Janusz JABŁOŃSKI
UNIWERSYTET ZIELONOGÓRSKI, WYDZIAŁ MATEMATYKI INFORMATYKI I EKONOMETRII

Efektywna implementacja podpisu cyfrowego opartego na RSA

Dr inż. Janusz JABŁOŃSKI

W latach 1994 – 2003 asystent i od 10.2003 adiunkt w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Od 2004 adiunkt Wydziału Matematyki, Informatyki i Ekonometrii UZ. Prowadzi zajęcia z Inżynierii Oprogramowania. Posiada również doświadczenie dydaktyczne w zakresie architektury komputerów oraz logiki programowalnej. Prowadzi badania związane z wykorzystaniem arytmetyki resztowej oraz układów FPGA w akceleracji obliczeń.

e-mail: J.Jablonski@wmie.uz.zgora.pl

Streszczenie

W prezentowanym opracowaniu zaproponowana została metoda poprawy efektywności przetwarzania dla kryptosystemu RSA również w realizacji podpisu cyfrowego. Rozwiązanie korzysta z zalet systemu resztowej reprezentacji liczb (ang. *Residue Number System*) jako metody prowadzącej do zrównoleglenia przetwarzania. Ponadto, analizowane są algorytmy potęgowania modularnego. W szczególności redukcji i mnożenia Montgomery'ego jako najefektywniejszych w implementacji programowej. Podkreślane są zalety implementacji algorytmów równoległych na komputerach PC wyposażonych w mikroprocesory wielordzeniowe. Efektem implementacji proponowanych rozwiązań jest wielokrotne przyspieszenie szyfrowania i podpisu cyfrowego korzystającego z RSA.

Abstract

In article was present method of improvement efficiency processing for RSA and digital signature - based at this scheme. Proposed method based at efficient solutions taken from Residue Number System (RNS) and variants of a Montgomery algorithm in effective of the modular reduction. The use RNS taken possibility to realize simultaneously processing in personal computers with multi-core microprocessors. This technology concept reduce disadvantages traditional methods of operation in cryptography and digital signature. Effect the implementation of proposed solution is the multiple of acceleration for RSA digital signature with this coding schema.

Słowa kluczowe: RNS, RSA, redukcja modularna, potęgowanie modularne, mnożenie Montgomery'ego

Keywords: RNS, RSA, modular reduction, Montgomery multiplication.

1. Wstęp

Bezpieczeństwo danych i ochrona informacji jest jednym z kluczowych elementów funkcjonowania we współczesnym zinformatyzowanym społeczeństwie. W kontekście upowszechnienia Internetu jako masowego środka komunikacji dla potrzeb transakcji handlowych przez transmisję satelitarną dla celów komercyjnych a na prostych komunikatorach kończąc wymagane są bezpieczne i efektywne mechanizmy przetwarzania informacji. Podpis cyfrowy i szyfrowanie informacji to uznane i skuteczne mechanizmy zabezpieczające zarówno dane jak również kanały komunikacyjne. Szczególnie interesujące w zakresie ochrony poufności czy też zabezpieczenia informacji przed niepowołanym dostępem są kryptosystemy asymetryczne zaproponowane w 1976r [1]. Uniwersalnym systemem kryptograficznym z kluczem publicznym jest RSA w 1978r zaproponowany również do podpisu cyfrowego [2]. Bezpieczeństwo RSA, jak również innych stosowanych w zabezpieczeniach systemów zabezpieczających opiera się na trudności faktoryzacji wielkich liczb [7]. Problem ten fascynuje badaczy od kilku tysiącleci i jak dotąd nie jest znany efektywny (wielomianowy) algorytm rozwiązujący to zagadnienie. Jednakże, zapewnienie wymaganego poziomu bezpieczeństwa wymaga stosowania coraz większych liczb obecnie jako bezpieczny uważany jest RSA z kluczem 2048bitów. Operacje arytmetyczne na tak dużych liczbach wymagają znaczących mocy obliczeniowa oraz zasobów pamięci. Niejednokrotnie, szczególnie w zastosowaniach komercyjnych wymagane jest szyfrowanie i deszyfrowanie informacji w czasie rzeczywistym. Wówczas, komercyjnie wdrażane rozwiązania korzystają z dedykowanych do obliczeń mikroprocesorów DSP (ang. *Digital Signal Processing*) lub specjalizowanych układów cyfrowych [3]. Są to rozwiązania efektywne lecz zbyt drogie dla większości niekomercyjnych użytkowników Internetu, którzy również wymagają poufności i zabezpieczania danych.

