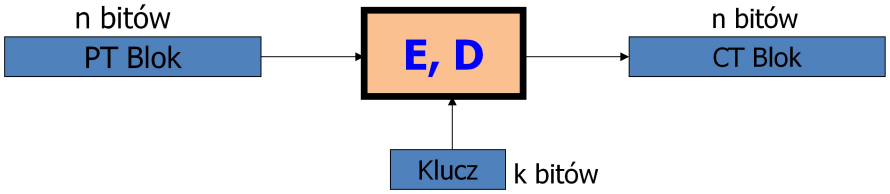
Kryptografia i bezpieczeństwo danych
- SZYFRY BLOKOWE I

Sławomir Samolej
ssamolej.kia.prz.edu.pl
ssamolej@prz.edu.pl

1

Definicja szyfru blokowego

Szyfry blokowe – zasada działania



Popularne przykładowe algorytmy:

1. 3DES: n= 64 bitów, k = 168 bitów
2. AES: n=128 bitów, k = 128, 192, 256 bitów

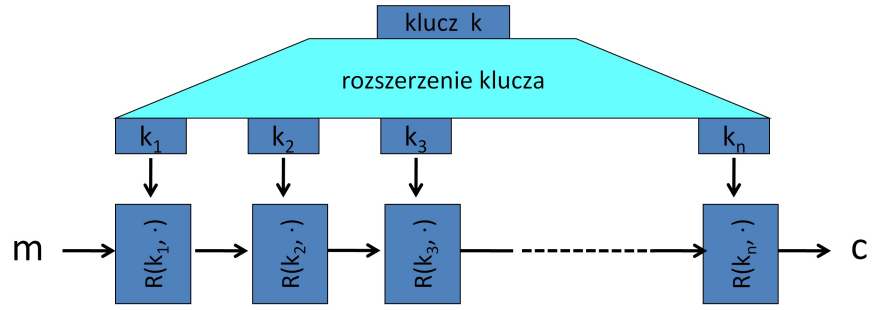
3

Szyfr blokowy składa się z dwóch algorytmów: E i D. Są to odpowiednio algorytmy szyfrujący i deszyfrujący. Oba algorytmy pobierają jako wejście klucz K. Zasada działania szyfru blokowego polega na tym, że bierze on N bitów danych do zaszyfrowania i wylicza dokładnie N bitów danych zaszyfrowanych. Następuje „zamapowanie" N bitów wejściowych na dokładnie N bitów wyjściowych.

W dolnej części slajdu zamieszono dwa reprezentatywne przykłady szyfrów blokowych. Pierwszym z nich jest „potrójny DES" (3DES, Triple Data Encryption Algorithm). W algorytmie potrójny DES jednorazowo jest szyfrowany blok 64 bitów a szyfrowanie odbywa się z zastosowaniem klucza o długości 168 bitów (algorytm potrójny DES jest uznany za przestarzały od 2018 roku i jego stosowanie będzie zabronione od 2023 roku).

Algorytm AES (Advanced Encryption Standard) jest nowszy. Został przyjęty jako standard w 2001 roku. Posiada inne parametry niż 3DES. Szyfruje pojedynczy blok o długości 128 bitów. Dopuszcza się 3 długości klucza: 128, 192 lub 256 bitów. Co do zasady im dłuższy jest klucz, tym dłużej wykonuje się algorytm, ale tym trudniejszy do złamania (bardzie bezpieczny).

Szyfry blokowe są budowane z zastosowaniem iteracji



R(k,m) jest nazywana funkcją rundy

dla 3DES (n=48), dla AES-128 (n=10)

4

Szyfry blokowe działają przez wykonanie szeregu iteracji. W pierwszej kolejności klucz K jest rozszerzany na klucze K1 do Kn nazywane kluczami rundy. Następnie wiadomość do zaszyfrowania jest wielokrotnie szyfrowana przez funkcje rundy. Wejściem każdej z funkcji jest kolejny klucz Kn oraz wyjście poprzednie funkcji rundy. Pierwszym wejściem algorytmu jest oczywiście wiadomość do zaszyfrowania i pierwszy klucz. Po każdej rundzie otrzymuje się blok wyjściowy zawsze o takiej samej długości jak dane wejściowe. Liczba rund zależy od algorytmu. W 3DES jest 48 rund, w AES – 10.

Porównanie wydajności

Crypto++ 5.6.0 [Wei Dai]

AMD Opteron, 2.2 GHz (Linux)

	Szyfr	Blok/roz. klucza	Szybkość (MB/sec)
strumieniowy	RC4		126
	Salsa20/12		643
	Sosemanuk		727
blokowy	3DES	64/168	13
	AES-128	128/128	109

5

Porównanie wydajności poszczególnych algorytmów na tym samym komputerze pokazano na obecnym slajdzie. Można zauważyć, że szyfry blokowe są wolniejsze od strumieniowych. Jednak w dalszej części wykładu zostanie pokazane, że z zastosowaniem szyfrów strumieniowych pewne problemy mogą być rozwiązane bardziej efektywnie. Pozwalają one między innymi zagwarantować integralność i wiele innych właściwości wymaganych we współczesnych systemach ochrony danych.

