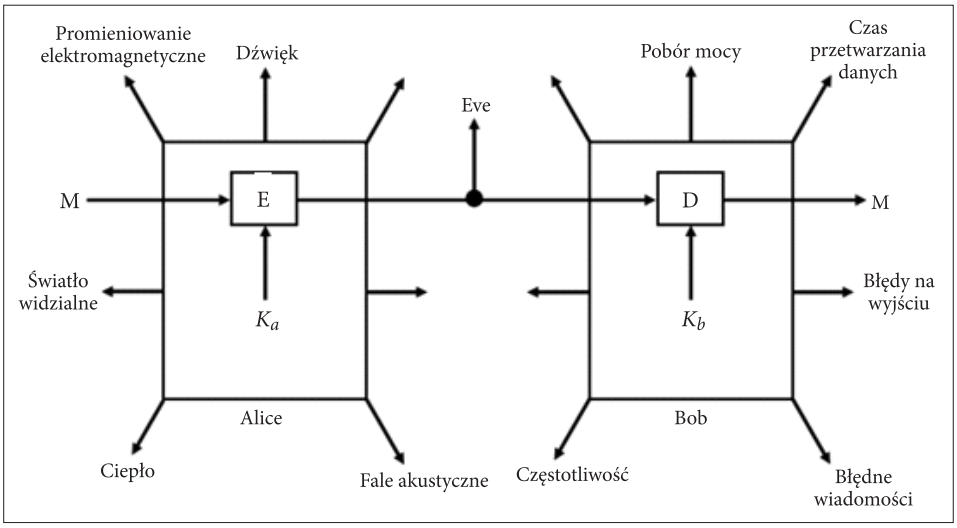
$$Q \equiv \underbrace{P + P + \ldots + P}_{k} \equiv \lfloor k \rfloor P \pmod{r}, \tag{5}$$

obliczenia te możemy wykonać bardzo szybko. Z kolei przy znajomości punktów P i Q wyznaczenie parametru k jest bardzo trudne. Możemy spróbować rozwiązać powyższy problem na przykład za pomocą metod rho Pollarda, lambda Pollarda oraz Gaudry'ego-Schosta. Złożoność obliczeniowa wymienionych metod jest jednak wykładnicza (patrz [2, 3]).

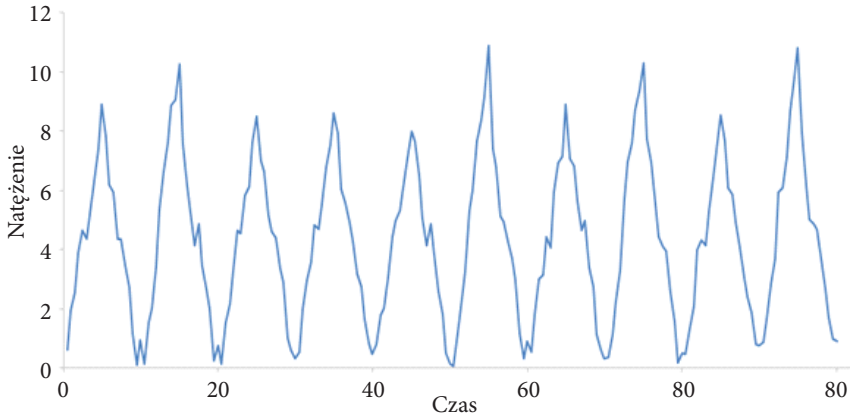
4. Ataki typu *side channel*

Pod pojęciem ataków typu *side channel* należy rozumieć analizę informacji emitowanych podczas pracy urządzeń szyfrujących/deszyfrujących. Zakłada się, że potencjalny atakujący ma dostęp do urządzenia kryptograficznego i zna zastosowany algorytm szyfrujący. Nie posiada on natomiast wiedzy na temat użytych kluczy kryptograficznych. Może próbować dokonać ataku, stosując metody oparte na znajomości tekstu jawnego, szyfrogramu lub par — tekst jawny i odpowiadający mu szyfrogram.

W przypadku dobrze zaprojektowanych algorytmów dane te nie są wystarczające do odtworzenia parametrów klucza. Należy zatem szukać dodatkowych informacji, skupiając się na analizie pracy urządzenia, np. czasie trwania algorytmu, charakterystyce poboru prądu, reakcji na błędy, ulocie elektromagnetycznym itp. Przykładowe dane, emitowane podczas pracy urządzenia, zostały zobrazowane na poniższym rysunku.