Obserwowane obecnie tendencje rozwoju rynku mikroprocesorów dla PC (ang. *Personal Computer*), do których zaliczane są również komputery przenośne, zmierzają w kierunku architektur wielordzeniowych. Taki kierunek rozwoju PC udostępnia nowe możliwości w zakresie zrównoleglenia przetwarzania szczególnie przetwarzania symetrycznego. Jednakże, efektywne wykorzystanie mocy obliczeniowych dostarczanych przez technologie wielordzeniowe wymaga opracowania metod i algorytmów przetwarzania podatnych na zrównoleglenie.

2. Efektywność implementacji RSA

Nazwa RSA pochodzi od nazwisk autorów opracowanego w 1977 roku przez Rona Rivesta, Adi Shamira i Leonarda Adlemana systemu kryptograficznego opartego na kluczu asymetrycznym. Może być on wykorzystywany w zakresie szyfrowania jak również w elektronicznym podpisywaniu dokumentów.

2.1. Analiza podstawowych operacji w RSA

Podstawowy schemat szyfrowania RSA przedstawia równanie:

$$C = M^e (\mathrm{mod} N), \tag{1}$$

gdzie C jest szyfrogramem wiadomości M - szyfrowanej kluczem publicznym $\{e, N\}$. Natomiast, N to iloczyn „dostatecznie" dużych – obecnie 2048bitowych, liczb pierwszych. W podstawowym schemacie czynnikami są dwie liczby pierwsze np. p i q. Losowo wybrana liczba $e < N$ i względnie pierwsza z $\varphi(N)$, w tym przypadku $\varphi(N) = (p\text{-}1)(q\text{-}1)$. Odtworzenie informacji M z szyfrogramu C wymaga deszyfrowania, opartego na analogicznych operacjach jak szyfrowanie, którego schemat przedstawia równanie:

$$M = C^d (\mathrm{mod} N) \tag{2}$$

gdzie, $de \equiv 1(\mathrm{mod}\varphi(N))$. Wartość d będąca odwrotnością multiplikatywną e modulo N podlega ścisłej ochronie jest to podstawowy element klucza prywatnego. W implementacji schematu szyfrowania wiadomość M dzielona jest na bloki wartości $m_i < N$, które są szyfrowane oddzielnie zgodnie ze schematem (1), jako:

$$c_i = m_i^e (\mathrm{mod} N) \tag{3}$$

Natomiast, tekst jawny z szyfrogramu otrzymywany jest na podstawie schematu deszyfrowania (2) w wyniku operacji:

$$m_i = c_i^d (\mathrm{mod} N) \tag{4}$$

Schematy (1, 2) oparte są na potęgowaniu modularnym jako operacji dominującej. Prosta lub inaczej klasyczna implementacja kryptosystemu RSA wymaga operacji potęgowania, lub wielokrotnego mnożenia zakończonego operacją dzielenia. Złożoność obliczeniowa mnożenia jak również potęgowania i dzielenia jest zależnością kwadratową względem rozmiaru bitowego argumentów. W kontekście realizacji operacji arytmetycznych są to czynności bardzo czasochłonne, szczególnie uwzględniając rozmiar argumentów rzędu 2000bitów. Klasyczna realizacja polegająca na wykonaniu potęgowania zakończonego dzieleniem, ze względu na rozmiar danych pośrednich, nie jest praktycznie wykorzystywana. Uwzględniając rozmiar argumentów wyniki przed redukcją modu-

lo zajmowałby aż (2048•2047)bitów ponad 4Mbit dla każdego bloku danych.