Uwagi o bezpieczeństwie - nieformalnie

- Wykonywanie poszczególnych faz szyfru blokowego (funkcje rund) można nazwać generowaniem kolejnych pseudolosowych permutacji (permutacja - ustawianie elementów zbioru w pewnej kolejności)
- Bezpieczeństwo szyfrów blokowych opiera się na tym, że wygenerowane permutacje muszą być nierozróżnialne od ciągów losowych.

6

Algorytm DES

Historia DES (Data Encryption Standard)

- Wczesna lata 70: Horst Feistel projektuje algorytm szyfrujący Lucifer (IBM)
  długość klucza = 128 bits ; długość bloku = 128 bits
- 1973: NBS prosi o przedłożenie propozycji szyfru blokowego.
  IBM wysyła wariant szyfru Lucifer.
- 1976: NBS przyjmuje DES jako standard federalny
  długość klucza = 56 bity; długość bloku = 64 bity
- 1997: DES jest złamane przez pełne przeszukiwanie
- 2000: NIST przyjmuje szyfr Rijndael jako AES jako następca DES

DES był szeroko zastosowany w bankowości i handlu.

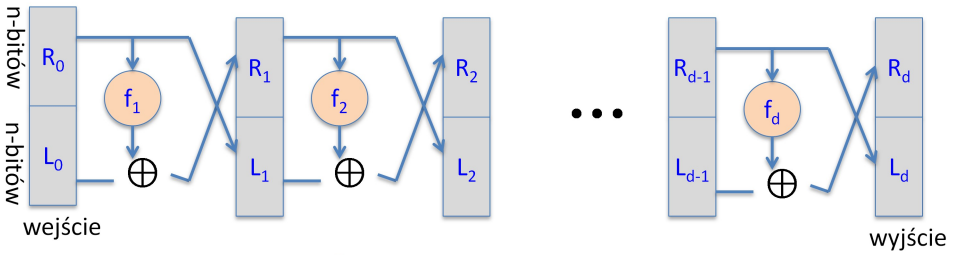
8

We wczesnych latach 70-tych IBM zdał sobie sprawę, że klienci firmy oczekują jakiejś formy szyfrowania danych. Została sformowana grupa, która miała się zająć wprowadzaniem rozwiązań kryptograficznych. Szefem grupy był Horst Feistel, który na początku lat 70-tych zaprojektował szyfr blokowy nazwany Lucifer. Szyfr miał wiele wariantów, ale w jednej z ostatnich wersji miał 128 bitowy klucz i 128 bitową długość bloku. W 1973 roku NBS (National Bureau of Standards) poprosiło o zaproponowanie szyfru blokowego, który miał się stać standardem federalnym. W 1976 roku Lucifer nazwano DES i stał się on federalnym standardem. W standardzie jednak przyjęto tylko 56 bitową długość klucza i 64 bitowa długość bloku. W 1997 roku szyfr został złamany przez brutalny atak (przeszukiwanie wszystkich 2^{56} kluczy). Uznano, że szyfr nie jest już bezpieczny i ogłoszono konkurs na opracowanie kolejnych rozwiązań. W 2000 roku przyjęto za standard szyfr blokowy Rijndael i nazwano go AES (Advanced Encryption Standard). DES był długo stosowany w bankowości i handlu. Od kilku lat jest zastępowany nowymi rozwiązaniami.

Podstawowy pomysł na DES:
Sieć Feistela

Mamy dane funkcje $f_1, \ldots, f_d$: $\{0,1\}^n \longrightarrow \{0,1\}^n$

Cel: zbudowanie odwracalnej funkcji $F: \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$



Symbolicznie:
$$\begin{cases} R_i = f_i(R_{i-1}) \oplus L_{i-1} \\ L_i = R_{i-1} \end{cases} \quad i = 1, 2, 3, \ldots, d$$

9

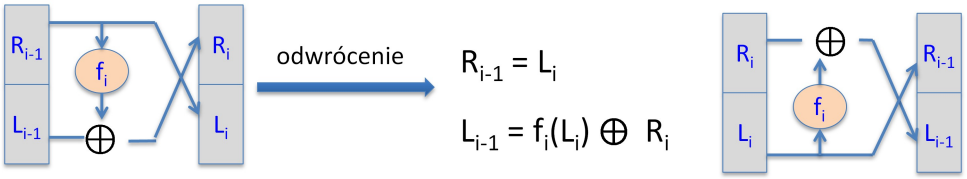
Główna idea szyfru DES opiera się na sieci Feistela opracowanej przez Horsta Feistela. To jest pomysł na zbudowanie szyfru blokowego z zastosowaniem ustalonych arbitralnie funkcji f_1 do f_d. Zakładamy, że mamy funkcje f_1 do f_d przekształcające n bitów w n bitów. Funkcje nie muszą być odwracalne (odwracalne to znaczy, że podając wynikowy ciąg z powrotem na wejście otrzymamy ciąg początkowy). Chcemy z tych funkcji zbudować funkcję F odwracalną ale przekształcające 2n bitów na 2n bitów.

Na wejście podajemy dwa bloki n-bitowe R odpowiada „prawemu", L „lewemu" (zwykle sieć jest opisywana od góry do dołu). W każdym przekształceniu nowy blok L_i jest niezmienioną wartością bloku R_{i-1} z poprzedniej fazy przekształcenia. Wartość R_i jest obliczana w następujący sposób. Brana jest wartość R_{i-1}, przekształcana za pomocą funkcji f_i, a następnie na otrzymanej wartości wykonywana jest operacja xor z wartością L_{i-1}.