Rys. 3. Model kryptograficzny z informacjami emitowanymi podczas pracy urządzenia (na podstawie [4])

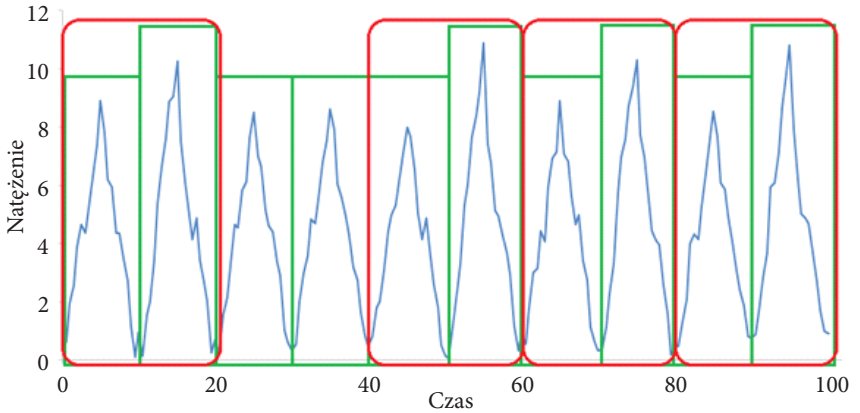


Rys. 4. Emitowany ulot elektromagnetyczny w trakcie pracy urządzenia obliczającego krotność punktu

Dysponując odpowiednio dobranym odbiornikiem, atakujący jest w stanie przechwycić informacje, a następnie z określonym prawdopodobieństwem wyznaczyć bity tajnego klucza.

Załóżmy, że udało nam się zarejestrować przebieg pracy urządzenia obliczającego krotność punktu, przedstawiony na rysunku 4. Korzystając z faktu, że operacja dodawania punktu jest dłuższa, wymaga wykorzystania większej liczby rejestrów oraz musi zostać poprzedzona wykonaniem operacji podwajania punktu, możemy przystąpić do analizy uzyskanych danych.

Na rysunku 5 kolorem czerwonym oznaczono operację dodawania punktu, natomiast kolorem zielonym operację podwojenia punktu. Wykorzystując te informacje, potencjalny atakujący w łatwy sposób może odtworzyć bity krotności k. W naszym przypadku parametr krotności jest równy $k = 100111_2 = 39_{10}$.



Rys. 5. Analiza danych uzyskanych podczas pracy urządzenia obliczającego krotność punktu

5. Sposoby obliczania logarytmu dyskretnego na krzywych eliptycznych z częściową informacją na temat parametru klucza

Bezpieczeństwo kryptosystemów opartych na krzywych eliptycznych wiąże się z trudnością rozwiązania problemu logarytmu dyskretnego w grupie punktów krzywych eliptycznych (ECDLP). Istnieje szereg metod rozwiązujących problem ECDLP. Najlepsza z nich oparta jest o metodę rachunku indeksów. Efektywność metody zależy od postaci krzywej eliptycznej, a co za tym idzie, nie można jej uogólnić dla dowolnego przypadku. W momencie kiedy dysponujemy informacją na temat części bitów parametru klucza (k), z odpowiednio wysokim prawdopodobieństwem, najlepsze wyniki można uzyskać, stosując następujące algorytmy: lambda (kangurów) Pollarda, wielowymiarową metodę Gaudry'ego-Schosta, czy też można przeszukać całą przestrzeń dostępnych rozwiązań.

Idea metody rho Pollarda (którą opisujemy bardzo ogólnie, dokładny opis i szczegóły zawarte są w [3, 5]) opiera się na paradoksie dnia urodzin. Wykorzystuje ona pojedynczą ścieżkę błądzenia przypadkowego aż do zamknięcia cyklu, czyli otrzymania tego samego elementu z grupy. Złożoność obliczeniowa i pamięciowa metody jest równa $\mathrm{O}\left(\sqrt{\frac{\pi \cdot n}{2}}\right)$, gdzie n oznacza rząd generatora, $\pi \approx 3{,}14$.

W przypadku metody lambda Pollarda zasadnicza idea się nie zmienia. W obu metodach można stosować zrównoleglenie obliczeń, wykorzystując wiele ścieżek. Wówczas każda z nich inicjalizowana jest punktem początkowym $P_0^{(1)}, P_0^{(2)}, \ldots, P_0^{(d)}$, a następnie korzystając z dowolnego odwzorowania f, wyznaczana zostaje sekwencja punktów. Punkty spełniające przyjęte kryteria przekazywane są do serwera. W momencie kiedy „natrafimy" na kolizję pomiędzy punktami pochodzącymi z różnych ścieżek, jesteśmy w stanie odtworzyć parametr k. Złożoność metody jest równa $\mathrm{O}\left(\sqrt{\frac{\pi \cdot n}{2}}/M\right)$, gdzie M oznacza liczbę wykorzystywanych procesorów (ścieżek).

