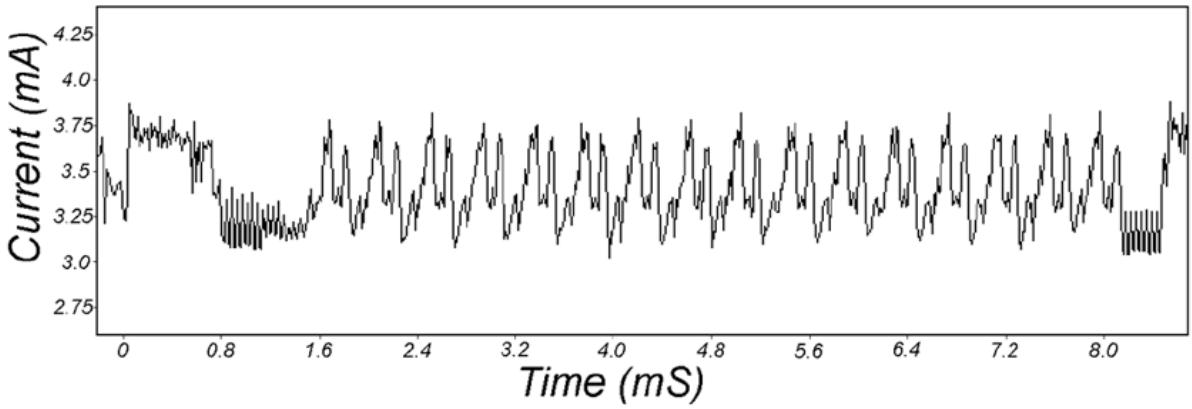
Wykład 8.

Temat: Różnicowa analiza mocy DES, analiza czasowa, różnicowa analiza błędów

8.1. Różnicowa analiza mocy DES

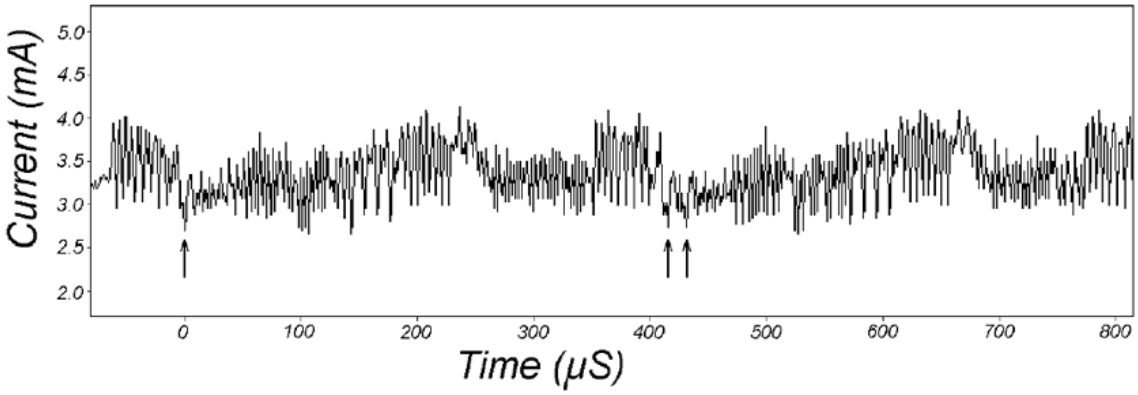
Wiele z technik kryptoanalizy pozwala na testowanie teoretycznej siły algorytmu. Przykładami takich technik są kryptoanaliza liniowa, czy tez kryptoanaliza różnicowa. Prawie każdy szyfr, oprócz implementacji programowej posiada również implementację sprzętową. Wiele z rozwiązań sprzętowych algorytmu, opartych jest na układach półprzewodnikowych. Możliwe jest zatem dokonanie pomiarów charakterystyk prądowo-napięciowych takich urządzeń. Okazuje się, że charakterystyki takie mogą dać wiele użytecznych informacji o realizowanych przez układ kryptograficznych operacjach, a nawet o kluczu.