W praktyce języki programowania wyposażone są w biblioteki realizujące szyfrowanie oparte na schemacie RSA w oparciu o algorytm iterowanego podnoszenia do kwadratu i redukcji modulo wyników pośrednich – często nazywanego algorytmem binarnym potęgowania modularnego [5]. W algorytmie tym liczba operacji mnożenia jest proporcjonalna do ilości jedynek w binarnej reprezentacji danych, natomiast dzieleń jest tyle ile bitów wykładnika. Algorytm binarny jest znacząco efektywniejszy od algorytmu klasycznego, skraca znacząco czas realizacji operacji potęgowania modularnego jak również znacząco zmniejsza wymagania pamięciowe, jednakże główną wadą jest brak podatności na zrównoleglenie [6]. Tak, więc poprawy efektywności implementacji RSA można poszukiwać w trzech obszarach: po pierwsze w zrównoleglenia przetwarzania, po drugie w ograniczaniu liczby mnożeń oraz po trzecie w eliminacji czasochłonnej operacji redukcji modulo lub zastąpieniu czasochłonnego dzielenia innymi mniej złożonymi czasowo działaniami.

2.2. Zrównoleglenie w implementacji RSA

Jednym z możliwych sposobów zrównoleglenia przetwarzania na poziomie argumentów operacji arytmetycznych jest resztowy system reprezentacji liczb (ang. *Residue Number System*, RNS). Jest to alternatywny w stosunku do systemów stałobazowych sposób przedstawiania liczb, którego cechą charakterystyczną jest podatność na przetwarzanie symetryczne. W systemie resztowym liczbę reprezentuje wektor reszt z jej dzielenia przez odpowiednio dobrane stałe naturalne tworzące zbiór B = $\{m_1, m_2, \ldots m_j\}$ względnie pierwszych modułów, nazywanych bazą systemu resztowego. Zakresem reprezentowanych w RNS wartości jest przedział będący iloczynem modułów bazy B i wynosi $M = m_1 m_2 \ldots m_i$. W RNS o bazie B reprezentacją liczby $X < M$ jest j – elementowy wektor reszt $X_i = X(\mathrm{mod} m_i)$ z dzielenia tej liczby przez kolejne m_i moduły bazy B. Wynik operacji arytmetycznych $Z = X \otimes Y$ jest reprezentowany przez wektor reszt $Z_i=(X_i \otimes Y_i)$ modm_i, a działania arytmetyczne, takie jak: dodawanie, odejmowanie, mnożenie i dzielenie w RNS są wykonywane na resztach – niezależnie w poszczególnych kanałach resztowych m_i. Umożliwia to dekompozycję pojedynczego łańcucha cyfr reprezentujących liczbę na wektor wartości z mniejszego zakresu, które mogą być reprezentowane również przez łańcuchy cyfr, lecz o znacznie mniejszej długości (ponieważ m_i<<M). Długość łańcucha oraz czas realizacji operacji arytmetycznych jest zdeterminowany przez rozmiar modułów m_i. Równomierny rozłożenie obciążeń wymaga aby moduły bazy miały porównywalny rozmiar bitowy. Wówczas dekompozycja może prowadzić do skrócenia czasu wykonania działań arytmetycznych – również potęgowania modularnego. Warunkiem uzyskania poprawy efektywności jest dysponowanie zasobami umożliwiającymi zrównoleglenie obliczeń oraz symetryczne przydzielenie jednostek liczących do realizacji operacji w kanałach resztowych. Otrzymane wyniki wymagają konwersji do systemu stałobazowego, co jest realizowane na podstawie Chińskiego Twierdzenia o Resztach (ang. *Chinese Remainder Theory*, CRT) oraz algorytmu Garnera przedstawionego w [5].

