SIGMA
CENTRUM MODELOWANIA MATEMATYCZNEGO

# Ataki na algorytm RSA

**Andrzej Chmielowiec**

29 lipca 2009

**Streszczenie**

Przedmiotem referatu są ataki na mechanizm klucza publicznego RSA. Wieloletnia historia wykorzystywania tego algorytmu naznaczona jest przykładami skutecznych ataków na konkretne implementacje. Szczególną uwagę chcę w tym artykule zwrócić na ataki z kanałem pobocznym (pomiar czasu, pomiar poboru mocy, ataki z generowaniem błędu, itp...), które stanowią bardzo poważne zagrożenie dla współczesnych implementacji.

## 1 Wprowadzenie

Ataki z wykorzystaniem kanału pobocznego są dziś jednym z głównych zagrożeń dla systemów kryptograficznych. Mierząc takie wielkości fizyczne jak: czas, moc, natężenie pola elektromagnetycznego, itp. kryptoanalityk uzyskuje dodatkowe informacje na temat przetwarzanych danych. W wielu przypadkach ta dodatkowa wiedza prowadzi do poznania bitów tajnego klucza. W przypadku algorytmu RSA wyciek nawet części bitów klucza prywatnego jest bardzo niebezpieczny. Związane jest to z wieloma algebraicznymi własnościami, które mogą być użyte do poznania całej informacji o kluczu prywatnym.

## 2 Ataki algebraiczne

### 2.1 Faktoryzacja modułu

Podstawową techniką algebraiczną, która pozwala na złamanie algorytmu RSA, jest faktoryzacja modułu. Jest to najbardziej czasochłonna metoda ataku, a czas potrzebny na przeprowadzenie ataku zależy od możliwości obliczeniowych i zastosowanych algorytmów. Najszybszą ogólną metodą rozkładu na czynniki jest aktualnie sito ciał liczbowych NFS (Number Field Sieve). Niemniej jednak istnieją metody, które pozwalają na szybką faktoryzację w przypadku zastosowania liczb pierwszych konkretnego typu. Z tego właśnie powodu wprowadza się dodatkowe warunki na parametry klucza RSA.

Aby wyeliminować możliwość szybkiej faktoryzacji dla określonego typu czynników zalecane jest generowanie liczb silnie pseudopierwszych. Liczby te mają własności, które uniemożliwiają praktyczne zastosowanie jakiejkolwiek szybkiej metody faktoryzacji. Niestety czas generowania parametrów RSA, spełniających dodatkowe warunki jest relatywnie długi. Dokładne wymagania co do sposobu generowania czynników modułu zostały określone w załączniku B normy FIPS 186-3 (wersja robocza). W szczególności wymaga się, aby czynniki $p$ i $q$ spełniały następujące warunki:

1. $p-1$ ma duży czynnik pierwszy,
2. $p+1$ ma duży czynnik pierwszy,
3. $q-1$ ma duży czynnik pierwszy,
4. $q+1$ ma duży czynnik pierwszy,
5. $|p-q| > 2^{\log_2 p - 100}$.

Ponadto prawdopodobieństwo zdarzenia, że liczby $p$ lub $q$ są złożone, powinno być nie większe niż $2^{-100}$. W praktyce oznacza to wykonanie 50 testów Rabina-Millera. Należy tutaj zaznaczyć, że załącznik B normy FIPS 186-3 zaleca generowanie liczb pierwszych spełniających dodatkowe warunki jedynie w przypadku 1024-bitowych kluczy RSA. Ma to duże znaczenie praktyczne w przypadku urządzeń o niewielkich zasobach obliczeniowych. Dodatkowe warunki wiążą się bowiem ze znacznym wydłużeniem czasu generowania klucza.