Jedną z takich technik analizy jest tzw. ***prosta analiza mocy, SPA*** (simple power analysis), której zasadniczym fundamentem jest interpretacja poboru mocy układu, podczas wykonywania operacji kryptograficznych. Przykładowy oscylogram zdjęty z karty realizującej szyfrowanie DES został przedstawiony poniżej.



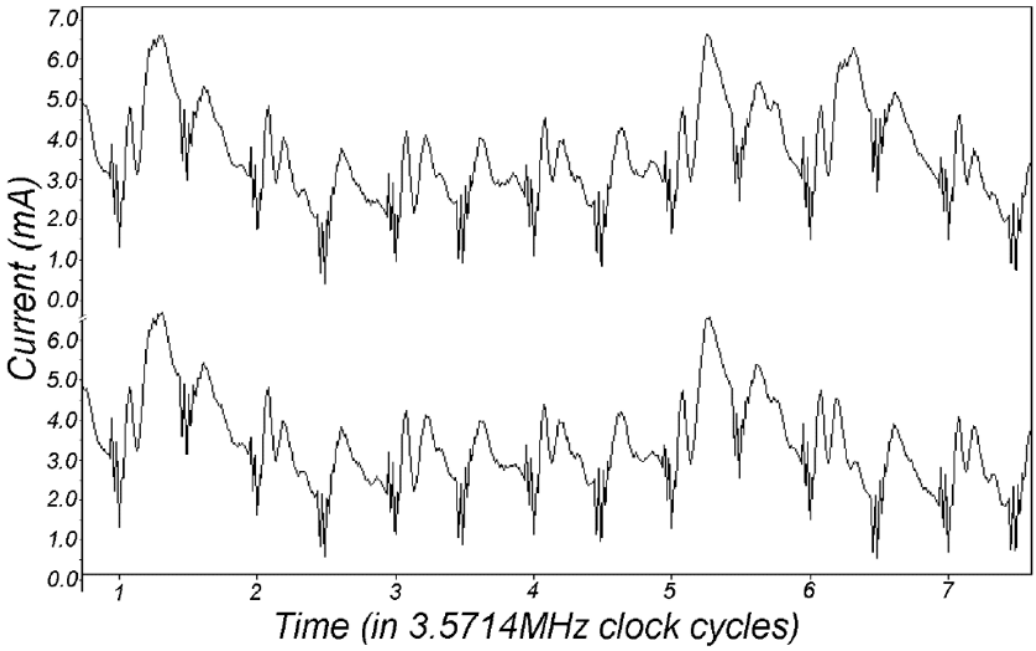
Rys. 8.1 Oscylogram pracy karty realizującej szyfrowanie DES

W oscylogramie tym można łatwo zauważyć 16 cykli pracy DES. Każdy jednak cykl ma nieco inny kształt, nieznacznie odbiegający od innych. Dokładniejszy obraz cyklu nr 2 i nr 3 został przedstawiony na rys. 8.2.



Rys. 8.2 Oscylogram pracy karty realizującej szyfrowanie DES - cykl 2 i 3

Analizując powyższy oscylogram, można zaobserwować znacznie więcej operacji. Przykładowo, w drugim cyklu 28-bitowe rejestry C i D (rozszerzenie klucza) są przesuwane o jedną pozycję, zaś w cyklu trzecim - o dwie pozycje, co zaznaczone zostało strzałkami. Wiele innych miejsc charakterystyk, stanowią wady SPA, wywołane skokami warunkowymi. Rys. 8.3 przedstawia oscylogram poboru mocy dla dwóch obszarów, każdy po siedem cykli zegarowych 3,5714MHz. Widoczne różnice pomiędzy cyklami zegarowymi wywołane są głównie przez różnice w poborze mocy mikroprocesora przy wykonywaniu różnych instrukcji. Górna charakterystyka przedstawia wykonywaną sekwencję mikroprocesora z wykonaniem skoku, zaś dolna taką samą sekwencję bez wykonania skoku - cykl 6.



