Silesian University of Technology
Faculty of Automatic Control, Electronics and Computer Science
Institute of Electronics
University of Rennes 1
IRISA

STRESZCZENIE

Arithmetic operators on $GF(2^m)$ for cryptographic applications: performance - power consumption - security tradeoffs

(Moduły operatorów arytmetycznych względem ciał skończonych $GF(2^m)$ dla zastosowań kryptograficznych: próba pogodzenia wydajności, poboru mocy i bezpieczeństwa)

Author:

Danuta Pamuła

Supervisors:
dr hab. inż. Edward Hrynkiewicz,
prof. nzw. w Politechnice Śląskiej (PL)
Arnaud Tisserand, CNRS researcher, HDR (FR)

Gliwice 2012

1. Wprowadzenie

W dzisiejszych czasach systemy cyfrowe i Internet obecne są w prawie wszystkich obszarach naszego życia. Odpowiedzialne są za komunikacje między ludźmi, instytucjami, rządami. Używane są do nadzoru systemów lotniczych, transportowych, zarządzania zasobami medycznymi, bankowymi, giełdowymi, wojskowymi. Zewsząd jesteśmy zalewani cyfrowymi danymi, które nie zawsze wiemy skąd pochodzą i które niełatwe są w identyfikacji, autoryzacji, zabezpieczaniu, zarządzaniu czy przechowywaniu. Większość zwyczajnych użytkowników systemów cyfrowych jest nieświadoma skutków niezabezpieczania, kradzieży, manipulacji ich cyfrowych danych lub, co gorsza kradzieży ich cyfrowej tożsamości. Dla wielu tożsamość cyfrowa nie jest traktowana na równi z tożsamością w rzeczywistym świecie reprezentowana przez dowód osobisty lub paszport. Skutki przejęcia lub utraty i jednej i drugiej tożsamości są identyczne, a cyfrowej mogą być nawet groźniejsze [22].

Na szczęście świadomość konieczności ochrony i zabezpieczania cyfrowych danych i tożsamości rośnie wraz ze wzrostem ilości usług dostępnych w globalnej sieci (Internecie). Każdy chce bezpiecznie dokonywać zakupów przez Internet, zarządzać zasobami swojego konta bankowego bez obawy o malwersacje, podpisywać dokumenty cyfrowe w sposób jednoznaczny, aby uniemożliwić fałszerstwo lub podszywanie się pod dana osobę. Zaczyna dostrzegać się potrzebę zabezpieczania cyfrowych danych takich jak np. dane medyczne, podatkowe, finansowe, prywatne dane, etc. W obecnych czasach, gdy większość usług, zadań można wykonać bez wychodzenia z domu korzystając z dostępnych cyfrowych rozwiązań rośnie potrzeba zabezpieczania przesyłanych i przechowywanych danych.

Dodatkowym problemem związanym z zabezpieczaniem danych jest wpływ mechanizmów zabezpieczających na pracę systemu, przesyłanie czy wymianę danych. Zbyt wolne lub znacznie obciążające system mechanizmy zabezpieczające dane (systemy kryptograficzne) powodują, że rezygnuje się z nich na rzecz płynnej pracy systemu. Z tego względu projektanci systemów cyfrowych zabezpieczając je muszą brać pod uwagę nie tylko skuteczność dodawanych zabezpieczeń ale także ich wpływ na wydajność systemu. Niestety wraz ze wzrostem

świadomości użytkowników, co do ochrony danych oraz wraz ze wzrostem ilości sposobów na ich zabezpieczanie rośnie również liczba możliwości ich kradzieży, uszkodzenia lub podsłuchania.

Nauka traktująca o ochronie danych to kryptografia [25]. Kryptografia jest jedną z gałęzi: kryptologii. Drugą gałęzią kryptologii rozwijającą się równolegle jest kryptoanaliza mająca na celu weryfikację zabezpieczeń dostarczanych przez kryptografów poprzez próbę opracowania skutecznych sposobów ich łamania.

Obecnie do zabezpieczania danych używa się właściwości NP trudnych problemów matematycznych i specjalnych zestawów zmiennych/stałych, które sprawiają, że rozwiązanie problemów jest bardzo kosztowne i czasowo nieopłacalne. Wraz ze wzrostem mocy obliczeniowej systemów cyfrowych NP trudne problemy matematyczne stają się rozwiązywalne w rozsądnym czasie i przy rozsądnych kosztach. Aby zwiększyć trudność ich rozwiązania proponuje się zwiększać rozmiar liczb, na których operują lub używać liczb o specjalnych właściwościach np.liczb pierwszych. Ze względu na to, że ataki matematyczne stają się coraz trudniejsze i bardziej kosztowne, zaczęto poszukiwać nowych (tańszych, łatwiejszych) metod łamania systemów kryptograficznych. Poszukując nowych metod na odkrycie sekretu (tajnych danych, klucza szyfrującego, itp.) kryptoanalitycy zaczęli się interesować urządzeniami wykonującymi zadania kryptograficzne. Analogicznie do łamaczy zamków mechanicznych podjęto próbę podsłuchania urządzenia w czasie pracy i analizy otrzymanych danych w celu stwierdzenia czy jest możliwe na tej podstawie złamanie systemu kryptograficznego. Próby podsłuchania urządzeń wykonujących zadania kryptograficzne okazały się być skuteczne. Kryptoanalitycy poprzez analizę czasu wykonywania operacji, mocy pobieranej przez urządzenie czy emitowanego pola elektromagnetycznego są w stanie złamać algorytm kryptograficzny.

Jeszcze do niedawna informacje wyciekające z urządzenia cyfrowego, takie jak chwilowy pobór mocy czy emitowane pole elektromagnetyczne uważane były za nieistotny szum, którego zmniejszeniem lub kontrolą się nie zajmowano. Obecnie twórcy systemów i rozwiązań kryptograficznych świadomi zagrożeń wynikających z podsłuchiwania urządzeń cyfrowych starają się im zapobiegać, poprzez kontrolowanie tych "nieistotnych szumów". Ataki, które wykorzystują takie informacje nazywane są atakami kanałem bocznym tzw. side-channel

attacks (SCA) [12]. Ataki te można podzielić ze względu na to, z jakiego kanału bocznego informacje wykorzystują, na ataki analizujące czas wykonywania operacji (timing analysis attacks), ataki analizujące chwilową moc pobieraną przez urządzenie (power analysis attacks) oraz ataki analizujące pole elektromagnetyczne urządzenia (electromagnetic analysis attacks).

Ataki typu side-channel polegające na analizie poboru mocy chwilowej przez urządzenia kryptograficzne są jednym z bardziej popularnych typów ataków, których opracowano bardzo wiele wariantów. Są one stosunkowo łatwe do przeprowadzenia, mało kosztowne i dosyć efektywne.

Dodawanie do układów kryptograficznych zabezpieczeń przed atakami fizycznymi nie jest łatwym zadaniem. Ze względu na dużą złożoność implementacyjną algorytmów kryptograficznych samo urządzenie bez zabezpieczeń jest już dość złożone. Rozbudowywanie urządzenia przez dodawanie elementów takich jak generatory szumów, zakłóceń, może skutkować opracowaniem ogromnego i powolnego urządzenia, co w dzisiejszych czasach jest nie do przyjęcia. Ponadto istnieją skuteczne techniki odfiltrowywania szumów z pozyskanych pomiarów, dodawanie więc do urządzenia układów generujących szumy w celu zakłócenia pozyskiwanych od niego sygnatur prądowych (wykresów mocy) jest metodą nieskuteczną jako samodzielny sposób zabezpieczania układu. Tego typu zabezpieczenie może jedynie być dodatkowym sposobem przeciwdziałania atakom. Zabezpieczenia układów kryptograficznych muszą polegać na takich modyfikacjach algorytmicznych i strukturalnych, które nie mają negatywnego wpływu na ich wydajność (zajętość zasobów i prędkość działania).

Dwoma najpopularniejszymi systemami kryptograficznymi są systemy kryptografii symetrycznej (kryptografia z kluczem tajnym) oraz systemy kryptografii asymetrycznej (kryptografia z kluczem publicznym). Oba typy systemów opierają swoje bezpieczeństwo na tajności klucza, którym operują. Tajność klucza zapewniona jest przez pewne problemy matematyczne o bardzo dużej złożoności obliczeniowej, niemożliwe do rozwiązania przy użyciu dostępnej mocy obliczeniowej w rozsądnym czasie.

W prowadzonych badaniach rozważano rozwiązania dedykowane dla technik kryptografii z kluczem publicznym (PKC). W PKC tajny klucz (klucz prywatny) można wyznaczyć jedynie poprzez rozwiązanie jednego z NP trudnych

problemów matematycznych, takich jak:

- problem faktoryzacji (system: RSA[1])
- problem logarytmu dyskretnego (DLP[2]) (system: ElGamal)
- problem logarytmu dyskretnego na krzywych eliptycznych (ECDLP[3]) (system: ECC[4] - kryptografia krzywych eliptycznych)

W tej pracy zdecydowano się zająć poprawą wydajności i bezpieczeństwa urządzeń wykonywujących zadania systemów ECC głównie ze względu na to, że dowiedziono że może być ona bardziej wydajna niż np. systemy RSA.

Systemy ECC uznawane są za potencjalnie wydajniejsze od np. systemów RSA ze względu na wielkość klucza, którego używają aby zapewnić określony poziom bezpieczeństwa. Przykładowo system RSA używający kluczy o rozmiarze 3072 bitów zapewnia bezpieczeństwo równoważne gwarantowanemu przez system ECC używający kluczy o rozmiarze 256 bitów. Porównanie wielkości kluczy gwarantujących ekwiwalentne bezpieczeństwo różnych systemów kryptograficznych przedstawione jest w Tabeli 1.

Tabela 1.: Rozmiary kluczy gwarantujących odpowiedni poziom zabezpieczeń dla różnych systemów kryptograficznych [27]

bezpieczeństwo (bity)	min. rozmiar (bity) klucza publicznego		
	DSA/DH	RSA	ECC
80	1024	1024	160
112	2048	2048	224
128	3072	3072	256
192	7680	7680	384
256	15360	15360	512

Systemy ECC wykorzystują właściwości krzywych eliptycznych zdefiniowanych nad ciałami skończonymi. Główne operacje protokołów ECC wykonywane są na punktach takich krzywych. Aby móc wykonać np. dodawanie punktów krzywej $P + Q$, należy wykonać operację na współrzędnych punktów P i Q. Współrzędne punktów są elementami ciała skończonego nad którym zdefin-

[1] Rivest-Shamir-Adleman
[2] discrete logarithm problem
[3] elliptic curve discrete logarithm problem
[4] elliptic curve cryptography

iowana jest dana krzywa. Z tych zależności wynika, że operacje protokołów krzywych eliptycznych mocno zależą od operacji wykonywanych na elementach ciał skończonych. Z tego względu założono, że od efektywności działania układów wykonujących operacje arytmetyczne na elementach ciał skończonych zależy efektywność działania całego systemu ECC. Dążąc do zwiększenia wydajności cyfrowego systemu ECC należy zaproponować szybkie i niewielkie układy sprzętowe realizujące operacje arytmetyczne w ciałach skończonych.

