Arytmetyka resztowa w szyfrowaniu RSA

Janusz Jabłoński

Streszczenie: W artykule przedstawiona została metoda poprawy efektywności szyfrowania *RSA*. Proponowane rozwiązanie korzysta z resztowej reprezentacji liczb (ang. *Residue Number System, RNS*) oraz konwersji z systemu resztowego do stałobazowego zaproponowanej przez Wang-a. *RNS* prowadzi do redukcji rozmiaru czynników oraz wprowadzenia zrównoleglenia przetwarzania na poziomie algorytmu. Natomiast *Małe Twierdzenie Fermata* zostało wykorzystane do redukcji wykładnika w schemacie *RSA*.

Słowa kluczowe: *RSA*, potęgowanie modularne, *RNS*.

1. WPROWADZENIE

Większość Informacji przechowywanych w systemach komputerowych lub przesyłanych za pośrednictwem infrastruktury teleinformatycznej ma swoją wartość lub chronione jest odpowiednimi przepisami. Dlatego coraz większą wagę przywiązuje się do bezpieczeństwa systemów informatycznych. W szczególności istotne jest zapewnienie poufności, nienaruszalności i integralności danych, ogólnie jest to ochrona przed nieuprawnionym dostępem. Najczęstszą metodą ochrony danych jest uczynienie informacji „nieczytelnej" dla nieuprawnionych do jej odczytania, w takich zastosowaniach podpis cyfrowy i szyfrowanie informacji to uznane oraz skuteczne mechanizmy zabezpieczające zarówno dane jak również kanały komunikacyjne. Szczególnie interesujące są systemy asymetryczne zaproponowane w 1976r. [1]. Takim systemem kryptograficznym z kluczem publicznym jest *RSA*, który w 1978r. zaproponowany został jako system podpisu cyfrowego [2]. Bezpieczeństwo *RSA*, jak również innych algebraicznych metod szyfrowania, opiera się na trudności rozwiązania problemu logarytmu dyskretnego oraz trudności faktoryzacji wielkich liczb [7]. Uwzględniając rozwój komputerów oraz doskonalenie metod kryptoanalizy zapewnienie wymaganego poziomu bezpieczeństwa systemu *RSA* wymusza stosowanie coraz bezpieczniejszych kluczy, co prowadzi do obliczeń na coraz większych liczbach. Aktualnie przyjmuje się, że szyfrowanie kluczem o rozmiarze 2048[bit] gwarantuje wystarczający poziom bezpieczeństwa. Obliczenia na tak dużych liczbach wymagają dużych mocy obliczeniowych oraz dużych pamięci. Szczególnie w zastosowaniach wymagających realizacji przetwarzania w czasie rzeczywistym.

2. EFEKTYWNOŚĆ IMPLEMENTACJI RSA

System *RSA* może być wykorzystywany w szyfrowaniu jak również w elektronicznym podpisywaniu dokumentów przy czym szybkość działań zależy od czasu realizacji potęgowania modularnego – jako operacji dominującej.

2.1. ANALIZA PODSTAWOWYCH OPERACJI W RSA

W szyfrowaniu *RSA* realizowane jest potęgowanie modulo N wiadomości W, szyfrogram S otrzymywany jest na podstawie równania 1:

$$S = W^E \bmod N. \tag{1}$$

Klucz szyfrujący, najczęściej klucz publiczny to dwie wartości $\{E, N\}$, gdzie N jest iloczynem „dostatecznie" dużych – obecnie 2048[bit] liczb pierwszych p i q, natomiast wymagane jest, aby $E < N$ było względnie pierwsze z $\varphi(N) = (p-1)(q-1)$. Odtworzenie wiadomości W z S wymaga deszyfrowania zgodnie z równaniem 2:

$$W = S^D \bmod N. \tag{2}$$

Klucz deszyfrujący, najczęściej tajny stanowią liczby $\{D, N\}$, gdzie jest odwrotnością multiplikatywną E modulo $\varphi(N)$. W rzeczywistych implementacjach szyfrowania wiadomość dzielona jest na bloki $W_i < N$, które są szyfrowane oddzielnie, deszyfrowanie realizowane jest analogicznie. Prosta implementacja to podniesienie do potęgi zakończone dzieleniem o łącznej złożoności $O((\log_2 n)^3)$, gdzie $\log_2 N$[bit] jest rozmiarem argumentów. Potęgowanie liczb rozmiaru 2048[bit] jest czasochłonne, wykonanie potęgowania zakończonego dzieleniem generuje dane pośrednie rozmiaru ponad 4[Mbit] dla każdego przetwarzanego bloku danych. W praktycznych implementacjach realizacja potęgowania modularnego korzysta z bibliotek języków programowania realizując szyfrowanie oparte na schemacie *RSA* w oparciu o algorytm mnożenia i podnoszenia do kwadratu oraz redukcji modulo wyników pośrednich – często nazywanego algorytmem *„potęgowania modularnego metodą binarną"* [5]:

Algorytm 1. Potęgowanie w Z_n metodą binarną

Dane: $a \in Z_N$, całkowite $0 < k < N$ gdzie $k = \sum_{i=0}^{t} k_i 2^i$

Wynik: a^k **mod** n

1. $b = 1$;
2. A = a;
3. if $k_0 = 1$ then $b = a$;
4. for i = 0 to t do
  4.1. A = $A^2 \bmod N$;
  2.2. if $k_i = 1$ then $b = A \cdot b \bmod N$;
5. return(b).

Metoda Binarna jest znacząco efektywniejsze od algorytmu klasycznego, skraca znacząco czas realizacji operacji potęgowania modularnego jak również znacząco zmniejsza wymagania pamięciowe, jednakże główną jego wadą jest brak podatności na zrównoleglenie [6]. Taka, nawet pobieżna analiza wskazuje, że poprawy efektywności implementacji *RSA* można poszukiwać w trzech obszarach:

1. zrównoleglenie przetwarzania,

2. ograniczenie liczby mnożeń,
3. eliminacji lub szybkiej realizacji redukcji modulo.

2.2. ZRÓWNOLEGLENIE W IMPLEMENTACJI RSA

Podstawową metodą zrównoleglenia w RSA jest wykorzystanie resztowego systemu 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 względnie pierwszych modułów $\{b_1, b_2, \ldots b_i\}$, często nazywanych bazą B systemu resztowego. Zakresem reprezentowanych w *RNS* wartości jest przedział będący iloczynem modułów B wynoszący $B = \prod b_i$. W RNS reprezentowane są liczby $X < B$, a reprezentacją X jest i – elementowy wektor reszt $\{x_1, x_2, \ldots x_i\}$ z dzielenia danej liczby przez kolejne moduły $x_i = X \bmod b_i$. Wynik operacji arytmetycznej: mnożenia, dodawania lub odejmowania jest reprezentowany przez wektor reszt $z_i = x_i \otimes y_i$. Operacje $\otimes$ są wykonywane na wartościach reszt niezależnie, na wartościach w poszczególnych kanałach b_i. *RNS* pozwala na zastąpienia lub dekompozycję pojedynczego łańcucha cyfr reprezentujących liczbę X wektorem wartości resztowych z mniejszego zakresu. Ponieważ $x_i << X$, więc wynik będzie również reprezentowany przez zbiór cyfr lecz o znacznie mniejszej rozmiarze. Zarówno długość łańcucha jak również czas realizacji operacji arytmetycznych, również potęgowanie modularne algorytmem 1, jest zdeterminowany przez rozmiar modułów b_i. Równomierne 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 możliwość korzystania z zasobów pozwalających na zrównoleglenie obliczeń w symetrycznie przydzielonych jednostkach liczących do poszczególnych kanałów przetwarzania, jest to jedna z przyczyn preferująca wykorzystanie tej metody w realizacjach sprzętowych. W systemie resztowym otrzymane wyniki potęgowania wymagają konwersji do systemu stałobazowego, co jest realizowane na podstawie *Chińskiego Twierdzenie o Resztach* (ang. *CRT, Chinese Remainder Theory*), najczęściej zgodnie z algorytmem zaproponowanym przez Garnera w [5]. Równanie 3 przedstawia schemat szyfrowania *RSA* wykorzystujący zastosowane oznaczenia oraz zrównoleglenie wprowadzane przez *RNS*.

$$S = W^E \bmod N = \left(\sum_i w_i z_i \right)^E \bmod N , \qquad (3)$$

Gdzie $z_i = N/n_i \cdot \left(N/n_i \right)^{-1} \bmod n_i$, a $w_i = W \bmod n_i$.

W *RSA* zakresem i bazą dla *RNS* jest $N=\{p, q\}$, zatem szyfrowanie realizowane jest zgodnie z równaniem 4:

$$S = \left(w_p \cdot z_p + w_q \cdot z_q \right)^E \bmod N , \qquad (4)$$

gdzie $Z_p = q \cdot q^{-1} \bmod p$, $Z_q = p \cdot p^{-1} \bmod q$ to odpowiednie odwrotności multiplikatywne. Taki sposób wykorzystania *RNS* w *RSA* pozwala – zostanie to wykazane dalszej trzeciej części opracowania, na zrównoleglenie przetwarzania z możliwością zmniejszenie rozmiaru podstawy w potęgowaniu.

