

Kryptografia i bezpieczeństwo systemów informatycznych

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

sem. letni 2024/2025

inf.ug.edu.pl/~amb/

Andrzej Borzyszkowski (Instytut Informatyki | Kryptografia i bezpieczeństwo systemów infor sem. letni 2024/2025 1 / 35

Funkcje pseudolosowe

- Ozn.: $[n]$ – zbiór ciągów bitów długości n , $\oplus$ dodawanie bitów mod 2
- Dana funkcja $F : [n] \times [n] \rightarrow [n]$, $F(k, m)$ – klucz i wiadomość
 - obliczalna PPT
 - problem, czy funkcję jednej zmiennej $F(k, _)$ da się odróżnić od losowej funkcji $f : [n] \rightarrow [n]$?
 - funkcji pierwszego typu jest tylko 2^n , drugiego typu $(2^n)^{2^n}$
 - sam zapis funkcji $f : [n] \rightarrow [n]$ wymaga wykładniczych zasobów
 - jeśli f jest losowa, to wartości $f(x)$ oraz $f(y)$, $x \neq y$, są niezależne
- Funkcja F jest pseudolosowa, jeśli przeciwnik o zasobach PPT nie odróżni $F(k, _)$, k losowe, od losowej funkcji f
 - nawet nie zna całej funkcji
 - korzysta jedynie z wyroczni obliczającej wartości funkcji
 - tzn.: podobnie jak w ataku z wybranym tekstem jawnym

Andrzej Borzyszkowski (Instytut Informatyki | Kryptografia i bezpieczeństwo systemów infor sem. letni 2024/2025 3 / 35

Szyfry blokowe

Andrzej Borzyszkowski (Instytut Informatyki | Kryptografia i bezpieczeństwo systemów infor sem. letni 2024/2025 2 / 35

Wykorzystanie funkcji pseudolosowych

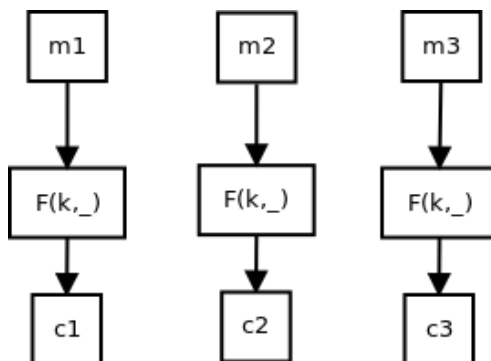
- Metoda naiwna
 - $Enc(k, m) = F(k, m)$
 - nie ujawnia niczego o tekście
 - ale jest deterministyczna, nie może być bezpieczna przeciwko atakowi z wybranym tekstem jawnym czy z zestawem tekstów jawnych
- Metoda słuszną
 - generowanie klucza: losowy wybór
 - funkcja szyfrowania: $Enc(k, m) = \langle r, F(k, r) \oplus m \rangle$, r jest losowo wybranym ciągiem $r \in [n]$
 - funkcja odszyfrowania: $Dec(k, \langle r, c \rangle) = F(k, r) \oplus c$
 - r nie może pojawić się powtórnie, prawdopodob. tego błędu jest zaniebdywalnie małe
- Tw.: jeśli funkcja jest pseudolosowa, to powyższy szyfr jest bezpieczny przeciwko atakowi z wybranym tekstem jawnym

Andrzej Borzyszkowski (Instytut Informatyki | Kryptografia i bezpieczeństwo systemów infor sem. letni 2024/2025 4 / 35

- Funkcja $F : [n] \times [n] \rightarrow [n]$, jest permutacją z kluczem, jeśli każda $F(k, _)$ jest bijekcją
 - jest efektywna, jeśli istnieją algorytmy PT obliczające $F(k, _)$ oraz funkcję odwrotną
- F jest pseudolosową permutacją, jeśli jest nieodróżnialna od losowych permutacji
 - równie dobrze jest nieodróżnialna od losowych funkcji
- Można żądać więcej, by $F(k, _)$ oraz funkcja odwrotna były nieodróżnialne od losowych permutacji

Tryby: ECB

- ECB (*electronic code book*)
 - każdy blok szyfrowany jest niezależnie:
 $c_j = \text{Enc}(k, m_j)$, $m_j = \text{Dec}(k, c_j)$
 - daje to szyfr deterministyczny $\text{Enc}(k, m) = F(k, m)$



