ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1999
Seria: INFORMATYKA z. 36 Nr kol. 1414

Katarzyna TRYBICKA-FRANCIK, Adam ZIĘBIŃSKI
Politechnika Śląska, Instytut Informatyki

METODA CERTYFIKACJI I ZASTOSOWANIA ALGORYTMÓW SZYFRUJĄCYCH I DESZYFRUJĄCYCH

Streszczenie. W pracy przedstawiono podstawy teoretyczne metody certyfikacji i zastosowania algorytmów szyfrujących i deszyfrujących. Ideą proponowanej metody jest wykorzystanie kolejnych znaków tekstu zaszyfrowanego do generacji kluczy. Naświetlono aspekty, jakie należy poruszyć w czasie dokonywania analiz, oraz opisano niektóre metody rozwiązania wynikających problemów przy użyciu reguł matematycznych.

METHOD OF CERTIFYING AND APPLICATION OF ENCRYPTION AND DECRYPTION ALGORITHMS

Summary. The paper presents theoretical foundations of certifying and application of encryption and decryption algorithms. Its main idea is to use consecutive characters of the cliphertext to generate the secret keys.
Same aspects that should be tacked about while during making such analyses were discussed. A few examples of solving problems using mathematical formulas were described.

1. Wstęp

Schyłek drugiego tysiąclecia to czas globalnego połączenia elektronicznego, wirusów i hackerów, elektronicznego podsłuchiwania i elektronicznych oszustw. Problem ochrony danych przed niepowołanym dostępem stał się kluczowym problemem. Tak więc, wraz z rozwojem systemów komputerowych i ich połączeń sieciowych rozwinęły się takie dyscypliny, jak kryptografia i ochrona danych w systemach komputerowych, co doprowadziło do stworzenia praktycznych, dostępnych programów służących do zabezpieczania informacji elektro-

nicznej. Jak grzyby po deszczu pojawiły się firmy obiecujące, że to właśnie ich produkt zapewnia najwyższy poziom zabezpieczeń, ale czy faktycznie?!

Potrzeba weryfikacji jakości narzędzi szyfrujących stała się równie nagląca jak konieczność ich powstawania. Strategie stosowane przy ich ocenie są różne i zależą od rodzaju systemu szyfrowania i dostępnych informacji.

W niniejszym artykule zostaną zaprezentowane wyniki prac nad bezpieczeństwem szyfrów blokowych, prowadzone w ramach grantów badawczych[1].

2. Ocena zgodności szyfratorów ze standardem DES

W pierwszym etapie prac został zaimplementowany w języku programowania MICROSOFT C++ szyfrator i deszyfrator DES w oparciu o algorytm zgodny ze standardem DES dany w postaci schematu blokowego i opisu słownego. Podczas tworzenia software'owego szyfratora/deszyfratora główny nacisk był skierowany na to by, był on w 100% zgodny z normą opisującą ów standard. Tak skonstruowany szyfrator/deszyfrator DES posłużył jako wzorzec dla badania zgodności dostępnych szyfratorów DES ze standardem.

W wyniku tych prac okazało się, że rozwiązania software'owe szyfratorów /deszyfratorów DES nie są między sobą kompatybilne. Różni producenci w różny sposób obudowują swoje pliki wynikowe (po zaszyfrowaniu) i by faktycznie móc ocenić ich zgodność ze standardem, trzeba znać strukturę plików wynikowych. Żadna z firm nie podaje jednak, w jaki sposób rozmieszcza dane pomocnicze w plikach wynikowych.

