WOJSKOWA AKADEMIA TECHNICZNA

**im. Jarosława Dąbrowskiego**

WYDZIAŁ CYBERNETYKI



##### PRACA DYPLOMOWA

###### STACJONARNE STUDIA I STOPNIA

Temat: **SPRZĘTOWA IMPLEMENTACJA WYBRANYCH ALGORYTMÓW MNOŻENIA W PIERŚCIENIACH Zn ORAZ ANALIZA ICH PRZYDATNOŚCI W PEWNYCH ZASTOSOWANIACH KRYPTOGRAFICZNYCH**

|  |  |
| --- | --- |
| Autor: | Kierownik pracy: |
| **Michał DASZCZUK** | **dr inż. Piotr BORA** |

W a r s z a w a 2015

Spis treści

[Wstęp 6](#_Toc438040269)

[Rozdział I. Podstawowe własności pierścieni Zn. 7](#_Toc438040270)

[1.1 Grupy 7](#_Toc438040271)

[1.2 Pierścienie 7](#_Toc438040272)

[1.3 Relacja przystawania modulo n 8](#_Toc438040273)

[1.5 Pierścienie Zn 9](#_Toc438040274)

[Rozdział II. Zastosowania pierścieni Zn w wybranych algorytmach kryptograficznych. 13](#_Toc438040275)

[2.1 Algorytm Rabin 13](#_Toc438040276)

[2.2 Algorytm Blum Blum Shub 14](#_Toc438040277)

[2.3 RSA 15](#_Toc438040278)

[Rozdział III. Sprzętowe algorytmy mnożenia w pierścieniach Zn. 18](#_Toc438040279)

[3.1 Modularne mnożenie klasyczne 18](#_Toc438040280)

[3.2 Modularne mnożenie Montgomery’ego 20](#_Toc438040281)

[3.3 Mnożenie Montgomery’ego z wykorzystaniem reprezentacji RNS (Residue Number System) 21](#_Toc438040282)

[Rozdział IV. Opis implementacji w strukturach programowalnych wybranego algorytmu kryptograficznego wykorzystującego mnożenie w pierścieniach Zn. 23](#_Toc438040283)

[Rozdział V. Analiza uzyskanych wyników. 24](#_Toc438040284)

[Zakończenie 25](#_Toc438040285)

[Bibliografia 26](#_Toc438040286)

# Wstęp

Ciągły rozwój techniki i komputeryzacja każdego z aspektów naszego życia ciągną za sobą potrzebę zabezpieczenia przekazywanych informacji. Z powodu szybkiego rozrastania się sieci Internet oraz dużej ilości wymienianych informacji przez otwarte sieci, systemy kryptograficzne zaczęły odgrywać szczególną rolę w życiu każdego użytkownika. Zaistniała konieczność zabezpieczenia wielu podstawowych działań jak na przykład komunikacja, płatności czy transfery danych. Ze względu na dużą liczbę operacji, jakich algorytmy kryptograficzne wymagają, często najefektywniejszym rozwiązaniem jest ich implementacja w strukturach programowalnych, np. FPGA.

Wiele współczesnych systemów kryptograficznych wykorzystuje operacje modularne na dużych liczbach całkowitych. Ich implementacja w układach programowalnych stawia przed projektantem wiele wyzwań optymalizacyjnych związanych z ograniczeniami sprzętowymi, ale także rodzi możliwości związane ze zwiększeniem ich wydajności.

Celem niniejszej pracy jest charakterystyka, opis implementacji oraz analiza przydatności wybranych sprzętowych algorytmów mnożenia w pierścieniach w zastosowaniach kryptograficznych.

# Rozdział I. Podstawowe własności pierścieni Zn.

Zacznijmy od zdefiniowania podstawowych pojęć niezbędnych do określenia pojęcia pierścienia .

## 1.1 Grupy

**Definicja 1.1.** Struktura algebraiczna jest nazywana grupą jeśli działanie w zbiorze spełnia następujące warunki:

1. łączność
2. istnieje element neutralny taki, że
3. istnieje element odwrotny (przeciwny) dla każdego elementu w

Jeżeli dodatkowo działanie jest:

1. przemienne  
   to jest grupą przemienną (abelową).

