Jak łamano RSA

1.Wprowadzenie

Kryptosystem RSA został opracowany przez Rona Rivesta, Adi Shamira i Lena Adlemana.ZOstał upubliczniony w sierpniu 1977 roku w Scientific American. Jest najczęściej wykorzystywany do ochrony prywatności i zapewnienia uwierzytelnienia danych cyfrowych. Jest używany przez serwery sieciowe i przeglądarki internetowe w celu zabezpieczenia ruchu, uwierzytelniania e-mail, bezpiecznych sesji zdalnego logowania, jest w centrum elektronicznych kart kredytowych systemów płatniczych. Jest używanym tam gdzie bezpieczeństwo danych cyfrowych jest istotne. Od czasu pierwszej publikacji, system RSA został zanalizowany pod względem podatności przez wielu naukowców. Mimo trzydziestu lat badań prowadzących do fascynujących prób jego złamania, żaden z nich nie był niszczący. Najczęściej ilustrują niebezpieczeństwo niewłaściwego użycia RSA. Rzeczywiści, bezpieczeństwo implementacji RSA nie jest zadaniem szablonowym. Naszym celem jest zbadanie niektórych z tych prób złamania RSA i opisanie podstawowych narzędzi matematycznych do tego użytych. Będziemy używali konwencjonalnego nazewnictwa takiego jak Alicja i Bob dla oznaczenia dwóch części próbujących się ze sobą skomunikować. Użyjemy Marcina dla oznaczenia złośliwego atakującego ,chcącego podsłuchiwać lub manipulować komunikacją między Alicją i Bobem. Zaczniemy od opisu uproszczonej wersji szyfrowania RSA. Niech N = pq będzie ilcozynem dwóch dużych liczb pierwszych o tym samym rozmiarze (n/2 bitów każda). Typowy rozmiar dla N = 1024 bity tj 309 czyfr dziesiętnych. Każdy ze współczynników ma 512 bitów. Niech e, d będą dwoma liczbami całkowitymio spełniającymi warunek ed = 1 mod φ(N) gdzie φ(N) = (p-1)(q-1) jest porządkiem grupy multiplikatywnej Z^*_N. Nazywamy N modułem RSA, e wykładnikiem szyfrowania a d wykładnikiem deszyfrowania. Para (N,e) jest kluczem publicznym. Jak sugeruje nazwa, jest on publiczny i używany do szyfrowania wiadomości. Para (N,d) jest nazywana kluczem tajnym lub kluczem prywatnym i jest znana odbiorcy szyfrowanej wiadomości. Tajny klucz pozwala na odszyfrowanie wiadomości. Wiadomość jest liczbą całkowitą $M \in Z^*_N$. Dla zasyzfrowania M obliczamy $C = M^e \bmod N$. Dla odszyfrowania wiadomości , uprawniony odbiorca oblicza $C^d \bmod N$. Rzeczywiści

$$C^d = M^{ed} = M \pmod N$$

gdzie ostatnia równość wynika z twierdzenia Eulera. Definiujemy funkcję RSA jako $x \to x^e \bmod N$. Jeśli podane jest d, funkcja może być łatwo odwrócona przy użyciu powyższego równania. Odnosimy się do d jako do pułapki umożliwiającej inwersję tej funkcji. W tym badaniu przestudiujemy trudności inwersji funkcji RSA bez tej pułapki. Odniesiemy się do tego jako łamania RSA. Precyzyjniej,przy danej trójce (N,e,C), pytamy jak trudno jest wyliczyć e-ty pierwiastek z C modulo N = pq, kiedy faktoryzacja N jest nieznana. Ponieważ Z^*_N jest zbiorem skończonym, można wyliczyć wszystkie elementy Z^*_N dopóki nie znajdzie się właściwe M. Niestety, to powoduje w algorytmie z czasem pracy rzędu N

mianowicie wykładniczy wzrost danych wejściowcyh ,który jest rzędu $\log_2 N$. Najczęściej jesteśmy zainteresowania alogrytmami, ze znacznie niższym czasem pracy, mianowicie rzędu n^c gdzie $n = \log_2 N$ a c jest jakąś małą stałą (powiedzmy mniejszą niż 5).
Tutaj zajmujemy się funkcją RSA w przeciwieństwie do kryptosystemu RSA. Krótko mówiąc, trudności w odr=wracaniu funkcji RSA w losowych wejściach implikuje ,że mając dane (N, e, C), atakujący nie może odkryć tekstu jawnego M. Jednak kryptosystem musi być odporny na bardziej subtelne ataki.Jeśli dane jest (N,e,C), powinno być trudno odzyskać informację o M. Jest to znane jako bezpieczeństwo semantyczne. Nie będziemy omawiać tych subtelnych ataków, ale wskażemy ,że RSA opisane powyżej nie jest semantycznie bezpieczne: przy danych (N,e,C), można łatwo wydedukować pewną informację o tekście jawnym M (na przykład, symbol Jacobiego M z N może być łatwo wydedukowany z C). RSA można stworzyć bezpiecznym semantycznie przez dodanie losowwości w procesie szyfrowania. Funkcja RSA $x \rightarrow x^e \bmod N$ jest przykładem funkcji jednokierunkowej z pułapką. Może być łatwo obliczona, ale nie można wystarczająco odwrócić bez pułapki d z wyjątkiem specjalnych warunków. Funkcja jednokierunkowa z pułapką może być używana dla podpisów cyfrowych. Podpisy cyfrowe zapewniają autentyczność i niezaprzeczalności elektronicznych dokumentów prawnych. Aby podpisać wiadomość $M \in Z^*_N$ używając RSA, Alicja stosuje swój prywatny klucz (N,d) do M i uzyskuje podpis $S = M^d \bmod N$. Przy danym (M,S), każdy może zweryfikować podpis Alicji w M przez sprawdzenie ,że $S^e = M$ nod N. Ponieważ tylko Alicja może wygenrować S, można oczekiwać ,że przeciwnik nie może sfałszować podpisu Alicji. Niestety rzecz nie jest tak prosta; dodatkowe środki są potrzebne dla właściwego bezpieczeństwa. Podpisy cyfrowe są ważnymi aplikacjami RSA. Niektóre z ataków były skierowane szczególnie przeciwko podpisom cyfrowym RSA. Para kluczy RSA jest generowana przez pobranie dwóch losowych liczb pierwszych n/2 bitowych i pomnożenie ich przez uzyskane N. Potem, dla danego wykładnika szyfrowania $e < \varphi(N)$, obliczamy $d = e^{-1} \bmod \varphi(N)$ używając rozszerzonego algorytmu Euklidesa. Ponieważ zbiór liczb pierwszych jest wystarczająco gęsty, losowa liczba pierwsza n/2 bitowa możę być szybko wygenerowana przez powtarzalne ponieranie losowych n/2 bitowych liczb całkowitych i testowanie każdej na pierwszość używając probablistycznych testów pierwszości.