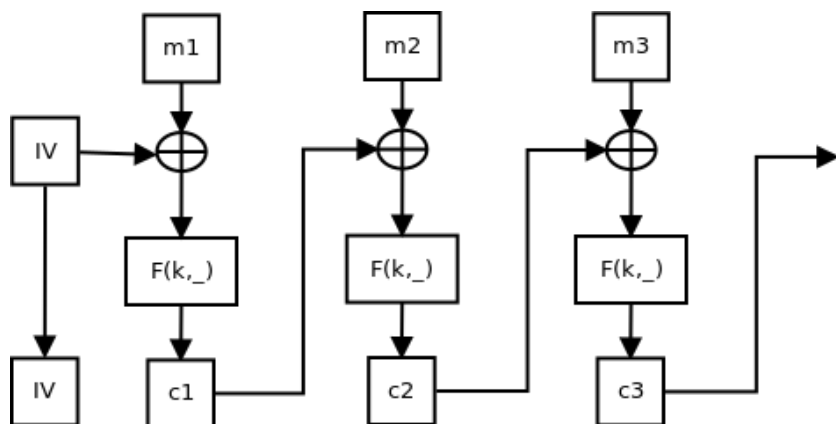
- Konieczność szyfrowania tekstów jawnych dłuższych niż blok
 - tekst jawny jest ciągiem bloków P_j , szyfrogram bloków C_j , $j = 1, 2, \dots$
 - być może trzeba wypełniać ostatni blok
 - standardowa metoda:
 - jeśli zabraknie jednego bajta, to wpisujemy ascii 1
 - jeśli dwóch, to dwa bajty ascii 2, itd.
 - jeśli nie ma potrzeby uzupełniania, to dodajemy cały blok z ascii 0
 - gdyby nie dodawać pustego bloku, to nie odróżnilibyśmy np. tekstu jawnego z ostatnim bajtem równym 1 od tekstu o jeden bajt krótszego

Własności ECB

- Jest to szyfr deterministyczny $\text{Enc}(k, m) = F(k, m)$
 - gdyby $\text{Enc}(k, m) = \langle r, F(k, r) \oplus m \rangle$
 - trzeba by dużej liczby losowych wektorów,
 - szyfrogram byłby dwa razy dłuższy niż tekst jawny
- Zalety:
 - niezawodność,
 - łatwość szyfrowania fragmentów (baza danych, zawartość dysku)
- Wady:
 - brak bezpieczeństwa każdego rodzaju, Ewa może rozpoznać, która z wiadomości została zaszyfrowana
 - niebezpieczeństwo rozpoznania stałych fragmentów, niebezpieczeństwo manipulacji fragmentami szyfrogramu
 - nie powinien być stosowany w ogóle

- CBC (*cipher block chaining*)

- $c_j = \text{Enc}(k, m_j \oplus c_{j-1})$, $m_j = \text{Dec}(k, c_j) \oplus c_{j-1}$, $c_0 = IV$
- takie same fragmenty tekstu są szyfrowane odmiennie



- Zalety:

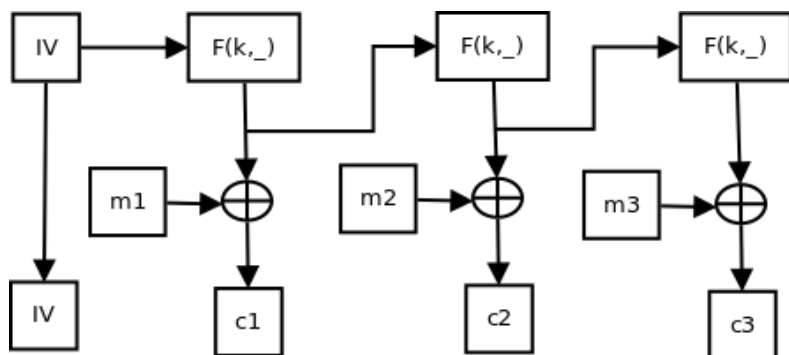
- jeśli F jest permutacją pseudolosową, a wektor początkowy jest losowy, to szyfr jest bezpieczny na atak z wybranym tekstem jawnym

- Wady:

- przetwarzanie musi być sekwencyjne, nie ma szansy na wykorzystanie ew. równoległości obliczeń
- błąd w jednym bicie kryptogramu powoduje błąd w dwóch odszyfrowanych blokach

- OFB (*output feedback*)

- strumień pseudolosowy jest generowany z szyfru blokowego:
 $r_j = \text{Enc}(k, r_{j-1})$, $r_0 = IV$
- szyfrowanie i odszyfrowywanie: $m_j \oplus r_j$, $c_j \oplus r_j$



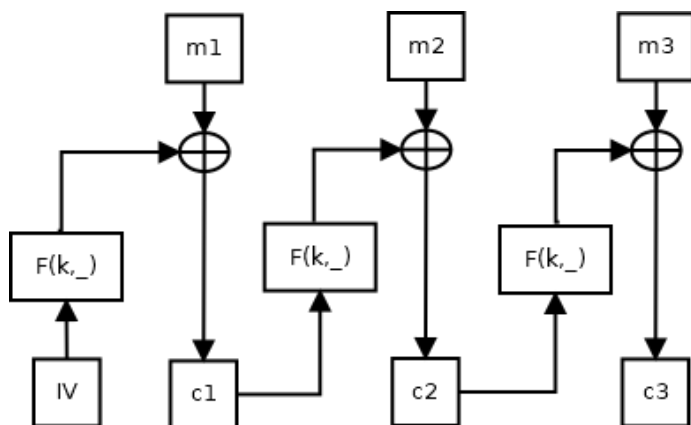
- Zalety:

- błędy w szyfrogramie nie propagują się wcale
- ciąg r_j można przygotować przed otrzymaniem szyfrogramu
- jeśli F jest funkcją pseudolosową, to szyfr jest bezpieczny na atak z wybranym tekstem jawnym
- pod warunkiem, że wektor początkowy jest losowy
- F nie musi być permutacją

- Wady:

- nie można wykorzystać przetwarzania współbieżnego

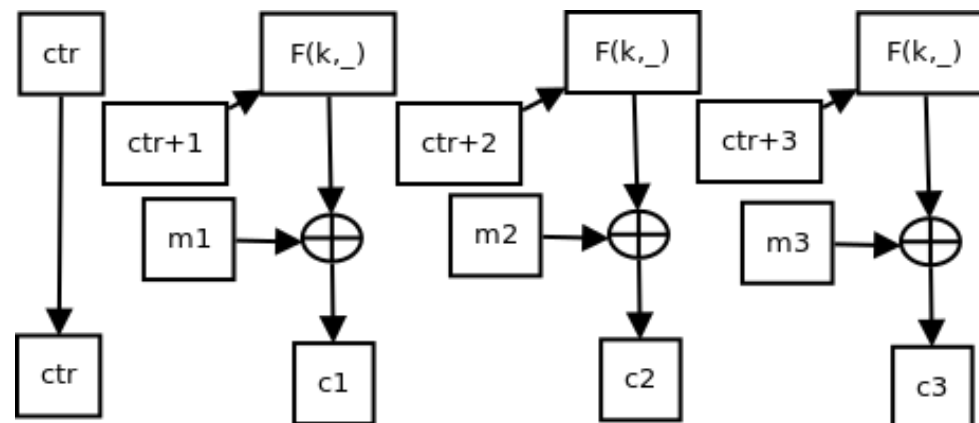
- CFB (*cipher feedback*)
 - $c_j = m_j \oplus \text{Enc}(k, c_{j-1})$, $m_j = c_j \oplus \text{Enc}(k, c_{j-1})$, $c_0 = IV$
- cechy podobne do CBC, sekwencyjne przetwarzanie, ew. błąd propaguje się na dwa bloki



Szyfry blokowe

- Bezpieczeństwo zależy od wielkości bloku
 - mniejszy blok zwiększa szansę na powtórzenie wektora losowego
 - *2połowa długości bloku* powinno być dużą liczbą
 - blok 64 bitowy dzisiaj jest za mały
- Szyfr blokowy może działać jak szyfr strumieniowy
 - jedyną zaletą szyfru strumieniowego jest wydajność
 - dzisiaj jest to coraz mniej znaczące (może telefony?)