## 1.2 Pierścienie

**Definicja 1.2.** Struktura algebraiczna jest nazywana pierścieniem, jeśli działanie addytywne oraz działanie multiplikatywne w zbiorze spełniają następujące warunki:

1. jest grupą abelową (grupa addytywna pierścienia ),
2. działanie jest łączne,
3. działanie jest rozdzielne względem działania +

Jeśli dodatkowo:

1. działanie jest przemienne, to określa się jako pierścień przemienny,
2. istnieje element neutralny działania (jedynka pierścienia )

to nazywamy pierścieniem z jedynką.

**Twierdzenie 1.1.** Niech będzie pierścieniem. Wtedy:

1. element neutralny jest unikalny,
2. element neutralny działania (zero pierścienia ) spełnia zależność:
3. elementy pierścienia oraz elementy do nich przeciwne spełniają zależność:

oraz

1. dla dowolnej wartości skalarnej zachodzi zależność:

## 1.3 Relacja przystawania modulo n

W zbiorze liczb całkowitych wprowadzamy relację przystawania modulo zdefiniowaną jako:

**Twierdzenie 1.2.** Relacja przystawania modulo jest relacją równoważności.  
Dowód.  
Niech . Wtedy:

1. zwrotność  
   ponieważ
2. symetria  
   wtedy
3. przechodniość  
   wtedy stąd odejmując od siebie równania otrzymujemy

Relacja przystawania modulo jest relacją równoważności, zatem dzieli zbiór liczb całkowitych na klasy abstrakcji (równoważności). Klasy równoważności są postaci

a zbiór klas abstrakcji to

## 1.5 Pierścienie Zn

Po ogólnym scharakteryzowaniu pierścieni możliwe jest wprowadzenie pojęcia pierścienia .

**Twierdzenie 1.3.** Struktura algebraiczna , gdzie

jest pierścieniem przemiennym z jedynką.

Dowód.  
Niech wtedy dla działania zachodzą następujące własności:

1. przemienność  
   gdzie - reszta z dzielenia modulo
2. łączność  
   Analogicznie  
   stąd
3. istnieje element neutralny
4. istnieje element przeciwny

oraz  
zatem

Stąd struktura jest grupą abelową.

Następnie dla działania zachodzą następujące własności:

1. przemienność
2. łączność
3. rozdzielność działania względem działania
4. istnieje element neutralny (jedynka pierścienia)

Z powyższych własności wynika, że struktura algebraiczna jest pierścieniem przemiennym z jedynką.

# Rozdział II. Zastosowania pierścieni Zn w wybranych algorytmach kryptograficznych.

Pierścienie ze względu na swoje własności zaprezentowane w poprzednim rozdziale znalazły zastosowanie głównie w algorytmach kryptograficznych opartych na problemie faktoryzacji i problemach pokrewnych.

## 2.1 Algorytm Rabin

Algorytm Rabin (patrz [1]) opiera swoje bezpieczeństwo na trudności obliczenia pierwiastka kwadratowego modulo liczba złożona, ponieważ nie istnieje efektywna metoda jego obliczania jeżeli nie znamy czynników pierwszych. Problem jest równoważny z problemem faktoryzacji, gdyż atakujący musi rozłożyć moduł na czynniki.

W celu zaszyfrowania wiadomości należy najpierw wygenerować dwie duże liczby pierwsze  i  kongruentne do 3 modulo 4. Liczby te są kluczem prywatnym, a ich iloczyn jest kluczem publicznym.

Sam proces szyfrowania polega tylko na podniesieniu wiadomości do kwadratu:

Deszyfrowanie wiadomości jest trochę bardziej skomplikowane. Znając tajne czynniki  i  odbiorca jest w stanie obliczyć wartość M korzystając z Chińskiego Twierdzenia o Resztach.

Wtedy obliczamy dwie dodatkowe wartości:

Następnie otrzymujemy cztery możliwe rozwiązania:

Jedno z rozwiązań jest wynikiem i jest równe .