## 2.2 Ataki na krótki wykładnik prywatny

Algorytm RSA stwarza wiele możliwości optymalizacji jego pracy i dostosowania do konkretnego zastosowania. Niestety nieodpowiednie optymalizacje mają negatywny wpływ na bezpieczeństwo algorytmu. W skrajnych przypadkach pozwala to na przeprowadzenie ataków i kompromitację kluczy kryptograficznych. Z taką sytuacją mamy na przykład do czynienia w przypadku, gdy długość wykładnika prywatnego $d$ jest zbyt mała. Zmniejszenie długości liczby $d$ byłoby korzystne z punktu widzenia wszystkich urządzeń o ograniczonych możliwościach obliczeniowych (na przykład tokeny, karty inteligentne, itp.). Niestety, jeżeli tylko spełniona jest nierówność $d < N^{0.3}$, to istnieje metoda efektywnego złamania systemu RSA.

Istnieje możliwość wybrania takiego wykładnika prywatnego $d$, który byłby duży modulo $\varphi(N)$, ale mały modulo $p-1$ i $q-1$. Wtedy wykorzystanie chińskiego twierdzenia o resztach (CRT) pozwala na znaczne przyspieszenie operacji kryptograficznej. Niestety nie wiadomo w jaki sposób wpływa to na bezpieczeństwo algorytmu.

## 2.3 Ataki na krótki wykładnik publiczny

Ataki na krótki wykładnik publiczny opierają się głównie na twierdzeniu Coppersmitha:

**Twierdzenie 1 (Coppersmith)** Niech $N$ będzie liczbą całkowitą, a $f(X) \in \mathbb{Z}_N[X]$ wielomianem stopnia $d$, którego współczynnik przy wyrazie $X^d$ wynosi 1. Ustalmy $x = N^{\frac{1}{d}-\epsilon}$ dla pewnego $\epsilon \geq 0$. Wtedy istnieje efektywny sposób wyznaczenia wszystkich liczb $x_0 < x$ takich, że $f(x_0) = 0 \bmod N$. Czas znalezienia pierwiastków zależy od wydajności algorytmu LLL dla kraty o wymiarze $O(w)$, gdzie $w = \min(1/\epsilon, \log_2 N)$.

Twierdzenie to wykorzystuje algorytm kratowy LLL do znalezienia pierwiastków wielomianu o współczynnikach całkowitych. Zaproponowane rozwiązanie jest tym wydajniejsze im mniejszy jest stopień wielomianu. Dlatego też wykorzystanie krótkiego klucza publicznego pozwala w niektórych przypadkach na odszyfrowanie wiadomości. W kolejnych paragrafach znajduje się krótki opis możliwych scenariuszy ataku.

Opisane dalej metody ataku działają jedynie w określonych sytuacjach. Warto jednak pamiętać, że w momencie generowania klucza nie zawsze wiadomo gdzie będzie on zastosowany. Dlatego też należy unikać krótkich wykładników publicznych (zwłaszcza 3 i 5) i jako najkrótszy wykładnik przyjmować $F_4 = 2^{2^4} + 1 = 65537$.

**Atak Hastada**

Idea ataku Hastada polega na wykorzystaniu faktu, że kilku użytkowników posiada bardzo krótki klucz publiczny RSA (na przykład 3).

**Twierdzenie 2 (Hastad)** Niech $N_1, \ldots, N_k \in \mathbb{Z}$ będą parami względnie pierwsze. Najmniejszą z liczb $N_i$ oznaczamy przez $N_{min}$. Jeśli wielomiany $g_i(X) \in \mathbb{Z}_{N_i}[X]$ mają stopień co najwyżej $k$ i istnieje jedna liczba $M < N_{min}$ spełniająca układ równań

$$g_i(M) = 0 \bmod N_i,$$

to liczba $M$ może być wyznaczona w sposób efektywny.

Wyobraźmy sobie teraz sytuację, w której nadawca wysyła tą samą wiadomość $M$ do trzech różnych osób, z których każda posługuje się kluczem publicznym postaci $\langle N_i, 3\rangle$. W takiej sytuacji wysyłane są 3 szyfrogramy:

1. $C_1 = M^3 \bmod N_1$,
2. $C_2 = M^3 \bmod N_2$,
3. $C_3 = M^3 \bmod N_3$.