- CTR (*counter*)
 - strumień pseudolosowy jest generowany z szyfru blokowego: $r_j = \text{Enc}(k, \text{ctr} + j)$, $\text{ctr} = IV$
 - rejestr jest licznikiem, umożliwia działanie współbieżne
 - można szyfrować niezależne fragmenty



Atak z wybranym kryptogramem

- Ewa ma zgadnąć, która wiadomość jest zaszyfrowana
 - ale ma prawo żądać szyfrowania i odszyfrowania dowolnej wiadomości oprócz dwóch konkursowych
- W trybach OFB, CFB, CTR – kryptogram jest sumą tekstu jawnego i pewnego ciągu
 - lekkie zaburzenie powoduje, że wynik jest zbliżony
- W trybie CBC zmiana bloku powoduje zmiany wszystkich dalszych, ale nie wcześniejszych
- Szyfr odporny na atak z wybranym kryptogramem powinien mieć własność odwrotną do odporności na zakłócenia
 - tzn. mała zmiana kryptogramu powoduje dużą zmianę lub niemożność odszyfrowania
- Atak z wybranym kryptogramem jest sens rozpatrywać, gdy przestrzeń tekstów jawnych czy kryptogramów jest ograniczona, może kilobajty, ale nie megabajty długości

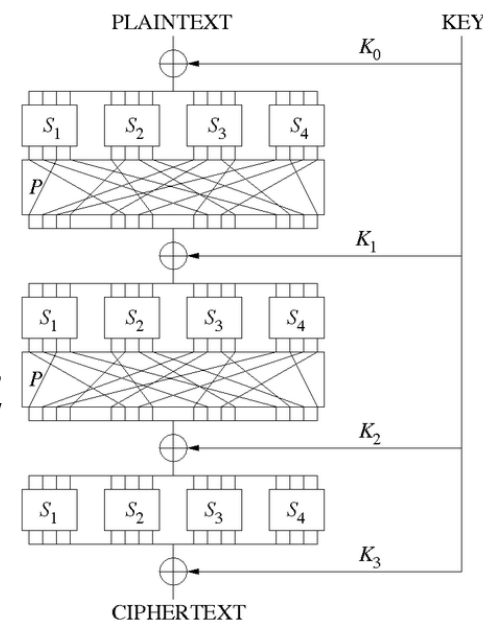
Implementacja szyfrów blokowych

Konfuzja i dyfuzja (Shannon)

- Konfuzja
 - de facto inna nazwa na permutację (bijekcję)
- Bijekcja $F(k, \cdot) : [n] \rightarrow [n]$ wymaga $\log(2^n!) \approx n \cdot 2^n$ bitów
 - nierealne dla $n = 64$
- Podział na mniejsze bloki, np.
 - $F(k, \langle x_1, \dots, x_8 \rangle) = \langle F_1(k, x_1), \dots, F_8(k, x_8) \rangle, x_i \in [8]$
 - dla $n = 64$ zamiast 2^{70} potrzeba $8 \cdot 8 \cdot 2^8 = 2^{14}$ bitów
 - na pewno nie jest pseudolosowa
 - bo zmiana jednego bitu powinna wpływać na wszystkie bity, nie tylko w obrębie podbloku
- Dyfuzja
 - trzeba wymienić bity pomiędzy małymi blokami
 - zmiana w jednym małym bloku wpływa na inne

Sieć podstawień i permutacji, przykład zmniejszony

- Każda z bijekcji $F_i(k, x_i)$ ma postać $S_i(k \oplus x_i)$
 - bijekcje S_i są ustalone
 - nazwa: podstawienie, S-box
 - zależność od klucza jest taka jak w szyfrach strumieniowych
- Dyfuzja
 - przestawia bity, nazwa: permutacja, nie mylić z bijekcją na zbiorze ciągów bitów
- Operacje $\oplus$ z kluczem, zastosowanie S-boksu oraz permutacja bitów są wielokrotnie iterowane
 - klucze k_i dla różnych rund są generowane z klucza k



ŹRÓDŁO: WIKIPEDIA

Zasady budowy sieci podstawień i permutacji

- Odwracalność
 - konieczność odszyfrowywania
 - dodawanie klucza jest odwracalne
 - podstawienia muszą być permutacjami (matematycznie)
 - permutacja bitów musi być permutacją (matematycznie)
- Efekt lawiny
 - zmiana jednego bitu na wejściu może zmienić każdy bit na wyjściu
 - w każdym S-boksie musi zmienić co najmniej dwa bity
 - te dwa bity są przekazane do dwóch różnych S-boksów
 - liczba rund musi być wystarczająco duża