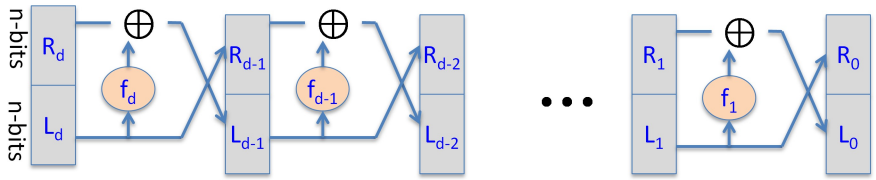
Opisana operacja ilustruje jedną rundę obliczeń. I jest wykonywana przez funkcję f_1. Kolejne rundy wyglądają tak samo, przy czym za każdym razem stosujemy inną funkcję f_i. Na koniec (po d rundach) otrzymujemy dwa ciągi bitów R_d i L_d.

Ważna właściwość sieci Feistela

- Przy sieci skonstruowanej jak na poprzednim slajdzie operacja obliczania poszczególnych rund jest **odwracalna!!!**
- Można sobie wyobrazić, że wykonujemy operacje na otrzymanym wyjściu w następujący sposób:



W konsekwencji otrzymujemy regułę deszyfrowania



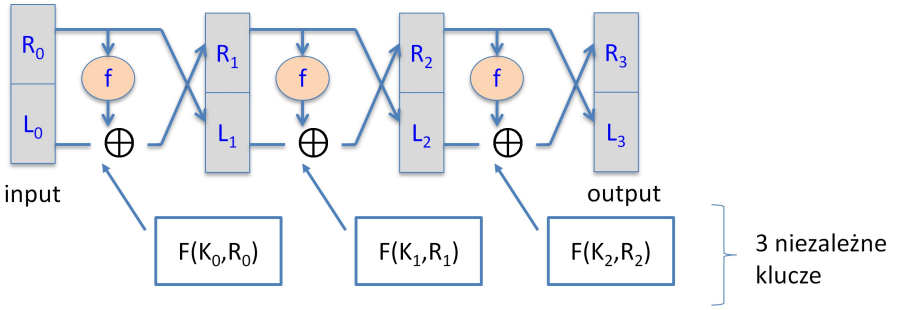
- Odwrócenie ma taką samą postać jak szyfrowanie, przy czym funkcje $f_1, \ldots, f_d$ są zastosowane w odwrotnej kolejności
- Otrzymujemy ogólną metodę na budowanie odwracalnych funkcji (szyfrów blokowych) z zastosowaniem ustalonego zbioru funkcji
- Podejście jest stosowane w wielu szyfrach blokowych … ale nie AES

11

Takie rozwiązanie jest atrakcyjne w rozwiązaniach sprzętowych, ponieważ szyfrowanie i deszyfrowanie ma taką samą strukturę. Jest stosowane w wielu szyfrach blokowych (ale nie AES). Nowe szyfry blokowe proponowano zmieniając tylko zestaw funkcji.

Ważne twierdzenie (Luby-Rackoff '85)

- Nieformalnie:
  Jeśli na wejście sieci Feistela podamy bezpieczny pseudolosowy ciąg danych, to po wykonaniu 3 rund obliczeń w drzewie otrzymamy bezpieczną pseudolosową permutację.
  - po 3 rundach szyfru blokowego otrzymujemy bezpieczny szyfr (oczywiście, jeśli na jego wejście podamy bezpieczny ciąg pseudolosowy)

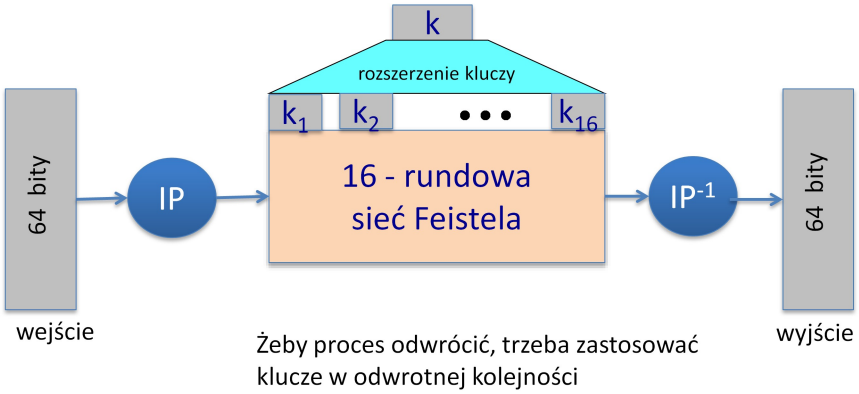


12

Ważna uwaga: w każdej rundzie musimy się posługiwać osobnym, niezależnym kluczem (w praktyce to jest podklucz wygenerowany z klucza pierwotnego, co może być słabym punktem systemu). Tworzy to teoretyczną podstawę do stosowania sieci Feistela w tworzeniu szyfrów blokowych.