Powyższy układ równań pozwala na odtworzenie wartości $M^3 \bmod (N_1 N_2 N_3)$. Ponieważ $M < N_i$, to wartość $M^3$ może być wyznaczona dokładnie. Teraz wystarczy wyciągnąć pierwiastek stopnia trzeciego w pierścieniu liczb całkowitych, aby otrzymać wartość przesyłanej wiadomości $M$.

**Atak Franklina-Reitera na powiązane wiadomości**

Metoda ta działa tylko w przypadku wykładnika publicznego o wartości 3. Pozwala na złamanie dwóch szyfrogramów $C_1 = M_1^3$ i $C_2 = M_2^3$, które zostały wysłane do tej samej osoby. Warunkiem koniecznym przeprowadzenia skutecznej kryptoanalizy jest znajomość liniowej zależności pomiędzy wiadomościami $M_1$ i $M_2$. Jeżeli atakujący zna funkcję $f(X) = aX + b \in \mathbb{Z}_N[X]$ ($b \neq 0$), dla której prawdziwa jest zależność $M_2 = f(M_1)$, to wykorzystując twierdzenie Coppersmitha jest w stanie odtworzyć wiadomości $M_1$ i $M_2$ na podstawie szyfrogramów $C_1$ i $C_2$.

**Atak Coppersmitha na wiadomości z krótkim dopełnieniem**

Coppersmith zaproponował znaczne wzmocnienie ataku Franklina-Reitera. Okazuje się, że zdeszyfrowanie wiadomości jest możliwe jeżeli zostanie ona wysłana dwukrotnie, a dopełnienie będzie zbyt krótkie. Jeżeli $e$ jest wykładnikiem publicznym, a $n$ oznacza liczbę bitów modułu, to określamy liczbę $m = \lfloor n/e^2 \rfloor$. Gdy atakujący przechwyci dwa szyfrogramy postaci:

1. $C_1 = (2^m M + r_1)^e$, gdzie $0 \leq r_1 < 2^m$,
2. $C_2 = (2^m M + r_2)^e$, gdzie $0 \leq r_2 < 2^m$,

to jest w stanie odzyskać z nich wiadomość $M$. Łatwo można wyliczyć, że w przypadku 1024-bitowego klucza i wykładnika publicznego równego 3 możliwe jest odzyskanie pierwszych 911 bitów $M$, o ile losowe dopełnienie jest nie dłuższe niż 113 bitów.

Jak widać atak może być zastosowany jedynie w przypadku, gdy dopełnienie umieszczane jest w najmłodszych bitach wiadomości. W związku z tym atak nie może być przeprowadzony dla wiadomości formatowanych zgodnie ze standardem PKCS#1. Drugą kwestią jest wielkość wykładnika publicznego $e$. Jeżeli jest on równy $F_4 = 2^{2^4} + 1 = 65537$, to atak nie ma szans powodzenia dla stosowanych w praktyce długości kluczy.

**Atak na częściowo jawny klucz prywatny**

Krótkie wykładniki publiczne pozwalają na znalezienie całego klucza prywatnego i faktoryzację modułu, jeżeli tylko znana jest część bitów klucza prywatnego. Jeżeli $N = pq$ jest modułem RSA o długości $n$-bitów, a wykładnik publiczny $e$ spełnia zależność $e < \sqrt{N}$, to można:

1. znaleźć wszystkie bity $d$ w czasie $e \log_2 e$, o ile znamy $\lfloor n/4 \rfloor$ najmłodszych bitów wykładnika prywatnego $d$,
2. znaleźć rozkład modułu RSA na czynniki, jeżeli znamy $\lfloor n/4 \rfloor$ najmłodszych lub najstarszych bitów czynnika $p$.

Atak ten ma duże znaczenie praktyczne. Może być bowiem wykorzystany do odzyskania kluczy kryptograficznych po ataku typu Cold Boot. Chwilowe odłączenie zasilania (element ataku Cold Boot) może bowiem spowodować utratę części bitów klucza RSA. Jak jednak widać zastosowanie odpowiednich metod algebraicznych pozwala na wydobycie kompletnej informacji o kluczu nawet w przypadku posiadania niewielkiej liczby bitów klucza.