Metoda Gaudry'ego-Schosta (patrz [7, 8]) opiera się na paradoksie dnia urodzin. Jednak przed przystąpieniem do opisu metody wprowadzimy następujące definicje i oznaczenia. Niech $T = \{k_1 \,|\, k_1 \in \{a,b\}\}$ oznacza zbiór punktów na drodze kangura oswojonego, $W = k + T$ — zbiór punktów na drodze kangura dzikiego (pojęcia oswojonych i dzikich kangurów pojawiły się najpierw w opisie metody lambda Pollarda, w celu łatwiejszego wyjaśnienia idei algorytmu; opis metody Gaudry'ego-Schosta przejął te określenia z metody lambda Pollarda). Dodatkowo zakładamy, że mamy do dyspozycji N_p procesorów. Połowa z nich odpowiada za realizację ścieżki kangura dzikiego, druga natomiast za ścieżkę kangura oswojonego. Przyjmijmy, że średni

rozmiar kroku wynosi $m/\theta = \sqrt{N}$, rozmiar ścieżki błądzenia przypadkowego wynosi U oraz przedstawmy współrzędną x_p jako:

$$x_p = c_{t-1}x^{t-1} + c_{t-2}x^{t-1} + \ldots + c_1 x + c_0, \tag{6}$$

gdzie $c_i \in \mathbb{Z}_p$ dla $i = 1,2,\ldots t-1$.

Wprowadźmy odwzorowanie S:

$$S\left(x_p, y_p\right) \equiv \left(c_{t-1}\right)_2 \| \left(c_{t-2}\right)_2 \| \ldots \| \left(c_1\right)_2 \| \left(c_0\right)_2 \ (mod\, n). \tag{7}$$

Schemat algorytmu można przedstawić następująco. Zakładamy, że droga kangura oswojonego została zainicjalizowana punktem $x_1 = \left[k_1\right]G$, gdzie $k_1 \in [a, b - m/\theta]$. Kolejne punkty wyznaczamy za pomocą odwzorowania:

$$x_{i+1} = f\left(x_i\right) = x_i + \left[U_{S(x_i)}\right]G. \tag{8}$$

W przypadku kangura dzikiego ścieżka błądzenia przypadkowego rozpoczyna się w punkcie $y_1 = Q + \left[k_1\right]G$ i jest kontynuowana w sposób analogiczny do reguły określonej równaniem (8). W momencie napotkania punktu spełniającego przyjęte kryteria do serwera przekazywana jest informacja dotycząca wartości punktu i odpowiadającej mu krotności. Algorytm kontynuujemy do napotkania kolizji na drodze kangurów oswojonego i dzikiego. Złożoność metody wynosi $\left(2{,}08\sqrt{n}\right)/N_p + \frac{1}{\theta}$.

Metody lambda Pollarda i Gaudry'ego-Schosta dobrze nadają się do poszukiwania logarytmu dyskretnego w przedziałach „zwartych". Algorytm Gaudry'ego-Schosta można wykorzystywać do poszukiwania logarytmu dyskretnego w przypadkach wielowymiarowych. Wówczas średnia złożoność algorytmu wynosi w przypadku pesymistycznym $O\left(2^{\frac{c}{2}}\sqrt{\pi n}\right)$, natomiast w przypadku średnim $O\left(\left(4-2\sqrt{2}\right)^c\sqrt{\pi n}\right) \approx O\left(1{,}17^c\sqrt{\pi n}\right)$, gdzie *c* oznacza liczbę wymiarów (bądź jak później pokażemy — przedziałów).

Szczegółowy opis metody Gaudry'ego-Schosta oraz wielowymiarowej metody Gaudry'ego-Schosta został przedstawiony w publikacji [8].

6. Ataki z częściową informacją o krotności

Załóżmy, że udało nam się przeprowadzić atak typu *side channel*, zdobywając informację na temat części bitów parametru klucza, z odpowiednio wysokim

prawdopodobieństwem. Przyczyną utraty informacji może być równoległa praca innego układu lub celowa manipulacja transmisji. Niestety, nie mamy możliwości przeprowadzenia ponownego ataku. Należy zatem przystąpić do obliczenia logarytmu dyskretnego, wykorzystując metody odpowiednie do uzyskanej wiedzy (patrz [9]). Możemy wyróżnić cztery różne warianty:

1) znamy m bitów najmniej znaczących,
2) znamy m bitów najbardziej znaczących i l bitów najmniej znaczących,
3) znamy m kolejnych bitów znajdujących się pomiędzy bitami najbardziej i najmniej znaczącymi, w szczególności mogą to być również bity najbardziej znaczące,
4) znamy m dowolnie rozmieszczonych bitów.