Rys. 8.3 Oscylogram pracy karty realizującej szyfrowanie DES - dla dwóch 7-cyklowych wariantów

Na podstawie wielu obserwacji i po analizie porównawczej wielu charakterystyk, dla różnych tekstów jawnych i kluczy, znając zasadę pracy DES, możliwe jest więc uzyskanie informacji dotyczących sekwencji wykonywanych instrukcji przy wykonywaniu przez mikroprocesor operacji kryptograficznych. Kolejność wykonywania tych instrukcji oraz ich rodzaj zależny jest od przetwarzanych danych. Przykładami mogą być:

- ***procedura rozszerzenia klucza*** - w DES związana jest ona z cyklicznym przesuwaniem zawartości rejestrów C i D. W zależności od wartości bitów, otrzymać można różne charakterystyki;
- ***permutacje DES*** - implementacje DES dokonują wielu permutacji. Warunkowe obejścia w wykonywanym kodzie przyczyniają się do różnych charakterystyk prądowych dla różnych wartości bitów;
- ***porównania*** - porównania łańcuchów lub obszarów pamięci zazwyczaj realizowane są przy użyciu warunkowych obejść;
- ***potęgowania*** - zwykłe potęgowanie modularne powoduje podnoszenie wartości do kwadratu przy wykonaniu każdej iteracji, co powoduje powtarzanie charakterystyk dla każdej iteracji. Na podstawie ich kształtu oraz liczby można zatem oszacować wielkość wykładnika.