# 3 Ataki czasowe

Czas wykonania poszczególnych instrukcji procesora może być zależny od jej argumentów wejściowych. Sytuacja taka ma miejsce w większości współczesnych procesorów, które wyposażone są w całą gamę mechanizmów optymalizujących wykonywanie kodu. Pomiar czasu wykonania danej operacji kryptograficznej jest więc w pewnym stopniu skorelowany z przetwarzanymi danymi (w szczególności kluczami kryptograficznymi). Generalnie atak czasowy skierowany jest przeciwko konkretnej implementacji, która uruchomiona jest na określonej rodzinie układów elektronicznych. Wykorzystując specyficzne cechy danej architektury kryptoanalityk uzyskuje pewne informacje o kluczach.

## 3.1 Atak czasowy Kochera

Atak polega na pomiarze czasu wykonania operacji prywatnej dla pewnej liczby argumentów wejściowych (proporcjonalnej do długości klucza RSA). Zakładamy przy tym, że dysponujemy również argumentami, które były przetwarzane oraz wiedzą na temat używanej implementacji. To wymaganie powoduje, że najbardziej wrażliwy na tego rodzaju atak jest podpis RSA (na przykład PKCS#1 v1.5 lub PSS). Po zgromadzeniu niezbędnej liczby próbek następuje faza analizy statystycznej, która daje informację na temat kolejnych bitów wykładnika prywatnego (począwszy od najmniej znaczących). W tym miejscu warto zauważyć, że dla wykładnika publicznego równego 3 i 5 wystarczy wyznaczyć jedynie około $1/4$ najmłodszych bitów. Pozostałe mogą być bowiem dość szybko znalezione metodami algebraicznymi.

Podstawową techniką zabezpieczającą przed tym atakiem jest tak zwane maskowanie. Polega ono na pomnożeniu argumentu wejściowego $x$ przez pewną losową wartość $r^e$, gdzie $e$ jest wykładnikiem publicznym. Wykonując operację prywatną na $xr^e$ otrzymujemy $x^d r^{ed} = x^d r \bmod N$. Na końcu wystarczy więc zdjąć maskę poprzez pomnożenie uzyskanego wyniku przez $1/r \bmod N$. Zaślepianie znacznie spowalnia operację prywatną, ale jest absolutnie konieczne w przypadku takich standardów jak podpis PKCS#1 v1.5. Niezastosowanie odpowiednich zabezpieczeń umożliwia bowiem złamanie klucza nawet tych usług, które działają w środowisku sieciowym. Wszystko jest tak naprawdę kwestią liczby zgromadzonych próbek. Oczywiście najbardziej narażone na tego typu atak są karty inteligentne. Wynika to z faktu, że sygnał zegarowy pobierany jest ze środowiska zewnętrznego. Należy tutaj podkreślić, że warunkiem koniecznym powodzenia ataku jest znajomość danych wejściowych. W związku z tym jego zastosowanie ogranicza się głównie do podpisów cyfrowych. Jeśli chodzi o łamanie kluczy służących do deszyfrowania, to atakujący musi mieć najpierw dostęp do jawnych tekstów wiadomości. Taki scenariusz można sobie dość łatwo wyobrazić. Wystarczy bowiem, że nawiązanie bezpiecznego połączenia z serwerem będzie inicjowane poprzez wysłanie zaszyfrowanego klucza symetrycznego do serwera. W takiej sytuacji klient będzie w stanie gromadzić dane, które pozwolą na przeprowadzenie analizy statystycznej i złamanie klucza.