Dwa najbardziej popularne ciała skończone wykorzystywane w kryptografii to ciała pierwsze: $GF(p)$ i grupa ciał o charakterystyce 2: $GF(2^m)$. Zdecydowano się zająć grupą ciał skończonych Ciała te uważane są za bardziej odpowiednie dla rozwiązań sprzętowych głownie ze względu na fakt, że ich elementy mogą być reprezentowane przez binarne wielomiany, a nie jak w przypadku ciał pierwszych liczb dziesiętnych. Unikamy w ten sposób problemów związanych z przeniesieniami [29].

Pierwszym celem prowadzonych badań jest opracowanie wydajnych układów realizujących operacje arytmetyczne w ciałach $GF(2^m)$. Za kryterium oceny wydajności przyjęto złożoność układową, prędkość działania i pobór mocy. W dalszej części tekstu dla uproszczenia układy te będziemy nazywali sprzętowymi operatorami arytmetycznymi dla ciał $GF(2^m)$.

Drugim celem pracy jest zabezpieczenie opracowanych sprzętowych operatorów arytmetycznych przed atakami typu side-channel analizującymi pobór mocy urządzenia kryptograficznego.

W protokołach kryptografii krzywych eliptycznych najważniejszą i najczęściej atakowaną operacją jest mnożenie punktu na krzywej przez bardzo duży, zazwyczaj tajny (często jest to klucz prywatny) skalar: $[k]P$. Operacja ta wykonywana jest jako ciąg elementarnych operacji na punktach krzywej ($P+Q$, $2P$). Najprostszym algorytmem wykonywującym te operacje jest algorytm "podwój-i-dodaj" (double-and-add), patrz Algorytm 1.

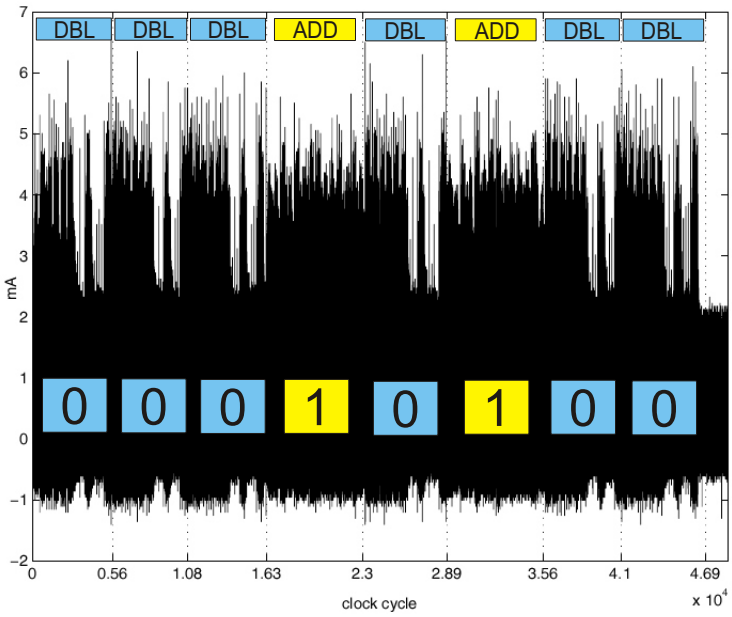
W algorytmie wykonanie operacji dodawania ściśle uzależnione jest od wartości bitu tajnego skalara k. Z tego względu, gdy możliwe jest rozróżnienie na wykresie chwilowego poboru mocy operacji dodawania i podwajania, możliwe jest w prosty sposób odkrycie skalara k (Rysunek 1).

Ponieważ operacje $P + Q$ i $2P$ są operacjami na współrzędnych punktów

Algorithm 1 Algorytm "podwój-i-dodaj" (double-and-add) [9]

Input: $k = (k_{t-1}, ..., k_1, k_0)_2$, $P \in E(\mathbb{F}_q)$
Output: $[k]P$

```
Q ← ∞
for i = 0 to t − 1 do
    if k_i = 1 then
        Q ← Q + P          // ADD //
    end if
    P ← 2P                 // DBL //
end for
return Q
```



Rysunek 1.: Przykład prostej analizy mocy (SPA) Algorytmu 1

krzywej, czyli operacjami na elementach ciał skończonych, to właśnie sygnatura tych operacji może pozwolić atakującemu na odróżnienie sygnatur prądowych operacji $2P$ i $P + Q$ od siebie.

Podjęta próba zabezpieczania operacji wykonywanych na elementach ciał skończonych jest pierwszą tego typu próbą. W dostępnej literaturze przed-

stawiono jedynie sposoby zabezpieczania operacji na poziomie krzywych eliptycznych, poprzez, np. ujednolicanie ilości wykonywanych operacji dodawania i podwajania (algorytm double-and-add always), wykonywanie redundantnych operacji, randomizacja/rekodowanie klucza k i punktu P (aby ciąg operacji $P + Q$ i $2P$ był za każdym razem inny). Poszczególne zabezpieczenia istniejąc samodzielnie są zazwyczaj wystarczające dla zabezpieczenia urządzeń przed atakami polegającymi na prostej analizie mocy (SPA[5]) jednak przed atakami polegającymi na różnicowej analizie mocy (DPA[6]) jest już je trudniej zabezpieczyć. Ataki SPA polegają na analizie pojedynczej sygnatury urządzenia, natomiast ataki DPA korzystają z bardzo dużej ilości próbek (kilka tysięcy), modelu urządzenia i metod statystycznych w celu złamania systemu. Uważa się, że zabezpieczenie wszystkich warstw operacji systemu ECC powinno uniemożliwić przeprowadzenie skutecznego ataku. Z tego względu zdecydowano się zbadać i zredukować źródła wycieku informacji, jakim mogą być operacje arytmetyczne wykonywane na elementach ciał skończonych, nad którymi krzywe są zdefiniowane. Zakłada się, że zredukowanie wycieku informacji powodowanego przez operatory arytmetyczne może w znacznym stopniu poprawić bezpieczeństwo systemów ECC. Zakłada się również, że możliwe będzie rozszerzenie opracowanych zabezpieczeń na zabezpieczenia przeciwko atakom analizującym emitowane przez urządzenie pole elektromagnetyczne.
Dwa wyżej wymienione cele prowadzą do zdefiniowania kolejnego. Zazwyczaj urządzenie bardzo dobrze zabezpieczone znacznie traci na wydajności. Celem badań jest zaproponowanie wydajnych układów, dlatego też należy znaleźć kompromis pomiędzy rozmiarem opracowanych zabezpieczeń, a spadkiem wydajności urządzenia. Dodane zabezpieczenia muszą jak najmniej negatywnie wpływać na wydajność urządzenia.

Zgodnie z wymienionymi celami sformułowano następującą tezę:
Możliwe jest opracowanie wydajnych i odpornych na wybrane ataki typu side-channel polegające na kryptoanalizie mocy, sprzętowych układów realizujących operacje arytmetyczne w ciałach $GF(2^m)$.

[5]Simple Power Analysis
[6]Differential Power Analysis

2. Przeprowadzone badania, napotkane problemy, zaproponowane rozwiązania

2.1. Sprzętowe realizacje operatorów arytmetycznych dla $GF(2^m)$.

W ciele $GF(2^m)$ zdefiniowane są dwie operacje: dodawanie i mnożenie. Wszystkie pozostałe operacje (dzielenie, inwersja, itp.) można przedstawić za pomocą dodawania i mnożenia. Celem prowadzonych badań jest zaproponowanie realizacji sprzętowych dla tych operacji arytmetycznych. Przed podjęciem się tego zadania należało zdecydować się na parametry operatorów i dobrać je odpowiednio do przyjętych założeń. Parametrami operatorów, które mają znaczny wpływ na sposób ich implementacji są: baza (sposób reprezentacji elementów ciała), generator ciała $f(x)$ i rozmiar ciała m (rozmiar argumentów na jakich będą wykonywane operacje).

Do reprezentacji elementów ciała wykorzystuje się wiele różnych baz zależnie od aplikacji. Najpopularniejszymi bazami wykorzystywanymi w kryptografii są baza potęgowa (standardowa), baza normalna i jej pochodne (optymalna baza normalna ONB, gaussowska baza normalna GNB) oraz bazy dualne. Po przepro-wadzeniu szerokiej analizy właściwości różnych baz zdecydowano się skorzystać z bazy potęgowej: $\{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\}$. Bazy dualne zostały odrzucone ze względu na to, że zazwyczaj stosowane są, gdy operuje się na niewielkich ciałach skończonych o $m = 8, 16, 32$, itp. (w tej pracy rozważano ciała o rozmiarach $m \cong$150–600).

Zarówno baza standardowa jak i bazy normalne posiadają wiele cech, które pozwalają na opracowanie wydajnych sprzętowych operatorów. Kierując się opiniami zawartymi w dostępnej literaturze dotyczącej przedmiotu stwierdzającymi że dla baz standardowych można otrzymać bardziej wydajne urządzenia zdecydowano się na użycie bazy standardowej. Aczkolwiek rozwiązania oparte na bazach normalnych wydają się warte rozważenia w przyszłości.

Kolejnym istotnym parametrem operatorów arytmetycznych dla ciał skończonych jest rozmiar ciała m i rodzaj użytego generatora ciała $f(x)$. Ze względu na to, że opracowywane operatory mają posłużyć rozwiązaniom dedykowanym systemom ECC, wykorzystywać będziemy rozmiary i generatory ciał zdefin-

iowane w standardach kryptograficznych [7].

Wszystkie opracowane rozwiązania dedykowane są dla układów FPGA. W badaniach wykorzystano układy produkowane przez firmę Xilinx takie jak: Spartan-3E XC3S1200E [32], Virtex-6 LX240T [2]. Do testowania zabezpieczeń opracowanych operatorów wykorzystano specjalną płytę SASEBO-G [23] zawierającą układ Virtex-II Pro XC2VP7 [31]. Opracowywane architektury opisane są w języku VHDL i syntezowane za pomocą środowiska ISE 12.2 i 9.2 firmy Xilinx. Symulacje zostały przeprowadzone przy użyciu wbudowanego symulatora ISim oraz symulatora ModelSim firmy Menthor Graphics.

Operator dodawania. Dodawanie w ciele $GF(2^m)$ jest względnie prostą operacją, operacją `XOR` przeprowadzaną na dwóch m-bitowych wektorach binarnych. Z drugiej strony nawet prosta operacja matematyczna wykonywana na bardzo dużych elementach może powodować problemy z synchronizacją (spowalniać działanie innych funkcji) i zajmować wiele zasobów. W celu zweryfikowania rzeczywistego wpływu operacji dodawania na inne elementy systemu ECC zaimplementowano kilka rodzajów operatorów dodawania.