Rozkład na czynniki dużych liczb całkowitych

Pierwszy atak na klucz publicznyu RSA (N,e), rozpatrzymy przez faktoryzację modułu N. Przy danej faktoryzacji N, atakujący może łatwo zbudować $\varphi(N)$, z którego odszyfruje wykładnik $d=e-1 \bmod \varphi(N)$. Odniesiemy się do faktoryzacji modułu jak ataku brute-force na RSA. Chociaż algorytm faktoryzacji stale się poprawia, przy obecnym stanie techniki, nadal nie istnieje zagrożenie bezpieczeństwa RSA, kiedy RSA jest użyte właściwie. Rozkład na czynniki pierwsze dużych liczb całkowitych jest najpiękniejszym probleme obliczeniowym w matematyce, ale nie o tym będziemy mówić. Obecnie najszybszym algorytmem faktoryzacji jest Ogólne Sito Ciała Liczbowego (GNFS).Jego czas pracy na n-bitowych liczbach

całkowitych to

$$\exp\left((c+o(1))n^{1/3}\log^{2/3} n\right)$$

dla c < 2. Ataki na RSA które trwają dłużej niż ten czas graniczny nie są interesujące. Są to ataki typu wyczerpujace wyszukiwanie dla M i inne starsze ataki opublikowane zaraz po publikacji RSA. Naszym celem jest zbadanie ataków na RSA, które deszyfrują wiadomości bez bezpośredniej faktoryzacji modułu N RSA. Jednak warto zauważyć ,że niektóre rzadkie zbiory modułu N=pq RSA , mogą być latwo rozłożone na czynniki. Na przykład, jeśli p-1 jest liczonym czynników pierwszych mniejszych niż B, wtedy N może być rozłożone na czynniki w czasie mniejszym niż B3. Niektóre implementacje wyraźnie odrzucają liczby pierwsze p, dla których p-1 jest iloczynem małych liczb pierwszych.

Jak wspomniano powyżej, jeśli istniej wydajny algorytm faktoryzacji, RSA jest niestabilne. Odwrotna sytuacja od dawna jest otwartym problemem: czy trzeba czynnika N a by skutecznie obliczyć e^{ty} pierwiastek modulo N? Czy złamanie RSA jest trudne jak faktoryzacja. Poniżej mamy otwarty konkrenty problem

Problem 1. Dane są liczby całkowite N i e sppełniające gcd(e,φ(N)) = 1,definiujemy funkjcę $f_{e,N} : Z^*_N \to Z^*_N$ przez $f_{e,N}(x) = x^{1/e} \bmod N$. Czy jest algorytm w czasie wielomianowym A , który oblicza faktoryzację N danego N i dostęp do $f_{e,N}(x)$ dla pewnego e? Dla f(x) szacujemy funkcję dla każdego wejściwoego x w jednostce czasu. Bonceh i Venkatesan dostarczyli dowód ,że dla małych e, odpowiedź na powyższy problem może brzmieć, "nie". Innymi słowy, dla małych e nie może istnieć czas wielomianowy redukcji z faktoryzacji do łamania RSA. Zrobili to przez pokazanie, że w pewnym modelu, odpowiedź pozytywna na problem małego e daje efektywny algorytm faktoryzacji. Zauważmy ,że pozytywna odpowiedź na Problem daje podstawę do "Ataku tekstem zaszyfrowanym" na RSA. Dlatego odpowiedź negatywna jest mile widziana.Następnie pokazujemy ,że ujawnienie klucza prywatnego d i rozkład na czynnik N jest równoważne. Dlatego nie ma sesnu ukrywanie faktoryzacji N przedkimś kto zna d.

Fakt 1 Niech (N,e) będą kluczem publicznym RSA. Przy danym kluczu prywatnym d, można skutecznie uzyskać współczynnik modułu N=pq. Odwrotnie, przy danej faktoryzacji N można skutecznie odzyskać d

Dowód. Rozkład na czynniki N daje φ(N) .Ponieważ e jest znane może odzyskać d. To udowadnia odwrotne polecenie. Teraz pokażemy ,ze przy danym d można uzysakć współczynnik N. Przy danym d, obliczamy k = de-1. Z definicji d i e wiemy ,że k jest wielokrotnością φ(N). φ(N) jest parzyste , $k = 2^t r$ z nieparzystym r i $t \geq 1$. Mamy $g^k=1$ dla każdego g € Z^*_N, dlatego $g^{k/2}$ jest pierwiastkiem kwadratowym jednostki modulo N. Z chińskiego twierdzenia reszt wynika że, 1 ma cztery pierwaistki kwdratowe modulo N = pq. Dwa z nich to +/- 1, pzostałe dwa to +/- x gdzie x spełania x = 1 mod p i x=-1 kod q. Korzystając z jednego z tych dwóch ostatnich pierwiastków kwadratowych, rozkład na czynniki N ujawnia sięprzez obliczenie gcd(x-1,N). Prosty argument pokazuje ,że jesli g jest wybrane losowa z Z^*_N wtedy z prawdopodobieństwem ½ jeden z tych elementów wsekwencji $g^{k/2}$, $g^{k/4}$, ... $g^{k/2t}$ mod N jest pierwiastkiem kwadratowym

jednostki, która ujawnia faktoryzację N. Wszystkie elementy w sekwnecji mogś być wydajnie wyliczone w czasie $O(n^3)$ gdzie $n = \log_2 N$

2.Ataki elementarne

Zaczniemy od opisu starych ,elementarnych ataków. Ataki te ilustrują rażące naruszenia RSA. Chociaż wiele z takich ataków istnieje , podamy tylko dwa przykłady

2.1 Wspólny moduł

Aby uniknąć generowania różnych modułów N=pq dla każdego użytkownik , można sobie zażyczyć stałego N dla wszytstkich. To samo N jest używane przez wszystkich. Zaufana jednostka centralna może dostarczyć uzytkownika i z unikalną parą e_i, d_i, z których użytkownik i formuje klucz publiczny (N, e_i) i klucz tajny (N, d_i). Na pierwszy rzut oka to może wydawać się że działą: tekst zaszyfrowany $C = M^{e_a} \bmod N$ przeznaczony dla Alicji nie może być rozszyfrowany przez Boba ponieważ Bob nie posiada d_a. Jednak, jest to niepoprawne w system wynikowy jest niezabezpieczony .Z Faktu 1 wynika, że Bob może użyć własnych wykładników e_b, d_b dla rozkładu modułu N. Kiedy N zostanie rozłożone, Bob może uzyskać prywatny klucz Alicji d_a z jej klucza publicznego e_a. Ta obserwacja pokazuje ,że moduły RSA nigdy nie powinny być używane więcej niż jeden raz.

2.2.Oślepienie

Niech (N,d) będzie prywatnym kluczem Boba a (N,e) jego odpowiednim kluczem publicznym. Przypuśćmy ,że przeciwnik Marcin chce podpisu Boba w wiadomości $M \in Z^*_N$. Nie będąc głupcem, Bob odrzuca podpis M. Marcin może próbować czegoś następującego: wybiera losowo $r \in Z^*_N$ i ustawia $M' = r^e M \bmod N$. Potem prosi Boba aby podpisał wiadomość losową M'. Bob może być skłonny dla dostarczenia podpisu S' na niewinnie wyglądajacym M'. Ale przypominam ,że $S' = (M')^d \bmod N$. Teraz Marcin próbuje wyliczyć $S = S'/r \bmod N$ i uzyskać podpis Boba S na oryginalnym M. Rezczywiście

$$S^e = (S')^e / r^e = (M')^{ed} / r^e \equiv M'/r^e = M \pmod N$$

Technika ta jest zwana oślepianiem, pozwala Marcinowi uzyskać poprawny podpis prosząc Boba aby podpisał losową "oślepiającą" wiadomość. Bob nie ma informacji jaką właściwie podpisuje wiadomość. Ponieważ większość schematów podpisów wykorzystuje "Hashowanie jednokierunkowe" wiadomości M przed podpisaniem, atak nie jest poważnym problemem.