DES: 16- rundowa sieć Feistela

$f_1, \ldots, f_{16}: \{0,1\}^{32} \longrightarrow \{0,1\}^{32}$, $f_i(x) = \mathbf{F}(k_i, x)$



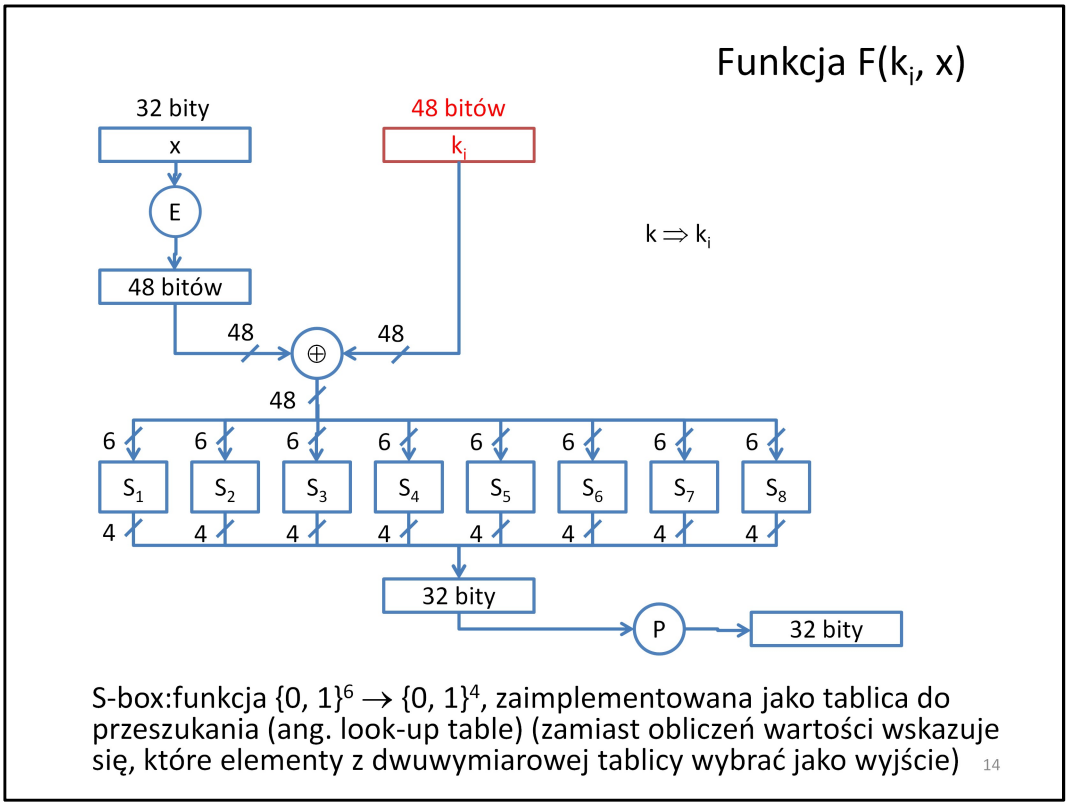
Żeby proces odwrócić, trzeba zastosować klucze w odwrotnej kolejności

13

DES jest 16-rundową siecią Feistela. Funkcje f_1 do f_{16} przekształcają 32 bity w 32 bity. Ostatecznie ze względu na to, że szyfrowanie ma postać sieci Feistela w czasie szyfrowania jednego bloku następuje przekształcenie 64 bitów w 64 bity. 16 funkcji f jest w rzeczywistości jedną funkcją F biorącą na wejście unikatowy klucz k_i (klucz danej rundy) otrzymany z rozszerzenia klucza szyfrowania k (56 bitów). W rezultacie otrzymujemy 16 różnych funkcji rundowych.

Szyfrowanie rozpoczyna się od obliczenia początkowej permutacji (IP) wiadomości do zaszyfrowania. Nie ma to znaczenia kryptograficznego, ale zostało ustalone na poziomie standardu. Następnie permutacja wiadomości przechodzi przez 16-rundową sieć Feistela. Po wyjściu z sieci dane są poddawane finałowej permutacji (IP^{-1}), która jest odwróceniem początkowej permutacji (be znaczenia dla bezpieczeństwa). Ostatecznie otrzymujemy wyjście algorytmu szyfrowania.

Do obliczenia funkcji poszczególnych rund, początkowy 56 bitowy klucz jest rozszerzany na 16 48-bitowych kluczy. Aby odzyskać zaszyfrowaną wiadomość wystarczy przepuścić szyfrogram przez tę samą strukturę, gdzie funkcje rund są podane od ostatniej do pierwszej.



Wejściem funkcji F jest 32 bitowa wartość x (połowa ciągu bitów do zaszyfrowania) i 48-biowy klucz rundy wygenerowany z klucza bazowego. Wartość x przechodzi przez blok rozszerzania (E), gdzie zostaje wydłużona z 32 bitów na 48 bitów. Blok rozszerzania replikuje niektóre bity a inne przemieszcza. Np. bit nr 1 ciągu x jest replikowany na pozycję 2 i 48 ciągu wyjściowego. Bit 2 jest przenoszony na pozycję 3 wyjścia. Podobne czynności wykonywane są dla innych bitów aby otrzymać 48-bitowy ciąg z ciągu 32-bitowego. Na rozszerzonej wartości x i kluczu rundy k_i wykonywana jest operacja xor. Następnie ta 48-bitowa wartość jest podzielona na 8 grup po 6 bitów. Każda z 6-bitowych wartości jest przepuszczana przez tzw. S-box (zasada działania zostanie pokazana za chwilę), który przekształca ją w ciąg 4-bitowy. Otrzymane ciągi są połączone w jeden 32-bitowy, który poddawany jest jeszcze permutacji P przestawiającej jego bity.