### 3.2 Atak czasowy na mnożenie Montgomery'ego i pamięć CACHE instrukcji

Podstawowa implementacja mnożenia Montgomery'ego zawiera pojedynczą instrukcję *if*, która jest wykonywana, gdy rezultat obliczeń jest większy od modułu. Okazuje się, że ta pojedyncza instrukcja może umożliwić atak, którego rezultatem będzie złamanie klucza. Całość ataku opiera się na założeniu, że kod takiej funkcji jest wykonywany na procesorze potrafiącym *przewidywać* instrukcję skoku. Okazuje się, że niewielkie różnice czasowe, które powstają podczas wykonywania takiego mnożenia mogą być użyte w analizie statystycznej do odzyskania klucza RSA. W ten sposób została pomyślnie zaatakowana między innymi implementacja biblioteki OpenSSL. Na szczęście aby przeprowadzić atak, musi być spełnionych kilka warunków. Pierwszym i najważniejszym z nich jest fizyczny dostęp do maszyny, na której wykonywane są operacje kryptograficzne. Drugim jest możliwość dokładnego pomiaru czasu wykonywanych operacji. Dopiero wtedy można przystąpić do zbierania danych na potrzeby analizy statystycznej.

W przypadku algorytmu RSA i mnożenia Montgomery'ego można uniknąć instrukcji warunkowej na końcu każdej funkcji mnożącej. Wystarczy tylko uwzględnić odpowiednią poprawkę w wyniku obliczeń.

## 4 Ataki z generacją błędu podczas operacji prywatnej wykorzystującej CRT

Jest to jeden z najbardziej efektywnych ataków na system RSA. Działa na implementacje, które wykonują operację podpisu z wykorzystaniem twierdzenia chińskiego o resztach (CRT). Aby dokładnie zrozumieć ideę ataku należy najpierw zapoznać się z podstawowym sposobem implementacji operacji prywatnej RSA z wykorzystaniem CRT. Jeżeli moduł jest postaci $N = pq$, wykładnik prywatny ma wartość $d$ i $d_p = d \bmod (p-1)$, $d_q = d \bmod (q-1)$ to operacja prywatna przebiega następująco:

1. $C_p = M^{d_p} \bmod p$,
2. $C_q = M^{d_q} \bmod q$,
3. $C = T_1 C_p + T_2 C_q \bmod N$, gdzie

$$T_1 = \begin{cases} 1 \bmod p \\ 0 \bmod q \end{cases}, \qquad T_2 = \begin{cases} 0 \bmod p \\ 1 \bmod q \end{cases}.$$

Atak polega na wywołaniu błędu procesora podczas pierwszej, albo drugiej fazy obliczeń. Należy tutaj zauważyć, że trzeci krok operacji prywatnej jest zaniedbywalnie krótki w stosunku do kroków 1 i 2. Zaburzenie pracy procesora może być wywołane na wiele różnych sposobów: skokowe zwiększenie lub zmniejszenie napięcia zasilającego, podanie zbyt dużej częstotliwości

zegara, impuls elektromagnetyczny, itp. Istotne jest, aby po wprowadzeniu zaburzenia powstał błąd obliczeń, który spowoduje, że powiedzmy wartość $C_q$ będzie różna od prawidłowej. Otrzymamy zatem podpis $\hat{C}$, który będzie spełniał zależności:

$$\hat{C}^e = M \bmod p \qquad \text{i} \qquad \hat{C}^e \neq M \bmod q.$$

Oznacza to, że gcd $\left(\hat{C}^e - M, N\right)$ jest nietrywialnym czynnikiem $N$. Oczywiście, aby atak zakończył się powodzeniem musimy znać wartość $M$. W praktyce wiąże się to z koniecznością używania przez ofiarę deterministycznego dopełnienia wiadomości (takiego jak PKCS#1 v1.5). W przypadku losowego dopełnienia wiadomości na przykład PKCS#1 PSS) atak nie ma szans powodzenia.

# 5 Podsumowanie

Ataki opisane w artykule stanowią jedynie krótki wstęp do technik, które mogą być zastosowane podczas ataków na algorytm RSA. Przygotowanie implementacji odpornej na wszystkie możliwe scenariusze ataków jest zadaniem bardzo trudnym, a w niektórych przypadkach wręcz niemożliwym. Na szczęście w wielu przypadkach niektóre zagrożenia można pominąć ze względu na bardzo małe prawdopodobieństwo przeprowadzenia danego typu ataku w praktyce. Niemniej jednak każda implementacja RSA powinna być poprzedzona analizą zagrożeń i wprowadzeniem odpowiednich mechanizmów zabezpieczających implementację.