Podstawową i bardzo mocną zaletą algorytmu Rabin jest efektywność szyfrowania i łatwość implementacji. Deszyfracja ma złożoność zbliżoną do RSA, wadą która uniemożliwia wprowadzenie szyfru do użycia jest konieczność rozpoznania, który z wyników jest pożądaną wiadomością. Jeżeli przesyłamy zaszyfrowany tekst weryfikacja jest mniej problematyczna, ale w przypadku przesyłania losowych ciągów bitów (np. klucza) nie jest możliwe odróżnienie go od fałszywych rozwiązań.

## 2.2 Algorytm Blum Blum Shub

Blum Blum Shub jest to generator liczb pseudolosowych (patrz [1]), którego działanie polega na podnoszeniu do kwadratu pewnej wartości początkowej modulo i zwracaniu najmniej znaczącego bitu wyniku jako bit ciągu losowego.

Na początku należy wybrać dwie duże liczby pierwsze i  kongruentne do 3 modulo 4 oraz obliczyć ich iloczyn . Następnie wybrać losową wartość względnie pierwszą z . Wartość

jest ziarnem tego generatora.

Kolejne bity oblicza się następująco:

Jedną z największych zalet tego generatora jest możliwość wyznaczenia -tego bitu wyjścia bez konieczności wykonywania iteracji. Znając współczynniki i  jest to możliwe przy pomocy następującej równości:

Ta własność oznacza, że generator liczb pseudolosowych Blum Blum Shub może być używany również jako silny kryptosystem strumieniowy.

Algorytm jest wolny, dlatego znajduje małą użyteczność wśród szyfrów strumieniowych, natomiast świetnie się sprawdza w zastosowaniach takich jak generacja kluczy, ponieważ atakujący nie znając czynników modułu nie jest w stanie przewidzieć kolejnych wartości bitów wyjściowych.

## 2.3 RSA

Asymetryczny algorytm z kluczem publicznym używany zarówno do szyfrowania, jak i do tworzenia podpisów cyfrowych. Do dzisiaj jest najbardziej popularnym algorytmem, nie tylko z powodu jego prostoty, ale także łatwości implementacji.

Bezpieczeństwo algorytmu opiera się na trudności faktoryzacji dużych liczb. Aby wygenerować klucze publiczny i prywatny, należy wybrać dwie duże liczby pierwsze,  i , a następnie obliczyć ich iloczyn . Klucz szyfrujący należy wybrać, tak aby był względnie pierwszy z . Wtedy, w celu obliczenia klucza deszyfrującego , rozwiązujemy równanie modularne postaci:

a zatem

Otrzymane w ten sposób pary oraz są odpowiednio kluczem publicznym i prywatnym.

Szyfrowanie z użyciem algorytmu RSA polega na podzieleniu tekstu jawnego w postaci binarnej na bloki długości , których wartość liczbowa jest mniejsza niż . Szyfrogram jest tej samej postaci i otrzymuje się go przez obliczenie:

Aby odszyfrować wiadomość należy wykonać podobne działanie z kluczem prywatnym:

Powyższe równości są spełnione ponieważ:

stąd również wynika, że tekst mógłby zostać zaszyfrowany kluczem i odszyfrowany kluczem

Bezpieczeństwo RSA zależy głównie od rozmiaru wybranego modułu, aczkolwiek wiele zależy od wyboru kluczy. Najszybsza obecna metoda faktoryzacji, ogólne sito ciała liczbowego, posiada wystarczająco dużą złożoność, aby moduły rozmiaru 1024 bitów wciąż gwarantowały bezpieczeństwo większości operacji wykonywanych w publicznych sieciach. Ważne przy tym, aby czynniki RSA spełniały dwa dodatkowe warunki:

1. posiadają duży czynnik pierwszy,
2. [2].