- Jedna runda
 - szyfr bezwartościowy, jedna para tekst jawny+kryptogram ujawnia klucz
- Dwie rundy
 - z góry wiadomo do jakich S-boksów wędrują zmiany na wejściu
 - np. dla bloku 64 bitowego i 4 bitowych S-boksów trzeba wypróbować $2^{16} \cdot 2^4$ kluczy i powtórzyć to 2^7 razy
 - złożoność ataku 2^{27} – dużo za mało
- Trzy rundy
 - na pewno można odróżnić od permutacji losowej
 - bo zmiany bitów ujawniają się w z góry znanych miejscach

DES (Data Encryption Standard)

- IBM, 1974
 - pierwsza wersja Lucifer
 - wersja obecna, 1975, standard od 1977
 - stracił pozycję w 2000, ale do dziś używany
- Szyfr blokowy (64 bitowy)
 - w pełnej wersji wymaga zdefiniowania trybu użycia szyfru blokowego
 - oparty o sieć Feistela
- Klucz 56 bitowy
 - dzisiaj możliwe jest złamanie brutalne, ale nie jest to trywialne
- Nie znaleziono innego ataku na ten szyfr
 - świadczy to o doskonałości projektu

- Przekształcenie Feistela
 - niech $f : [\ell] \times [n] \rightarrow [n]$ jest dowolną funkcją, definiujemy $F(k, \langle x, y \rangle) = \langle x, y \oplus f(k, x) \rangle$, $F : [\ell] \times [2n] \rightarrow [2n]$
 - wówczas $F(k, F(k, \langle x, y \rangle)) = \langle x, y \oplus f(k, x) \oplus f(k, x) \rangle = \langle x, y \rangle$, tzn. F jest odwrotna do siebie
 - w praktyce lepiej zamienić argumenty $F(k, \langle y, x \rangle) = \langle x, y \oplus f(k, x) \rangle$ funkcja odwrotna zaczyna od zamiany, wielokrotne złożenie F jest mocno mieszające: $x, y \oplus f(k, x), x \oplus f(k, y \oplus f(k, x)), \dots$
- Sklejanie i rozkład bloków
 - blok $2n$ bitów to dwa bloki n -bitowe
 - $L_i = R_{i-1}$, $R_i = L_{i-1} \oplus f_i(R_{i-1})$, f_i może zależeć od rundy
 - odwrotne odwzorowanie ma podobny wzór
- Funkcja f nie musi być bijekcją

Trzy składniki DES-a

- Permutacja rozszerzająca: $[n] \rightarrow [m]$, $m \geq n$, zapisywana jest w postaci tabeli o m pozycjach o wartościach $1, \dots, n$, np. 12434356 pokazuje jak przenieść 6 bitów na 8 bitów wyjściowych
- S-box (*substitution box* – tabelaryczny zapis funkcji)
 - $S : [m] \rightarrow [n]$, $m \geq n$
 - w tabeli jest 2^m pozycji, każda zapisywana n bitami
 - np.

101,	010,	001,	110,
011,	100,	111,	000,
001,	100,	110,	010,
000,	111,	101,	011
 - jest to S-box $[4] \rightarrow [3]$, każda wartość na wyjściu powtarza się dokładnie 2 razy
- Klucze dla rund, np. z klucza $x_1, \dots, x_9$ ciąg dziewięciu kluczy 8-bitowych: każdy zaczyna się od kolejnego bitu

- Bloki są 12 bitowe
 - dzielone są na dwa po 6 bitów, L_0, R_0
- W jednej rundzie $L_i = R_{i-1}, R_i = L_{i-1} \oplus f(k_i, R_{i-1})$
- Występuje wiele rund, w ostatniej jeszcze zamiana L_n i R_n
- Więc odwzorowanie odwrotne ma klucze w odwrotnej kolejności, poza tym jest identyczne
- $f : [8 + 6] \rightarrow [6]$ jest zdefiniowane $f(k, R) = S_1 \times S_2(E(R) \oplus k)$, gdzie
 - $E : [6] \rightarrow [8]$ jest ustaloną permutacją rozszerzającą
 - k jest kluczem 8-bitowym, innym dla każdej rundy
 - S_1 i S_2 są dwoma ustalonymi S-boksami, tzn. funkcjami $[4] \rightarrow [3]$
 - blok 8-bitowy jest dzielony na dwa, każda z funkcji jest stosowana osobno, wyniki są sklejone do bloku 6 bitów