3.Niski prywatny wykładnik

Dla zredukowania czasu deszyfracji (lub czasu generowania podpisu), możemy użyć małej wartości d zamiast losowegod. Ponieważ modułowe potęgowanie wymaga czasu liniowego w $\log_2 d$, małe d może poprawić wydajność o conajmniej współczynnik 10 (dla 1024 bitowego modułu). Niestety, atak M.Wienera pokazał ,że małe d daje w wyniku całkowite złamanie kryptosystemu.

Twierdzenie 2 (M.Wiener) Niech N = pq z $q < p < 2q$. Niech d <

$1/3N^{1/4}$. Przy danym ed=1 mod φ(N) Marcin może skutecznie odzyskać d.

Dowód

Dowód się opiera na podstawie przybliżania ułamków. Ponieważ ed=1 mod φ(N), istnieje takie k ,że ed-kφ(N) = 1 .Dlatego

$$\left|\frac{e}{\varphi(N)} - \frac{k}{d}\right| = \frac{1}{d\varphi(N)}.$$

Zatem k/d jest przybliżeniem e/φ(N).Chociaż Marcin nie zna φ(N), może użyć N do jego przybliżenia. Rzeczywiście, ponieważ φ(N)= N-p-q+ 1 i p+q-1 < 3√N, mamy |N -φ(N)| < 3√N. Używając N w miejsce φ(N) uzyskujemy

$$\begin{aligned}\left|\frac{e}{N} - \frac{k}{d}\right| &= \left|\frac{ed - k\varphi(N) - kN + k\varphi(N)}{Nd}\right| \\ &= \left|\frac{1 - k(N - \varphi(N))}{Nd}\right| \leq \left|\frac{3k\sqrt{N}}{Nd}\right| = \frac{3k}{d\sqrt{N}}.\end{aligned}$$

Teraz kφ(N) = ed-1 < ed/ Ponieważ e < φ(N) , widzimy ,że k < d < $1/3N^{1/4}$. Zatem otrzymujemy

$$\left|\frac{e}{N} - \frac{k}{d}\right| \leq \frac{1}{dN^{1/4}} < \frac{1}{2d^2}.$$

Jest to klasyczna relacja przybliżenia. Liczba ułamków k/d z d < N to w przybliżeniu e/N więc ściśle jest ograniczona przez $\log_2 N$. Faktycznie, wszystkie takie ułamki są uzyskiwane jako konwergencja ułąmka łańcuchowego e/N. Wszystko co trzeba zrobić to obliczyć log N konwergencji ułamka łańcuchowego dla e/N. Jeden z nich będzie równy k/d. Ponieważ ed-kφ(N) = 1, mamy gcd(k,d) = 1 a zatem k/d jest ułamkiem nieskracalnym. Jest to algorytm czasu liniowego dla odkrycia tajnego klucza d.

Ponieważ zazwyczaj N ma 1024 bity, wynika ,że d musi być przynajmniej długi na 256 bitów aby uniknąć takiego ataku. Jest to niefortunne dla energooszczędnych urządzeń takich jak "karty z mikroporcesorem", w których małe d powoduje duże oszczędności. Ni wszystko jednak stracone. Wiener opisuje kilka technik, które pomagają szybkiej deszyfracji i nie są podatne na ten atak.

Duże e : Załóżmy,że zamiast redukcji e modulo φ(N), użyjemy (N,e') dla kucza publicznego, gdzie e' = e+t* φ(N)dla dużego t. Wyraźne e' może być użyte w miejsce e dla szyfrowania waidomości. Jednak, kiedy duża wartość e jest używana, k w powyższym dowodzie nie jest dużo małe. Proste obliczenie pokazuje ,że jeśli $e' > N^{1.5}$ wtedy nie ważne jak małe jest d, powyższy atak nie może zostać zmontowany. Niestety, duże wartości e w wyniku zwiększają czas szyfrowania.

Użycie CRT: Alternatywnym podejście jest użycie chińskiego twierdzenia o resztach (CRT). Załóżmy ,że wybraliśy d takie ,że zarówno d_p = d mod (p-1) a d_q = d mod (q-1)są małe, powiedzmy 128 bitów każde. Wtedy szybko deszyfrujemy tekst zaszyfrowany C w

następujący sposób: najpierw obliczamy $M_p = C^{dp} \bmod p$ i $M_q = C^{dq} \bmod q$. Potem użyjemy CRT do obliczenia unikalnej wartości $M \in Z_N$ spełaniające $M = M_p \bmod p$ i $M = M_q \bmod q$. Wynik M spełniające $M = C^d \bmod N$ jest wymagany.Chcodzi o to ,że chociaż d_p i d_q są małe, wartość d mod φ(N) może być duża tj. na zlecenie φ(N). W wyniku tego ,atak z Twierdzenia 2 nie ma zastosowania. Zauważmy ,że jeśli (N,e) jest dane, istnieje atak pozwalający przeciwnikowi rozłożyć na czynniki N w czasie O (min ($\sqrt{d_p}$,$\sqrt{d_q}$). Zatem d_p i d_q nie mogą być zbyt małe.
Nie wiemy czy któraś z tych metod jest bezpieczna. Wszyscy wiemy ,że atak Wiener jest nieskuteczny wobec nich. Twierdzenie 2 zostało ostatnio ulepszone przez Boneha i Durfee, którzy pokazali ,że tak długo jak $d < N^{0.292}$, przeciwnik może skutecznie odkryć d z (N,e). Te wyniki pokazują ,że granica Wienera nie jest napieta. Jest prawdopodobne ,że poprawna granica to $d < N^{0.5}$.
Otwarty problem 2: Niech N = pq i $d < N^{0.5}$. Jeśli Marcin ma dane (N,e) z ed=1 mod φ(N) i e< φ(N), czy może skutecznie odkryć d?

4.Niski publiczny wykładnik

Aby zredukować czas szyfrowanie lub werydfikację podpisu jest w zwyaczaju używanie małego wykładnika publicznego e. Najmnijesza możliwa wartość dla e to 3, ale aby pokonć pewne ataki ,zalecana jest wartość $e = 2^{16}+1 = 65537$. Kiedy jest używana wartość $2^{16}+1$ weryfikacja podpsiu wymaga 17 mnożeń w przeciwieństwie do prawie 1000 kiedy używane jest losowe $e \leq \varphi(N)$. W przeciwieństwie do ataków z poprzedniej sekcji, kiedy stosuje się małe e są dalekie od całkowitego złamania.