Badania nad bezpiecznym rozmiarem klucza prywatnego, jakie przeprowadził M. Wiener (patrz [3]) i w latach późniejszych D. Boneh oraz G. Durfee (patrz [4]), wykazały, że klucz powinien być większy niż . Natomiast przy wyborze klucza publicznego należy zadbać, aby , ponieważ w przeciwnym wypadku wyciek ( - liczba bitów modułu ) najmłodszych bitów lub , prowadzi do odzyskania odpowiednio tych wartości. Również klucze , według [5], pozwalają na określenie dolnej granicy dla liczby najbardziej znaczących bitów, które należy odkryć, aby sfaktoryzować moduł w czasie wielomianowym.

Kolejną możliwość ataku daje wybór małej wartości (np. 3, 5 lub 7) w celu optymalizacji czasu szyfrowania (lub podpisu). Wtedy możliwy jest atak Hastad’a korzystający z Chińskiego Twierdzenia o Resztach ujawniający zaszyfrowaną wiadomość. Innym podobnym atakiem na klucz jest atak Franklin’a-Reiter’a, w którym konieczna jest znajomość liniowej zależności pomiędzy wiadomościami postaci dla pewnej znanej funkcji (na podstawie [6]).

Kryptosystem RSA mimo wielu lat ciągłych prób złamania wciąż cieszy się niesłabnącą popularnością, a dobra implementacja gwarantuje jego bezpieczeństwo.

# Rozdział III. Sprzętowe algorytmy mnożenia w pierścieniach Zn.

Kryptografia jest fundamentalnym elementem zabezpieczania przepływu informacji. Jednak algorytmy kryptograficzne wymagają ogromnej mocy obliczeniowej, która może być wąskim gardłem całej komunikacji. Stąd wynika potrzeba osiągnięcia jak największej szybkości przetwarzania danych. Ponadto, aby nadążyć za różnorodnością i szybkimi zmianami wśród algorytmów oraz standardów, musi istnieć łatwy sposób aktualizacji zaimplementowanego rozwiązania. Najlepszym wyjściem jest użycie adaptatywnego procesora, który mógłby zapewnić elastyczność oprogramowania i wydajność sprzętową.

Technologia FPGA (Field-Programmable Gate Array) rozwiązuje postawione problemy, oferując wydajność porównywalną z układami ASIC oraz elastyczność procesorów. Tworzy to możliwość produkcji układów sprzętowych dedykowanych dla aplikacji, umożliwiających łatwą integrację oraz spełniających wymagania obliczeniowe [7].

Potrzeba implementacji systemów kryptograficznych w postaci modułów sprzętowych przyczyniła się również do konieczności rozwoju algorytmów dedykowanych dla środowisk sprzętowych. Zostały przy tym zoptymalizowane tak, aby jak najlepiej wykorzystywać możliwości oferowane przez te urządzenia. W niniejszym rozdziale zostaną opisane sprzętowe algorytmy mnożenia w pierścieniach Zn.

## 3.1 Modularne mnożenie klasyczne

Algorytm mnożenia modularnego wykorzystujący proste przekształcenie zwane schematem Hornera.

Niech będą dwiema dużymi liczbami całkowitymi mniejszymi od modułu , oraz  podstawą systemu liczbowego. Ponadto liczby będą długości słów rozmiaru . Wtedy

Następnie posługując się schematem Hornera otrzymujemy

Przy założeniu, że i ,  
. Z powyższego wynika następujący algorytm.

**Algorytm 3.1 Mnożenie z przeplotem mnożenie-redukcja**



Wartość można oszacować korzystając np. z metody Barreta, która wykorzystuje następujące przekształcenia:

gdzie jest liczbą bitów modułu , a wartość jest obliczona wcześniej [8].

W przypadku algorytm upraszcza się do postaci:

3. .

## 3.2 Modularne mnożenie Montgomery’ego

Metoda [8] obliczania dla dużych liczb całkowitych mniejszych od , która redukuje czas wykonywania dużej liczby mnożeń modularnych z tym samym modułem przy małej liczbie czynników.

Algorytm w zasadzie polega na zamianie redukcji modulo na redukcję modulo , gdzie i . Ponieważ R jest potęgą 2, dzielenie modulo upraszcza się do przesunięć binarnych. Wykorzystywana jest własność

Przed wykonaniem mnożenia należy wykonać tzw. transformację Montgomery’ego czynników:

oraz dodatkowo obliczyć wartość pomocniczą .