Stosując różne klucze rundy otrzymujemy 16 różnych funkcji rundy. Funkcja S przekształca 6 bitów w 4 bity i jest implementowana w postaci tablic do przeszukania (w tablicy są zgromadzone pewne wartości, algorytm wykonuje operacje tylko na indeksach tablicy i na koniec wskazuje pewne elementy z tablicy, jest to sposób na przyspieszenie wyliczenia wartości przez komputer).

Przykładowy S-box

$S_i: \{0,1\}^6 \longrightarrow \{0,1\}^4$

S_5		Middle 4 bits of input															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
Outer bits	**00**	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

0 1101 **1** → 1001

15

Tablica przeszukiwania ma 64 wartości odpowiadające 2^6 możliwym wejściom. Na slajdzie pokazana jest postać jednej z tablic (S_5). Wyjście S-box'a otrzymuje się na skrzyżowaniu odpowiedniego wiersza i kolumny. Wejście składa się z pierwszego bitu pierwszej kolumny, potem 4 bitów znajdujących się w pierwszym wierszu i drugiego bitu pierwszej kolumny (proszę porównać rysunek). Przykładowo, dla wejścia **0**11011 wyjście wynosi 1001.

Przykład: zły dobór tablicy S-box'a

Załóżmy:

$S_i(x_1, x_2, \ldots, x_6) = (x_2 \oplus x_3,\ \ x_1 \oplus x_4 \oplus x_5,\ \ x_1 \oplus x_6,\ \ x_2 \oplus x_3 \oplus x_6)$

Stosując równoważny zapis: $S_i(\mathbf{x}) = A_i \cdot \mathbf{x} \pmod 2$

$$\begin{bmatrix} 0&1&1&0&0&0 \\ 1&0&0&1&1&0 \\ 1&0&0&0&0&1 \\ 0&1&1&0&0&1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} x_2 \oplus x_3 \\ x_1 \oplus x_4 \oplus x_5 \\ x_1 \oplus x_6 \\ x_2 \oplus x_3 \oplus x_6 \end{bmatrix}$$

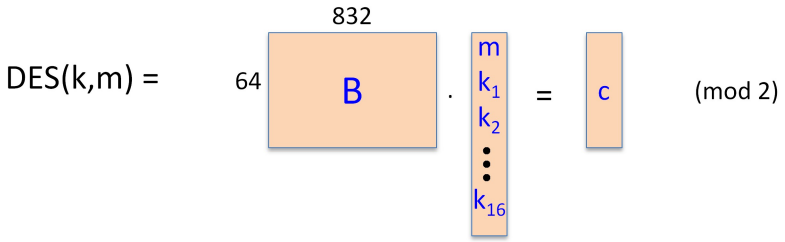
Mówimy, że S_i jest funkcją liniową.

16

Powstaje pytanie, w jaki sposób wybrać taką tablicę. Zaczynamy od złego przykładu. Załóżmy, że 6 bitów $x_1, x_2, \ldots, x_6$ zostanie przekształcone w 4-bitowy ciąg poprzez obliczenie jakiejś ich liniowej kombinacji z zastosowaniem operatora xor, jak na pokazanym slajdzie. Zapis z górnej części slajdu można zamienić na postać macierzową, jak w środkowej części slajdu. Przy takim podejściu do budowania funkcji S-box system szyfrowania nie będzie bezpieczny.

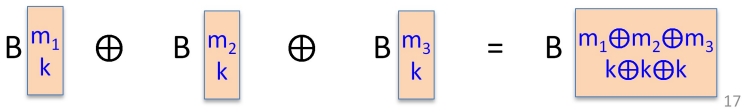
Analiza złego doboru S-box'a

W takim przypadku cały DES byłby liniowy:



Ale wtedy:

$$DES(k,m_1) \oplus DES(k,m_2) \oplus DES(k,m_3) = DES(k, m_1 \oplus m_2 \oplus m_3)$$



17

Przy takim doborze S-box'a cały szyfr DES stałby się liniową funkcją. Można by wyznaczyć jedną macierz B o rozmiarach 64 x 823 wynikających z liniowych operacji przeprowadzonych w poszczególnych rundach i etapach szyfrowania, która mnożona przez wektor złożony z wiadomości do podpisania oraz kluczy poszczególnych rund dałaby zaszyfrowaną wiadomość.

Gdybyśmy rozważyli obliczenie xor trzech zaszyfrowanych wiadomości utworzonych odpowiednio z wiadomości m_1, m_2 i m_3, okazałoby się, że jest ono równoważne zaszyfrowaniu xor tych wiadomości ($m_1 \oplus m_2 \oplus m_3$). Takie przekształcenie na pewno nie jest funkcją losową i nie spełnia warunku bezpieczeństwa. I można łatwo uzyskać wartość klucza szyfrowania (wystarczy 832 par wejście-wyjście).