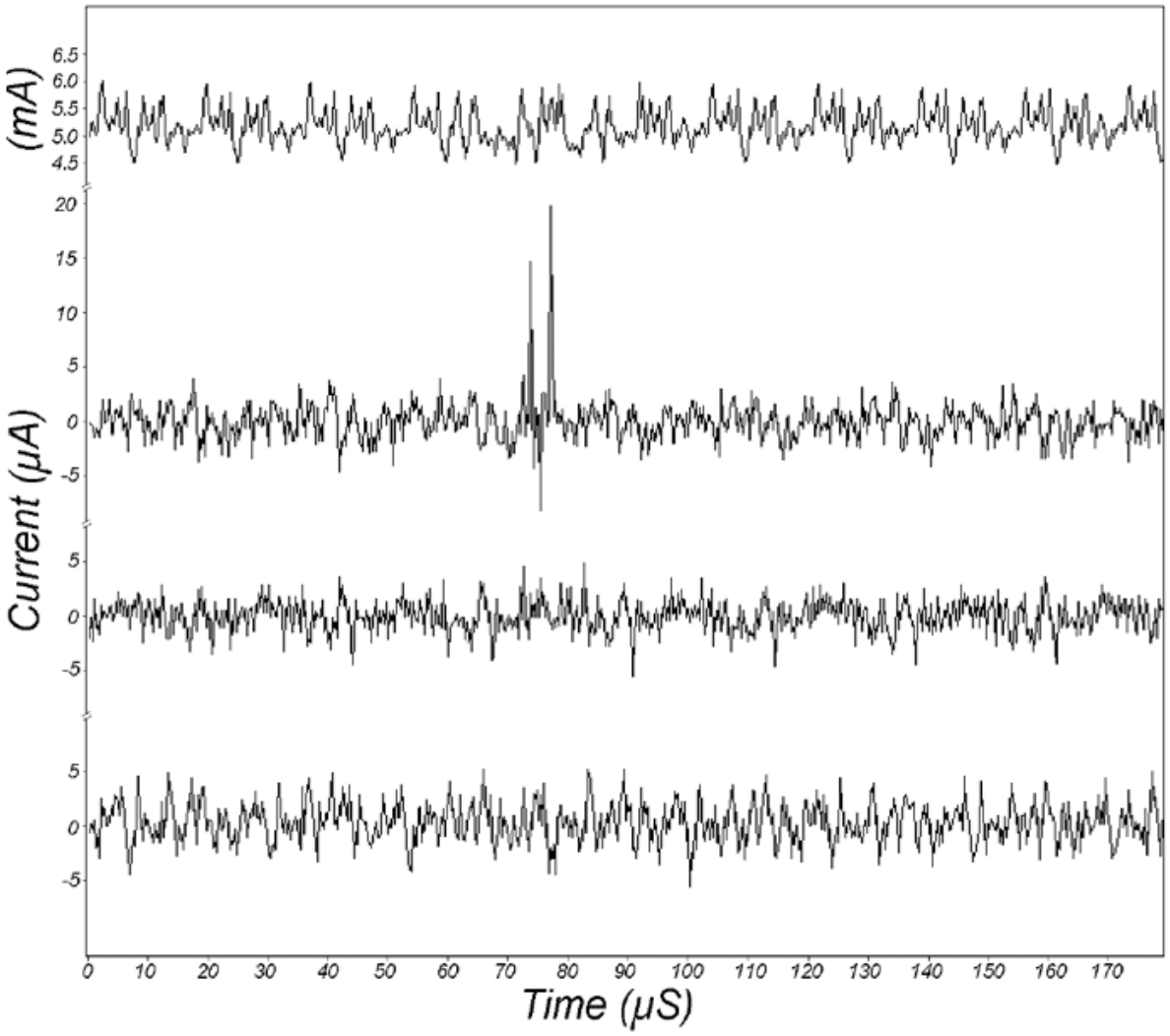
Drugą techniką jest ***różnicowa analiza (poboru) mocy DPA*** (differential power analysis). Atak na DES za pomocą DPA składa się z dwóch faz. W pierwszej fazie należy dokonać pomiarów charakterystyk ok. 1000 operacji DES. Druga faza opiera się na tworzeniu charakterystyk różnicowych powyższych pomiarów i ich analizie. W tym celu konstruowana jest funkcja wybierająca $D(K_i, C, b)$, zależna od wartości klucza oraz szyfrogramu. Wartość tej funkcji zależy od wartości bitu $0 \le b \le 32$ w bloku L na początku szesnastego cyklu deszyfrowania szyfrogramu C, zaś 6 bitów klucza wchodzących do S-boxa przyporządkowanego bitom b jest liczbą $0 \le K_i \le 64$. Następnie generowany jest przebieg różnicowy, poprzez znalezienie różnicy w przebiegach, dla których funkcja selekcji ma wartość zero oraz w przebiegach, dla których funkcja selekcji ma wartość 1. Jeżeli wartość K_i jest niepoprawna, wartość bitu obliczona za pomocą funkcji wybierającej różni się dla polowy używanych szyfrogramów. Przebieg różnicowy ma wówczas wartość zbliżoną do zera z pewnymi fluktuacjami. Jeżeli K_i jest poprawne, wówczas wartość obliczona za pomocą funkcji wybierającej jest równa szukanemu bitowi b. Funkcja selekcji jest więc skorelowana z wartością bitu b podlegającego manipulacji w szesnastym cyklu. Rezultatem tego jest wzrost poboru mocy w momencie przetwarzania tego bitu. Ponieważ wartość poboru mocy zależy od wartości jaką mają przetwarzane bity, na przebiegu różnicowym w momentach przetwarzania tych bitów zauważyć można gwałtowne skoki poboru prądu. Na podstawie tego można zidentyfikować poprawne wartości klucza.

Poniżej znajduje się rysunek (rys. 8.4) obrazujący cztery przebiegi sporządzone przy badaniach karty chipowej szyfrującej DES. Pierwszy z nich przedstawia średni pobór prądu przy operacjach DES. Przebiegi nr 2, 3, 4 to przebiegi różnicowe. Przebieg nr 2 został zdjęty przy użyciu właściwej wartości klucza K_i, przebiegi nr 3 i nr 4 - przy użyciu niepoprawnej wartości K_i.

DPA może zostać użyta nie tylko do kryptoanalizy DES. Uniwersalność jej oraz możliwości adaptacyjne czynią ją dobrym narzędziem do kryptoanalizy algorytmów asymetrycznych przy ich implementacjach hardwarowych. Według jej twórców[1], możliwe jest za jej pomocą dokonywanie *reverse engineeringu* nieznanych szyfrów lub protokołów. Dodatkowym atutem jest niski koszt jej przeprowadzenia, oraz względna prostota.