Procesor ECC, w którym zaprojektowane operatory zostaną wykorzystane przewiduje przesyłanie danych w słowach (16, 32-bitowych). Z tego względu zaproponowano następujące rozwiązania operatorów dodawania:

1: Dodawanie "w locie" kolejnych słów wektorów a, b i odkładanie częściowych wyników w rejestrze c (rozwiązanie 1);

2: Dodawanie "w locie" kolejnych słów wektorów a, b i odkładanie częściowych wyników w pamięci (rozwiązanie 2);

3: Oczekiwanie na wszystkie słowa wektorów a, b, dodanie otrzymanych wartości (rozwiązanie 3);

W Tabeli 2 zaprezentowane są uzyskane wyniki, nie prezentujemy jedynie parametrów rozwiązania 3, ze względu na to, że jest ono bardzo proste, można powiedzieć, że jest to translacja wejść do wyjść układu. Otrzymane wyniki wykazały, że operacja dodawania jest operacją prostą mającą niewielki wpływ na inne układy z którymi zostanie zintegrowana. Zaproponowane rozwiązania są małe, zajmują od 20-30 LUTów i szybkie. Wraz ze wzrostem rozmiaru argumentów zajętość rośnie w niewielkim stopniu, a prędkość działania niez-

Tabela 2.: Operatory dodawania (Virtex-6)

rozmiar ciała	rozwiązanie 1		rozwiązanie 2	
m	[LUT]	[MHz]	[LUT]	[MHz]
163	21	771	26	562
233	21	771	26	562
283	22	767	28	560
409	22	767	28	560
571	24	578	31	558

nacznie spada. W praktyce dodawanie w systemach ECC wykonuje się zazwyczaj równolegle do innych operacji ze względu na niski stopień złożoności.

Operator mnożenia w ciele $GF(2^m)$. Mnożenie w ciele skończonym [4] uważane jest za dosyć skomplikowaną operację. Można je traktować jako operację modularną ze względu na to, że w celu uzyskania właściwego iloczynu dwóch elementów ciała należy je pomnożyć, a następnie zredukować otrzymany iloczyn modulo generator ciała, czyli nieredukowalny wielomian $f(x)$. W ten sposób w wyniku mnożenia dwóch elementów pewnego ciała otrzymujemy element $c(x) = a(x)b(x) \bmod f(x)$, który jest również elementem tego samego ciała.

Algorytmy mnożenia w ciele $GF(2^m)$ można podzielić na: **algorytmy dwustopniowe**, w których mnożenie i redukcję wykonuje się w dwóch osobnych, kolejnych krokach i **algorytmy naprzemienne**, w których przeplata się lub łączy mnożenie z redukcją.

Podczas badań przeanalizowano wiele istniejących algorytmów mnożenia i ich modyfikacji, aby odkryć takie ich właściwości (sposoby przetwarzania, reprezentacji danych, zrównoleglania operacji, sposoby optymalizacji), które pozwoliłyby na opracowanie własnych sprzętowych operatorów mnożenia. W tym paragrafie przedstawiono pokrótce najważniejsze otrzymane wyniki i wnioski z analizy algorytmów opisanych w dostępnej literaturze.

Zaproponowane rozwiązania nie będą całkowicie nowe gdyż będą oparte na dobrze znanych matematycznych teoriach. Celem badań jest zmodyfikowanie lub złożenie właściwości istniejących algorytmów aby móc opracować jak na-

jbardziej wydajne sprzętowe operatory. Należy też znaleźć naj-lepszy sposób na przeniesienie poszczególnych części algorytmów w sprzęt.

Algorytmy dwustopniowe - najciekawsze wyniki

Podczas badań osobno rozpatrywano metody mnożenia oraz redukcji.

Krok pierwszy: mnożenie. Najbardziej znaną metodą mnożenia dwóch wielomianów jest metoda "przesuń-i-dodaj" (shift-and-add), nazywana również metodą szkolną (schoolbook). Mając dane dwa wielomiany $a(x)$ i $b(x)$ stopnia $(m-1)$ iloczyn $d(x)$ stopnia $(2m-2)$ otrzymujemy mnożąc wielomian $a(x)$ przez kolejne współczynniki wielomianu $b(x)$, czyli przesuwając wektor a, reprezentujący wielomian $a(x)$, o odpowiednią ilość bitów wskazaną przez współczynniki wektora b (wielomian $b(x)$), otrzymane częściowe wyniki kolejno dodajemy.

Metoda ta jest dosyć prosta i pozwala tworzyć układy o regularnych strukturach. Jednak wykorzystana w niezmienionej postaci do mnożenia bardzo dużych wektorów może okazać się mało wydajna szczególnie pod względem ilości zajmowanych zasobów sprzętowych. Większość algorytmów mnożenia wielomianów jest zazwyczaj jakiegoś rodzaju modyfikacją tego algorytmu. Różnice pomiędzy różnymi wersjami polegają na sposobach reprezentowania, kolejności przetwarzania danych, itp. W przypadku mnożenia wielkich liczb wykorzystuje się np. sposób podziału danych metoda "dziel-i-zwyciężaj", reprezentacje argumentów z użyciem macierzy, proponuje się specyficzne sposoby przetwarzania danych. Najciekawszymi optymalizacjami wydają się być: metoda wykorzystująca zasadę "dziel-i-zwyciężaj" i optymalizację Karatsuby-Ofmana oraz metoda macierzowo-wektorowa.

Celem metody "dziel-i-zwyciężaj" jest podzielenie dużego problemu na mniejsze podproblemy. W naszym przypadku rozłożenia problemu mnożenia wielkich liczb na zestaw mnożeń mniejszych liczb, czyli podziału wektorów wejściowe na mniejsze i wykonanie mnożenie na mniejszych argumentach. Częściowe wyniki składa się w celu uzyskania wyniku ostatecznego. Najbardziej popularną i wykorzystywaną modyfikacją tej metody jest optymalizacja Karatsuby i Ofmana [10], która minimalizuje liczbę mnożeń niezbędnych do wykonania na mniejszych argumentach.

Metoda macierzowo-wektorowa [4] zakłada zapisanie wielomianu $a(x)$ jako

macierzy A o rozmiarze $(2m-1) \times m$, w której każda kolumna reprezentuje kolejne przesunięcia w lewo wektora $a(x)$ ($a << 2$). Element $b(x)$ zapisany jest w postaci wektora o rozmiarze m, iloczyn $d(x)$ jest także wektorem ale rozmiaru $(2m-1)$.

Różne sposoby reprezentacji liczb, na których operujemy silnie wpływają na architekturę sprzętowych rozwiązań. Niektóre pozwalają przyśpieszyć działanie układu, ale zwiększają ilość zasobów sprzętowych przez niego zajętych, niektóre znacznie zmniejszają powierzchnie układu ale za to powodują zmniejszenie prędkości jego działania. Przy projektowaniu układów zazwyczaj należy szukać kompromisu pomiędzy zajętością zasobów, a prędkością układu. W celu zaprojektowania wydajnych operatorów mnożenia niezbędne było przestudiowanie wielu istniejących rozwiązań. Niestety bardzo często proponowane w literaturze optymalizacje algorytmów są rozważaniami teoretycznymi, wiele z tych optymalizacji gdy przeniesiono w sprzęt okazało się dawać wyniki gorsze od oryginału lub dokładnie takie same. Zaimplementowano więc bardzo wiele teoretycznych rozwiązań w celu dostrzeżenia ich właściwości, które mogą być korzystne dla rozwiązań sprzętowych. Szczegółowa analiza różnych rozwiązań dla układów mnożących została zaprezentowana na konferencjach lokalnych i miedzynarodowych [18, 17, 19].

Dla ułatwienia porównywania przedstawionych rozwiązań wprowadzono miarę AT (współczynnik wydajności), zdefiniowany następująco:
$AT =$ (zajętość $\times$ czas wykonywania operacji).

Wyniki badań wskazują, że najlepsze rozwiązania dla dwustopniowych algorytmów można osiągnąć przy implementacji wykorzystujących reprezentację macierzowo-wektorową argumentów oraz sztuczkę Karatsuby-Ofmana. Dokładniej, budując układy mnożące złożone z trzech podjednostek mnożących argumenty o rozmiarach o połowę mniejszych niż pierwotne. Wyniki dla operatorów mnożenia dla ciał o rozmiarach 233, 283, 409, 571 przedstawione są w Tabeli 3.

Krok drugi: redukcja. Generalnie istnieją dwie metody redukcji: klasyczna i macierzowa. Klasyczna metoda jest metodą iteracyjną a metoda macierzowa wykorzystuje specjalną macierz redukcji R (patrz [4]). Dla obydwóch metod zaproponowano już wiele optymalizacji, większość optymalizuje liczbę oper-

Tabela 3.: 233, 283, 409 i 571-bitowe operatory mnożenia (Virtex-6)

Układ mnożący	Zajętość [LUT]	Max.f(#cykli) [MHz]	AT
233-bitowy operator zbudowany z trzech 117-bitowych podjednostek mnożących	2625	520 MHz (234)	1181.3
283-bitowy operator zbudowany z trzech 142-bitowych podjednostek mnożących	3381	535 MHz (284)	1794.8
409-bitowy operator zbudowany z trzech 206-bitowych podjednostek mnożących	4834	535 MHz (412)	3722.6
571-bitowy operator zbudowany z trzech 286-bitowych podjednostek mnożących	7095	522 MHz (572)	7774.6

acji niezbędnych do wykonania dla konkretnego nieredukowalnego wielomianu $f(x)$. Istnieją specjalne typy generatorów ciała $f(x)$ takie jak trójmiany, wielomiany pięcioelementowe lub np. wielomiany typu ESP[7], które pozwalają na znaczne uproszczenie algorytmów redukcji. W pracy zaproponowano własny sposób optymalizacji względem nierdukowalnego wielomianu aby zmniejszyć złożoność operacji redukcji.
Drugą wspomnianą metodą redukcji jest redukcja z użyciem macierzy redukcji R [4]. Metoda ta pozwala znacznie przyśpieszyć proces redukcji modulo $f(x)$. Macierz redukcji R jest zdefiniowana przez współczynniki nierozkładalnego wielomianu $f(x)$. Powszechnie uważa się, że metoda redukcji z użyciem macierzy R jest bardzo wydajna. Podczas prowadzonych badań tę metodę również zoptymalizowano ze względu na używany generator ciała $f(x)$.