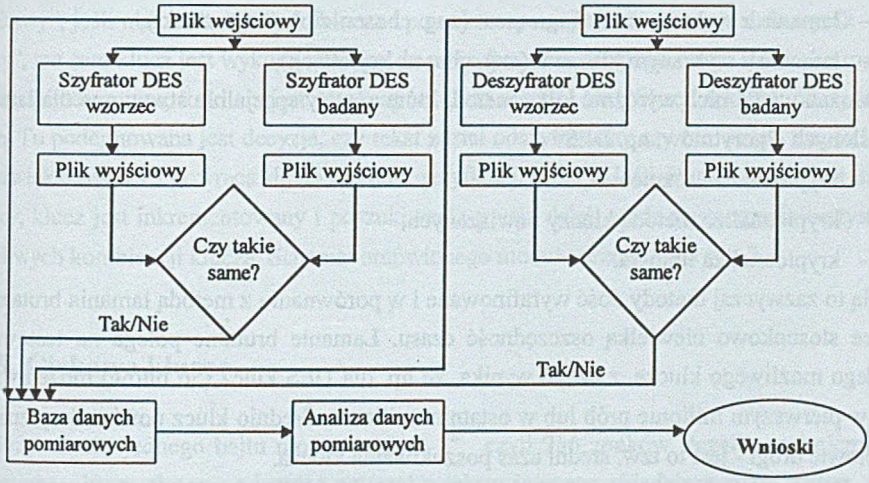
Testy prowadzone na wersji szyfratora DES firmy SOTEL wykazały, że oba rozwiązania są spójne. Dla celów testowych wykorzystano również ogólnodostępną aplikację szyfratora/deszyfratora DES, rozpowszechnianą przez Bruce'a Schneiera. Wyniki tych testów były również satysfakcjonujące. Na podstawie tych badań doszliśmy do wniosku, że nasze rozwiązanie jest zgodne ze standardem DES i może posłużyć jako wzorzec dla celów testowych.

Rys. 1 przedstawia metodę badania zgodności szyfratora/ deszyfratora ze standardem DES.

W chwili gdy analizowaliśmy problem zgodności szyfratorów DES ze wzorcem, podejrzewaliśmy, że wynikiem prac będzie zbiór plików testowych, które umożliwią w łatwy sposób sprawdzenie, czy proponowane rozwiązanie jest istotnie szyfratorem/deszyfratorem DES. Z czasem uznaliśmy jednak, że lepszym rozwiązaniem będzie losowy wybór plików, a dowód niezaprzeczalności zgodności ze wzorcem wykazany zostanie na podstawie plików testo-

[1] Badania finansowane przez KBN w ramach grantu dla młodych naukowców nr 8T11C 033 14, przyznanego K. Trybickiej-Francik, oraz grantu promotorskiego nr 8T11C 021 15, przyznanego A. Ziębińskiemu.

wych. Gdy znane są pliki testowe, nic nie stoi na przeszkodzie, by aplikację napisać tak, by spełniała oczekiwania osoby testującej.



Rys. 1. Schemat blokowy modułu weryfikacji algorytmów
Fig. 1. The block diagram of verification algorithms

Inną zaletą przemawiającą za jest fakt, że każdy z użytkowników mógłby do testów wybrać taki typ plików, jaki z jego punktu widzenia jest strategiczny.

W testach przyjmowaliśmy, że jeżeli przy stu losowo dobranych plikach, długości od kilku KBajtów do kilku MBajtów, nie wystąpiła niezgodność, to proponowany szyfrator/deszyfrator DES jest zgodny ze standardem.

3. Bezpieczeństwo algorytmu

Szyfr jest przełamywalny, jeśli istnieje możliwość odtworzenia tekstu jawnego bądź klucza na podstawie tekstu zaszyfrowanego albo określenie klucza na podstawie znajomości tekstów jawnego i zaszyfrowanego. Przy łamaniu szyfru zakłada się zwykle, że kryptoanalityk zna szczegóły algorytmu szyfrowania. Nie jest to co prawda założenie prawdziwe w wielu realnych sytuacjach, lecz często się tak przyjmuje w kryptoanalizie naukowej. Jeśli bezpieczeństwo systemu zależy od tajności algorytmu, to jest to kiepskie zabezpieczenie.

Wyróżnia się sześć klas metod łamania szyfrów:

- łamanie z szyfrogramami (ang. ciphertext-only attack),
- łamanie ze znanym tekstem jawnym (ang. known-plaintext attack),

- łamanie z wybranym tekstem jawnym (ang. chosen-plaintext attack),
- łamanie z adaptacyjnie wybranym tekstem jawnym (ang. adaptive-chosen-plaintext-attack),
- łamanie z wybranym szyfrogramem (ang. chosen-ciphertext attack),
- łamanie z wybranym kluczem (ang. chosen-key attack),

Można wśród nich wyróżnić kilka metod, które zostały specjalnie stworzone dla łamania określonych algorytmów, np. DES:

- kryptoanaliza różnicowa,
- kryptoanaliza metodą kluczy powiązanych,
- kryptoanaliza liniowa.

Są to zazwyczaj metody dość wyrafinowane i w porównaniu z metodą łamania brutalnego dające stosunkowo niewielką oszczędność czasu. Łamanie brutalne polega na testowaniu każdego możliwego klucza, z czego wynika, że np. dla DES klucz (56 bitów) możemy znaleźć w pierwszym milionie prób lub w ostatnim milionie. Średnio klucz powinniśmy znaleźć w połowie drogi - jest to tzw. średni czas poszukiwania klucza.