2.3. Szybkie potęgowanie dla RSA

Krótka analiza schematu RSA zawarta w pkt. 2.1 pokazuje, iż potęgowanie jest podstawą operacją arytmetyczną wykonywaną w algorytmie i implementacja tej operacji ma kluczowe znaczenie dla efektywności kryptosystemu. Programowa implementacja może być oparta na klasycznym wielokrotnym mnożeniu, jest to jednak metoda nieefektywna i zasobochłonna. Znaczną redukcję liczby mnożeń można uzyskać implementując potęgowanie oparte na schemacie Hornera, które jest podstawą metod mnożenia opartych na wstępnym przeglądzie wykładnika – wielomianu będącego Binarną reprezentacją wartości wykładnika [8]. Metoda, ta jest często implementowana w bibliotekach operujących na dużych liczbach języków programowania (C++, Java) i określana jest jako algorytm potęgowania binarnego. Jest to „najprostszy" z algorytmów korzystających z metody skanowania wykładnika. W algorytmie sprawdzane są kolejne bity wykładnika i dla kolejnej wartość równej zero wykonywane jest podnoszenie do kwadratu podstawy, jeśli kolejny jest jedynką następuje podnoszenie do kwadratu i mnożenie. Dla n-bitowego wykładnika metoda ta redukuje liczby k mnożenia, gdzie k – liczba jedynek w wykładniku. Jednak metoda ta wymaga n operacji podnoszenia do kwadratu. Przy, czym operacja podnoszenia do kwadratu można wykonana dwa razy szybciej aniżeli mnożenie [5]. Usprawnieniem metody binarnej jest sprawdzanie w jednym kroku m – bitów i analogicznie jak w metodzie binarnej realizacji odpowiednich działań, jest to potęgowanie m– arne [5]. W metodzie przesuwającego się okna rozdzielane są sekwencje bitów o wartości zero i jeden, tworząc „okna" zer i jedynek, które w zależności od metody zakładają, iż liczba „okien" może być stała i stałej długości lub zmienna i różnej długości. Analizy przeprowadzona w [9] wykazuje, iż metoda o zmiennych „okien" – nazywana metodą przesuwającego się okna, wymaga średnio 5% mniej mnożeń aniżeli pozostałe metody oparte na wstępnej analizie bitów wykładnika.

2.4. Efektywna metody redukcji modularnej

Redukcja modularna jest najbardziej czasochłonną operacją w systemach kryptograficznych opartych na logarytmie dyskretnym. Klasyczna metoda realizacji polega na dzieleniu z resztą argumentów będących wynikiem potęgowania dużych liczb. Efektywniejszą metodą, jednakże również wymagającą czasochłonnego dzielenia, jest połączenie kroku w mnożeniu binarnym z dzieleniem jako redukcja modularna. Algorytmem nie wymagającym dzielenia jest algorytm redukcji Montgomery'ego [4], który jako dane wejściowe przyjmuje liczby całkowite m, $R=b^k$ oraz x, gdzie $R < m$ i $0 \le x < mR$, przy czym R i m są względnie pierwsze. Wstępnie wyznaczana jest wartość $-m_0^{-1}$mod$b$, gdzie b =2 jest podstawą systemu, więc operacja $x$mod2 może być zrealizowana jako przesunięcie logiczne o jedną pozycję bitową. Wynikiem redukcji jest liczba całkowita xR^{-1}mod$m$. Mnożenie Montgomery'ego przedstawia Algorytm 1 wynikiem $Mont(x,y)$ jest liczba całkowita xyR^{-1}mod$m$.

Algorytm 1. Funkcja *Mont*(*x*,*y*) Mnożenia Montgomery'ego

Dane: liczby $m=(m_{n-1}\ldots m_1 m_0)_b$, $x=(x_{n-1}\ldots x_1 x_0)_b$ oraz $y=(y_{n-1}\ldots y_1 y_0)_b$, takie, że: $0 \le x,y < m$, $R=b^n$, oraz $NWD(m,b)=1$ i $m'=-m^{-1}$mod$b$
Wynik: xyR^{-1}**mod**m

1. $A=0$;
2. for i=0 to (n-1) do
   2.1. $u_i = (a_0 + x_i y_0)m'$mod$b$;
   2.2. $A=(A + x_i y + u_i m)/b$;
3. if $A \ge m$ then $A = A - m$;
4. return(A)

Algorytm 2 to schemat potęgowania modularnego korzystający z funkcji $Mont(x,y)$ i przystosowany do algorytmu potęgowania binarnego.

Algorytm 2. Potęgowanie Montgomery'ego

Dane: $m=(m_{n-1}\ldots m_1 m_0)_b$, $R=b^n$, $m'=-m^{-1}$**mod**b,$e=(e_t\ldots e_1 e_0)$, $e_t=1$
liczba całkowita $1 \le x < m$
Wynik: x^e**mod**m

1. $x' = Mont(x,R^2 \mathrm{mod} m)$, $A=R$mod$m$;
2. for $i=t$ down to 0 do
   2.1 $A=Mont(A,A)$;
   2.2 if e_i=1 then $A=Mont(A,x')$;
3. $A=Mont(A,1)$;
4. return (A);

Aby zakończyć obliczenia i otrzymać wynik xymod$m$ należy otrzymaną liczbę przemnożyć przez wcześniej wyznaczoną stałą Rmod$m$.