Dobór S-box'ów

- Okazuje się, że dobór losowych S-box'ów również nie tworzy bezpiecznego szyfru blokowego (można uzyskać klucz po analizie ok. 2^{24} wyjść).
- Reguły do wyboru S-box'ów :
  - Wyjście nie może być bliskie liniowej funkcji na wejściowych bitach
  - …

Ataki siłowe

19

Atak siłowy w celu uzyskania klucza szyfru blokowego

- Zostało wykazane matematycznie, że dysponując dwiema parami: wiadomość/szyfr można dokonać ataku siłowego.
- Przeprowadzenie ataku siłowego oznacza więc odkrycie klucza na podstawie co najmniej dwóch par wiadomość – zaszyfrowana wiadomość
- Dokonuje się tego przez podawanie na wejście wszystkich możliwych kluczy.
- Problem jest do rozwiązania. Stawianym pytaniem jest w **jakim czasie** i **przy pomocy jakich środków** można tego dokonać.

Wyzwanie DES (DES challenge)

msg = "The unknown messages is: XXXX … "

CT = c_1 c_2 c_3 c_4

Cel: znaleźć $k \in \{0,1\}^{56}$ taki, że $DES(k, m_i) = c_i$ dla i=1,2,3

1997: Przeszukiwanie z zastosowaniem Internetu -- **3 miesiące**
1998: Maszyna EFF (deep crack) -- **3 dni** (250 tys. $)
1999: Internet + EFF -- **22 godziny** (nowe wyzwanie)
2006: COPACOBANA (120 FPGAs) **-- 7 dni** (10 tys. $)

21

⇒ Szyfry 56 bitowe nie mogą być już używane!!

⇒ Zastosowanie 128-bitowego klucza wymagałoby 2^{72} dni do złamania

W celu sprawdzenia, ile czasu zajmie brutalne złamanie szyfru DES firma RSA zaproponowała następujące wyzwanie. Firma opublikowała serię zaszyfrowanych wiadomości, ale dla trzech z nich podała teksty jawnych wiadomości. Dokładnie powiedziano, że zaszyfrowana wiadomość ma postać „The unknown message is:XXXXXXXX …". Jawny tekst to trzy 64 bitowe sekwencje, dla których opublikowano szyfry c_1, c_2 i c_3. Zaszyfrowane dalsze części teksu również podano do wiadomości. Zadanie polegało na znalezienie klucza, który pozwoliłby odtworzyć resztę zaszyfrowanej wiadomości (c_4, c_5, …).

Historia złamania DES wygląda następująco. Do przeszukania jest 2^{56} kluczy, ale zwykle wystarczy przeszukać połowę z nich, aby odkryć ten właściwy. W 1997 roku zastosowano zasoby Internetu do przeszukiwania i technologię distributed.net aby złamać DES w ciągu 3 miesięcy. Fundacja EFF (Electronic Frontier Foundation) skontaktowała się z Paulem Kocheram w celu zbudowania dedykowanego rozwiązania sprzętowego do łamania DES. Kosztowało ono 250 000 USD i złamało wyzwanie DES w 3 dni. W 1999 roku RSA zaproponowała nowe wyzwanie i powiedziała, że dokona złamania DES w czasie o połowę krótszym od 3 dni z zastosowaniem maszyny EFF i przeszukiwania w Internecie i dokonała tego w 22 godziny. Można już z tego wywnioskować, że DES jest w obecnej chwili bezużyteczny i nie gwarantuje bezpieczeństwa. Ostatnim gwoździem do trumny było opracowanie układu złożonego z 120 standardowych FPGA w ramach projektu COPACABANA, który był w stanie złamać RSA w 7 dni przy kosztach ok. 10 tys. USD.

DES był bardzo popularnym szyfrem i powstało pytanie, czy można dokonać jego modyfikacji w celu zwiększenia jego bezpieczeństwa. Warto również nadmienić, że po złamaniu DES podjęto poszukiwania w celu opracowania zupełnie nowego szyfru blokowego, czego wynikiem był później standard AES.

Wzmocnienie DES przeciw atakowi siłowemu

Metoda 1: **Potrójny-DES**

- Niech $E : K \times M \longrightarrow M$ będzie szyfrem blokowym
- Definiujemy **3E**: $K^3 \times M \longrightarrow M$ jako

$$\mathbf{3E}\big((k_1,k_2,k_3), m\big) = E(k_1, D(k_2, E(k_3, m)))$$

Jeśli $k_1=k_2=k_3$ otrzymujemy DES

Dla 3DES: rozmiar klucza = 3×56 = 168 bity.
szyfrowanie trwa 3×wolnie niż DES.

(prosty atak siłowy twa $\approx 2^{118}$)

22

Pierwszym pomysłem było wykonanie procesu szyfrowania tej samej wiadomości kilka razy. Mamy trzy klucze, którymi trzykrotnie szyfrujemy jawny tekst. W przypadku 3DES zdecydowano się na interesujące połączenie algorytmów szyfrowania. Wiadomość jest najpierw szyfrowana jednym kluczem, potem deszyfrowana drugim kluczem, a następnie ponownie szyfrowana trzecim kluczem. Jeśli zastosujemy taki sposób szyfrowania, to możemy podać na wejście trzy takie same klucze i otrzymać wiadomość zaszyfrowaną za pomocą jednokrotnego DES. Będzie to działało 3 razy dłużej, ale ten sam układ sprzętowy/program może być zastosowany do szyfrowania za pomocą dwóch różnych algorytmów szyfrowania.

W 3DES w konsekwencji klucz szyfrowania jest połączeniem trzech kluczy poszczególnych algorytmów DES i jego długość 168 bitów zapobiega skutecznie jego załamaniu przy pomocy prostego ataku siłowego. Czas potrzebny do złamania siłowego wynosi 2^{168} i wymaga pracy wszystkich komputerów na świecie przez 10 lat.