W tym miejscu można się pokusić o próbę zdefiniowania pojęcia mocy algorytmu szyfrującego. W tych rozważaniach przyjęto, że moc algorytmu szyfrującego to różnica:

$$\Delta T = T_2 - T_1$$

gdzie: T_1 - to czas życia informacji,

T_2 - to czas potrzebny na złamanie tekstu zaszyfrowanego.

Szyfr uznaje się za:

- Bezwzględnie mocny, gdy: $T1 < T_{2optymistyczne}$.
  W praktyce przypadek nieosiągalny, gdyż oznaczałoby to, że czas życia informacji musiałby być krótszy od czasu jej deszyfracji.
- Mocny, gdy: $T_{2optymistyczne} < T_1 << T_{2średnie}$.
- Słaby, gdy: $T_{2średnie} << T_1 < T_{2pesymistyczne}$.

Jak łatwo zauważyć, dwa parametry określają szybkość łamania brutalnego: liczba kluczy do przetestowania i szybkość każdego testu.

Po krótkiej analizie problemu łamania brutalnego można dojść do wniosku, że mamy wpływ na liczbę kluczy użytych do testów. Postanowiono stworzyć moduł weryfikacji kluczy, którego zadaniem jest wybór potencjalnie „ciekawego" klucza, tzn. takiego, który rokuje nadzieje na to, że przy jego użyciu można odczytać szyfrogram. Należy tu podkreślić, że nie szukamy odpowiedzi na pytanie, jaki klucz przełamuje szyfrogram, lecz staramy się odpowiedzieć na pytanie, które z badanych kluczy na pewno się do tego nie nadają.

Zaletą tego rozwiązania jest to, że właściwy program deszyfrujący i dokonujący analizy rozszyfrowanych tekstów musi sprawdzić tylko ułamek wszystkich możliwych kluczy.

4. Działanie modułu weryfikacji kluczy

Moduł odszyfrowywuje jeden blok za pomocą klucza i sprawdza, czy wynik operacji jest „ciekawy", jeśli nie, to dodaje 1 do klucza i powtarza operację. Jeśli operacja dała wynik „ciekawy", ten sam klucz jest wykorzystywany do odszyfrowania drugiego bloku. Jeśli oba rozwiązania są interesujące, to klucz jest przekazywany do modułu weryfikacji tekstów odszyfrowanych. Tu podejmowana jest decyzja, czy tekst został odszyfrowany czy nie. Jeżeli nie, to klucz jest zwiększany o 1 i powraca się do modułu weryfikacji kluczy. Jeśli drugi blok nie jest interesujący, klucz jest inkrementowany i poszukiwania trwają dalej, aż do wyczerpania wszystkich możliwych kombinacji klucza. Schemat omówionego modułu przedstawia rys. 2.

5. Ciekawe klucze

Przy użyciu jednego bajtu można zapisać 2^8, czyli 256 znaków. Jeżeli przyjmiemy, że szyfrogram, który chcemy odczytać powstał z tekstu jawnego składającego się z liter, cyfr, spacji i znaków interpunkcyjnych, to okaże się, że liczba interesujących nas znaków jest znacznie mniejsza aniżeli 256. Dla języka polskiego będzie to około 76 znaków.