4.1.Twierdzenie Coppersmitha

Najpopularnijeszym atakiem na niski wykładnik publiczny RSAQ jest oparty na twierdzeniu Coppersmitha. Twierdzenie Coppersmitha ma wiele aplikacji

Twierdzenie 2 (Coopersmith) Niech będzie liczbą całkowitą a $f \in Z[x]$ będzie wielomianem uformowanym stopnia d. Ustawiamy $X = N^{1/d-\epsilon}$ dla pewnego $\epsilon \geq 0$. Wtedy,przy danym (N,f) Marcin może skutecznie znaleźć wszystkie liczby całkowite $|x_0| < X$ spełaniającego $f(x_0) = 0 \bmod N$. Czas uruchamiania jest zdominowany przez czas potrzebny do uruchomienia algorytmu LLL na siatce o wymiarze O(w) z $w = \min(1/\epsilon, \log_2 N)$ .Twierdzenie dostarcza algortymu dla skutecznego znajdowania wszystkich pierwiastkó f modulo N , które są mniejsze niż $X = N^{1/d}$. Ponieważ X maleje, zmniejsza się czas uruchamiania algorytmu. Siłą twierdzenia jest jego zdolność do znajdowania małych pierwiastków wiloeminau modulo złożonego N. Kiedy pracujemy z modulo liczby pierwszej, nie ma powodu aby użyć twierdzenia Coppersmitha ponieważ istnieją lepsze algorytmy.MAmy szkic głównych idei twierdzenia Coppersmitha. Stosujemy uproszczone podejście Howgravea-Grahama. Dany jest wielomian $h(x) = \sum a_i x^i \in Z[x]$, definiujemy $||h||^2 = \sum_i |a_i|^2$. Dowód opiera się na poniższej obserwacji.
Lemat 4 Niech $h(x) \in Z[x]$ będzie wielomianem stopnia d i niech X będzie dodatnią liczbą całkowitą. Załóżmy ,ze $||h(xH)|| < N/\sqrt{d}$.

Jeśli $|x_0| < X$ spełnia $h(x_0) = 0 \bmod N$, wtedy $h(x_0) = 0$ przterzymuje liczba całkowita.

Dowód Z nierówności Schwarza mamy

$$\begin{aligned} |h(x_0)| &= \left|\sum a_i x_0^i\right| = \left|\sum a_i X^i \left(\frac{x_0}{X}\right)^i\right| \leq \sum \left|a_i X^i \left(\frac{x_0}{X}\right)^i\right| \\ &\leq \sum \left|a_i X^i\right| \leq \sqrt{d}\,\|h(xX)\| < N. \end{aligned}$$

Ponieważ $h(x_0) = 0 \bmod N$, uzyskujemy $h(x_0) = 0$

Lemat stanowi ,że jeśli h jest wielomianem z niską normą, wtedy wszystkie małe pierwiastki h mod N są również pierwiastkami h liczb całkowitych. Lemat sugeruje ,że aby znaleźć mały pierwiastek x_0 f(x) mod N powinniśmy szukać innego wielomianu $h \in Z[x]$ z małą normą mające ten same pierwiastki jak f modulo N. Wtedy x_0 będzie pierwiastkiem h w ciągu liczb całkowitych i może być łatwo znalezione. Aby to zrobić, możemy przeszukać wielomian $g \in Z[x]$ taki ,że h = gf ma niską normę, mniejszą niż N. Te ilości wyszukiwania dla kombinacji liniowej liczb całkowitych wielomianów $f, xf, x^2f, \ldots, x^rf$ z niską normą .Niestety najczęściej nie ma nietrywialnych kombinacji liniowych z normą wystarczająco małą. Coppersmith znalazł sztuczkę dla rozwiązania tego problemu: jeśli $f(x0) = 0 \bmod n$, wtedy $f(x_0)k = 0 \bmod N^k$ dla dowolnego k. Ogólnie definiujemy wielomiany:

$$g_{u,v}(x) = N^{m-v} x^u f(x)^v$$

dla predefiniowanych m. Wtedy x0 jest pierwiastkiem $g_{u,v}(x)$ modulo N^m dla dowolnego $u \geq 0$ i $0 \leq v \leq m$. Aby użyć lematu 4 musimy znaleźć kombinację liniową liczby liczby całkowitej h(x) wielomianu $g_{u,v}(x)$ taką że h(xX) ma normę mniejszą niż N^m. (przypominam ,że X jest górną granicą x_0 spełaniająca $X \leq N^{1/d}$). Dzięki swobodnej górnej granicy normy (N^m zmiast N), można powiedzieć ,że dla dostatecznie dużych m, zawsze istnieje kombinacja liniowa h(x) spełniająca wymaganą granicę. Po znalezieniu h(x), lemat 4 implikuje ,że ma x_0 jako pierwiastek w ciągu liczb całkowitych. W związku z tym x_0 może być łatwo znalezione. Pozostaje pokazać jak skutecznie znaleźć h(x). Aby to zrobić musimy najpierw ustanowić kilka faktów o siatcec w Z^w. Niech $u_1 \ldots u_w \in Z^w$ będzie wektorem liniowo niezależnym. Siatka L łączona przez $(u_1 \ldots u_w)$ jest zbiorem wszystkich całkowitych kombinacji liniowych $u_1 .. u_w$. Wyznacznik L jest definiowany jako wyznacznik macierzy kwadratowej w x w , której wiersze są wektorami $u_1 .. u_w$. W naszym przypadku, widzimy wielomian $g_{u,v}(xX)$ jako wektory i zbadamy siatkę L łączoną przez nie. Niech v = 0....,m a u = 0....,d-1,a zatem siatka ma wymiar w=d(m+1). Na przykład, kiedy f jest kwadratowym wielomianem unormowanym a m=3, siatka wynikowa jest łączona przez wiersze następującej macierzy:

$$
\begin{array}{c}
 \\
g_{0,0}(xX) \\
g_{1,0}(xX) \\
g_{0,1}(xX) \\
g_{1,1}(xX) \\
g_{0,2}(xX) \\
g_{1,2}(xX) \\
g_{0,3}(xX) \\
g_{1,3}(xX)
\end{array}
\begin{array}{c}
\begin{array}{cccccccc} 1 & x & x^2 & x^3 & x^4 & x^5 & x^6 & x^7 \end{array} \\
\left[\begin{array}{cccccccc}
N^3 & & & & & & & \\
 & XN^3 & & & & & & \\
* & * & X^2N^2 & & & & & \\
 & * & * & X^3N^2 & & & & \\
* & * & * & * & X^4N & & & \\
 & * & * & * & * & X^5N & & \\
* & * & * & * & * & * & X^6 & \\
 & * & * & * & * & * & * & X^7
\end{array}\right]
\end{array}
$$

Wyrazy * odpowiadają współczynnikom wielomianów, które ignorujemy. Wszystki puste wyrazy są zerami. Ponieważ macierz jest trójkątna, jej wyznacznik jest iloczynem elementów przekątnej. Naszym celem jest znalezienie krótkiego wektora w tej siatce. Klasyczny wynik Hermitea stanowi ,że dowolna siatka L wymiaru w zawiera niezerowy punkt v ϵ L którego norma L_2 spełania $||v|| \leq \gamma_w \det(L)^{1/w}$, gdzie γ_w jest stałe zależne tylko do w. Granica Hermite'a może być używana do pokazania że dla dość dużych m nasza siatka zawiera wektory o normach mniejszych niż N^m, jakie jest wymagane. Pytanie brzmi czy możemy skutecznie skonstruować krótki wektor w L którego długość nie jest większa niż granica Hermite'a. Algortym LLL jest wydajnym aklgorytmem , który robi to dość precyzyjnie.