[1] P.C.Kocher, J.Jaffe, B.Jun. Differential Power Analysis, Crypto 1999

Istnieją techniki pozwalające DPA w wysokim stopniu skomplikować. Są to techniki zarówno programowe jak i sprzętowe. Wpływają one jednak znacząco na złożoność a tym samym koszt rozwiązań kryptograficznych. Pozwala to wysunąć tezę, że o sile rozwiązania kryptograficznego decyduje nie tylko doskonałość zastosowanych technik kryptograficznych, ale także ich odpowiednia implementacja.



Rys. 8.4 Oscylogram pracy karty realizującej szyfrowanie DES - 4 warianty

8.2. Analiza czasowa

Obok analizy poboru mocy, istnieje jeszcze drugi rodzaj ataku, związany z bezpośrednią implementacją szyfru, atak oparty na pomiarze czasu wykonywanych operacji i dedukcji bitów klucza na podstawie otrzymanych charakterystyk czasowych – tzw. ***analiza czasowa (timing attack)***.

Wiadomym jest fakt, iż w kryptosystemach czas przetwarzania danych, często w dużej mierze zależy od ich wartości. Składa się na to wiele przyczyn, składających się na optymalizację szybkości działania, takich jak: obejścia, instrukcje warunkowe, rodzaj zastosowanego algorytmu; oraz przyczyn wynikających bezpośrednio z konstrukcji określonego sprzętu - różnego czasu wykonywania poszczególnych instrukcji. Znając zatem pewne charakterystyki czasowe, możliwe jest na ich podstawie uzyskanie pewnych informacji o danych wejściowych - np. kluczu, takich jak waga Hamminga czy też przybliżony rząd wielkości klucza. Informacje te w znacznym stopniu przyczyniają się do ułatwienia ostatecznego ataku.

Najczęstszym celem ***ataku na RSA*** jest wartość prywatnego klucza d. Operacją związaną z użyciem klucza prywatnego jest deszyfrowanie, lub też podpisywanie wiadomości opisane $m^d \bmod n$, w przypadku gdy za pomocą RSA dokonujemy podpisu elektronicznego. Zakłada się, że atakujący zna projekt i implementację algorytmiczną atakowanego systemu.

Załóżmy więc, że obliczenie $m^d \bmod n$ jest wykonywane z użyciem algorytmu Montgomery, za pomocą mnożenia i podnoszenia do kwadratu:

$x = m$
FOR $i = n - 2$ DOWNTO 0
    $x = x^2$
    IF ($d_j = 1$) THEN
    $x = x * m$
ENDFOR
RETURN x

Czas mnożenia za pomocą algorytmu Montgomery jest stały, niezależny od czynników, chyba, że pośredni rezultat mnożenia jest większy niż moduł n, wówczas na końcu mnożenia wykonywane jest dodatkowe odejmowanie, zwane redukcją. Oznacza to, że dla niektórych czynników operacja mnożenia będzie wykonywana nieznacznie dłużej. Można więc dokonać ataku bazującego na