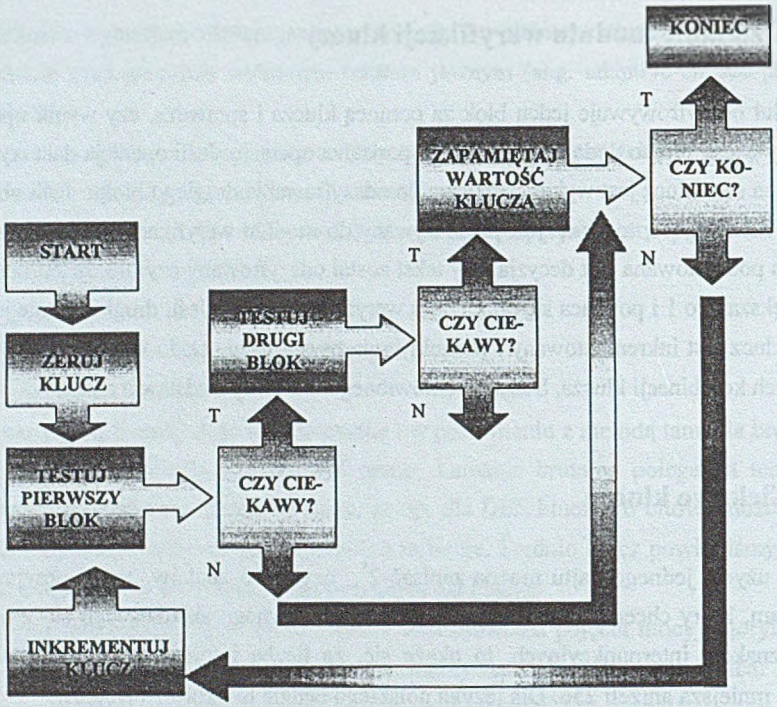
Wynika stąd, że prawdopodobieństwo wystąpienia na jednym bajcie interesującego nas znaku jest równe: P(bajt) = 76/256 ≈ 0,29. By usprawnić obliczenia, przyjmiemy, że prawdopodobieństwo P(bajt) = 0,25 = 2^{-2}. Załóżmy, że rozpatrujemy algorytm DES, blok składa się z ośmiu bajtów, to prawdopodobieństwo, że wszystkie osiem bajtów będzie interesującymi nas znakami wynosi: P(blok) = P(bajt)8 = (2-2)8 = 2-16.

Oznacza to, że średnio co 65.536 klucz byłby przez moduł weryfikacji kluczy uznawany za "ciekawy". Dla klucza o długości 2^{56} bitów taka selekcja nie daje jeszcze zbyt wiele (do sprawdzenia pozostanie średnio 2^{40} kluczy, czyli ponad jeden bilion kombinacji).

Dlatego też przez moduł weryfikacji klucza sprawdzane są dwa bloki szyfrogramu, co daje prawdopodobieństwo trafienia na interesujący nas blok równe: P(2 bloki) = $P(blok)^2=(2^{-16})^2 = 2^{-32}$.

Do sprawdzenia pozostanie więc około 224 kluczy, czyli blisko 17 mln. Jest to liczba nadal bardzo duża, ale nieporównywalnie mniejsza od 2^{56}.

To, że w ciągu sekundy potrafimy przetestować 225 000 bloków, oznacza , iż w tym czasie zostanie zweryfikowanych 112 500 kluczy. Wynika z tego, że średni czas testowania kluczy dla algorytmu DES, na komputerze klasy IBM_PC, z procesorem PENTIUM II, z zegarem 400 MHz wyniósłby 10 155 lat, a w przypadku pesymistycznym 20 310 lat.



Rys. 2. Schemat blokowy modułu weryfikacji kluczy
Fig. 2. The block diagram of verification keys

Wynik nie jest zbyt zachęcający, ale jeżeli weźmiemy pod uwagę rozwiązanie sprzętowe, to w dobie bumu elektronicznego nie trudno wyobrazić sobie urządzenie, które korzystając z powyższego algorytmu poradziłoby sobie ze złamaniem DES w kilkanaście dni.

Istnieją układy scalone, które potrafią szyfrować i deszyfrować dane z szybkością 1Gbit/s (Digital Equipment Corporation; układ zbudowany z 50 tysięcy tranzystorów na podłożu GaAs; 1994 rok), co jest równe 15,6 mln bloków/s, czyli w ciągu sekundy sprawdza ok. 7,8 mln kluczy. Z tego wynika, że średni czas łamania DES'y wynosi 146 lat, a pesymistyczny 292 lat. Jeżeli do pracy zaprzęgniemy 64 takie procesory, to czas łamania w pesymistycznym przypadku wyniesie 4,5 roku. Jeżeli z tych samych układów zbudujemy macierz 64 układy na 64 układy, to DES będzie przełamywany w 27 dni w wariancie pesymistycznym. Jak widać, jest o co walczyć. Postanowiono podjąć wyzwanie. W chwili gdy powstaje ten artykuł, prowadzone są badania nad zaimplementowaniem powyższego rozwiązania w matryce programowalne FPGA.

Pojawia się kolejny problem, w jaki sposób podjąć decyzję, który z rozszyfrowanych tekstów (a może ich być 2^{24}, czyli blisko 17 mln) jest interesującym rozwiązaniem z punktu widzenia kryptoanalizy.

Należało więc stworzyć mechanizmy pozwalające w sposób automatyczny odrzucać rozwiązania zdecydowanie złe. W swych rozważaniach nadal zakładamy, że mamy do czynienia z tekstem napisanym w dowolnym języku, którego alfabet jest alfabetem łacińskim.