W celu porównania metod redukcji zaimplementowano jej wersję klasyczną, zoptymalizowaną i macierzową. Wyniki przedstawiono w Tabeli 4. Jak można zauważyć jednostka oparta na klasycznej, niezoptymalizowanej metodzie jest jednostką potrzebującą najwięcej cykli zegarowych do wykonania operacji oraz jednostką zajmującą dosyć dużo zasobów układu rekonfigurowalnego. W przy-

[7] equally spaced polynomials

Tabela 4.: Wyniki implementacji algorytmów redukcji ($m = 233$) (Virtex-6)

Typ algorytmu	Zajętość [LUT]	Max. f(#cykli) [MHz]	AT
Klasyczny	3528	209 MHz (600)	10128.2
Klasyczny zoptymalizowany	1165	571 MHz (8)	16.3
Macierz redukcji (sekwencyjny)	466	1264* MHz (2)	0.74
Macierz redukcji (kombinacyjny)	233	1.13 ns	0.26

*wyniki przedstawione w tabeli są wynikami podanymi przez środowisko Xilinx ISE, w tym wypadku otrzymany wynik mówi nam, że jednostka po integracji z innym układem nie będzie miała prawie żadnego wpływu na jego prędkość.

padku zoptymalizowanego algorytmu wykonywującego redukcję liczba operacji niezbędnych do wykonania silnie zależy od formy nierozkładalnego wielomianu $f(x)$. Dla $m = 409$ liczba operacji pozostaje taka sama jak dla $m = 233$, dla tych dwóch rozmiarów ciał istnieją trójmiany. Natomiast dla $m = 163, 283, 571$ liczba operacji, które należy wykonać wzrasta do 14 (co nadal jest małą liczbą) ponieważ dla tych ciał minimalnymi wielomianami nierozkładalnymi są wielomiany o pięciu niezerowych współczynnikach. Podsumowując można stwierdzić, że jeżeli znany jest generator ciała na którym operujemy powinno się ze względu na niego zoptymalizować metodę redukcji.

Najbardziej wydajnymi wydają się być rozwiązania korzystające z macierzy redukcji R. W przypadku zoptymalizowania jej pod względem nierozkładalnych wielomianów okazuje się, że dla wielomianów o trzech i pięciu elementach macierz redukcji R zawiera wiele zer. Z tego powodu skomplikowaną i czasochłonną operację redukcji można zastąpić kilkoma operacjami `XOR`. Im więcej elementów niezerowych posiada generator tym więcej operacji `XOR` należy wykonać. W przypadku projektowania operatorów arytmetycznych do zastosowań kryptograficznych nie wyznaczamy generatorów ciał sami lecz są one zdefiniowane w standardach. W ten sposób dla każdego używanego rozmiaru ciała znamy jego generator i możemy przeprowadzić niezbędne optymalizacje.

Podsumowanie analizy algorytmów dwustopniowych. Przeprowadzona analiza zaowocowała zaprojektowaniem pełnego operatora mnożenia dla $GF(2^m)$. W oparciu o otrzymane wyniki aby stworzyć kompletny operator zdecydowano, że najlepszą wydajność gwarantuje użycie:

- jako **jednostki mnożącej**: jednostkę opartą na algorytmie macierzowo-wektorowym, zbudowaną z trzech podjednostek, wykorzystującą optymalizację Kartasuby-Ofmana i podział argumentów wejściowych na pół $(m/2 \pm 1)$
- jako **jednostki wykonywującej redukcję**: jedostkę opartą na algorytmie klasycznym zoptymalizowanym ze względu na wielomian nierozkładalny lub jednostkę wykorzystującą macierz redukcji R.

Przykładowe wyniki implementacji dla $GF(2^{233})$ zaprezentowane są w Tabeli 5.

Tabela 5.: Operatory mnożenia w ciele $GF(2^{233})$ (Virtex-6)

Układ mnożący m=233	Zajętość [LUT]	Max. f [MHz]	# cykli zeg.	AT
operator z klasyczną redukcją	3638	302	264	3.18×10^3
operator z redukcją macierzową	2862	302	238	2.25×10^3

Algorytmy naprzemienne (interleaved) - najciekawsze wyniki

W tego typu algorytmach operację mnożenia wykonujemy naprzemiennie z redukcją. Z tej grupy algorytmów najbardziej interesujące wydają się być: algorytm wykorzystujący macierz Mastrovito i algorytm Montgomery'ego.

Metoda wykorzystująca macierz Mastrovito. Metoda macierzy Mastrovito jest rozwinięciem klasycznej metody macierzowo-wektorowej. W mnożeniu z użyciem macierzy Mastrovito zamiast kolejno mnożenia i redukcji wykonujemy: $c = Mb$, gdzie M to tzw. *macierz Mastrovito.* Macierz Mastrovito jest zbudowana z macierzy A (reprezentującej $a(x)$) i macierzy redukcji R [4]. W wyniku badań opracowano skuteczną optymalizację algorytmu. Zaproponowano aby zamiast operować na całej macierzy M o rozmiarze $(2m-1) \times m$

podzielić ją na podbloki o rozmiarze 16×16 bitów, przykładowy podział dla ciała o rozmiarze $m = 233$ przedstawiony jest na Rysunku 2. Podbloki wym-

M	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M1	M1	M1	M1	Mc
1	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M1	M1	M1	Mc
2	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M1	M1	Mc
3	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M1	Mc
4	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	M4	M4	M4	M4	Mc
5	M2	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	M4	M4	M4	Mc
6	M0	M2	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	M4	M4	Mc
7	M0	M0	M2	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	M4	Mc
8	M0	M0	M0	M2	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	Mc
9	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	M3	M3	M3	M3	Mc
10	M0	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	M3	M3	M3	Mc
11	M0	M0	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	M3	M3	Mc
12	M0	M0	M0	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	M3	Mc
13	M0	M0	M0	M0	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	Mc
14	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR

Rysunek 2.: Ilustracja podziału macierzy Mastrovito dla $m = 233$

nażane są przez odpowiedni fragment wektora b przy użyciu dedykowanych podjednostek mnożących. Na Rysunku 2 można zaobserwować podbloki korzystając z tych samych podjednostek (podbloki o tym samym oznaczeniu).

Dodatkowo aby zmniejszyć rozmiar układu założono, że współczynniki podbloków macierzy M obliczane będą "w locie" podczas mnożenia (nie zachodzi wtedy potrzeba przechowywania ich). Cały proces mnożenia kontrolowany jest za pomocą automatu stanów skończonych (FSM).

Aby znaleźć jak najlepsze rozwiązanie wykorzystujące macierz Mastrovito wykonano analizę wielu wariantów tego typu rozwiązania. Zmieniano ilość stanów automatu kontrolującego mnożenie, zmnieniano ilość wykorzystywanych instancji podjednostek, itd. Ze względu na ogromne ilości wariantów tego rozwiązania nie przetestowano wszystkich możliwych opcji. W wyniku przeprowadzonej analizy stwierdzono, że najbardziej wydajne jest rozwiązanie korzystające

z automatu kontrolującego o większej ilości stanów ale wykorzystujące pojedyncze instancje podjednostek mnożących podbloki macierzy.

Tabela 6.: Implementacje rozwiązań wykorzystujących macierz Mastrovito

Virtex-6 XC6VLX240T	$(A^L b + A^H R)b$	Mb
zajętość (A)[LUT]	5014	3760
max.f	297 MHz	276 MHz
czas operacji (T)[cykle zegarowe]	65	75
wydajność AT	1097	1021

Algorytm Montgomery'ego. Drugim bardzo popularnym algorytmem naprzemiennym jest metoda Montgomery'ego [15]. Metoda oparta jest na założeniu, że przy wykonywaniu ciągu operacji korzystamy ze specjalnej reprezentacji liczb, reprezentacji Montgomery'ego. Aby wykonać operacjmodularną zamiast $c(x) = a(x)b(x) \bmod f(x)$ wykonujemy $c(x) = a(x)b(x)r^{-1}(x) \bmod f(x)$, co przy odpowiednim doborze parametru $r(x)$, znacznie skraca czas i zmniejsza złożoność wykonywanej operacji. W celu pozyskania właściwego wyniku należy na koniec operacji wykonywanych w reprezentacji Mongomery'ego powrócić do oryginalnej reprezentacji liczb. Aby przy jednokrotnym mnożeniu z użyciem tego algorytmu otrzymać wynik taki jak w przypadku poprzednich algorytmów należy użyć algorytmu dwukrotnie. Pełny algorytm wykonujący mnożenie w ciele $GF(2^m)$ zaprezentowany jest poniżej, patrz Algorytm 2.

Algorithm 2 Mnożenie modularne - metoda Montgomery'ego [11]

Input: $a(x), b(x), r(x), f(x), f'(x), r^2(x) \bmod f(x)$
Output: $c(x) = a(x)b(x) \bmod f(x)$

1: $t(x) = a(x)b(x)$
2: $u(x) = t(x)f'(x) \bmod r(x)$ // $MontMult(a, b)$ //
3: $d(x) = [t(x)$ `XOR` $u(x)f(x)]/r(x)$
4: $t(x) = d(x)(r^2(x) \bmod f(x))$
5: $u(x) = t(x)f'(x) \bmod r(x)$ // $MontMult(d, r^2 \bmod f)$ //
6: $c(x) = [t(x)$ `XOR` $u(x)f(x)]/r(x)$
7: **return** c

W algorytmie wykorzystywane są trzy dodatkowe wartości: $r(x)$, $r^2(x)$ mod $f(x)$, $f'(x)$. Element $r(x)$ według [11] dla ciał $GF(2^m)$ jest prostym jednomianem x^m. Jeżeli znamy, tak jest w tym przypadku, nieredukowalny wielomian $f(x)$, który będzie używany, wartość $f'(x)$ i $r^2(x)$ mod $f(x)$ można wyznaczyć.

W Tabeli 7 przedstawiono przykładowe wyniki implementacji. Najbardziej złożoną operacją tego algorytmu jest operacja wykonywana w kroku drugim, czyli mnożenie dwóch dużych wektorów a, b. Z tego powodu parametry jednostki mnożącej wykonującej tę operację mają znaczący wpływ na prędkość i rozmiar urządzenia.

Tabela 7.: Rozwiązania dla algorytmu Montgomery'ego (Virtex-6)

	Zajętość [LUT]	max.f [MHz]	# cykli	jednostka mnożąca	AT
1	3197	338	270	233-bitowy operator zbudowany z 233x16-bitowych jednostek mnożących zajętość: 2308 LUT, max.f: 323MHz	2554
2	3730	302	244	233-bitowy operator zbudowany z 3 117-bitowych jednostek mnożących zajętość: 2625 LUT, max.f: 302MHz	3014

2.2. Podsumowanie i wnioski

Porównanie otrzymanych rozwiązań z już istniejącymi nie jest łatwe. Dostępna literatura nie zawsze zawiera wszystkie niezbędne do porównania dane. Prezentowane w niej rozwiązania nie zawsze są odpowiednio opisane, brakuje czasu wykonywania operacji albo prędkości lub też zajętość podana jest w innych niż powszechnie przyjęto jednostkach. Dodatkowo w wielu przypadkach wydajność urządzeń podana jest w postaci teoretycznych wyliczeń ilości potrzebnych bramek `AND` czy `OR` lub też skomplikowanych matematycznych wyliczeń przewidujących opóźnienia lub prędkość układu. Jako, że wyniki implementacji zależą nie tylko od użytego algorytmu ale też od sposobu jego opisu w języku HDL, własne implementacje opisanych algorytmów mogą dać inne wyniki niż te otrzymane przez ich autorów.