Zacznijmy od przypadku 1. Do obliczenia logarytmu dyskretnego dla takiego wariantu możemy wykorzystać jednowymiarową metodę Gaudry'ego-Schosta. Na początku zobrazujmy dane parametrów klucza w następujący sposób:

Pierwszym krokiem będzie wyznaczanie przestrzeni poszukiwań. Zakładając, że l bitów klucza jest nieznanych, rozmiar tej przestrzeni wynosi 2^l. W konsekwencji rozwiązanie należy do przedziału $[0,2^l - 1]$. Generatorem dla tak zdefiniowanego przedziału jest punkt G. Można łatwo pokazać, że przypadek 3) można sprowadzić do przypadku 1. Złożoność jednowymiarowej metody Gaudry'ego-Schosta jest w takim przypadku równa $O\left(2,08 \cdot \sqrt{2^l}\right) = O\left(2,08 \cdot 2^{\frac{l}{2}}\right)$, a po pewnych usprawnieniach można uzyskać złożoność niższą niż $O\left(1,72 \cdot 2^{\frac{l}{2}}\right)$, patrz [8].

Jeżeli chodzi o przypadek 2), to w niektórych przypadkach można go sprowadzić do problemu rozmiaru 2^l. Przypadek ten jest dokładnie opisany w [9].

Zastanówmy się teraz nad przypadkiem 4), w którym nieznane bity są dowolnie rozmieszczone. Z uwagi na fakt, że podczas ataku side channel mogą występować zakłócenia w transmisji, przypadek ten wydaje się być najbardziej prawdopodobny. Do obliczenia logarytmu dyskretnego możemy wykorzystać wielowymiarową metodę Gaudry'ego-Schosta lub, w niektórych przypadkach, kiedy mamy do czynienia z bardzo dużą liczbą rozłącznych przedziałów, przeszukać przestrzeń dostępnych rozwiązań metodą brutalną.

Przechwycony ciąg będziemy rozpatrywać w przedziałach. W pierwszym kroku nieznane bity krotności wypełniamy zerami, natomiast pozostałą część bitów pozostawiamy bez zmian. W ten sposób otrzymujmy wartość k_p zadaną następująco:

Aby przedziały „uzwarcić", będziemy operować na innych generatorach:

gdzie k_i jest następującej postaci: $2^{\sum_{i=1}^{c}(m_i+l_i)}$.

Krotność początkowa U =

ZNANE (m_c)	00000...00000	ZNANE (m_{r-1})	00000...00000	...	ZNANE (m_1)	00000...00000

$$G_1 = G$$
$$G_2 = [k_1]G$$
$$\dots$$
$$G_c = [k_{c-1}]G,$$

Tym samym punkt Q stanowiący k-tą krotność generatora wyrażony będzie równaniem:

$$Q = [k]G = [U]G + \sum_{i=1}^{c}[d_i]G_i, d_i \in \{0,\dots,2^{l_i}-1\}, \tag{9}$$

gdzie d oznacza liczbę nieznanych przedziałów $c_i \in \{0,\dots,2^{l_i}-1\}$, $i = \overline{1,d}$.

Złożoność ataku wynosi w przybliżeniu $O\left(2^{\frac{d}{2}}\sqrt{\pi n}\right)$ w przypadku pesymistycznym dla metody Gaudry'ego-Schosta i $O\left(\left(4-2\sqrt{2}\right)^{d}\sqrt{\pi n}\right)$ w przypadku średnim.

7. Opis architektury koprocesora

Głównym założeniem projektowym jest analiza możliwości konstrukcji i realizacji w strukturach programowalnych koprocesora przeznaczonego do obliczania logarytmu dyskretnego na krzywej eliptycznej NIST*P*-192, ponieważ atak na krzywą tej długości jest jeszcze realny. W przypadku zastosowania krzywych eliptycznych o większych rzędach grupy punktów, poszukiwanie logarytmu dyskretnego, przy wykorzystaniu aktualnie dostępnych mocy obliczeniowych, jest bardzo trudne.

Cechą charakterystyczną koprocesora będzie wykorzystanie wielu podukładów zdolnych do obliczania krotności punktu, ale o stosunkowo niewielkiej złożoności, tak aby uzyskać jak największą przepustowość. Zakładamy, że udało się odtworzyć część bitów parametru krotności k, przeprowadzając atak typu *side channel*. Jeżeli uzyskane bity występują kolejno po sobie, to wówczas będziemy preferowali wykorzystanie jednowymiarowej metody Gaudry'ego-Schosta. Jeżeli mamy do czynienia z rozłącznymi przedziałami (ale nie jest ich bardzo dużo), to będziemy wykorzystywać wielowymiarową metodę Gaudry'ego-Schosta. W przypadku wielu

rozłącznych przedziałów (np. jeżeli znamy co drugi bit) zostanie zastosowana metoda brutalna, ale nie będziemy się tym przypadkiem w niniejszej pracy zajmować.