6. Moduł weryfikacji plików

Jednostka decyzyjna w module weryfikacji pliku oparta jest na znajomości liczby statystycznych wystąpień pojedynczych liter, digramów i trigramów, a także na grupie słowników pozwalających w przybliżony sposób określić, czy mamy do czynienia z sensownym tekstem, a jeżeli tak, to w jakim języku. Rys. 3. przedstawia schemat działania modułu weryfikacji pliku.

7. Długość krytyczna

Długość krytyczna jest to najmniejsza wartość N, dla której $H_C(K)$[1] jest bliskie zera; innymi słowy, jest to ilość tekstu zaszyfrowanego niezbędna do jednoznacznego określenia klucza. Szyfr jest bezwarunkowo bezpieczny, jeżeli $H_C(K)$ nigdy nie osiągnie wartości 0, nawet dla bardzo dużej wartości N. Oznacza to, że klucz nie może być jednoznacznie określony niezależnie od tego, jak dużo tekstu zaszyfrowanego zostanie przechwycone.

Większość szyfrów jest zbyt złożona, aby można było określić prawdopodobieństwa niezbędne do ustalenia długości krytycznej. Shannon wykazał jednak, że dla pewnych szyfrów bazujących na modelu szyfru losowego jest możliwe jej przybliżone określenie. Hellman uzyskał taki sam efekt stosując nieco inne podejście.

Idąc drogą rozważań Hellmana okaże się, że dla systemów bliskich szyfrom losowym, długość krytyczną, jako ilość tekstu niezbędną do złamania szyfru, można określić wzorem:

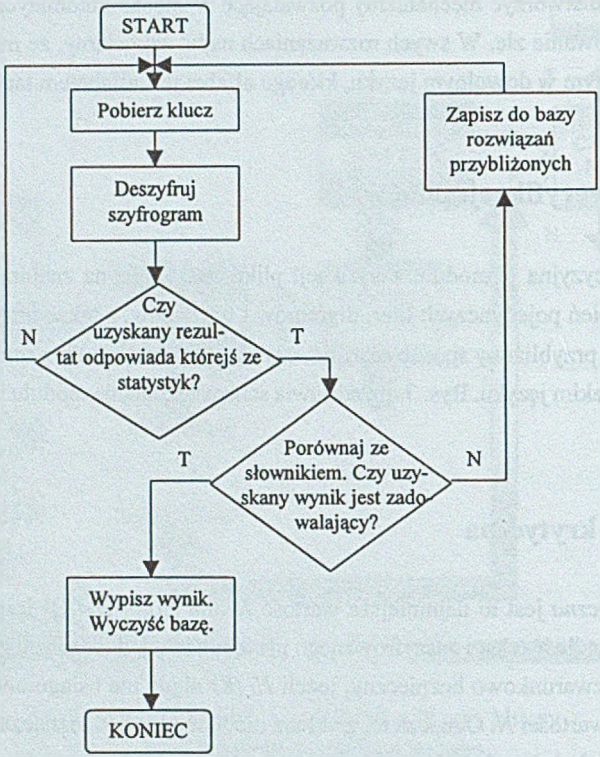
$$N = \frac{H(K)}{D}$$

gdzie: N - długość krytyczna,

$H(K)$ - entropia klucza (liczba bitów w kluczu),

D - nadmiarowość języka.

[1] Shannon określił poufność szyfru w kategoriach entropii warunkowej klucza $H_C(K)$ dla danego tekstu zaszyfrowanego C, tj. wielkości w K przy danym C.



Rys. 3. Schemat działania modułu weryfikacji plików
Fig. 3. The block diagram of verification files

Rozważmy teraz system DES, w którym 64-bitowe bloki szyfruje się używając 56-bitowych kluczy. System ten jest bliskim przybliżeniem szyfru losowego. Przyjmijmy, że H(K)=56 oraz D=3,2. Stąd długość krytyczna

$$N = \frac{56}{3{,}2} = 17{,}5 \ ,$$

czyli nieco ponad dwa bloki.

W rzeczywistości, w dniu pisania artykułu, przy użyciu stworzonej przez nas aplikacji testowej, klucz można jednoznacznie określić dla szyfrogramu o długości równej co najmniej 6 bloków.