W Tabeli 8 przedstawiamy zaprojektowane w wyniku badań rozwiązania.

Tabela 8.: Zaproponowane operatory mnożenia dla $GF(2^{233})$

Algorytm	Zajętość [LUT]	f [MHz]	cykle zegarowe	AT
Klasyczny 1	3638	302	264	3.18
Klasyczny 2	2862	302	238	2.25
Mastrovito	3760	297	75	0.95
Montgomery (pełny)	3197	338	270	2.55

W Tabeli 9 prezentujemy parametry rozwiązań opisanych w dostępnej literaturze. Porównując zawartości obu tabel możemy stwierdzić, że zaproponowane w wyniku prowadzonych badań rozwiązania są dosyć szybkie i nieduże.

3. Zabezpieczanie sprzętowych operatorów arytmetycznych dla ciał $GF(2^m)$

Drugim celem prowadzonych badań jest zabezpieczenie opracowanych operatorów arytmetycznych przed atakami typu side-channel polegającymi na kryptoanalizie mocy pracującego urządzenia. Wyniki prezentowane poniżej zostały przedstawione na konferencji WAIFI 2012 [20].

Kryptanaliza mocy rozważa dwa typy wycieków informacji:

- *wyciek wynikającu z liczby przełączeń* - związany z liczbą bitów zmieniających swój stan w jednostce czasu.
- *wyciek wynikający ze zmian odległości Hamminga HW* - związany z liczbą bitów równych 1 przetwarzanych w jednostce czasu.

Ponieważ w układach VLSI moc chwilowa związana jest z liczbą przełączeń występujących w jednostce czasu rozważano wyciek wynikający z liczby "użytecznych" przełączeń w układzie ($Hamming\ distance = HW(t+1) - HW(t)$). Przez użyteczne przełączenia rozumiemy liczbę zmian stanów bitów rejestrów i sygnałów urządzenia podczas wykonywania operacji arytmetycznych, w czasie jednostki czasu. Interesuje nas liczba zmian stanów, która nastąpiła pomiędzy dwoma cyklami zegarowymi. Tak zdefiniowane użyteczne przełączenia będziemy w dalszej części nazywać *użyteczną aktywnością układu*. Aby zidentyfikować

Tabela 9.: Rozwiązania opisane w dostępnej literaturze

	m	Układ	Zajętość	Max.f /opóźnienie	T
[3]	256	Virtex II 2000-6	5267 LUT	44.91 MHz	5.75 us
	1033				23.07 us
[28]	1024	XCV2000E-6	4355 CLB	100.4 MHz	-
		XC40150XV-7	8339 CLB	44.4 MHz	-
		XC4VFX100-10	2793 CLB	150.5 MHz	-
[5]	233	XC2V-6000-4	415 slices	-	2.42 us
[16]	233	Stratix EP1S40F780C5	3728 LE	4.04 ns	12 cycles
	283		3396 LE	3.66 ns	20 cycles
[30] by [16]	233	Stratix EP1S40F780C5	3353 LE	6.91 ns	16 clock
	283		3118 LE	6.95 ns	20 clock
[8]	233	XC2V-6000 FF1517-4	37296 LUT	77 MHz	-
			11746 LUT	90.33 MHz	-
			36857 LUT	62.85 MHz	-
			45435 LUT	93.20 MHz	-
[24]	191	XCV2600E	8721 CLB	-	82.4 us
[1]	88	Altera EP2S60	6644 ALUTS	-	
[6]	163	Virtex XCVL330	201,989 LUTs 214,703 DFFs	241 MHz	-
			1471 LUTs 982 DFFs	241 MHz	- -
[13]	283	Virtex 4 XC4VFX140	1781 CLB 2156 FF 3367 LUT	246.670 MHZ	-
	1132		25,955 CLB 32,578 FF 48,591 LUT	248.447 MHz	-

źródło wycieku informacji istniejące w opracowanych układach, zaproponowano sposoby jego ewaluacji, a następnie przeprowadzono wnikliwą analizę.

Przeprowadzając analizę mocy urządzenia i znając algorytm na podstawie

którego jest ono zbudowane atakujący stara się znaleźć korelacje istniejące pomiędzy danymi, na których ono operuje lub rodzajami wykonywanych operacji, a zmianami w mocy chwilowej pobieranej przez urządzenie [14].

W celu ewaluacji stopnia zabezpieczeń operatorów (ewaluacji użytecznej aktywności urządzeń) zaproponowano następującą metodę. Zaproponowano by monitorować każdy rejestr i każdy sygnał urządzenia poprzez dołączenie do niego tzw. *monitora aktywności* [26].

Monitor aktywności zlicza liczbę przełączeń stanów bitów układu w następujący sposób. Przerzutik typu `D` zapamiętuje poprzedni stan sygnału i przy użyciu bramki `XOR` jest on porównywany z obecnym stanem sygnału. Bramka `XOR` daje na wyjściu `1` w momencie gdy nastąpiło przełączenie. Liczba przełączeń zliczana jest przez k-bitowy licznik. Liczba bitów k licznika zależy od ilości monitorowanych sygnałów.

Zważywszy na specyfikę układów VLSI, okazuje się, że taki sposób ewaluacji aktywności układu daje wyniki bardzo podobne do uzyskiwanych poprzez wykonanie pomiarów prądowych z użyciem sondy i oscyloskopu [26]. Po omówieniu istniejących zródeł wycieków informacji i sposobów ich redukcji przedstawimy porównanie pomiędzy danymi otrzymanymi z monitorów, a danymi otrzymanymi poprzez zmierzenie prądu chwilowego pobieranego przez urządzenie.

Jak zaprezentowano powyżej w wyniku analizy algorytmów dla operacji w ciele $GF(2^m)$ przygotowano trzy typy układów mnożących: klasyczny, Mastrovito i Montgomery. Wszystkie przeanalizowano ze względu na odporność na ataki typu side-channel.

3.1. Weryfikacja poziomu zabezpieczeń, identyfikacja problemu, proponowane modyfikacje

Przeprowadzono bardzo wiele testów, w których wykorzystano wiele różnych zestawów losowych danych o równomiernym rozłożeniu zer i jedynek. Tutaj prezentujemy typowe wyniki badań aktywności. Mimo, iż zauważono, że zależność między otrzymanymi kształtami wykresów, a danymi wejściowymi jest niewielka, niemożliwym było uśrednienie otrzymanych krzywych ze względu na to, że mogłoby to zafałszować wyniki, np. poprzez spłaszczenie lub ujednolice-

nie otrzymywanego wykresu aktywności urządzenia. W pracy przedstawiono analizę dla $m = 233$ dla pozostałych rozmiarów ciał zdefiniowanych w standardach wyniki są podobne.

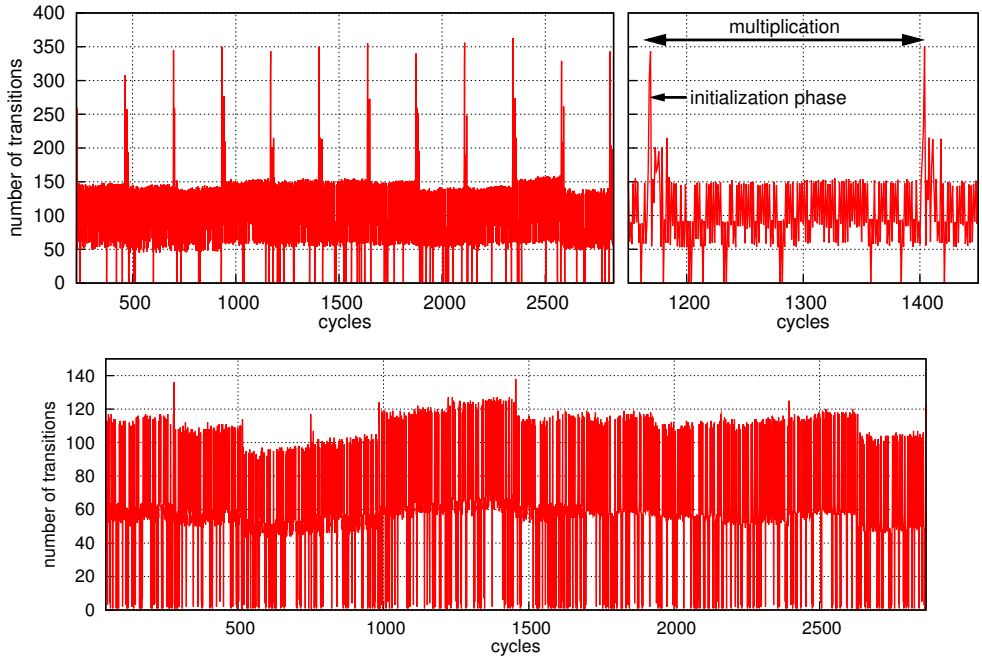
Rozwiązanie oparte na algorytmie klasycznym Zaproponowano dwa typy rozwiązań opartych na algorytmie klasycznym: wykorzystujące klasyczną zoptymalizowaną metodę redukcji i wykorzystujące macierz redukcji R.

Po przeanalizowania próbek aktywności dla obydwóch typów rozwiązań stwierdzono, że redukcja nie ma znaczącego wpływu na kształt krzywej przedstawiającej aktywność urządzenia. Dla obydwóch rozwiązań krzywe są niemal identyczne (dla tych samych danych testowych). Zdecydowano się więc na analizę rozwiązania korzystającego ze zoptymalizowanej metody redukcji

Na Rysunku 3 (rysunek górny) przedstawiono wykres aktywności urządzenia niezabezpieczonego dla sekwencji mnożeń wykonanych na losowych argumentach. Łatwo można zaobserwować co może być potencjalnym źródłem przydatnych informacji dla atakującego, co może ułatwić wyznaczenie granic czasowych poszczególnych operacji mnożenia, czyli stwierdzić np. ile operacji mnożenia zostało wykonanych.

Na początku każdej operacji występuje nagły wzrost aktywności, który pozwala na wyznaczenie początku operacji mnożenia. Ten nagły wzrost aktywności występuje ze względu na sposób inicjalizacji/reinicjalizacji urządzenia i przeładowywania wartości argumentów na których urządzenie operuje.