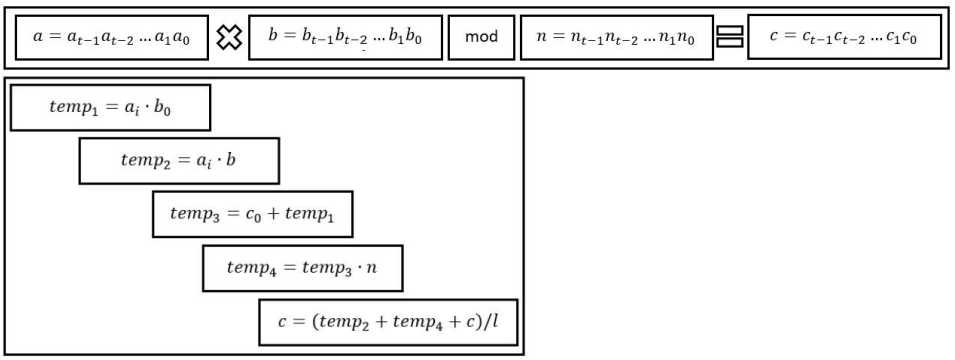
Należy zwrócić uwagę, że istotny wpływ na efektywność rozwiązania ma wybór algorytmu mnożenia modularnego. W tym celu zastosowano algorytm mnożenia Montgomery'ego ze zmienną długością kroku. Przeprowadzono badania pozwalające uzyskać optymalną długość kroku, dla którego zaprojektowane moduły są stosunkowo szybkie, przy nieznacznej zajętości układu.

7.1. Opis zastosowanych algorytmów

Arytmetyka krzywych eliptycznych została zaimplementowana w oparciu o algorytmy dodawania, podwajania punktu przedstawione w publikacji [1]. Krotność punktu obliczana jest za pomocą metody binarnej opisanej w rozdziale 2. Zastosowany algorytm mnożenia modularnego można przedstawić następująco. Niech $m,R,T \in \mathbb{Z}: R > m \wedge \gcd(m,R) = 1 \wedge 0 \le T \le mR$. Wtedy $TR{-}1(mod\ n)$ nazywa się redukcją Montgomery'ego T modulo m przy podstawie R. Należy zauważyć, że dla m zapisanego w postaci $(m_{n-1} \ldots m_0)_b$ optymalnym wyborem R będzie b^n, gdzie $b,n \in \mathbb{N}$. Załóżmy, że $a,b \in Z : 0 \le a,b < m$ oraz $\hat{a} \equiv aR \ (mod\ m), \hat{b} \equiv bR(mod\ m)$. Redukcję Montgomery'ego dla $\hat{a} \cdot \hat{b}$ wyraża się jako:

$$\hat{a} \cdot \hat{b} \cdot R^{-1}(mod\ m) \equiv a \cdot R \cdot b \cdot R \cdot R^{-1}(mod\ m) \equiv a \cdot R \cdot b(mod\ m). \tag{10}$$

Opis algorytmu wraz ze wskazówkami implementacyjnymi znajduje się w publikacji [10]. Podstawowe operacje realizowane przez układ mnożący zostały przedstawione na poniższym schemacie.



Rys. 6. Schemat bloku realizującego operację mnożenia Montgomery'ego w pojedynczym cyklu

Parametry a, b, c, n, l oznaczają kolejno argumenty mnożenia, wynik mnożenia, modulnik oraz długość kroku mnożenia. Parametr t jest wynikiem operacji *długość liczby w bitach*/*długość kroku mnożenia*. Przykładowo, dla kroku mnożenia długości $l = 64$ parametr $t = 192/64 = 3$.

Przystępując do wyznaczenia optymalnej długości kroku mnożenia, wykonano szereg testów, analizując struktury programowalne z rodziny Stratix IV oraz Cyclone IV. Otrzymane rezultaty przedstawiono w poniższych tabelach.

Tabela 1

Analiza uzyskanych rezultatów dla struktury z rodziny STRATIX IV

Długość kroku mnożenia	Zajętość układu			Wymagana liczba taktów zegara na wykonanie mnożenia Montgomery'ego	Częstotliwość taktowania zegara [MHz]
	ALM	Rejestry	DSP (18×18)		
1	**714**	**1324**	**0**	**192**	**153,17**
8	1278	1021	15	24	78,41
16	1367	1074	15	12	65,14
32	1471	1098	30	6	57,48
64	2147	1129	56	3	46,31