Fakt 5 (LLL) Niech L będzie siatką łączoną przez $(u_1...u_w)$.Kiedy $(u_1...u_w)$ jest dane jako dana wejściowa, wtedy dane wyjściowe algorytmu LLL wskazując v ϵ L spełniające

$$\|v\| \leq 2^{w/4} \det(L)^{1/w}.$$

Algorytm LLL (nazwany na cześć twórców L.Lovasza, A.Lenstra i H.Lenstra Jr) ma wiele aplikacji zarównow w obliczweniowej teorii liczb i kryptografii. Jego odkrycie w 1982 roku dało wydajny algorytm dla faktorowania wielomianów w ciągu liczb całkowitych,ogólniej, przez pierścienie liczb. LLL jest często wykorzystywany do atakowania różnych kryptosystemów. Na przykład, wiele krytposystemów opiera się na "problemie plecakowym" złamany przy użyciu LLL. Używając LLL możemy całkowicie udowodnić twierdzenie Coppersmitha. Aby upewnić się ,że wektor stworzony przez LLL spełnia granice lematu 4 potrzebujemy

$$2^{w/4} \det(L)^{1/w} < N^m/\sqrt{w},$$

gdzie w=d(m+1) jest wymiarem L. Podprogram obliczeniowy pokazuje ,że dla dość dużego m, granica jest spełniona. Rzeczywiście, kiedy $X = N^{1/d- \epsilon}$, wystarczy zwrócić m=O(k/d) z k = min(1/ϵ, log N). W konsekwencji, czas uruchamiania jest zdominowany przez uruchomienie LLL w siatce o wymiarze O(k), co

jest wymagane. Naturalnym pytaniem jest czy twierdzenie Coppersmitha może być stosowane do wielomianów dwuwymiarowych i wielowymiarowych . Jeśli dla danego $f(x,y) \in Z_N[x,y]$ jest dany pierwiastek (x_0,y_0) z $|x_0y_0|$ odpowiednio ograniczonymi, czy Marcin oże skutecznie znaleźć (x_0,y_0)? Chociaż ta sama technika pojawiła się przy pracy z wielomianami dwuwymiarowymi, aktualnie jest problemem do udowodnienia. Ponieważ coraz więcej wyników zależy od dwuwymiarowego rozszerzenia twierdzenia Coppersmitha, będzie bardzo przydatny rygorystyczny algorytm.

Otwarty Problem 3 Znajdź ogólne warunki na jakich twierdzenie Coppersmitha może być uogólnione na wielomiany dwuwymiarowe.

4.2 Atak Hastada

Jako pierwsze zastosowanie twierdzenia Coppersmitha zaprezentyjemy poprawiony stary atak Hastada. Załóżmy ,że Bob życzy sobie wysłać zaszyfrowną wiadomość M do kilku jednostek $P_1,P_2,....P_k$. Każda jednostka ma swój własny klucz RSA (N_i,e_i).Załóżmy ,że M jest mniejsze niż wszystkie N_i. Naiwnością byłoby wysyłanie M szyfując przez Boba używając każdego publicznego klucza i wysłanie i-te zaszyfrowanych tekstów do P_i. Atakujący Marcin musi podsłuchać połączenia Boba i zebrać k transmitowanych tekstów zaszyfrowanych. Dla uproszczenia, załóżmy ,że wszystki publiczne wykładniki e_i są równe 3. Prosty argument pokazuje że Marcin może odzyskać M jeśli $k \geq 3$. Rzeczywiście. Marcin uzyska C_1,C_2,C_3 , gdzie

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

Możemy założyć ,że $\gcd(N_i,N_j) = 1$ dla wszystkich $i \neq j$ ponieważ w przeciwnym razie Marcin może rozłożyć na czynniki pewne Ni. Zatem, stosując chińskie twierdzenie o resztach (CRT) do C_1,C_2,C_3 daje $C' \in Z_{N1N2N3}$ spełaniające $C' = M3 \bmod N_1N_2N_3$ .Ponieważ M jest mniejsze niż wszystkie Ni , mamy $M^3 < N_1N_2N_3$. Wtedy $C'=M^3$ przechowywane jest w ciągu liczb całkowitych. Zatem, Marcin może odkryć M przez obliczenie rzeczywistego pierwiastka sześciennego z C'. Ogólnie, jeśli wszystkie publiczne wykładniki są równe e, Marcin może odkryć M gdy $k \geq e$. Atak jest możliwy tylko jeśli używamy małego e. Hastad opisał o wiele silniejszy atak. Aby ukierunkować wynik Hastada, rozważmy naiwną obronę przeciw powyższemu atakowi. Zamiast transmisji szyfrowania M, Bob może "wypełnić' wiadomość przed szyfrowaniem. Na przykład jeśli M jest długa na m bitów, Bob może wysłać $Mi = i2^m+M$ do jednostek P_i. Ponieważ Marcin uzyskuje zaszyfrowane różne wiadomości, nie może zmontować ataku. Niestety Hastad wykazał ,że to liniowe wypełnienie nie jest bezpieczne. Faktycznie, udowodnił ,że zastosowanie dowolnego stałego wielomianu do wiadomości przed szyfrowanie nie zabezpiecza przed atakiem. Załóżmy ,że dla każdego uczestnika $P_1....P_k$,Bob ma stały publiczny wielomian $f_i \in Z_N[x]$. Dal przesłania wiadomości M, wysyła zaszyfrowane $f_i(M)$ do jednostki P_i. Przez podsłuchiwanie, Marcin uczy się ,żę $C_i=f_i(M)^{e_i} \bmod Ni$ dla i = 1...k. Hastad wykazał ,że jeśli dość jednostek jest zaangażowanych, Marcin moze odzyskać tekst jawny M ze wszystkich tekstów zaszyfrowanych. Poniższe

twierdzenie jest silniejszą wersją oryginalnego wyniku Hastada

Twierdzenie 6 (Hastad) Niech $N_i...N_k$ będą parą względnie liczb pierwszych i ustawiamy $N_{min} = min_i(N_i)$. Niech $g \in Z_{ni}[x]$ będzie k wielomianami maksymalnego stopnia d. Załóżmy ,że istnieje unikalne $M < N_{min}$ spełaniajce

$$g_i(M) = 0 \bmod N_i \text{ dla wszystkich } i = 1...k$$

Przy założeniu ,że k > d, można skutecznie znaleźćM przy danym $(N_i, g_i)^k{}_{i=k}$.

Dowód. Niech $\overline{N} = N_1...N_k$. Możemy założyć ,że wszystkie g_i są unormowane. Przez pomnożenie każdego g_i przez właściwą potęgę x, możemy założyć, że wszystkie mają stopień d. Zbudujemy wielomian

$$g(x) = \sum_{i=1}^{k} T_i g_i(x), \quad \text{gdzie } T_i = \begin{array}{ll} 1 \bmod N_j & \text{if } i = j \\ 0 \bmod N_j & \text{if } i \neq j. \end{array}$$

T_i są liczbami zwanymi współczynnikami chińskiej reszty. Wtedy g9x) musi być unormowane ponieważ jest unormowanym modułem wszystkich N_i. Jego stopień to d. Co więcej, wiemy ,że $g(M) = 0 \bmod \overline{N}$ .Twierdzenie 6 teraz idzie za Twierdzeniem 3 ponieważ

$$M < N_{min} \leq \overline{N}^{1/k} < \overline{N}^{1/d}.$$

To twierdzenie pokazuje ,że system jednowymiarowych równań modulo liczb względnie pierwszych kompozytów można skutecznie rozwiązać, zakładając ,że dostarczono wystarczającą ilość równań. Przez ustawienie $g_i = f^{e_i}{}_i - C_i \bmod N_i$, zobaczymy ,że Marcin może odzyskać M z danego tekstu zaszyfrowanego kiedy liczba jednostek to przynajmniej d, gdzie d jest maksymalnym $e_i deg(f_i)$ przez wszystkie i =1....k. W szczególności, jeśli wszystkie e_i są równe e a Bob wysyła liniowo powiązane wiadomości, wtedy Marcin może odzyskać tekst ajwny jeśli k > e. Oryginalne twierdzenie Hastada jkest słabsze niż omówione powyżej. Zamiast d wielomianów, Hastad wymaga d(d+1)/2 wielomianów. Dowód Hastada jest podobny do twierdzenia Coppersmitha. Jednak Hastad nie używa potęg g w siatce i w konsekwencji uzyskuje słabsze granice.