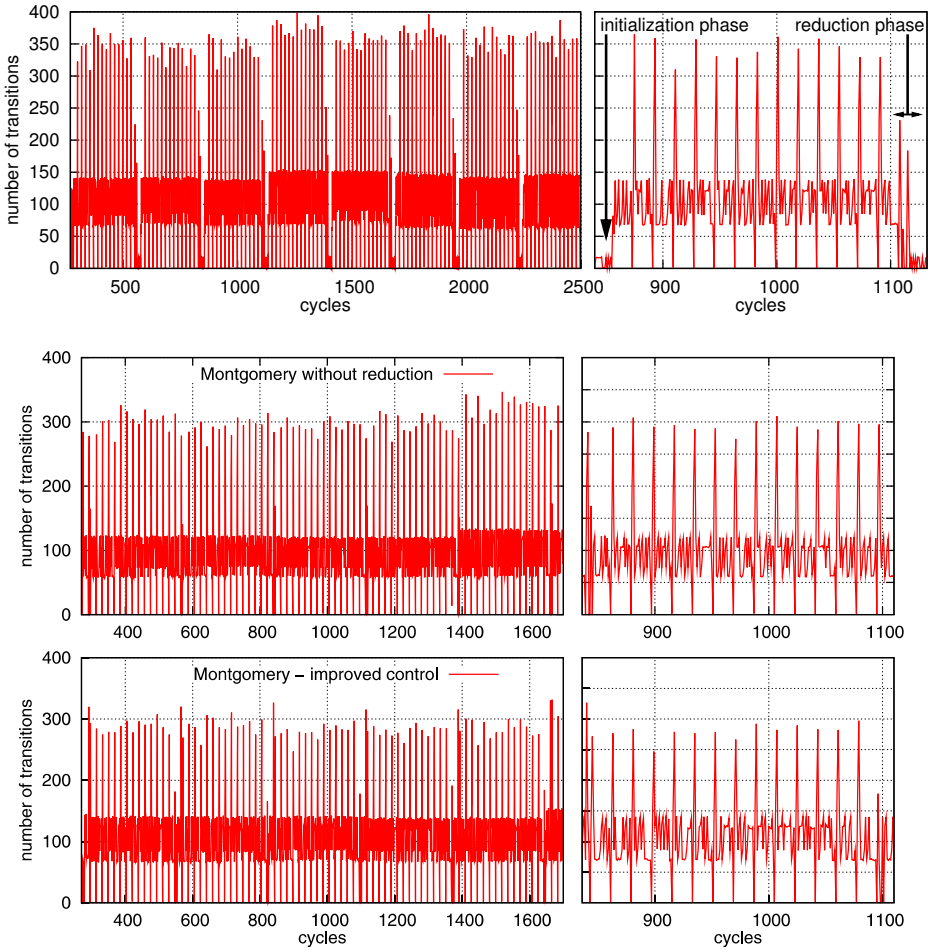
Aby uniknąć nagłego wzrostu aktywności spowodowanego inicjalizacją urządzenia, przebudowano kontroler układu i sposób resetowania układu. Początkowo by zapewnić prawidłową pracę resetowano i przeładowywano wartości wszystkich wykorzystywanych rejestrów na początku pracy układu. Teraz resetuje się poszczególne rejestry lub przeładowywuje ich zawartość tuż przed użyciem, rozkładając proces inicjalizacji na kilka cykli zegarowych. W ten sposób unika się zmiany stanów większości rejestrów w jednej jednostce czasu. Na Rysunku 3, część dolna, przedstawiono wykres aktywności układu po zaproponowanych modyfikacjach. Podczas modyfikacji działania układu i jego architektury, zaproponowano wiele różnych sposobów reinicjalizacji. Tu prezentujemy wykres otrzymany dla najlepszej według nas modyfikacji. Przeprowad-



Rysunek 3.: Wykres użytecznej aktywności układu mnożącego wykorzystującego algorytm klasyczny

zone modyfikacje pozwoliły na zauważenie dodatkowych sposobów optymalizacji układu i wprowadzenie pewnych ulepszeń. Jak można zaobserwować, liczba przełączeń zmalała, obecnie waha się w granicach od 100 do 120 przełączeń, w wersji niezmodyfikowanej wynosiła od 100 do 150 przełaczeń. Optymalizacje minimalizujące w pewien sposób liczbę przełączeń minimalizują również nieznacznie moc pobieraną przez urządzenie. Analizując kształt krzywej otrzymanej dla zabezpieczonej wersji operatora widać brak charakterystycznych wzrostów aktywności. Dodatkowo np. w przedziale pomiędzy 1500, a 2400 cykli trudno wyznaczyć granice kolejnych operacji, trudno określić ile mnożeń zostało wykonanych.

Rozwiązanie oparte na algorytmie Montgomery'ego Wyniki pomiaru aktywności dla urządzenia niezabezpieczonego realizującego Algorytm 2 przedstawione są na Rysunku 4 (rysunek górny). Dla właściwego porównania algorytmu



Rysunek 4.: Wykres użytecznej aktywności układu mnożącego wykorzystującego algorytm Montgomery'ego.

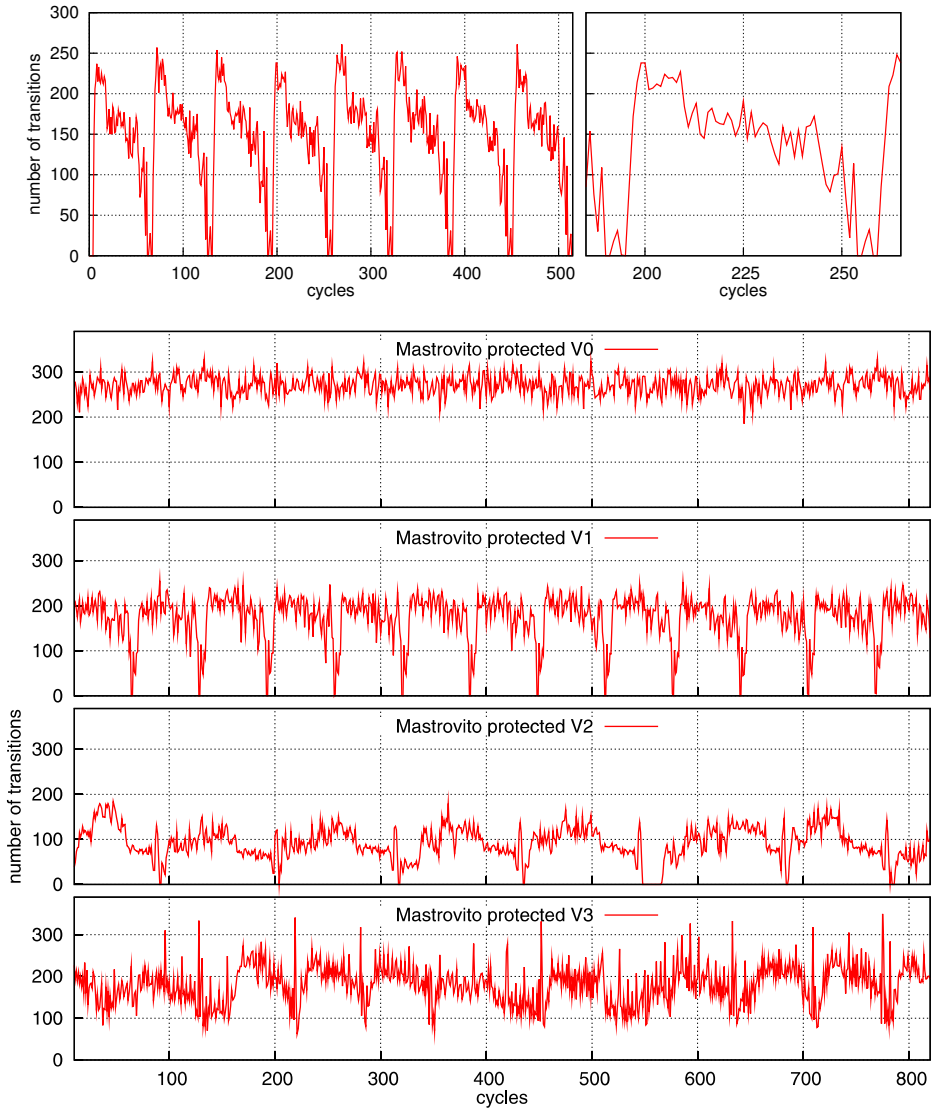
z innymi przedstawiono jego pełną wersję. Każde mnożenie zawiera końcową konwersję wyniku z reprezentacji Montgomery'ego.

W wypadku tego rozwiązania można zaobserwować spadek aktywności pod koniec operacji mnożenia. Stwierdzono, że spadek ten spowodowany jest operacją konwersji z reprezentacji Montgomery'ego i działaniem kontrolera układu mnożącego, który przed podaniem nowych wartości argumentów zatrzymuje

pracę urządzenia. W drugiej części Rysunku 4 (dolnej części) przedstawione są dwa wykresy aktywności, jeden dla układu wykonującego pełne mnożenia ale z ulepszonym kontrolerem (improved control), a drugi z ulepszonym kontrolerem i z pominięciem konwersji z reprezentacji Montgomery'ego (without reduction). Jeżeli nie bierzemy pod uwagę konwersji z reprezentacji Montgomery'ego, czyli działamy zgodnie ze wszystkimi założeniami algorytmu Montgomery'ego, wtedy wariacje aktywności algorytmu są dosyć jednolite. Wykres aktywności nie zawiera drastycznych spadków lub wzrostów aktywności, które pomogłyby atakującemu w zlokalizowaniu/rozróżnieniu kolejnych operacji.

Rozwiązanie wykorzystujące właściwości macierzy Mastrovito Rysunek 5 (górny) przedstawia wykres aktywności układu mnożącego korzystającego z właściwości macierzy Mastrovito.
Jak można zaobserwować kolejne mnożenia mają bardzo charakterystyczny "schodkowy" kształt, który może dostarczyć atakującemu wielu przydatnych informacji. Należało tak zmodyfikować architekturę układu aby wyeliminować ten specyficzny kształt. Ze względu na zaprojektowaną architekturę układu (podział na podmacierze) zaproponowano dwa cele modyfikacji: uniformizację (ujednolicenie) kształtu aktywności dla poszczególnych mnożeń i randomizację kształtu poszczególnych mnożeń, sprawienie że każde mnożenie będzie posiadało względnie różny kształt aktywności. W pierwszym przypadku dąży siąo tego aby trudno było zlokalizować poszczególne operacje, a w drugim aby atakujący nie był w stanie powiedzieć jaka operacja jest w danym momencie wykonywana, czy jest to mnożenie czy może inna operacja.