2.3. REDUKCJA LICZBY MNOŻEŃ DLA *RSA*

Potęgowanie modularne jest podstawą operacją arytmetyczną wykonywaną w *RSA* 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 czasochłonna. Znaczną redukcję liczby mnożeń można uzyskać implementując potęgowanie oparte na schemacie Hornera, które jest podstawą metod mnożenia opartych na wstępnym przeglądzie wielomianu będącego binarną reprezentacją wartości wykładnika [8]. Metoda, ta jest często implementowana w bibliotekach języków programowania (C++, Java) operujących na dużych liczbach. Jest to „najprostszy" z algorytmów korzystających z metody skanowania wykładnika. W algorytmie sprawdzane są kolejne bity wykładnika i dla „1" wykonywane jest dodatkowo – oprócz modularnego podnoszenia do kwadratu, mnożenie i dzielenie modulo. Dla n-bitowego wykładnika metoda ta redukuje mnożenia do liczby jedynek w wykładniku. Jednakże, metoda ta wymaga n operacji podnoszenia do kwadratu i dzielenia modulo. Przy, czym operacja podnoszenia do kwadratu jest wykonywana dwa razy szybciej aniżeli mnożenie [5]. Usprawnieniem metody binarnej jest sprawdzanie w jednym kroku m – bitów i analogicznie jak w metodzie binarnej realizacja odpowiednich działań nazywanych potęgowaniem m – arnym [5]. Rozdzielenie sekwencji jednakowej wartości bitów tworzy „okna" zer i jedynek. Liczba „okien" może być stała i stałej długości lub zmienna i różnej długości. Analiza przeprowadzona w [9] wykazuje uśrednioną 5% redukcję liczby mnożeń w porównaniu z innymi metodami opartymi na wstępnej analizie bitów wykładnika.

2.4. REDUKCJA DZIELENIA MODULO

Redukcja modularna to również czasochłonna operacja arytmetyczna realizowana 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 dzielenia, jest połączenie pojedynczego mnożenia z redukcją modularną. Natomiast, 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 $-m_0^{-1} \bmod b$, gdzie b=2 to podstawa systemu liczbowego, więc operacja $x \bmod 2$ może być zrealizowana jako przesunięcie logiczne o jedną pozycję bitową. Wynikiem redukcji jest $xR^{-1} \bmod m$. Mnożenie Montgomery'ego przedstawia Algorytm 2 wynikiem $Mont(x,y)$ jest liczba całkowita $xyR^{-1} \bmod m$.

Algorytm 2. *Mont*(*x*,*y*) Mnożenia Montgomery'ego

Dane: $m=(m_{n-1}\dots m_1m_0)_b$, $x=(x_{n-1}\dots x_1x_0)_b$, $y=(y_{n-1}\dots y_1y_0)_b$, $0\le x,y<m$, $R=b^n$, oraz $NWD(m,b)=1$ i $m'=-m^{-1}\bmod b$
Wynik: $xyR^{-1}\bmod m$
1. $A=0$;
2. for $i=0$ to $(n-1)$ do
2.1. $u_i=(a_0+x_iy_0)m'\bmod b$;
2.2. $A=(A+x_iy+u_im)/b$;
3. if $A\ge m$ then $A=A-m$;
4. return(A).

Algorytm 3 przedstawia schemat potęgowania modularnego korzystający z *Mont*(*x*,*y*) i przystosowany do algorytmu potęgowania binarnego. Aby zakończyć obliczenia i otrzymać wynik $xy\bmod m$ należy otrzymaną liczbę przemnożyć przez wcześniej wyznaczoną stałą $R\bmod m$. Oprócz redukcji Montgomery'ego w *RSA* wykorzystywana jest redukcja Baretta [3]. Algorytm Redukcji Baretta przedstawiony w [5] również nie wymaga dzielenia, jednakże wymagane są czasochłonne dodatkowe operacje – wstępne oraz końcowe, dlatego nie może być on efektywnie implementowany w pojedynczych działaniach modulo. Jednakże, przetwarzanie *RSA* wymaga wielokrotnych – wykonywanych „seryjnie" mnożeń modularnych, wówczas czas wstępnych obliczeń nie jest tak istotny i algorytm Baretta może również znaleźć swe zastosowanie.