- Bloki są 64 bitowe, występuje 16 rund
- Klucz jest 56 bitowy, podzielony na dwa bloki C_0, D_0
 - w każdej rundzie nowe bloki są $C_i = P_i(C_{i-1})$, podobnie D_i
 - P_i jest przesunięciem bitów o jeden lub dwa bity (określa to tabela 1122222212222221 – 1, 2, 9 i ostatnia runda ma przesunięcie o 1)
 - z 56 bitów wybieranych jest 48 według ustalonego wzorca
- Na wstępie wykonywana jest dodatkowa ustalona permutacja na zakończenie permutacja odwrotna
- $f : [48 + 32] \rightarrow [32]$ wykorzystuje ustaloną permutację rozszerzającą $E : [32] \rightarrow [48]$, osiem S-boksów $[6] \rightarrow [4]$ i dodatkową ustaloną permutację bitów na koniec:
 - $f(R, K) = C(S_1 \times \dots \times S_8(E(R) \oplus K))$
- Permutacje bitów, S-boksy, przesunięcia klucza są częścią standardu

DES

- rysunek z dokumentu FIPS46–3, czyli standardu opisującego DES

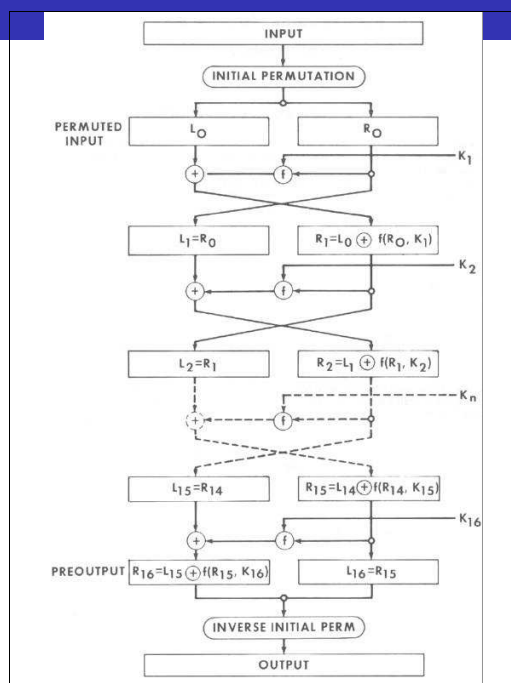


Figure 1. Enciphering computation.

Zasady projektu DES

- S-boksy są funkcjami dalekimi od liniowych
 - jeśli na wejściu do S-boksu jest różnica jednego bitu, to na wyjściu różnią się dwoma bitami
 - ustalamy ciąg 6 bitów z i rozpatrujemy 32 pary (x, y) takie, że $x \oplus y = z$ obliczamy $Sx \oplus Sy$, najwyżej 8 par ma prawo dać ten sam wynik
 - utrudnia/uniemożliwia to kryptoanalizę różnicową
- Permutacja na początku i na końcu – nie wiadomo dlaczego!
- Tw.: DES nie jest grupą, tzn. $E(k_1); E(k_2)$ nie jest szyfrowaniem DES (w przeciwieństwie do szyfru Cezara, afinicznego, Hilla)
- dyfuzja: zmiana jednego bitu w tekście jawnym zmienia dużo w szyfrogramie (ok. połowy bitów w bloku)

- Pierwsze wątpliwości (za krótki klucz) natychmiast, 1975
- Zatwierdzenie standardu w 1983, 1988 i 1993
- Propozycja zbudowania komputera o specjalnej architekturze, 1993
- Propozycja wykorzystania obliczeń rozproszonych (łatwa dostępność sieci tysięcy komputerów), 1996
- Złamanie klucza, 1997 – kilka miesięcy obliczeń
 - 1998, wystarczyło 39 dni
 - 1999, czas liczony w godzinach
- DES Cracker – zakłada, że tekst jawny jest literą ascii 7-bit
 - szansa, że losowy klucz wyprodukuje 8 liter jest 1/60tys większość kluczy jest szybko odrzucana
 - maszyna ma 1500 procesorów, łamie każdy klucz w 4 dni
- Cost-Optimized PArallel COde Breaker (2006/2008): 1 dzień