Oczywistym jest, że schodkowy kształt aktywności wynika z nierównej liczby przełączanych rejestrów w jednostce czasu.Uniformizacja ma na celu wyrównanie liczby bitów przełączanych w jednostce czasu, natomiast randomizacja zakłada przełączanie losowej ilości bitów w jednej jednostce czasu. Przeprowadzono wiele różnych modyfikacji aby skutecznie zabezpieczyć operatory. Tutaj prezentujemy najciekawsze wyniki. Najlepszy wynik uzyskany dla uniformizacji bez modyfikowania automatu kontrolującego przedstawiony jest na Rysunku 5, krzywa V1. Po modyfikacji działania kontrolera i zaproponowaniu kolejnego sposobu na uniformizację uzyskano wykres aktywności V0. W wyniku

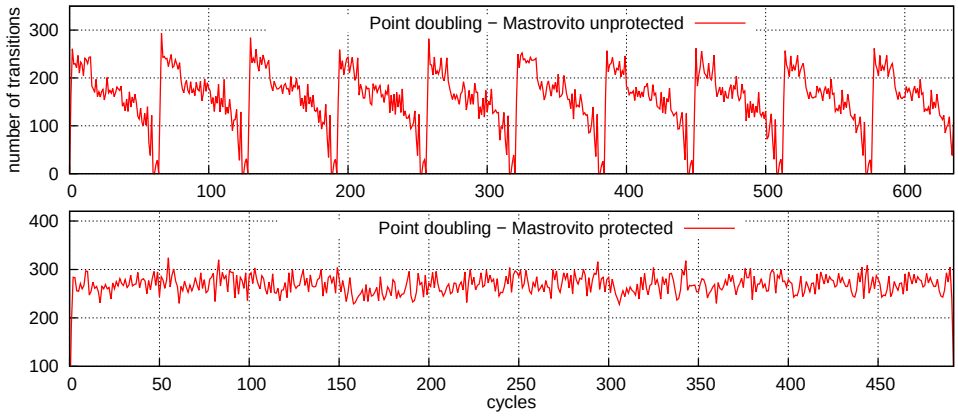


Rysunek 5.: Wykres użytecznej aktywności układu mnożącego wykorzytującego macierz Mastrovito (4 wersje).

zmian dokonanych w celu randomizacji uzyskano wykresy V2 i V3. W wyniku randomizacji udało się uzyskać zmienne czasy wykonywania operacji mnożenia. Dla wersji o wykresie V2, minimalną liczbą cykli zegarowych jest 98,

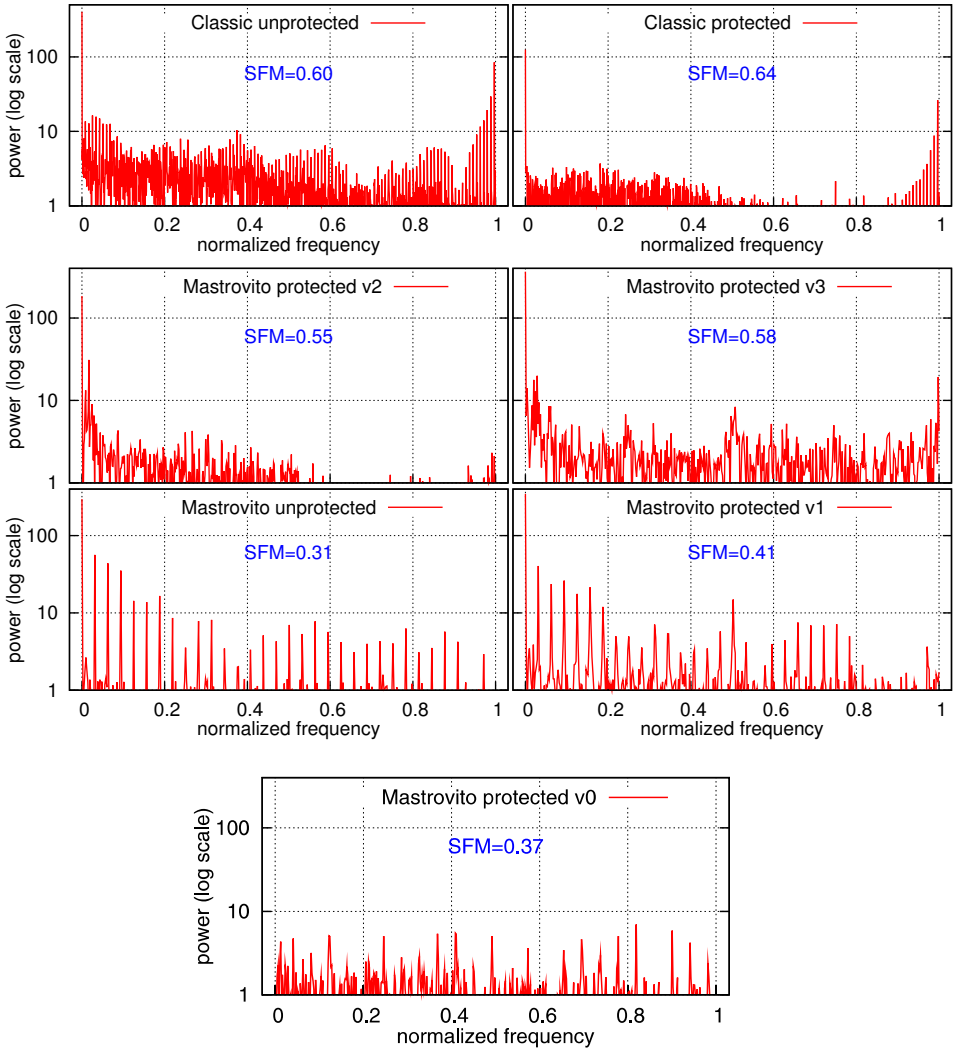
a maksymalną 126. W przypadku wersji V3, minimalna liczba cykli niezbędna do wykonania operacji to 64, a maksymalna 108.

Przeprowadzone modyfikacje miały na celu zamaskowanie charakterystycznych kształtów operacji mnożenia. Aby zweryfikować dodane zabezpieczeń przenalizowano kilka popularnych wariantów kryptoanalizy mocy prowadzących do uzyskania informacji o danych na jakich pracuje urządzenie. Przeanalizowano np. zależność pomiędzy kształtem krzywej aktywności, a zmianą jednego lub kilku bitów danych na których operujemy, przeanalizowano rownież ciągi operacji niezbędne np. do wykonania operacji $2P$ lub $P + Q$. Wyniki analizy dla niezabezpieczonego i zabezpieczonego operatora przedstawione są na Rysunku 6.



Rysunek 6.: Wykres użytecznej aktywności dla operacji $2P$ dla niezabezpieczonego i zabezpieczonego operatora mnożenia

Dodatkowo wykorzystano narzędzia używane w dziedzinie przetwarzania sygnałów (analiza FFT). Przy użyciu transformacji Fouriera dokonano przejścia z dziedziny czasu do dziedziny częstotliwości i na podstawie otrzymanych danych wyznaczono tzw. współczynnik SFM [21], czyli miarę płaskości widma. Rysunek 7 prezentuje otrzymane wartości. Im SFM bliższy wartości 1 tym bardziej równomiernie rozłożona jest moc widma. Obserwując otrzymane wyniki można zauważyć znaczną poprawę równomierności rozłożenia mocy np. dla ulepszonego układu Mastrovito.

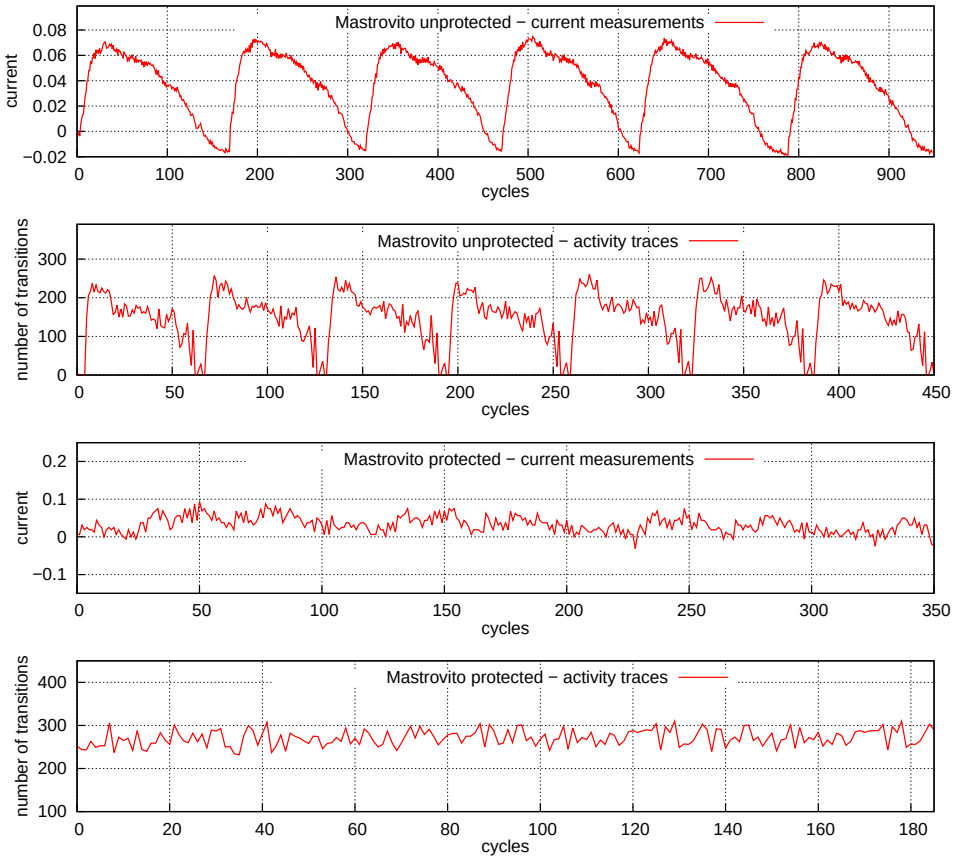


Rysunek 7.: Analiza FFT użytecznej aktywności dla zaproponowanych niezabezpieczonych i zabezpieczonych operatorów mnożenia

Porównanie otrzymanych wykresów aktywności z pomiarami prądowymi

Ostatnim krokiem w weryfikacji dodanych zabezpieczeń było porównanie ich z pomiarami prądowymi urządzeń. Układy mnożące zaimplementowano w układzie Virtex-II Pro zamontowanym na płycie SASEBO-G i przy użyciu szybkiego os-

cyloskopu LeCroy Waverunner 104Xi-a i sondy prądowej Tektronix CT1 zmierzono prąd pobierany przez urządzenie podczas wykonywania mnożeń. Aby pomiary zawierały jak najmniej zakłóceń do zasilania płyty użyto niskoszumowy zasilacz HP E3610A. Rysunek 8 przedstawia porównanie uzyskanych pomiarów z wykresami aktywności. Pierwsze dwa wykresy (od góry) pokazują wyniki uzyskane dla operatora niezabezpieczonego, natomiast dwa dolne wykresy pokazują wyniki uzyskane dla operatora zabezpieczonego. Analizując otrzymane wyniki



Rysunek 8.: Porównanie wykresów aktywności i pomiarów prądowych dla układu wykorzystującego macierz Mastrovito w wersji: niezabezpieczonej – 5 kolejno wykonanych mnożeń i zabezpieczonej (uniformizacja) – 3 kolejno wykonane mnożenia

można stwierdzić, że zaproponowany sposób ewaluacji aktywności urządzenia

odzwierciedla jego rzeczywistą aktywność. W Tabeli 10 przdstawiamy wyniki implementacji zabezpieczonych operatorów. Dla ułatwienia porównania z niezabezpieczonymi operatorami, wprowadzono współczynnik α: `zabezpieczony` $=$ $\alpha \times$ `niezabezpieczony`.