4.3 Atak Franklina-Reitera

Franklin i Reiter znaleźli sprytny atak keidy Bob wysyłaAlicji zazyfrowaną wiadomość używając tego samego modułu. Niech (N,e) będzie kluczem publicznym.Załóżmy ,żę $M_1 M_2 \in Z^*_N$ są dwoma odrębnymi komunikatami spełniającymi $M_1 = f(M_2) \bmod N$ dla publicznie znanego wielomianu $f \in Z_N[x]$. Aby wysłać M_1 i M_2 do Alicji, Bob może zaszyfrować i przesłać wynikowy tekst zaszyfrowany $C_1 C_2$. Pokażemy ,że przy danych $C_1 C_2$, Marcin może łatwo uzyskać M_1, M_2. Chociaż atak działa dla małych e, ustanowimy poniższy lemat dla e = 3 aby uprościć dowód.

Lemat 7 (FR) Usatwiamy e = 3 i niech (N,e) będzie kluczem publicznym RSA. Niech $M1 \neq M2 \in Z^*_N$ spełniającego $M_1 = f(M_2) \bmod N$ dla pewnego liniowego wielomianu $f = ax+b \in Z_N[x]$ z $b \neq 0$.Wtedy, rzy danych (N, e, C_1, C_2, f), Marcin może odzyskać M_1, M_2 w czasie kwadratowym w log N.

Dowód. Ustawimy tą część ogólnym dowodem, ustanowimy użycie arbitralnego e (zamiast ograniczonego do e=3) .Ponieważ C1 = M^e_1 mod N, wiemy ,że M_2 jest pierwiastkiem wielomianu $g_1(x) = f(x)^e - C_1 \in Z_N[x]$. Podobnie M2 jest pierwiastkiem $g_2(x) = x^e - C_2 \in Z_N[x]$. Liniowy współczynników $x-M_2$ dzieli oba wielomiany. Dlatego ,Marcin może użyć algorytmu Euklidesa do obliczenia gcd g_1 i g_2. Jeśli gcd okazuje się być liniowy, M_2 jest znalezione. gcd może być obliczone w czasie kwadratowym w e i log N. Pokazujemy ,że kiedy e = 3, gcd musi być liniowy. Wielomian x^3-C_2 dzieli współczynniki modulo zarówno p i q na czynnik liniowy i nieredukowalny czynnik kwadratowy (pamiętaj ,że gcd(e,φ(N))=1 a zatem x^3-C_2 ma tylko jeden pierwiastek kwadratowy w Z_N).Ponieważ g_2 nie może dzielić g1, gcd musi być liniowy. Dla e > 3 gcd jest prawie zawsze liniowy. Jednak dla pewnych rzadkich M_1,M_2 i f możliwe jest uzyskanie nieliniowego gcd, a w takim przypadku atak nie wypali. Dla e > 3 atak zabiera czas kwadratowy w e. W konsekwencji, może być stosowany tylko kiedy mały publiczny wykładnik e jest używany. Dla dużego e praca przy obliczeniu gcd jest zaporowa. Jest interesującym pytaniem opracowanie takiego ataku dla dowolnych e. W szczególności czy może powyższe gcd g_1 i g_2 w czasie wielomianowym w log e?

4.4 Atak Coppersmitha krótkim wypełnieniem

Atak Franklina-Reitera może wydawać się nieco sztuczny. Po tym wszystkim, dlaczego Bob wysyła Alicji zaszyfrowaną powiązaną . Wiadomość? Coppersmith wzmocnił tak i okazał ważny wynik w wypełnianiu. Naiwny losowy algorytm wypełanienia może wypełnić tekst jawny M przez dołączenie kilku losowych bitów do jednego z konców. Poniższy atak wykazuje niebezpieczeństwo takiego uproszczonego wypełnienia. Załóżmy ,że Bob wysła właściwie wypełnione zaszyfrowane M do Alicji. Atakujący Marcin, przechwytuje tekst zaszyfrowany i zabezpiecza przed osiągnięcie przez niego miejsca docelowego. Bob zauważ ,że Alicja nie odpowiada na jego wiadomość i postanawia ponownie wysłać M do Alicji. Losowo wypełnia M i przesyła wynikowy tekst zaszyfrowany. Teraz Marcin ma dwie zaszyfrowane teksty odpowiadające dwóm szyfrowaniom tej samej wiadomości przy użyciu dwóch różnych losowych wypełnień. Poniższe twierdzenia pokazuje ,że chociaż nie zna użytych wypełnień, Marcin może uzyskać tekst jawny.

Twierdzenie 8 Niech (N,e) będzie kluczem publicznym RSA gdzie N jest n-bitowej długości. Ustawiamy m = $[n/e^2]$.Niech $M \in Z^*_N$ będzie wiadomością o długości co najmniej n − m bitów. Definiujemy $M_1=2^mM+r_1$ i $M_2=2^mM+r_2$, gdzie r_1 i r_2 są różnymi liczbami całkowitymi z $0 \le r_1, r_2 < 2^m$. Jeśli Marcin ma dane (N,e) i zaszyfrowane C_1,C_2 z M_1, M_2 (ale nie ma danego r_1 lub r_2), może skutecznie odkryć M.

Dowód. Definiujemy $g_1(x,y) = x^e-C_1$ i $g2(x,y) = (x+y)^e-C_2$. Wiemy ,że kiedy $y = r_2-r_1$, wielomiany te mają M_1 jako wspólny pierwiastek. Innymi słowy, $\Delta = r_2-r_1$ jest pierwiastkiem "wynikowym" $h(y) = res_x(g_1,g_2) \in Z_N[y]$. Stopień h jest co najmniej e^2. Co więcej, $|\Delta| < 2m < N^{1/e2}$. Zatem, Δ jest małym pierwiastkiem h modulo N, a Marcin może skutecznie go znaleźć używając twierdzenia Coppersmitha (Twierdzenie 3). Kiedy Δ jest znane, atak Franklina -Reitera z poprzedniej części może być zastosowany do odzyskania M_2, a w

konsekwencji M. Kiedy e=3, atak może być montowany tak długo jak długość wypełnienia jest mniejsza niż $1/9^{n\text{-tej}}$ długości wiadomości. Jest to ważny wynik. Zwróć uwagę że dla zalecanej wartości e=65537, atak jest bezużyteczny przeciwko standardowym rozmiarom modułu.

4.5. Atak częściową ekspozycją klucza

Niech (N,d) będzie prywatnym klczuem RSA. Załóżmy ,że w jakiś sposób Marcin jest w stanie udostępnić częścbitów d, powiedzmy , jedną czwartą z nich. Czy może zrekonsrtuować resztę d? Niespodziewanie odpowiedź jest pozytywna kiedy odpowiedni klucz prywatny jest mały. Boneh,Durfee i Frankel wykazali ,że tak długo jak e < √N, możliwe jest zrekonstruowanie całego d z fragmentów jego bitów. Te wyniki ilustrują ważność zabezpieczania całego klucza prywatnego RSA.