8. Klucze słabe i półsłabe

Chociaż potencjalna liczba kluczy algorytmu wynosi 2^{56}, to w ataku polegającym na przeszukiwaniu wyczerpującym wystarczy przebadać, co wynika z twierdzenia 1 (niżej w tekście), 2^{55} kluczy, a średnio 2^{54} kluczy. Jest to ciągle liczba bardzo duża.

W dalszej części tego opracowania skoncentrujemy się na wyznaczeniu zbioru kluczy słabych i półsłabych [1]. Wyznaczenie zbioru tego typu kluczy pozwala ustrzec się przed sytuacją, w której wiadomość jawna pozostałaby nie zmieniona, albo zmieniona pozornie poprzez zanegowanie wszystkich bitów.

W tym celu konieczne jest przyjęcie kilku założeń.

Przedstawmy dowolne przekształcenie DES_k w postaci konkatenacji osiemnastu słów:

$$L_0R_0R_1R_2...R_{15}R_{16},$$

gdzie

$$L_i = R_{i-1} \text{ dla } i = 1, 2,...,16.$$

Jeżeli przyjmiemy, że:

$$n_0 = L_0,$$

oraz że:

$$n_1 = R_0,$$

to dla $i = 2,...,17$,

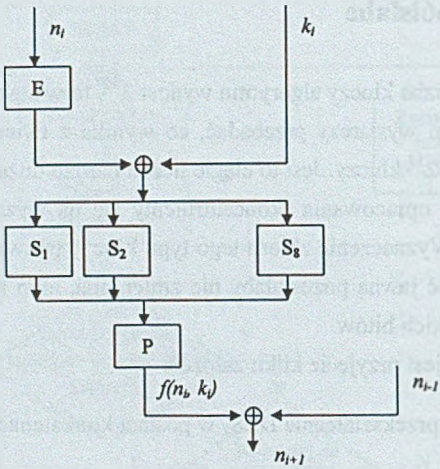
$$n_i = R_{i-1}.$$

Zatem mamy, że: n_0n_1 jest wiadomością jawną po dokonaniu permutacji początkowej IP, zaś $n_{16}n_{17}$ wiadomością zaszyfrowaną przed dokonaniem permutacji końcowej IP^{-1}.

Natomiast dla $i = 1, 2,...,16$:

$$n_{i+1} = n_{i-1} \oplus f(n_i, k_i).$$

Na rysunku 4 przedstawiono tę zależność.



Rys. 4. Ilustracja zależności $n_{i+1} = n_{i-1} \oplus f(n_i, k_i)$
Fig. 4. The dependence $n_{i+1} = n_{i-1} \oplus f(n_i, k_i)$

Twierdzenie 1. Jeśli $n_i = n_j \oplus U$, gdzie U jest dowolnym 32-bitowym słowem, a ponadto $k_i = k_j \oplus E(U)$, to dla $i \neq j$

$$f(n_i, k_i) = f(n_j, k_j).$$

Skoro dowolny 32-bitowy blok U można rozszerzyć za pomocą przekształcenia E do bloku 48-bitowego, to istnieje dokładnie 2^{32} par (n_j, k_j), gdzie $n_j = n_i \oplus U$ oraz $k_j = k_i \oplus E(U)$, do których stosuje się wynik twierdzenia 2 (niżej w tekście). Jednakże tych 2^{32} par jest to liczba w dużym stopniu zawyżona, co będzie wykazane poniżej.

Niech k i k^* będą kluczami odwrotnymi, tj. takimi, że $DES_{k^*}(m) = DES_k(m)$ dla każdej wiadomości jawnej m. Skoro proces deszyfrowania za pomocą algorytmu DES przebiega w taki sam sposób jak proces szyfrowania z uwzględnieniem odwrotnego porządku kluczy pomocniczych, to warunek wystarczający zachodzenia równości $f(n_i, k_i) = f(n_j, k_j)$ z twierdzenia 2 mógłby być zapisany w postaci

$$k_i^* = k_{17-i}, \text{ gdzie } 16 \geq i \geq 1.$$

Twierdzenie 2. Dla klucza k istnieje klucz odwrotny k^* spełniający warunek, że $k_i^* = k_{17-i}$ dla każdego $i \in \{1, 2, \ldots, 16\}$ wtedy i tylko wtedy, gdy k jest jednym z 16 kluczy, w których wszystkich 14 bitów, określonych przez każdy z czterech podzbiorów A, B, C, D, jest jednakowych, przy czym

A = {1, 2, 3, 17, 18, 19, 33, 34, 35, 36, 49, 50, 51, 52},

B = {4, 5, 6, 7, 20, 21, 22, 23, 37, 38, 39, 53, 54, 55},

C = {9, 10, 11, 25, 26, 27, 41, 42, 43, 44, 57, 58, 59, 60},

D = {12, 13, 14, 15, 28, 29, 30, 31, 45, 46, 47, 61, 62, 63}.