Tabela 2

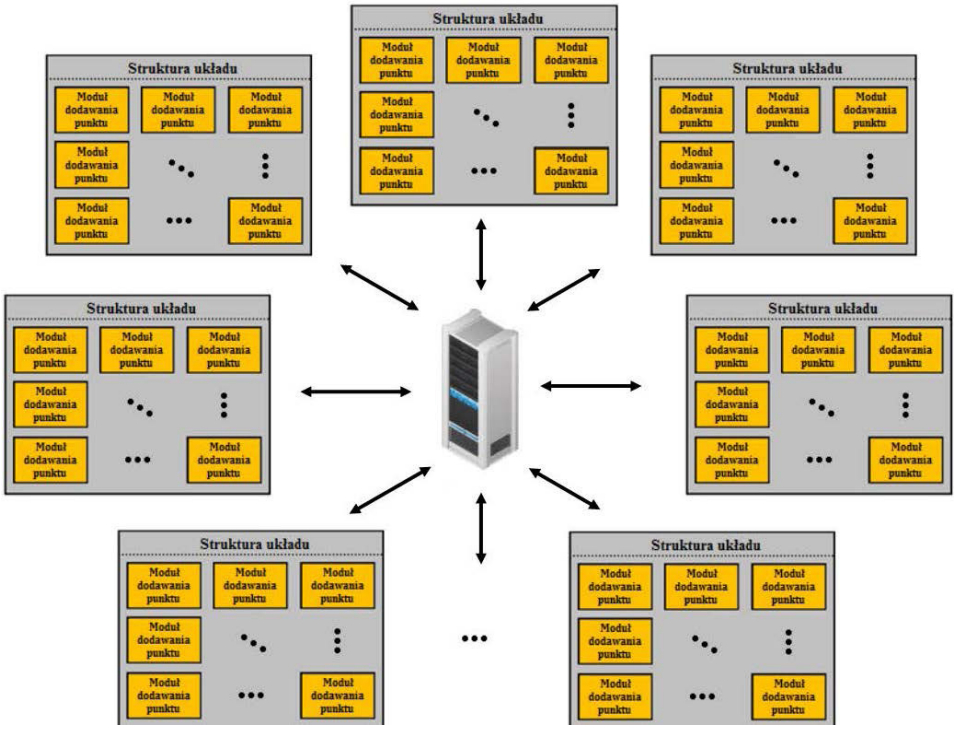
Analiza uzyskanych rezultatów dla struktury z rodziny CYCLONE IV

Długość kroku mnożenia	Zajętość układu			Wymagana liczba taktów zegara na wykonanie mnożenia Montgomery'ego	Częstotliwość taktowania zegara [MHz]
	ALM	Rejestry	DSP (9×9)		
1	**1871**	**1406**	**0**	**192**	**71,32**
8	2078	1012	45	24	55,74
16	2114	1076	46	12	49,98
32	2941	1198	94	6	41,32
64	4735	1232	196	3	32,87

Zgodnie z oczekiwaniami, najniższą zajętością oraz najwyższą częstotliwością taktowania zegara charakteryzuje się moduł z krokiem mnożenia wynoszącym 1 bit. Dodatkowo dla powyższego układu w trakcie realizacji obliczeń nie są wykorzystywane bloki DSP (szybkie multiplikatory), które znacznie zwiększają przepustowość rozwiązania. Z uwagi na dużą liczbę taktów zegara (192) wymaganą do otrzymania wyniku, optymalnym wyborem będzie długość kroku wynosząca 16 lub 32 bity.

7.2. Schemat architektury koprocesora

Struktura układu realizującego obliczanie logarytmu dyskretnego na krzywych eliptycznych charakteryzuje się dużą liczbą modułów odpowiedzialnych za poszukiwanie rozwiązania (zrównoleglenie obliczeń wpływa na efektywność rozwiązania). Każdy z modułów zostaje zainicjowany punktem początkowym oraz otrzymuje unikalną część przestrzeni klucza k. Zakładamy, że punkty początkowe dla modułów oraz punkty będące generatorami w nieznanych przedziałach zostały obliczone i przekazane do struktury układu przez komputer centralny. Na poniższym rysunku przedstawiono schemat zaprojektowanego rozwiązania.



Rys. 7. Schemat układu obliczającego logarytm dyskretny

Moduły rozpoczynają pracę po zainicjalizowaniu punktem początkowym oraz fragmentem przestrzeni klucza. Komponenty odpowiedzialne są za dodawanie punktów we współrzędnych afinicznych (aby możliwe było porównywanie uzyskanych punktów na podstawie wartości współrzędnej x).

7.3. Analiza uzyskanych rezultatów

Przeprowadzając analizę wyników implementacji, skorzystano z układów EP4SGX70, EP4S40G2 z rodziny Stratix IV oraz EP4CGX150 z rodziny Cyclone IV. Kompilacje przeprowadzono w oparciu o środowisko QUARTUS firmy ALTERA w wersji 13.1. Parametry układów przedstawiono w poniższej tabeli.