Algorytm 3. Potęgowanie Montgomery'ego

Dane: $m=(m_{n-1}\dots m_1m_0)_b$, $R=b^n$, $m'=-m^{-1}$**modulo***b*,
$e=(e_t\dots e_1e_0)$, $e_t=1$ liczba całkowita $1\le x<m$
Wynik: $x^e\bmod m$
1. $x'=Mont(x,R^2\bmod m)$, $A=R\bmod m$;
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);

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

Tabela. 1
Porównanie redukcji dzielenia modulo

	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

3. METODA POPRAWY EFEKTYWNOŚCI RSA

W 2000r. Wang [8] zaproponował uproszczoną wersję *CRT* konwersji liczb z systemu resztowego – dwumodułowego, do systemu stałobazowego. Schemat tej metody z zastosowaniem oznaczeń przedstawia równanie:

$$X=x_2+b_2\cdot\left[z_{2,1}\cdot(x_1-x_2)\right]\bmod b_1, \qquad (5)$$

gdzie $z_{2,1}=\left(B/b_1\right)^{-1}\bmod b_1=b_2^{-1}\bmod b_1$.

Metoda konwersji Wang'a w odniesieniu do szyfrowania *RSA* została przedstawiona w równaniu 6:

$$S=\left(w_q+q\cdot\left[\left(N/p\right)^{-1}\cdot\left(w_p-w_q\right)\right]\bmod p\right)^E. \qquad (6)$$

Ponieważ, w_q i w_p w równaniu 6 są składnikami wyniku S i są to reszty, więc równanie 6 można zapisać w poniższej postaci:

$$S=s_1+q\cdot\left(q^{-1}\bmod p\right)\cdot s_2, \qquad (7)$$

gdzie

$$\begin{aligned} s_1&=w_q^E\bmod q,\\ s_2&=\left(w_p-w_q\right)^E\bmod p. \end{aligned} \qquad (8)$$

W taki (równanie 7) sposób wykorzystany *RNS* w *RSA* przynosi poprawę efektywności, ponieważ potęgowanie modularne w obliczaniu s_1 i s_2 mogą być realizowane równolegle. Jednakże, realizacja wprost obliczeń dla s_2 jest mało efektywna, ponieważ wymaga wyliczenia E-tej potęgi różnicy w_p-w_q a następnie redukcji modulo p. Ponadto, uwzględniając $q<p$ doszłoby do dublowania obliczeń lub układów realizujących $(w_q^E\bmod q)\bmod p$. Można zauważyć, że obliczenia s_2 stanowią operacje w grupie addytywnej klas reszt modulo p – liczba pierwsza, stanowiące ciało. Zatem, obliczając s_2 można skorzystać z rozdzielności mnożenia względem dodawania. Wówczas równanie na s_2 przyjmuje postać przedstawioną w równaniu 8:

$$s_2=w_p^E\bmod p-(w_q^E\bmod q)\bmod p. \qquad (9)$$

Ponieważ, $q<p$, więc operacje modulo p odniesione do drugiego składnika w równaniu 9 mogą zostać pominięte. Dla składników równania 9 zapisanego w postaci s_1 oraz s_2 można wykorzystać *Małe Twierdzenie Fermata* dla redukcji wykładników, wówczas $a^{p-1}\bmod p\equiv 1\bmod p$, a zależności na s_1 i s_2 przyjmą postać przedstawiona w równaniu 10:

$$\begin{aligned} s_1&=w_q^{E\bmod q-1}\bmod q\\ s_2&=w_p^{E\bmod p-1}\bmod p-s_1 \end{aligned} \qquad (10)$$

W ten sposób operacje potęgowania modularnego w *RSA* nie będą się dublowały i mogą być realizowane równolegle – jeśli tylko zasoby na to pozwalają. Taka, oparta na równaniu 10 implementacja sprzętowa potęgowania modularnego w *RSA* operuje na mniejszych argumentach i wymaga znacznie mniej zasobów sprzętowych.

Sprawdzenie przydatności proponowanych metod można przedstawić na przykładzie konkretnych wartości:

Przykład Szyfrowania RSA

Dane: W = 981, $N=pq=149*113=16837$, $E=15301$,
Wynik: szyfrogram $S=W^E\bmod N$
Metoda konwencjonalna:

$$S=981^{15301}\bmod 16837=7788$$

Metoda oparta na *CRT*:

Obliczenia wstępne:

$$w_p = W \bmod p = 981 \bmod 149 = 87,$$

$$w_q = W \bmod q = 981 \bmod 113 = 77$$

$$z_p = q \cdot q^{-1} \bmod p = 113 \cdot 120 = 13560$$

$$z_q = p \cdot p^{-1} \bmod q = 149 \cdot 22 = 3278$$

Wynik: $S = \left(w_p \cdot z_p + w_q \cdot z_q\right)^E \bmod N$

$$S = \left(87 \cdot 13560 + 77 \cdot 3278\right)^{15301} \bmod 16837 = 7788,$$

Korzystając uproszczeń (równanie 9):
obliczenia wstępne:

$$E_p = E \bmod p = 15301 \bmod 113 = 69,$$

$$E_q = E \bmod q = 15301 \bmod 149 = 57,$$

Operacje po uproszczeniach:

$$S = \left(77^{69} \cdot 3278 + 87^{57} \cdot 13560\right) \bmod 16837 = 7788$$

Metoda proponowana

Obliczenia wstępne:

$$e_p = E \bmod p - 1 = 15301 \bmod 148 = 57,$$

$$e_q = E \bmod q = 15301 \bmod 112 = 69,$$

$$w_p = W \bmod p = 981 \bmod 149 = 87,$$

$$w_q = W \bmod q = 981 \bmod 113 = 77$$

$$z_{2,1} = q^{-1} \bmod p = 120$$

Wynik: $S = s_1 + q \cdot \left[z \cdot s_2 \bmod p\right]$

Operacje po uproszczeniach:

$$s_1 = w_q^{e_q} \bmod q = 77^{69} \bmod 113 = 104$$

$$s_2 = w_p^{e_p} \bmod p - s_1 = 40 - 104 = -64$$

$$S = 104 + \left[\left(120 \cdot (-64)\right) \bmod 149\right] \cdot 113 = 7788$$

---

4. PODSUMOWANIE

Przedstawione wykorzystanie zastosowania *RNS* w *RSA* oparte na konwersji z systemu resztowego do pozycyjnego zaproponowanej przez Wanga poprawia efektywność w implementacji programowej. Jednakże, główną zaletą proponowanych rozwiązań jest znaczące zmniejszenie rozmiaru czynników operacji szyfrowania co w implementacji sprzętowej poprawia efektywność, ale przede wszystkim znacznie zmniejsza zasoby sprzętowe niezbędne do realizacji obliczeń. W porównaniu rozwiązaniami bazującymi na klasycznym zastosowaniu *CRT* (równanie 3, 4), gdzie potęgowanie modularne i redukcja modularna realizowana jest na argumentach o rozmiarze $\log_2 N$, zaproponowane rozwiązanie oparte na równaniu 6 implementowane z uwzględnieniem równania 10 operuje na argumentach rozmiaru $\log_2 p$. Wówczas złożoność opisuje zależność $O((\log_2 N/2)^3)$, wymagając o połowę mniejszych arytmometrów. Zaproponowane rozwiązanie sprawdza się również w realizacji potoku przetwarzania dla szyfrowania *RSA*, ze względu na mniejsze argumenty mnożenia i krótszy czas realizacji poszczególnych stopni potoku, pozwala to na uzyskanie wielokrotnie większej przepustowości – uwzględniając również krótszy czas napełniania potoku. Dodatkowa zaletą proponowanego rozwiązania jest możliwość łączenia go z innymi metodami np. proponowanymi przez Montgomerye'go czy Barett'a.

LITERATURA

[1] Whitfield D. and Hellman M., "New directions in cryptography". IEEE Transactions on Information Theory, 197.

[2] Rivest R., Shamir A. and Adleman A., „A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". Communications of the ACM, February 1978.

[3] Barrett P.D., "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] Montgomery, P., "Modular multiplication without trial division," Mathematics of Computation, Vol. 44, 1985.

[5] N. Ferguson, B. Schneier, *Kryptografia w praktyce*, Helion, Gliwice 2004

[6] Menezes, A., Oorschot, P., Vanstone S., *Kryptografia stosowana,* Warszawa, WNT, 2005.

[7] Koblitz, N. *Wykład z teorii liczb i kryptografii*, Warszawa: WNT, 1995.

[8] Wang, Y. „Residue-to-binary converters based on New Chinese Remainder Theorems",IEEE Transactions on Circuits and SystemsPart II: Analog and Digital Signal Processing **47**(3): 197–205.

[9] Koc, C. K., High-speed RSA implementation, Version 2.0. *RSA Laboratories*, November 1994.

dr inż. Janusz Jabłoński
Uniwersytet Zielonogórski, Wydział Matematyki, informatyki i ekonometrii
Zakład Zastosowań Informatyki

ul. Prof. Z. Szafrana 4a
65-516 Zielona Góra
tel.: 68 3282868
e-mail: Jablonski@wmie.uz.zgora.pl