W tabelach 1 i 2, w zapisie szesnastkowym, zostały zebrane klucze słabe i półsłabe.

Tabela 1

Klucze słabe

Pierwotny ciąg słabego klucza	Faktyczny ciąg klucza
0101 0101 0101 0101	0000000 0000000
FEFE FEFE FEFE FEFE	FFFFFFF FFFFFFF
1F1F 1F1F 0E0E 0E0E	0000000 FFFFFFF
E0E0 E0E0 F1F1 F1F1	FFFFFFF 0000000

Tabela 2

Klucze półsłabe

1FE0 1FE0 0EF1 0EF1
01E0 01E0 01F1 01F1
1FFE 1FFE 0EFE 0EFE
011F 011F 010E 010E
E0FE E0FE F1FE F1FE
FE01 FE01 FE01 FE01
E01F E01F F10E F10E
E001 E001 F101 F101
FE1F FE1F FE0E FE0E
1F01 1F01 0E01 0E01
FEE0 FEE0 FEF1 FEF1

Tabela 3

Klucze częściowo słabe

1F 1F 01 01 0E 0E 01 01	1F FE E0 01 0E FE F0 01	1E FE 01 E0 0E FE 01 F1
01 1F 1F 01 01 0F 0E 01	1F E0 FE 01 0E F1 FE 01	01 FE 1F E0 01 FE 0E F1
1F 01 01 1F 0E 01 01 0F	01 FE FE 01 01 FE FE 01	1F E0 01 FE 0E F1 01 FE
01 01 1F1F 01 01 0E 0E	1F E0 E0 1F 0E F1 F1 0E	01 E0 1F FE 01 F1 0E FE
E0 E0 01 01 F1 F1 01 01		
FE FE 01 01 FE FE 01 01	01 FE EO 1F 01 FE F1 0E	01 01 E0 E0 01 01 F1 F1
FE E0 1F 01 FE F1 0E 01	01 E0 FE 1F 01 F1 FE 0E	1F 1F E0 E0 0E 0E F1 F1
E0 FE 1F 01 F1 FE 0E 01	1F FE FE 1F 0E FE FE 0E	1F 01 FE E0 0E 01 FE F1
FE E0 01 1F FE F1 01 0E	0E 01 01 E0 F1 01 01 F1	01 1F FE E0 01 0E FE F1
E0 FE 01 1F F1 FE 01 0E	FE 1F 01 E0 FE 0E 01 F1	1F 01 EO FE 0E 01 F1 FE
E0 E0 1F 1F F1 F1 0E 0E	FE 01 1F E0 FE 01 0E F1	01 1F E0 FE 01 0E F1 FE
FE 1F E0 01 FE 0E F1 01	E0 1F 1F E0 F1 0E 0E F1	01 01 FE FE 01 01 FE FE
	FE 01 01 FE FE 01 01 FE	1F 1F FE FE 0E 0E FE FE
	E0 1F 01 FE F1 0E 01 FE	
E0 1F FE 01 F1 0E FE 01		
FE 01 E0 1F FE 01 F1 0E	E0 01 1F FE F1 01 0E FE	FE FE E0 E0 FE FE F1 F1
E0 01 FE 1F F1 01 FE 0E	FE 1F 1F FE FE 0E 0E FE	E0 FE FE E0 F1 FE FE F1
01 E0 E0 01 01 F1 F1 01		FE E0 E0 FE FE F1 F1 FE
		E0 E0 FE FE F1 F1 FE FE

9. Podsumowanie

W ramach prac badawczych nad algorytmem DES zostały zrealizowane praktycznie wszystkie zakładane cele:

1. Oceniono zgodności szyfratorów DES ze standardem.
2. Zdefiniowano pojęcie mocy szyfratorów.
3. Zaprojektowano narzędzia ułatwiające ocenę mocy algorytmów.
4. Zrealizowano moduł decyzyjny pozwalający na automatyczną ocenę jakości plików zdeszyfrowanych.
5. Wyznaczono grupę kluczy słabych, półsłabych i częściowo słabych.

Wszystkie moduły programowe, mimo że realizowane pod kątem projektu dotyczącego algorytmu DES, są łatwo adaptowalne dla celów testowania innych algorytmów.