Trzeba wiedzieć, że istnieje prostsza metoda ataku, która pozwala złamać 3DES w czasie 2^{118}. Jednak każdy atak na szyfr, który wymaga więcej niż 2^{90} jest uważany za wystarczająco bezpieczny.

Dlaczego nie podwójny DES?

- Definiujemy $2E((k_1,k_2), m) = E(k_1, E(k_2, m))$

długość klucza = 112 bitów

Atak:
dysponujemy: $M = (m_1,\ldots, m_{10})$,
$C = (c_1,\ldots,c_{10})$.

Poszukujemy pary kluczy (k_1, k_2), takich że $E(k_1, E(k_2,M)) = C$
Równoważnie:
$E(k_2,m) = D(k_1,c)$

- krok 1: budujemy tablicę.
  sortujemy po 2 kolumnie

k^0 = 00…00	$E(k^0, M)$
k^1 = 00…01	$E(k^1, M)$
k^2 = 00…10	$E(k^2, M)$
⋮	⋮
k^N = 11…11	$E(k^N, M)$

2^{56} wejść

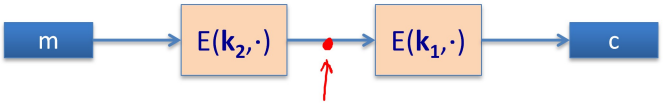
23

3DES jest 3 x wolniejszy od DES, dlaczego więc nie można zastosować podwójnego DES? Dla podwójnego DES klucz miałby długość 112 bitów, jego siłowe złamanie dalej będzie trwało zbyt długo. Okazuje się, że taka konstrukcja jest nie jest bezpieczna.

Załóżmy, że dysponujemy zbiorem wiadomości $M=(m_1, \ldots,m_{10})$ i zbiorem ich szyfrogramów $C=(c_1,\ldots,c_{10})$. Poszukujemy takiej pary (k_1, k_2), takich że $E(k_1, E(k_2,M)) = C$. Przy takiej strukturze kryptograficznej można zauważyć, że jest ono równoważne $E(k_2,M) = D(k_1,C)$ (wystarczy zastosować algorytm deszyfracji z kluczem k_1 „po obu stronach wyrażenia"). W otrzymanym wyrażeniu nastąpiło odseparowanie kluczy k_1 i k_2. To zwykle oznacza, że na taką konstrukcję szyfrującą istnieje szybszy atak niż siłowy. Ten atak nazywa się „spotkanie po środku" (ang. Meet-in-the-middle attack) i jest skierowany na miejsce pomiędzy pierwszym a drugim etapem szyfrowania. Szukamy osobnych kluczy, z których jeden zaszyfruje nam wiadomość, drugi odszyfruje szyfrogram, a rezultat obu operacji będzie ten sam.

Atak może być przeprowadzony w następujący sposób. Na początku trzeba zbudować tablicę wszystkich możliwych wartości klucza k_2 i zaszyfrować wiadomość z tymi wszystkimi kluczami (trzeba wygenerować 2^{56} szyfrogramów). Następnie należy posortować ją według drugiej kolumny (szyfrogramów). Jak dotąd złożoność obliczeniowa tych dwóch operacji wynosi $2^{56}*\log(2^{56})$.

Atak „spotkanie w środku”



Atak: $M = (m_1,\ldots, m_{10})$, $C = (c_1,\ldots,c_{10})$

$k^0 = 00\ldots00$	$E(k^0, M)$
$k^1 = 00\ldots01$	$E(k^1, M)$
$k^2 = 00\ldots10$	$E(k^2, M)$
⋮	⋮
$k^N = 11\ldots11$	$E(k^N, M)$

- Krok 1: zbuduj tablicę.
- Krok 2: dla wszystkich $k \in \{0,1\}^{56}$ wykonaj:
  sprawdź, czy $D(k, C)$ znajduje się w 2 kolumnie.

  jeśli tak, to $E(k^i,M) = D(k,C) \Rightarrow (k^i,k) = (k_2,k_1)$

24

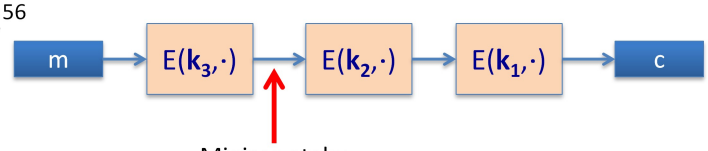
Teraz postępujemy „od drugiej strony”. Deszyfrujemy szyfrogram za pomocą wszystkich możliwych kluczy k_1. Potem poszukujemy, gdzie rezultat deszyfracji znajduje się w posortowanej tablicy. Jeśli znajdziemy tę wartość, wiemy, że wiadomość zaszyfrowana kluczem k_2 wygląda dokładnie jak szyfrogram odszyfrowany kluczem k_1. Znaleźliśmy parę kluczy k_1 i k_2 stanowiących klucz podwójnego DES.