Następnie otrzymane czynniki należy zapisać w postaci słów rozmiaru ,  
 oraz wykonać kroki algorytmu 3.2 opisanego poniżej.

**Algorytm 3.2 Mnożenie Montgomery’ego**



Kroki algorytmu wynikają z następujących przekształceń [8]:

Korzystając z zależności

otrzymujemy  
Wtedy

## 3.3 Mnożenie Montgomery’ego z wykorzystaniem reprezentacji RNS (Residue Number System)

Najszybsze znane implementacje algorytmu RSA opierają się na systemie RNS, tzn. systemie reprezentacji liczb zgodnie z bazą wzajemnie względnie pierwszych modułów rozmiaru , gdzie jest licznością bazy.

Liczba całkowita reprezentowana przez ciąg słów , gdzie  
. Chińskie Twierdzenie o Resztach zapewnia unikalność takiej reprezentacji dla każdego w przedziale , gdzie . Konwersja do pierwotnej postaci polega na obliczeniu:

gdzie .

Z powyższej reprezentacji wynikają bezpośrednie korzyści związane z możliwością wykonywania operacji dodawania, odejmowania i mnożenia równolegle ze stałą złożonością czasową. Niech będą reprezentowane w RNS odpowiednio przez , wtedy:

Jedyną wadą tej reprezentacji jest brak możliwości wykonywania porównań, stąd również problem z dzieleniem i wykrywaniem przepełnień. Operacje kryptograficzne wykonywane w pierścieniu lub ciele skończonym eliminują problem przepełnień, a opisane w poprzednim podrozdziale mnożenie Montgomerego rozwiązuje kwestię trudności dzielenia [10].

Algorytm modularnego mnożenia Montgomerego z użyciem reprezentacji liczb w RNS jest obecnie najszybszym rozwiązaniem sprzętowym mnożenia w pierścieniach . Niniejsza praca nie obejmuje implementacji sprzętowej mnożenia w RNS ze względu na jego skomplikowanie (algorytm ten znaczenie wykracza poza zakres studiów inżynierskich), stąd jego opis został pominięty. Szczegóły algorytmu można znaleźć w [10], [11].

# Rozdział IV. Opis implementacji w strukturach programowalnych wybranego algorytmu kryptograficznego wykorzystującego mnożenie w pierścieniach Zn.

# Rozdział V. Analiza uzyskanych wyników.

# Zakończenie

# Bibliografia

|  |  |
| --- | --- |
| [1] | B. Schneier, "Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C", New York: John Wiley & Sons, Inc., 1996. |
| [2] | A. Chmielowiec, „Ataki na RSA,” Centrum Modelowania Matematycznego Sigma. |
| [3] | M. J. Wiener, „Cryptanalysis of short RSA secret exponents,” *IEEE Transactions on Information Theory,* pp. 553-558, 1990. |
| [4] | D. Boneh i G. Durfee, „Cryptanalysis of RSA with private key d less than N^0,292,” *IEEE Transactions on Information Theory,* pp. 1339-1349, 2000. |
| [5] | M. Ernst, E. Jochemsz, A. May i B. de Weger, „Partial key exposure attacks on RSA up to full size exponents,” *Proc. of Eurocrypt,* pp. 371-386, 2005. |
| [6] | J. Wang, „Attacks aganist RSA Cryptosystems in Thirty Years,” 2011. |
| [7] | V. Prasanna i A. Dandalis, „FPGA-based Cryptography for Internet Security,” Los Angeles. |
| [8] | H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen i F. Vercauteren, "Handbook of Elliptic and Hyperelliptic Curve Cryptography", Boca Raton: Taylor & Francis Group, 2006. |
| [9] | H. Warren, Jr., „Montgomery Multiplication,” 2012. |
| [10] | J.-C. Bajard, L.-S. Didier, P. Kornerup i F. Rico, „Some Improvements on RNS Montgomery Modular Multiplication”. |
| [11] | L. Imbert i J.-C. Bajard, „A Full RNS Implementation of RSA,” IEEE Transactions on Computers, 2002. |