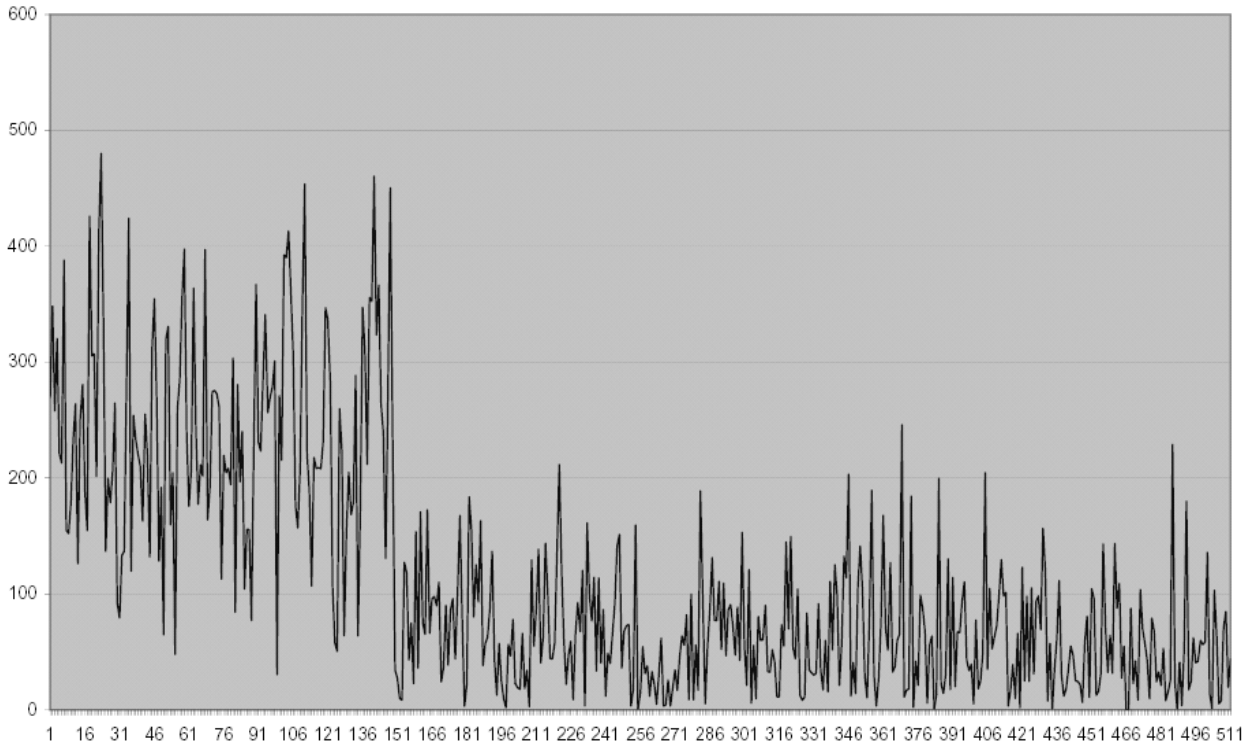
pomiarze czasu wykonywanych operacji, w oparciu o mnożenie liczb w algorytmie.

Załóżmy, że pierwszy bit klucza prywatnego d_1 ma wartość 1. Atak można zacząć więc od drugiego bitu klucza d_2. Obserwując algorytm krok po kroku, można zauważyć, że jeśli wartość tego bitu będzie wynosić 1, wykonywana będzie instrukcja $x = x*m$, a tym samym obliczana będzie wartość $m*m^2$. Dla niektórych wiadomości m (dla tych dla których pośredni rezultat mnożenia będzie większy niż moduł n), podczas mnożenia będzie wykonywana dodatkowa redukcja, dla niektórych zaś nie. Jeżeli zbiór badanych wiadomości M będzie wystarczająco liczny, wówczas badane próbki można podzielić na dwa podzbiory: M_1 - zbiór wiadomości, dla których przy mnożeniu $m*m^2$ dodatkowa redukcja będzie wykonywana, oraz M_2 - zbiór wiadomości, dla których, przy mnożeniu $m*m^2$ redukcja nie będzie wykonywana. Jeżeli wartość bitu d_2 będzie rzeczywiście 1, wówczas można się spodziewać, że czasy obliczeń dla wiadomości z M_1 będą odpowiednio wyższe niż czasy obliczeń wiadomości z M_2.

Z drugiej strony, jeśli wartość bitu d_2 będzie równa zero, wówczas mnożenie $m*m^2$ nie będzie wykonywane. Podział na dwa zbiory jest wówczas bezsensowny: nie ma powodu, dla którego m powodowało by redukcję przy wykonywaniu $m*m^2$, dlatego też dwa zbiory będą wyglądały losowo i nie będzie możliwe zauważenie znaczących różnic w czasach wykonywania obliczeń. Analizując przebiegi czasowe można więc wydedukować na ich podstawie bit klucza d_i.. Analogicznie zestaw powyższych operacji można przeprowadzić w celu znalezienia kolejnych bitów klucza.