Prace nad opisanym tu rozwiązaniem będą nadal kontynuowane. W ich wyniku powinien powstać prototyp urządzenia kryptoanalitycznego opartego na matrycach programowalnych

FPGA, a także ulepszona jednostka weryfikacji plików, tzn. moduł weryfikacji plików, który nie będzie ograniczony tylko do plików tekstowych.

LITERATURA

1. Stokłosa J.: Algorytmy kryptograficzne. OWN, Poznań 1994.
2. Lexar Corporation, An Evaluation of the DES, Sep 1976.
3. Morris R., Soane N. J. A., and Wyner A. D.: Assessment of the NBS Proposed Data Encryption Standard. Cryptologia, v. 1, n. 3, Jul 1977, pp.281-291.
4. Brandstad D. K., Gait J., and Katzke S.: Report on the Workshop on Cryptography in Support of Computer Security. NBSIR 77-1291, National Bureau of Standards, Sep 21-22, 1976, September 1977.
5. Simmons G. J.: The Prisoner's Problem and the Subliminal Channel. Advances in Cryptology: Proceedings of Crypto 83, Plenum Press, 1984, pp. 51-67.
6. Tuchman W.: Hellman Presents No Shortcut Solutions to DES. V. 16, n. 7, Jul 1979, pp. 40-41.
7. Shamir A.: On the Security of DES. Advances in Cryptology-CRYPTO'85 Proceedings, Berlin: Springer-Verlag, 1986.
8. Brickell E. F., Moore J. H., and Purtill M. R.: Structure in the S-Boxes of the DES. Advances in Cryptology-CRYPTO'86 Proceedings, Berlin: Springer-Verlag, 1987.
9. Davio M., Desmedt Y., Fosseprez M., Govaerts R., Hulsborsch J., Neutjens P., Piret P., Quisquater J. J, Vandewalle J., and Wouters S.: Analytical Characteristics of the DES. Advances in Cryptology: Proceedings of Crypto 83, Plenum Press, 1984.
10. Davio M., Desmendt Y., and Quisquater J. J.: Propagation Characteristics of the DES. Advances in Cryptology-EUROCRYPT'84 Proceedings, Berlin: Springer-Verlag, 1985.
11. Davio M., Desmendt Y., and Quisquater J. J.: Dependence of Output on Input in DES: Small Avalanche Characteristics. Advances in Cryptology-CRYPTO'84 Proceedings, Berlin: Springer-Verlag, 1985.
12. Stokłosa J.: Kryptograficzne metody ochrony danych. Wydawnictwo Politechniki Poznańskiej, Poznań, 1992.
13. Moore J. H. and Simmons G. J.: Cycle Structure of the DES with Weak and Semi -Weak Keys. Advances in Cryptology-CRYPTO'86 Proceedings, Berlin: Springer-Verlag, 1987.

14. Advances in Cryptology-CRYPTO'98 Proceedings, Berlin: Springer-Verlag, 1998.
15. Advances in Cryptology-EUROCRYPT'98 Proceedings, Berlin: Springer-Verlag, 1998.

Recenzent: Dr inż. Andrzej Białas

Wpłynęło do Redakcji 29 marca 1999 r.

Abstract

The research activities consist of three parts.

1. **Methods of estimation of compatibility of the commercially available DES implementation with the standard**

   In this part, tools have been implemented which allow to easily prepare test files as well as to carry out the comparative studies. With this in mind, an encryption and a decryption modules have been implemented, with efficiency of 17,5 MB per second, which are fully compatible with the DES standard,.

2. **Definition of the power of an algorithm. Deciding if the cipher has been broken.**

   Sophisticated cryptoanalytic methods developed for the DES algorithm give relatively few advantages if compared with the method of brutal breaking, which consists of testing every possible key. So a module has been created for brutal DES breaking, in which a function of fast key verification had been developed. It allows to check, if there is a chance that a given key is the correct one. Although the software realisation does not give promises for breaking the cipher in acceptable time, a hardware solution based on the same idea should succeed in time shorter than 27 days. Brutal breaking module, besides of the fast key verification function, is provided with an engine for estimation of output file quality. It automatically decides whether a given file is a correct decrypted file. This engine should succeed with each file not exceeding 56 bits.

3. **An analysis of appearance of weak and semi-weak keys**

   This part is devoted to determination, on the theoretical basis, the appearance of weak and semi-weak keys, applying a set of theorems addressing the rules of the DES algorithm.