Twierdzenie 9 (BDF) Niech (N,d) będzie kluczem prywatnym RSA w którym N jest długie n bitów. Przy danych [n/4] mniej znaczących bitów d, Marcin może zrekonstruować całe d w czasie liniowym $e\log_2 e$.

Dowód opiera się na innym twierdzeniu Coppersmitha

Twierdzenie 10 (Coppersmith) Niech N = pq będzie n-bitowym modułem RSA. Wtedy przy danych n/4 mniej znaczących bitach p lub n/r bardziej znaczących bitach p, można skutecznie rozłożyć na czynniki N

Twierdzenie 9 wynika z Twierdzenia 10. Faktycznie, przez zdefiniowanie e i d, istnieje liczba całkowita k taka ,że

$$ed - k(N - p - q + 1) = 1.$$

Ponieważ d < φ(N), musimy mieć 0 < k ≤ e. Redukując równanie modulo $2^{n/4}$ i ustawiajac q = N/p uzyskujemy

$$(ed)p - kp(N - p + 1) + kN = p \pmod{2^{n/4}}.$$

Ponieważ Marcin ma dane n/4 mniej znaczących bitów d, zna wartość ed mod $2^{n/4}$. W konsekwencji, uzyskuje równanie w k i p. Dla każdego e możliwą wartość k, Marcin rozwiązuje równanie kwadratowe w p i uzyskuje klika wartości kandydujących do p mod $2^{n/4}$. Dla każdej z tych kandydujących wartości, uruchamia algorytm z Twierdzenia 10 próbując rozłożyć na czynniki N. Można wykazac ,że całkowita liczba kandydujących wartości dla p mod $2^{n/4}$ to co najmniej $e\log_2 e$. Zatem po $e\log_2 e$ próbach N będzie rozłożone na czynniki.

Twierdzenie 9 jest znane jako atak częściowej ekspozycji klucza. Podobne ataki istnieją dla większych wartości e tak długo jak e < √N. Jednak te techniki są trochę bardziej złożone. Interesujące jest ,że kryptosystemy oparte na logarytmie dyskretnym, takie jak system klucza publicznego ElGamal nie wydają się podatne na częściową ekspozycję klucza. Rzeczywiście, jeśli g^x mod p i stały ułamek bitów są dane, nie jest znany algorytm czasu wielomianowego do obliczenia reszty x. Na koniec pokażę ,że kiedy wykładnik szyfrowania e jest mały, system RSA ujawnia połowę bardziej znaczących bitów odpowiedniego klucza prywantego d. Aby to zobaczyć, rozważmy pnownie równanie ed-k(N-p-q+1) = 1 dla liczby całkowitej 0 < k ≤ e. Przy danym k, Marcin może łatwo obliczyć

$$\widehat{d} = \lfloor (kN+1)/e \rfloor$$

Wtedy

$$|\widehat{d} - d| \leq k(p+q)/e \leq 3k\sqrt{N}/e < 3\sqrt{N}.$$

Zatem $\widehat{d}$ jest dobrym przybliżeniem dla d. Granica pokazuje ,że dla większości d, połowa bardziej znaczących bitów $\widehat{d}$ jest równa tym z d. Ponieważ są tylko e możliwe wartości dla k, Marcin może konstruować mały zbiór o rozmiarze e taki ,że jeden z elementów w tym zbiorze jest równy połowie bardziej znaczących bitów d. Przypadek e=3 jest szczególnie interesujący W tym przypadku można wykazać ,że zawsze k=2, a zatem system całkowicie przepuszcza połowę bardziej znaczących bitów z d.

5. Ataki na implementację

Zwrócimy uwagę na zupełnie inną klasę ataków. Zamiast atakować podstawową strukturę funkcji RSA, ataki te skupiają się implementacji RSA

5.1 Atak czasowy

Rozważmy kartę chipową, która przechowuje prywatny klucz RSA. Ponieważ karta jest zabezpieczona, atakujący Marcin nie może zbadać jej zawartości i wyeksponować klucza. Jednak sprytny atak Kochera pokazuje ,że przez precyzyjne zmierzenie czasu potrzebnego do wykonania kary deszyfrowania RSA (lub podpisu) Marcin może szybko odkryć prywatny wykładnik dezyfrowania .Wyjaśnimy jak zmontować atak przeciwko prostej implementacji RSA używając "algorytmu powtarzania kwadratury". Niech d = $d_n d_{n-1},...d_0$ będzie binraną reprezentacją d to jest

$$d = \sum_{i=0}^{n} 2^i d_i$$

z di ϵ {0,1}. Algorytm powtarzającej się kwadratury oblicza C = M^d mod N używając co najmniej 2n modularnych mnożeń. Jest to oparte na obserwacji ,że

$$C = \prod_{i=0}^{n} M^{2^i d_i} \bmod N.$$

Algorytm działa jak następuje:
Ustawiamy z równe M i C równe 1. Dla i=0,...n, wykonujemy następujące kroki:
(1)jeśli di = 1 ustawiamy C równe C*z mod N,
(2)ustawiamy z równe z^2 mod N
Na koniec, C ma wartość M^d mod N
Zmienna z przebiega przez zbiór wartości $M^{2^i} \bmod N$ dla i=0,..n. Zmienna C "zbiera" właściwe potęgi w zbiorze, uzyskując M^d mod N.
Montując atak, Marcin prosi kartę o wygenerowanie podpisów na

dużej ilości losowych wiadomości $M_1,..M_k \in Z^*_N$ i mierzy czas T_i potrzebny karcie na wygenerowanie każdego podpisu. Atak odkrywa bity d po kolei zaczynając od najmniej znaczących bitów. Wiemy że d jest nieparzyste. Zatem $d_0=1$. Rozważmy drugą iteracje. Początkowo $z = M^2 \bmod N$ a C = M. Jeśli $d_1 ==1$, karta wylicza iloczyn $C*z = M*M^2 \bmod N$. W przeciwnym razie nie. Niech t_i będzie czasem potrzebnym karcie na obliczenie $M_i*M^2 \bmod N$.t_i różni się jedno od drugiego ponieważ czas obliczenia $M_i*M_i^2 \bmod N$ zależy od wartości M_i (prosty algorytm redukcji modularnej pobiera różne ilości czasu w zależnosci od wartości będącej redukowaną) Marcin mierzy t_i offline przed zmontowaniem ataku kiedy uzyskuje fizyczną specyfikację karty. Kocher zaobserwował ,że kiedy $d_1=1$,$\{t_i\}$ i $\{T_i\}$ są skorelowane. Na przykład, jeśli dla jakiegoś i, t_i jest dużo większe niż oczekiwano, wtedy T_i jest również większy niż oczekiwano. Z drugiej strony, jeśli $d_1=0$ wtedy $\{t_i\}$ i $\{T_i\}$ zachowują się jak niezależne zmienne losowe. Przez zmierzenie korelacji,Marcin może określić czy d_1 to 0 czy 1. Kontynuując w ten sposób może odkryć d_2, d_3 itd. Zauwa że kiedy niski publiczny eksponent e jest używany, atak częściową ekspozycją klucza z poprzedniej sekcji pokazuje ,że atak czasowy Kochera musi być zatrudniony do odkrycia jednej czwartej bitów d. Są dwa spsoby obrony przeciwko temu atakowi. Najprostszym jesy dodanie właściwego opóźnienia tak aby potęgowanie modularne zawsze zabierało stałą ilość czasu. Drugie podejście, ze względu na Rivesta, jest oparte na oślepianiu. Przed deszyfrowaniem M karta wybeira losowo $r \in Z^*_N$ i oblicza $M'=M*r^e \bmod N$. Potem stosuje d do M' i uzyskuje $C'=(M')^d \bmod N$ .W końcu karta ustawia $C =C'/r \bmod N$. Przy takim podejściu, karty stosują d do losowej wiadomości M' nieznanej Marcinowi .W wyniku tego Marcin nie może zmontować ataku. Kocher odkrył inny atak w tym kierunku nazwany kryptoanaliza mocy. Kocher wykazał ,że przez precyzyjne zmierzenie zużycia energii karty podczs generowania podpisu, Marcin może często łatwo odkryć tajny klucz. Jak się okazuje, podczas mnożenia o dużej preczyji, pobór energii karty jest większy niż normalnie. Mierząc długość okresów wysokiej konsumpcji, Marcin może łatwo określić czy w danej iteracji karta wykonuje jedno czy dwa mnożenia, odsłaniając bity d.