Pomiaru charakterystyk czasowych można również dokonywać dla innych operacji algorytmu, takich jak np. podnoszenie do kwadratu. Charakterystyki te mogą się okazać bardziej miarodajne. Przytoczony przykład oparty na mnożeniu miał na celu tylko charakter zapoznawczy.

Interesującą własnością tego rodzaju ataku jest własność detekcji błędów. Atak ten polega na symulacji obliczeń do pewnego momentu. Następnie na podstawie analizy podejmowana jest decyzja jaką ma wartość szukany bit klucza. Każdy krok ataku jest w pewnym stopniu związany z poprzednim. Załóżmy więc, że dla bitu klucza d_i została podjęta błędna decyzja. W kolejnym kroku, obliczenia nie będą mogły zostać przeprowadzone prawidłowo i nie będzie można stwierdzić, czy dodatkowa redukcja była wykonywana czy też nie. Kryterium decyzyjne stanowią średnie różnice czasów obliczeń dla wiadomości z dwóch zbiorów. Po podjęciu nieprawidłowej decyzji, charakterystyki czasowe dla obydwu zbiorów będą zbliżone. Sytuację taką przedstawia rys. 8.5.



Rys. 8.5 Detekcja błędu dla 512 bitowego klucza

Analizując rys. 8.5, widać że błąd w wyznaczaniu wartości bitu wystąpił około 150-ego bitu klucza. Z powyższej charakterystyki również widać, że w niektórych przypadkach, mimo iż właściwie określono wartości bitów, niektóre z wartości różnic czasu, stanowiących kryterium decyzyjne, są bardzo małe. Sugeruje to, że do detekcji błędu, znacznie lepiej używać kryteriów zbudowanych w oparciu o wiele bitów, niż w oparciu o jeden bit. Aby zatem prawidłowo wyznaczyć wartości bitów, liczba próbek badanych powinna być stosunkowo duża. Dla klucza długości 128 bitów, powinna ona wynosić ok. 20000, dla klucza długości 512 bitów - 350000, co znacznie zwiększa nakład pracy koniecznej do przeprowadzenia ataku. Korzystną rzeczą może być więc zastosowanie w takim przypadku specjalnej procedury korekcji błędów. Dla G.Hachez i F.Koeune[2] zastosowanie takiej procedury pozwoliło zredukować liczbę badanych próbek blisko dwukrotnie.

[2] G.Hachez, F.Koeune, J.J.Quisquarter. Timing Attack: What can be achieved by a powerful adversary ?, UCL Crypto Group, 1999

Atak bazujący na pomiarze czasu wykonywanych operacji możliwy do przeprowadzenia jest również dla innych szyfrów. Poniżej zostanie przedstawiony tego rodzaju ***atak na szyfr Rijndael***.

Każdy cykl szyfru ***AES***, z wyjątkiem ostatniego składa się z czterech transformacji: SubBytes, ShiftRows, MixColumn oraz AddRoundKey. Warto jedynie przypomnieć postać macierzową MixColumn

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

Wielomian c(x) jest równy $c(x) = 03x^3 + 01x^2 + 01x + 02$. Mnożenie odbywa się zgodnie z zasadami mnożenia w $GF(2^8)$, czyli modulo nieredukowalny wielomian $x^8 + x^4 + x^3 + x + 1$ ($11B_x$). Redukcja współczynników iloczynu wielomianu jest modulo $x^4 + 1$. Patrząc jednak na wartości współczynników można zauważyć, że $03 = 02 + 01$, dlatego też w praktyce istnieje konieczność wykonania tylko jednego mnożenia - przez 02_x. Mnożenie przez 02_x w $GF(2^8)$ może być wykonywane w dwóch krokach:

1) przesunięcia bitów o jedną pozycję w lewo,
2) w przypadku gdy stopień wielomianu wynikowego przekracza 7 (występuje przeniesienie), sumy XOR wyniku z wartością $11B_x$ - redukcji.