Dane historyczne

- Pierwszy IBM PC: 4.7 MHz, 8 bitowy procesor
 - szyfrował w tempie 370 bloków (3 kB) na sekundę
 - 66 MHz 486, 16 bitowy procesor: 100 razy szybciej
 - 2GHz Pentium4, 32 bitowy procesor: 20 razy szybciej
- Hasła w systemie UNIX (historia)
 - 8 bajtów hasła tworzy klucz (56 bitów)
 - przechowuje się wynik szyfrowania kluczem ciągu zer
 - można dodać 12 bitów ziarna, zmieniają one nieco przebieg algorytmu DES
 - dłuższe hasła są niewykorzystane, krótsze uzupełniane
 - dziś stosuje się inne funkcje skrótu

- Analiza różnicowa
 - odkryta w 1980
 - wymaga 2^{49} wybranych tekstów jawnych – za dużo
 - była znana i tajna w IBM
 - DES był zaprojektowany z myślą o odporności na ten atak
- Analiza liniowa
 - odkryta w 1993
 - wymaga 2^{43} wybranych tekstów jawnych, też dużo
 - nie była znana projektantom DES'a
- Połączenie ataków i usprawnienia
 - wymaga 2^{16} wybranych tekstów jawnych i 2^{29} obliczeń

Ratowanie DES-a

- Szyfrowanie dwukrotne: $E(k_1, E(k_2, m))$, przestrzeń kluczy jest iloczynem kartezjańskim, tzn. jest ich 2^{112}
- dany jest tekst jawny m i zaszyfrowany j.w.
 - $m = D(k_2, D(k_1, c))$ gdzie $c = E(k_1, E(k_2, m))$ czyli $D(k_1, c) = E(k_2, m)$
 - atak urodzinowy: można przygotować listę 2^{56} szyfrogramów $E(k, m)$ oraz 2^{56} tekstów odszyfrowanych $D(k, c)$ i sprawdzać, czy jest para identycznych
 - a więc dwukrotne szyfrowanie daje lepszą ochronę tylko przy założeniu, że barierą jest złożoność pamięciowa
 - trzykrotne szyfrowanie ma w tym sensie moc 2^{112}

- 3DES – trzykrotne szyfrowanie
 - moc co najmniej taka jak szyfru 112 bitowego
 - zdefiniowano wersję $E(k_1, k_2, m) = E(k_1, D(k_2, E(k_1, m)))$
 - użycie D zamiast E nie zmienia siły, a ułatwia kompatybilność, $k_1 = k_2$ oznacza zwykłe szyfrowanie DES
 - uwaga: trzy różne klucze nie zwiększają mocy, klucze w innej kolejności zabijają ideę
- DESX – $E(k_1, k_2, k_3, m) = k_3 \oplus E(k_2, k_1 \oplus m)$
 - moc jest taka jak przestrzeń trzech kluczy
 - nie widać ataku podobnego do poprzedniego

- 1997 – wezwanie NIST do zmiany szyfru
- założenia: klucze długości 128, 192 i 256 bitów
 - praca na procesorach 8-bitowych (karty inteligentne)
 - 15 kandydatur rozpatrywanych
- 5 finalistów:
 - MARS (IBM)
 - RC6 (RSA)
 - Rijndael (2 naukowców z Belgii)
 - Serpent (3 naukowców z USA)
 - Twofish (m.in. Schneier)
- obecny standard: AES=Rijndael

Rijndael

- Sieć podstawień i permutacji
 - blok 128 bitów (macierz 4×4 bajtów)
 - każdy bajt jest podstawiany za pomocą tego samego S-boksu
 - permutacją jest przesunięcie bajtów w wierszach macierzy
 - dodatkowo operacja algebraiczna na całej macierzy
- Oparty o teorię ciał skończonych (256 elementowych)
 - elementy przedstawiane są jako formalne wielomiany 8 stopnia, mnożenie wykonywane jest modulo pewien ustalony wielomian
 - pozwala to traktować bajty jak liczby, definiować operacje na wektorach i macierzach, są one odwracalne