Tabela 10.: Wyniki implementacji zabezpieczonych operatorów mnożenia

Algorytm	zajętość LUT ($\times\alpha$)	f MHz ($\times\alpha$)	cykle zegarowe ($\times\alpha$)
Klasyczny	2868 ($\times$0.79)	270 ($\times$0.89)	260 ($\times$0.98)
Montgomery	2099 ($\times$0.96)	323 ($\times$1.00)	264 ($\times$0.98)
Mastrovito v0	3889 ($\times$1.04)	225 ($\times$0.75)	48 ($\times$0.64)
Mastrovito v1	3463 ($\times$1.09)	414 ($\times$1.39)	75 ($\times$1.00)
Mastrovito v2	3700 ($\times$1.02)	306 ($\times$1.03)	avg. 116 ($\times$1.55)
Mastrovito v3	3903 ($\times$1.03)	319 ($\times$1.07)	avg. 80 ($\times$1.07)

4. Podsumowanie i wnioski

W prowadzonych badaniach zajmowano się sprzętowymi układami wykonywującymi operacje arytmetyczne w ciele $GF(2^m)$. Badania miały na celu zaproponowanie wydajnych sprzętowych operatorów arytmetycznych odpornych na ataki typu side-channel polegające na kryptoanalizie mocy.

Pierwszym celem było opracowanie wydajnych, dedykowanych dla układów rekonfigurowalnych (FPGA) i dla aplikacji kryptograficznych (dla systemów ECC) układów arytmetycznych wykonywujących operacje w ciałach $GF(2^m)$. W tym celu przeprowadzono szeroką analizę istniejących rozwiązań podczas której zaimplementowano i zweryfikowano wiele istniejących rozwiązań i optymalizacji. Założono, że operatory powinny być szybkie, niewielkie (zajmować rozsądną ilość zasobów sprzętowych) oraz operować na bardzo dużych liczbach (150–600 bitów). Przeprowadzone badania pozwoliły na znalezienie, połączenie i ulepszenie takich cech istniejących rozwiązań, które pozwoliły na opracowanie operatorów zgodnie z założeniami.

Drugim celem było zabezpieczenie opracowanych operatorów arytmetycznych

przed bardzo popularnymi obecnie atakami typu side-channel polegającymi na analizie mocy pobieranej przez urządzenie. Zgodnie ze znanymi źródłami literaturowymi jest to pierwsza próba zabezpieczania operacji na najniższym poziomie systemu ECC, czyli operacji wykonywanych na elementach ciał skończonych, nad którymi zdefiniowane są krzywe wykorzystywane w systemie. Do tej pory zabezpieczenia dodawano na wyższym poziomie operacji systemów ECC, czyli na poziomie operacji na punktach krzywych eliptycznych.

Potencjalne źródła wycieku informacji zostały zidentyfikowane na dwa sposoby: poprzez wykorzystanie tzw. monitorów aktywności i narzędzia ChipScope oraz poprzez pomiar prądu chwilowego pobieranego przez urządzenie przy użyciu płyty SASEBO-G i specjalistycznego sprzętu pomiarowego. W wyniku analizy otrzymanych pomiarów stwierdzono bardzo specyficzne kształty aktywności urządzenia pozwolające wyznaczyć liczbę wykonanych operacji mnożenia lub zidentyfikować operację $2P$ lub $P + Q$. Aby zredukować wyciek informacji wprowadzono modyfikacje algorytmiczne i strukturalne w zaproponowanych urządzeniach. Zrobiono to na bardzo niskim poziomie aby uniknąć degradacji wydajności urządzenia.

Trzecim celem realizowanym równolegle z drugim było znalezienie kompromisu pomiędzy wydajnością, a bezpieczeństwem urządzenia. W momencie gdy dodane zabezpiecznie zbyt negatywnie wpływało na wydajność operatora, przebudowywano je lub rezygnowano z niego. Tabela 10 prezentuje uzyskane ostateczne parametry zabezpieczonych operatorów.

Podsumowując, w wyniku prowadzonych badań osiągnięto następujące oryginalne wyniki:

- zaproponowano **wydajne sprzętowe układy realizujące operacje arytmetyczne w ciele** $GF(2^m)$ dedykowane dla systemów ECC ;
- zaproponowano **skuteczne sposoby zabezpieczania** stowrzonych układów przed atakami typu side-channel polegającymi na kryptoanalizie mocy;
- znaleziono **kompromis** pomiędzy rozmiarami i skutecznością dodanych zabezpieczeń, a ich wpływem na wydajność zaproponowanych układów arytmetycznych,

które potwierdziły słuszność założonej tezy.

Bibliografia

[1] Ch. W. Chiou, J.-M. Lin, Ch.-Y. Lee, and Ch.-T. Ma. Novel Mastrovito Multiplier over $GF(2^m)$ Using Trinomial. In *Proc. 5th International Conference on Genetic and Evolutionary Computing*, ICGEC '11, pages 237–242, Washington, DC, USA, 2011. IEEE Computer Society.

[2] Xilinx Corporation. Virtex-6 family overview (product specification), 2012.

[3] F. Crowe, A. Daly, and W. Marnane. A scalable dual mode arithmetic unit for public key cryptosystems. In *Information Technology: Coding and Computing (ITCC)*, volume 1, pages 568–573, April 2005.

[4] S. S. Erdem, T. Yanik, and C. K. Koc. Polynomial Basis Multiplication over GF(2^m). *Acta Applicandae Mathematicae*, 93(1-3):33–55, September 2006.

[5] E. Ferrer, D. Bollman, and O. Moreno. A fast finite field multiplier. In *Proc. 3rd International Conference on Reconfigurable Computing: Architectures, Tools and Applications*, ARC'07, pages 238–246. Springer, 2007.

[6] A. P. Fournaris and O. Koufopavlou. Applying systolic multiplication-inversion architectures based on modified extended Euclidean algorithm for $GF(2^k)$ in elliptic curve cryptography. *Computers & Electrical Engineering, Elsevier*, 33(5-6):333–348, September 2007.

[7] P. Gallagher. FIPS PUB 186-3 Federal Information Processing Standards Publication Digital Signature Standard (DSS), 2009.

[8] C. Grabbe, M. Bednara, J. Teich, J. von zur Gathen, and J. Shokrollahi. FPGA designs of parallel high performance $GF(2^{233})$ multipliers [cryptographic applications]. In *Proc. International Symposium on Circuits and Systems (ISCAS)*, volume 2, pages 268–271, May 2003.

[9] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve*

Cryptography. Springer, 2004.

[10] A. Karatsuba and Y. Ofman. Multiplication of Multi-Digit Numbers on Automata (in Russian). *Doklady Akad. Nauk SSSR*, 145(2):293–294, 1962. Translation in Soviet Physics-Doklady, 44(7), 1963, p. 595-596.

[11] C. K. Koc and T. Acar. Montgomery Multiplication in GF(2^k). *Designs, Codes and Cryptography*, 14(1):57–69, April 1998.

[12] F. Koeune and F.-X. Standaert. A Tutorial on Physical Security and Side-Channel Attacks. In *FOSAD*, pages 78–108, 2004.

[13] H. Li, J. Huang, P. Sweany, and D. Huang. FPGA implementations of elliptic curve cryptography and Tate pairing over a binary field. *Journal of Systems Architecture, Elsevier*, 54(12):1077–1088, December 2008.

[14] S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards.* Springer, 2007.

[15] P. L. Montgomery. Modular Multiplication Without Trial Division. *Mathematics of Computation*, 44(170):519–521, April 1985.

[16] A. H. Namin, W. Huapeng, and M. Ahmadi. Comb Architectures for Finite Field Multiplication in $F(2^m)$. *IEEE Transactions on Computers*, 56(7):909–916, July 2007.

[17] D. Pamula, E. Hrynkiewicz, and A. Tisserand. Multiplication in $GF(2^m)$: area and time dependency/efficiency/complexity analysis. In *Proceedings of 10th International IFAC Workshop on Programmable Devices and Embedded Systems (PDES)* , pages 43–48, 2010.

[18] D. Pamula, E. Hrynkiewicz, and A. Tisserand. Analiza algorytmów mnozenia w ciele $GF(2^m)$. *Pomiary Automatyka Kontrola (PAK) (Measurement, Automation and Monitoring)* , 57(01/2011):58–60, 2011.

[19] D. Pamula, E. Hrynkiewicz, and A. Tisserand. Analysis of $GF(2^m)$ multipliers regarding Elliptic Curve Cryptosystem applications. In *Proceedings of 11th IFAC/IEEE International Conference on Programmable Devices and Embedded Systems (PDES)* , pages 252–257, 2012.

[20] D. Pamula and A. Tisserand. $GF(2^m)$ Finite-Field Multipliers with Reduced Activity Variations. *WAIFI 2012, LNCS 7369. Springer*, pages 152–167.

[21] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing.* Prentice Hall, 1996.

[22] S. Radack. Guide to protecting personally identifiable information. ITL Bulletin for April 2010.

[23] Research Center for Information Security National Institute of Advanced Industrial Science and Technology. Side-channel Attack Standard Evaluation Board. SASEBO-G. Specification. Version 1.0 , 2008.

[24] F. Rodríguez-Henríquez, N. A. Saqib, and A. Díaz-Pérez. A fast parallel implementation of elliptic curve point multiplication over $GF(2^m)$. *Microprocessors and Microsystems*, 28(5-6):329–339, 2004.

[25] B. Schneier. *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C.* John Wiley & Sons, Inc., New York, USA, 1995.

[26] A. Tisserand. Fast and Accurate Activity Evaluation in Multipliers. In *Proc. 42nd Asilomar Conference on Signals, Systems and Computers*, pages 757–761, Pacific Grove, California, U.S.A., October 2008. IEEE.

[27] S. Vanstone. ECC holds key to next generation cryptography. *[Online].Available: http://www.design-reuse.com/articles/7409/ecchold-key-to-next-gen-cryptography.html*, March 2006.

[28] J. Wang and A. Jiang. A high-speed dual field arithmetic unit and hardware implementation. In *ASIC, 2007. ASICON '07. 7th International Conference on*, pages 213–216, October 2007.

[29] J. Wolkerstorfer. Dual-Field Arithmetic Unit for $GF(p)$ and $GF(2^m)$. *CHES 2002, LNCS 2523. Springer*, pages 500–514.

[30] H. Wu, M. A. Hasan, I. F. Blake, and S. Gao. Finite field multiplier using redundant representation. *IEEE Transactions on Computers*, 51(11):1306–1316, November 2002.

[31] Xilinx Corporation. Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet (Product Specification), 2007.

[32] Xilinx Corporation. Spartan-3E FPGA Family: Data Sheet (Product Specification), 2009.