Oprócz redukcji Montgomery'ego w RSA wykorzystywana jest redukcja Baretta [3]. Algorytm Redukcji Baretta przedstawiony

w [5] również nie wymaga dzielenia. Zarówno redukcja Montgomery'ego jak również redukcja Baretta wymagają dodatkowych wstępnych i końcowych operacji, dlatego nie mogą być efektywnie implementowane w jednostkowych działaniach modulo. Jednakże, w prezentowanych algorytmach potęgowania modularnego działania modulo realizowane są wielokrotnie – „seryjnie", wówczas czas wstępnych obliczeń nie jest tak istotny.

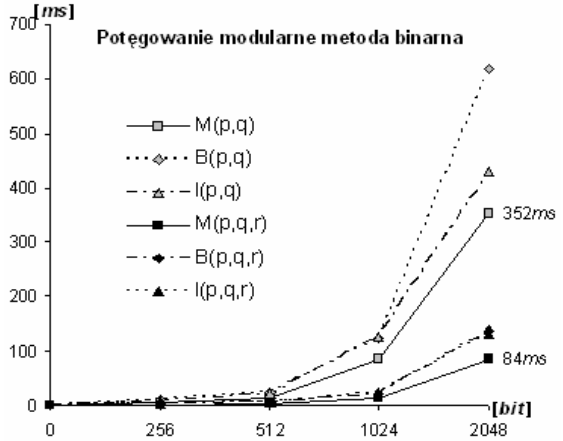
3. Analiza proponowanych rozwiązań

W tabeli 1 zestawione są wymagane operacje dla omawianych algorytmów potęgowania modularnego.

Tab. 1. Porównanie operacji realizowanych w algorytmach
Tab. 1. Algorithm with operations comparison

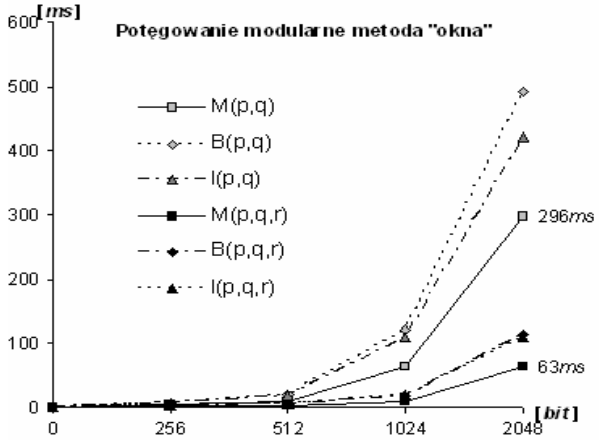
	Algorytmy		
	Klasyczny	Montgomery	Barrett
Ograniczenia	Brak	$X < mb^k$	$x < b^{2k}$
Mnożenia	$k(k+2)$	$K(k+1)$	$k(k+4)$
Dzielenia	K	0	0
Wstępne	normalizacja	$-m_0^{-1} \bmod b$	$\mu = \left\lfloor \frac{b^{2k}}{m} \right\rfloor$
Zakończenie	denormalizacja	Redukcja	Brak

W celu weryfikacji efektywności przedstawionych w pkt. 2 rozwiązań przygotowana została wielowątkowa aplikacja z biblioteką funkcji w języku Java. Dla każdego z algorytmów zaproponowane zostało rozwiązanie korzystające z dwu i trzymodułowego RNS.