W przypadku, gdy implementacja szyfru zostanie wykonana, jak powyżej, wówczas czas wykonania powyższej operacji nie będzie stały. Kiedy wystąpi konieczność redukcji stopnia wyniku (sumy XOR z $11B_x$), operacja mnożenia będzie wykonywana dłużej. Przy takiej implementacji, w transformacji MixColumn, bajty będą przynajmniej raz mnożone przez wartość 02. Zdarzenie to oznaczone zostanie jako ξ. Jak zostało zauważone powyżej proces ten może trwać dłużej, jeśli pierwszy bit bajta ma wartość równą jeden.

Atak składa się z dwóch faz:

1. Fazy inicjacji - zbudowania macierzy, w której dla każdej możliwej wartości pierwszego bajta klucza *i* dla *N* różnych możliwych wartości pierwszego bajta tekstu jawnego[3], której elementy będą zależne od tego, czy mnożenie ξ wymaga redukcji (dodatkowej XOR), czy też nie:
   $\vee\ 0 \le i \le 255,\ 0 \le j < N$, T[*i*][*j*] = 1 jeśli pierwszy bit ByteSub(*i* XOR *j*) = 1,
   0 w przeciwnym razie.
   Każdy wiersz i odnosi się do możliwej wartości pierwszego bajta klucza cyklu R_1, każda kolumna j odnosi się do różnych wartości pierwszego bajta tekstu jawnego.
2. Fazy pomiarów – w tym celu zbudowane zostanie *N* zbiorów składających się z *M* wiadomości, w których pierwszy bajt każdej wiadomości z danego podzbioru S_i jest równy wartości *i*, pozostałe bajty są losowe. Dzięki temu, dla każdej wiadomości z podzbioru S_i, mnożenie ξ będzie identyczne.

Niech wiadomości te zostaną zaszyfrowane i zmierzony zostanie czas obliczeń. Jeśli *M* jest dość duży, można się spodziewać, że średni czas obliczeń dla podzbioru S_i odzwierciedla czas mnożenia ξ. Tym sposobem, można orzec, z pewnym prawdopodobieństwem błędu, czy dla *i* w [0, *N*-1] pierwszy z bitów operacji ByteSub(*i XOR j)* ma wartość równą jeden.

Aby wyznaczyć R_1, należy wynik porównać z całą tablicą T. Wiersz, który w najlepszy sposób odzwierciedla przewidywane wartości, odnosi się do poprawnej wartości pierwszego bajta R_1. Pozostałe bajty pierwszego klucza cyklu można uzyskać w identyczny sposób. Specyficzny algorytm rozszerzenia klucza pozwala, dzięki znajomości pierwszych N_K bitów kluczy cykli, wygenerować pozostałe klucze dla cykli pozostałych.

Analiza ta, dokonana przez F.Koeune[4], pozwoliła dla ***Rijndaela*** operującego na bloku i kluczu długości 128 bitów, uzyskać, przy ilości 3000 próbek na bajt klucza, wszystkie bity klucza głównego, przy rozsądnym nakładzie kosztów. Atak ten był możliwy do przeprowadzenia, tylko i wyłącznie dzięki nierozsądnej implementacji algorytmu szyfrującego.

Powyższy atak stanowi kolejny dowód, oprócz doskonałego bezpieczeństwa i wysokiej odporności szyfru na teoretyczne ataki, o sile kryptosystemu decyduje również odpowiednia implementacja. Atak ten, zaprezentowany przez P. Kochera[5] może również być używany do ***ataków na implementacje DES***

---

[3] Możliwe jest zbudowanie tej macierzy dla $N = 256$. Eksperymenty przeprowadzone przez autorów (zob. X.Lai, J.L.Masses. Markov Ciphers and Differential Cryptanalysis, Eurocrypt 91, Springer Verlag 1998) pokazały jednak, że atak może zostać przeprowadzony już dla $N = 20$

[4] F.Koeune, J.J.Quisquarter. A Timing Attack Against Rijndael, UCL Crypto Group, Technical Report CG-1999/1

[5] P.C.Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems, 1998

(procedura rozszerzenia klucza), RC6, oraz różnego rodzaju kryptosystemy z kluczem publicznym.