5.2 Błędy losowe

Deszyfrowanie implementacji RSA i podpisów często używa chińskiego twierdzenia o resztach dla przyspieszenia obliczenia $M^d \bmod N$ .Zamiast działania modulop N, podpisujący Bob najpierw oblicza podpisy modulo p i q a potem łączy wyniki używając chińskiego twierdzenia o resztach. Precyzyjniej, Bob najpierw oblicza

$$C_p = M^{d_p} \bmod p \text{ i } \quad C_q = M^{d_q} \bmod q,$$

gdzie $d_p = d \bmod (p-1)$ a $d_q = d \bmod (q-1)$. Potem uzyskuje podpis C przez ustawienie

$$C = T_1 C_p + T_2 C_q \pmod N,$$

gdzie

$$T_1 = \left\{ \begin{array}{l} 1 \bmod p \\ 0 \bmod q \end{array} \right\} \qquad \text{i} \qquad T_2 = \left\{ \begin{array}{l} 0 \bmod p \\ 1 \bmod q \end{array} \right\}$$

Czas uruchomienia ostatniego kroku CRT jest niewielki w stosunku do dwóch potęgowań. Zauważ ,że p i q są połową długości N. Ponieważ proste implementacje mnożenie pobierają czas kwadratowy, mnożenie modulo p jest cztery razy szybsze niż modulo N. Co więcej, d_p jest połową długości d i w konsekwencji obliczenie

$M^{d_p} \bmod p$ jest osiem razy szybsze niż obliczenie M^d mod N. Całkowity czas podpisu jest zatem zredukowany przez współczynnik cztery. Wiele implementacji używa tej metody do poprawy wydajności. Boneh, DeMilo i Lipton zaobserwowali ,że jest nieodłączne niebezpieczeństwo użycia metody CRT. Załóżmy ,że podczs generowania podpisu, usterka na komputerze Boba powoduje złe obliczenie w jednej instrukcji. Powiedzmy podczas kopiowania rejestru z jednego miejsca do drugiego, jedne z bitów został odwrócony. Przy danym niepoprawnym podpisie, Marcin może łatwo rozłożyć na czynniki modulo N Boba. Zaprezentowaliśmy wersję ataku opisanego przez A.K. Lenstra. Załóżmy ,że pojedynczy błąd wystąpił kiedy Bob generował podpis. W wyniku tego, dokładnie jedno C_p lub C_q będzie obliczone niepoprawnie. Powiedzmu C_p jest poprawne ale $\widehat{C}_q$ już nie .Podpis wynikowy to $\widehat{C} = T_1 C_p + T_2 \widehat{C}_q$ .Kiedy Marcin odbierze $\widehat{C}$, wie ,że jest to fałszywy podpis ponieważ $\widehat{C}^e \neq M \bmod N$. Jednak zauważ ,że

$$\widehat{C}^e = M \bmod p \qquad \text{podczas gdy} \qquad \widehat{C}^e \neq M \bmod q.$$

Jako wynik $\gcd(N, \widehat{C}^e - M)$ eksponuje nietrywialne współczynniki N. Aby atak zadziałał, Marcin musi mieć pełną wiedzę o M. Mianowicie, zakładamy ,że Bob nie użył procedury losowego wypełnienia. Losowe wypełnienie przed podpisem udaremnia atak. Prostszą obroną jest dla Boba sprawdzenie wygenrowanego podpisu przed wysłaniem go w świat. Sprawdzenie jest szczególnie ważne kiedy używamy przyspieszonej metody CRT. Błędy losowe są niebezpieczne dla wielu systemów kryptograficznych. Wiele systemów obejmujących implementację RSA nie-CRT może być zaatakowanych przy użyciu błędów losowych. Jednak te wyniki są bardziej teoretyczne.

5.3 Atak Bleichenbachera na PKCS 1

Niech N będzie n-bitowym modułem RSA a M będzie m bitową wiadomością z m < n. Przed zastosowaniem szyfrowania RSA naturalne jest wypełnienie wiadomości M do n bitów przez dołączenie losowych bitów do niej. Stara wersja standardu jest znana jako Public Key Cryptography Standard 1 (PKCS 1) używa tego podejścia. Po wypełnieniu wiadomość wygląda tak:

02	Random	00	M

Wiadomość wynikowa jest długa na n bitów i jest bezpośrednio zaszyfrowana z RSA. Początkowy blok zawiera "02" i jest długi na 16 bitów i jest tam aby wsklazywać ,ze losowe wypełnienie zostało dodane do wiadomości. Kiedy wiadomość PKCS 1 jest odbierana przez komputer Boba, aplikacja (np,. Przeglądarka internetowa) odszyfrowuje ją, sprawdzając blok początkowy i wydziela losowe wypełnienie. Jednak, niektóre aplikacje sprawdzają blok początkowy dla "02" i jeśli go nie ma odsyłają komunikat błędu "nieporawny tekst zaszyfrowany". Bleichenbacher wykazał ,ze ten komunikat błędu moze prowadzić do poważnych konsekwencji: używając komunikatu błędu, Marcin może odszyfrować tekst zaszyfrowany. Załóżmy ,że Marcin przechwycił tekst zaszyfrowany C skierowany do Boba i oczekuje na jego odszyfrowanie. Montując atak, Marcin wybiera losowo $r \in Z^*_N$ i, wylicza C'=rC mod N,i wysyła C' do Boba. Ap[likacja uruchomiona na maszynie Boba odbiera C' i próbuje ją zdeszyfrować. I odpowiada komunikatem błędu lub nie odpowiada wcale (jeśli C' wydaje się być właściwie sformatowane) Zatem, Marcin nauczył się ,że najbardziej znaczące 16 bitów deszyfrowania C' jest równe 02. W efekcie, Marcin ma wyrocznię, która testuje dla niego czy 16 najbardziej znaczących bitów z deszyfrowania rc mod N jest równe 02, dla r z jego wyboru. Bleichenbacher wykazał ,ze taka wyrocznia jest wydajna dla odszyfrowywania C

.