Rys. 1. Implementacja RSA metodą binarną
Fig. 1 Binary metod in RSA implementation

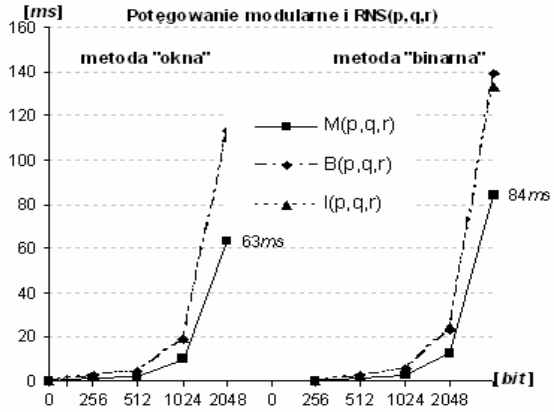
Dla stopnia zrównoleglenia równego dwa i trzy jako dwa i trzy moduły bazy RNS odpowiednio: *M* – redukcja Montgomery'ego, *B* - redukcja Baretta, *I* – algorytm iteracyjny oraz odpowiednio (*p*,*q*) – dwa moduły bazy RNS, (*p*,*q*,*r*) – trzy moduły bazy RNS.



Rys. 1. Implemntacja RSA metodą okna
Fig. 1 The window method in RSA implementation

Wyniki przedstawione na wykresie z rys. 1, 2 oraz 3 dotyczą odpowiednio uśrednionych wyników pomiaru czasu realizacji RSA. Badania zostały przeprowadzone w środowisku MS Windows XP-profesional z SP2, na przenośnym komputerze klasy PC z procesorem Centrino Duo T7400 i 2GB RAM.

Zbiorcze zestawienie porównawcze dla trzech modułów bazy przedstawia rys.3.



Rys. 1. Porównanie implemntacji RSA
Fig. 1 Comparison the RSA implementations

4. Podsumowanie

Korzystając w szyfrowaniu RSA ze standardowych bibliotek Java dla *N* rozmiaru 1024bitów i 16-to bitowego klucza wymagany jest czas około 250ms, natomiast deszyfrowanie wymaga około 500ms [10]. Wykorzystanie RNS jako metody zrównoleglenia obliczeń oraz mikroprocesorów wielordzeniowych jako narzędzia realizacji zaproponowanych algorytmów umożliwia wprowadzenia zrównoleglenia i przetwarzanie symetryczne. Takie, rozwiązanie pozwala na kilkukrotne skrócenie czasu przetwarzania w programowej realizacji szyfrowania RSA. Potwierdzają to wyniki przeprowadzonych testów (rys.3) wskazując, iż najkorzystniejsze rezultaty można osiągnąć implementując mnożenie Montgomer'ego połączone z metodą przesuwającego się okna, chociaż teoretyczne algorytm Barett'a oferuje najkorzystniejsze rozwiązanie. Otrzymane rezultaty pozwalają na szyfrowanie w czasie rzeczywistym transmisji 30kb/s za pomocą standardowego PC. Na podkreślenie zasługuje możliwość wykorzystania zalet arytmetyki resztowej. Dotychczas rozwiazania oparte na arytmetyce resztowej były implementowane układowo – wymagały powielenia jednostek wykonawczych, natomiast architektury wielordzeniowe w PC pozwalają na nowo odkrywać zalety RNS. W omawianym zastosowaniu można pokusić się o stwierdzenie, iż pozwala to popularnym komunikatorom na zaoferowanie telekonferencji szyfrowanej 2048-bitowym RSA.

5. Literatura

[1] Whitfield Diffie and Martin E. Hellman. *New directions in cryptography*. IEEE Transactions on Information Theory, 1976.
[2] R. Rivest, A. Shamir and L. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.* Communications of the ACM, February 1978.
[3] P.D. Barrett, "Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor," *Advances in Cryptology, Proc. Crypto'86, LNCS 263*, A.M. Odlyzko, Ed., Springer-Verlag, 1987.
[4] P.L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, Vol. 44, 1985.
[5] N. Ferguson, B. Schneier, Kryptografia w praktyce, Helion, Gliwice 2004
[6] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, Kryptografia stosowana, WNT, W-wa 2005
[7] N. Koblitz, Wykład z teorii liczb i kryptografii, WNT Warszawa 1995
[8] S.Y. Yan, Teoria liczb w informatyce, PWN Warszawa 2006
[9] Koç, Ç. K. (1994). High-speed RSA implementation, Version 2.0. *RSA Laboratories*, November 1994.
[10] http://www.securecottage.com/demo/rsa2.html

Title: Efficient implementations of digital signature based at RSA

Artykul recenzowany