Tabela 3

Parametry wybranych struktur programowalnych

Struktury programowalne	ALM	LE	Rejestry	DSP	Cena [$]
EP4S40G2	91 200	–	182 400	1288	10 000
EP4SGX70	29 040	–	58 080	384	800
EP4CGX150	–	149 760	149 760	720	300

Dla struktur programowalnych EP4SGX70, EP4S40G2 optymalnym rozwiązaniem jest wybór modułu z 32-bitowym krokiem mnożenia Montgomery'ego. W obrębie układu EP4SGX70 można zaimplementować trzy moduły obliczające logarytm dyskretny. Dla struktury programowalnej EP4S40G2 liczba modułów wynosi 10. Z kolei dla EP4CGX150 najlepszą przepustowość otrzymano dla modułu z krokiem mnożenia wynoszącym 16 bitów. Liczba dostępnych modułów w tym przypadku jest równa 7. Czynnikami decydującymi o liczbie modułów są komórki logiczne ALM/LE oraz bloki DSP.

Tabela 4

Liczba modułów, które można umieścić w obrębie struktury EP4SGX70 z rodziny Stratix IV

Krok mnożenia	Częstotliwość taktowania zegara [MHz]	Przepustowość pojedynczego modułu [liczba operacji $\times 10^6$/s]	Liczba modułów	Całkowita przepustowość l modułów [10^6/s]
32	54,81	0,020	3	0,060
	53,12	0,019	3	0,058
16	55,78	0,011	3	0,033
	54,91	0,011	3	0,033

Tabela 5

Liczba modułów, które można umieścić w obrębie struktury EP4S40G2 z rodziny Stratix IV

Krok mnożenia	Częstotliwość taktowania zegara [MHz]	Przepustowość pojedynczego modułu [liczba operacji $\times 10^6$/s]	Liczba modułów	Całkowita przepustowość l modułów [10^6/s]
32	55,17	0,020	10	0,202
	54,16	0,020	10	0,199

cd. tabeli 5

16	55,63	0,011	10	0,110
	54,89	0,011	11	0,119

Tabela 6

Liczba modułów, które można umieścić w obrębie struktury EP4CGX150 z rodziny Cyclone IV

Krok mnożenia	Częstotliwość taktowania zegara [MHz]	Przepustowość pojedynczego modułu [liczba operacji $\times 10^6$/s]	Liczba modułów	Całkowita przepustowość l modułów [10^6/s]
32	30,45	0,011	3	0,033
	30,01	0,011	3	0,033
16	42,11	0,008	7	0,058
	39,15	0,008	7	0,054

Największą przepustowością charakteryzują się struktury programowalne EP4S40G2. Cena układu to ok. 10 000 dol., dopuszczalna liczba modułów jest równa 10, co skutkuje kosztem pojedynczego modułu na poziomie 1000 dol. Pewne jest, że posiadając odpowiednią liczbę modułów, jesteśmy w stanie znaleźć logarytm dyskretny w stosunkowo krótkim czasie.

Postaramy się teraz przedstawić rozwiązanie bardziej adekwatne pod względem efektywności do ceny. Zakładamy, że dysponujemy budżetem wynoszącym 100 000 dol., natomiast czas ważności zaszyfrowanej wiadomości wynosi 30 dni. Jako koszt rozwiązania uwzględniamy koszty struktur programowalnych, pomijamy koszt pracy i połączenia podzespołów ze sobą. Do porównania urządzeń wykorzystamy parametr oznaczający liczbę operacji na sekundę przy koszcie jednego dolara. Otrzymane wyniki przedstawiono w poniższej tabeli. Przyjęto, że mamy do czynienia z jednym przedziałem występujących po sobie nieznanych bitów. Uzyskane rozwiązanie można łatwo dostosować do przypadku, gdy mamy do czynienia z kilkoma rozłącznymi przedziałami nieznanych bitów, ale należy pamiętać, że układ będzie posiadał odrobinę większą zajętość i czas poszukiwania rozwiązania będzie dłuższy, w zależności od liczby przedziałów.

Tabela 7

Dopuszczalna liczba nieznanych bitów krotności dla problemu ECDLP

Struktura programowalna	Częstotliwość taktowania zegara [MHz]	Liczba operacji na sekundę/koszt struktury	Liczba bitów — metoda Gaudry'ego-Schosta
EP4S40G2	55,17	22	83
EP4SGX70	54,81	75	87
EP4CGX150	**42,11**	**194**	**90**

Analizując otrzymane rezultaty, dochodzimy do wniosku, że optymalnym wyborem jest struktura programowalna EP4CGX150. Przy zastosowaniu jednowymiarowej metody Gaudry'ego-Schosta jesteśmy w stanie odtworzyć 90 bitów krotności k.

8. Podsumowanie

Zgodnie z przyjętymi założeniami udało się zaprojektować koprocesor przeznaczony do obliczania logarytmu dyskretnego na krzywych eliptycznych, przy założeniu znajomości części kolejnych bitów poszukiwanej krotności. Skupiono się na krzywych eliptycznych nad ciałem $GF(p)$, a badania wykonano na konkretnym przypadku NIST P–192.

Przeprowadzono badania pozwalające uzyskać efektywne moduły przeznaczone do obliczania dodawania punktów na krzywych eliptycznych w odniesieniu do zagadnienia logarytmu dyskretnego na krzywych eliptycznych. Rozpatrywano w tym celu mnożenie Montgomery'ego z różną długością kroku. Dobrano optymalną wartość długości kroku, dla której moduły obliczające logarytm dyskretny są stosunkowo szybkie, ale jednocześnie zajmują mało komórek logicznych i bloków DSP. Dla urządzeń z rodziny Stratix IV najlepszym wyborem jest 32-bitowy krok mnożenia. W przypadku rodziny Cyclone IV jest to krok 16-bitowy.

Dokonano analizy parametrów układów, dla których zaproponowane rozwiązanie zapewnia największą liczbę operacji na sekundę przy koszcie jednego dolara. Dla rodziny Stratix IV strukturą programowalną spełniającą nasze kryteria jest EP4SGX70 (koszt 800 dol.), natomiast dla rodziny Cyclone IV struktura EP4CGX150 (koszt 300 dol.).

Oszacowano maksymalny rozmiar problemu, który jesteśmy w stanie rozwiązać, wykorzystując jednowymiarową metodę Gaudry'ego-Schosta. Realnym ograniczeniem czasu niezbędnego do uzyskania rozwiązania są ograniczenia finansowe.

Zaproponowane rozwiązanie umożliwia praktyczną realizację poszukiwania logarytmu dyskretnego.

Artykuł opracowany na podstawie referatu pt. *Realizacja koprocesora wspierającego kryptoanalizę problemów opartych na krzywych eliptycznych* wygłoszonego przez Michała Kędzierskiego i Michała Wrońskiego na XLIV Konferencji Zastosowań Matematyki w Zakopanem — 19 września 2015 r.
W dopracowanie ostatecznych założeń i koncepcję realizacji badań znaczny wkład miał Michał Misztal.

Artykuł wpłynął do redakcji 10.05.2016 r. Zweryfikowaną wersję po recenzjach otrzymano 27.11.2017 r.

LITERATURA

[1] Blake I., Seroussi G., Smart N., *Krzywe eliptyczne w kryptografii*, WNT, 1999.

[2] Chmielowiec A., *Wydajne metody generowania bezpiecznych parametrów algorytmów klucza publicznego*, Warszawa, 2012.

[3] Sliverman J.H., *The Arithmetic of Elliptic Curves*, USA, 2009.

[4] Zhou Y.B., Feng D.G., *Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing*, https://eprint.iacr.org, 2005.

[5] Rosen K.H., *Elliptic Curves Number Theory and Cryptography. Second Edition*, LLC, 2008.

[6] Vredendaal Ch., *Rank Estimation Methods in Side Channel Attacks*, Stanford Security Seminar, 2014.

[7] Menezes A.J., van Oorschot P.C., Vanstone S.A., *Handbook of Applied Cryptography*, 1CRC Press, 1, 1996.

[8] Ruprai R.S., *Improvements to the Gaudry-Schost Algorithm for Multidimensional discrete logarithm problem and Applications*, Department of Mathematics, Royal Holloway University of London, 2010.

[9] Gopalakrishnan K., Theriault N., Yao Ch.Z., *Solving Discrete logarithms with Partial Knowledge of the Key*, Springer, 2007.

[10] Montgomery P.L., *Modular Multiplication without trial division*, JSTOR, 1985.

[11] *Mathematical routines for the NIST prime elliptic curves*, *https://www.nsa.gov.*

M. KĘDZIERSKI, M. MISZTAL, M. WROŃSKI

Realization of coprocessor which supports counting of discrete logarithm on elliptic curves with partial knowledge

Abstract. In this paper we analyse realization of a coprocessor which supports counting of discrete logarithm on elliptic curves over the field $FG(p)$, where p is the large prime, in FPGA. Main idea of the realization is based on using modules which are able to add the points and have relatively small resources' requirements. We showed the simplified case in which we know l most significant bits of key k and we used one-dimensional Gaudry–Schost method. We also generalize that case and analyse the case when unknown bits are given in many disjoint intervals. To do this we propose using a multidimensional Gaudry–Schost method. At the end of this article we show the solution which provides best trade-off between throughput and price of a device.

Keywords: cryptology, elliptic curves, discrete logarithm on elliptic curves (ECDLP), attacks with partial knowledge, multi-dimensional Gaudry–Schost algorithm

DOI: 10.5604/01.3001.0010.8185