Złożoność obliczeniowa ataku

Czas = $2^{56}\log(2^{56}) + 2^{56}\log(2^{56}) < 2^{63} << 2^{112}$,

Rozmiar tablicy do przeszukania ≈ 2^{56}

Taki sam atak można wykonać na 3DES: czas = 2^{118} ,
r. tablicy do przesz ≈ 2^{56}



Miejsce ataku
na 3 DES

25

Przyjrzyjmy się złożoności obliczeniowej ataku. $2^{56}\log(2^{56})$: utworzenie tablicy i jej posortowanie + $2^{56}\log(2^{56})$: przeszukanie tablicy. To jest mniej niż 2^{63} więc znacznie mniej niż 2^{112} (koszt brutalnego ataku na 2DES). Omówiony atak jest poważny i mógłby być realnie obecnie wykonany. Jego złożoność jest porównywalna z brutalnym atakiem na 1DES. Algorytm wymaga przygotowania dużych tablic, ale jest to obecnie wykonalne.

Podobny atak można przeprowadzić na algorytm 3DES. Jego złożoność czasowa wynosi 2^{118}, a przestrzeń do zgromadzenia tablic – 2^{56}. Złożoność czasowa ataku na 3DES uznawana jest za na tyle dużą, że nie stanowi realnego zagrożenia bezpieczeństwa.

Uważna uwaga: Algorytm 3DES jest uznany za przestarzały od 2018 roku i jego stosowanie będzie zabronione od 2023 roku.

Metoda 2: DESX (nieustandaryzowany przez NIST)

$E : K \times \{0,1\}^n \longrightarrow \{0,1\}^n$ szyfr blokowy

Definicja EX to $EX((k_1,k_2,k_3), m) = k_1 \oplus E(k_2, m \oplus k_3)$

Dla DESX: długość klucza = 64+56+64 = 184 bity

… ale go można łatwo zaatakować w czasie $2^{64+56} = 2^{120}$

Proszę zwrócić uwagę, że operacje xor:
$k_1 \oplus E(k_2, m)$ and $E(k_2, m \oplus k_1)$
tak naprawdę nie mają wpływu na zwiększenie bezpieczeństwa szyfru

26

Modyfikacja DESX zakłada zastosowanie 3 kluczy. Blok wiadomości jest szyfrowany z zastosowaniem xor z kluczem k_3. Potem na takim szyfrogramie wykonuje się cały algorytm szyfrowania DES (z kluczem k_2), a potem jego wyjście jest szyfrowane trzecim kluczem (k_1)znowu z zastosowaniem operacji xor. Ta metoda jest szybsza od 3DES, bo sam algorytm DES jest wykonywany tylko raz, a w między czasie wykonywanie są tylko operacje xor. Pomimo, że długość klucza wynosi 184 bity, jest możliwy atak na taką konstrukcję w czasie 2^{120}. Są opublikowane analizy, że taka konstrukcja jest odporna na siłowy atak.

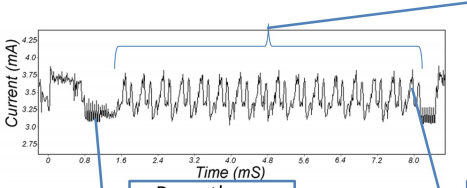
Inne ataki na szyfry blokowe

27

Ataki na implementację

1. Ataki przez boczny kanał:
   - Pomiar **czasu** podczas pracy algorytmu,
   - Pomiar **mocy** podczas pracy algorytmu

smartcard



[Kocher, Jaffe, Jun, 1998]

2. Ataki na błędy:
   - Spowodowanie błędu w ostatniej rundzie ujawnia ukryty klucz

⇒ Proszę nie **implementować** struktur kryptograficznych samodzielnie …
Należy posługiwać się gotowymi bibliotekami/urządzeniami, których kod i implementacja były opublikowane i poddane atakom i te ataki były opublikowane

28

Ataki na szyfry mogą przebiegać nie tylko na drodze matematycznej i informatycznej. Pierwszy przykład ataku zakłada, że blokowy algorytm szyfrowania jest zaimplementowany na karcie chipowej. Karta może być zastosowana do np. do płatności. Może zawierać wbudowany ukryty klucz do uwierzytelnienia płatności podczas podłączenia karty do terminalu. Osoba atakujące może przejąć taką kartę i zabrać do laboratorium. Karta może być zastosowana do szyfrowania/deszyfracji jakiś danych. W warunkach laboratoryjnych można dokładnie zmierzyć np. czas potrzebny karcie do przeprowadzenia szyfrowania i deszyfrowania. Czas tych operacji zależy od długości zastosowanego klucza i w ten sposób można wejść w posiadanie pewnych informacji o kluczu, a nawet wyekstrahować klucz z karty. Jest wiele przykładów takich procedur, gdzie mierząc czas operacji algorytmów szyfrowania uzyskuje się tajny klucz.

Innym typem ataków są ataki na błędy (fault attacks). Atakując kartę chipową można doprowadzić do jej nieprawidłowego działania, np. poprzez przyspieszenie zegara, lub podgrzewanie. Zmusza się wtedy procesor do popełniania błędów i generowania błędnych danych wyjściowych. Okazuje się, że jeśli doprowadzi się do błędu w ostatniej rundzie szyfrowania blokowego z otrzymanego szyfrogramu można wyodrębnić klucz. Ochrona algorytmu szyfrującego przed takimi atakami polega na uzupełnieniu oprogramowania/sprzętu w taki sposób, że sprawdza się, czy otrzymano prawidłowy wynik. Nie jest to zagadnienie trywialne, choćby z tej przyczyny, że nie wiadomo, czy nie wystąpił błąd w algorytmie sprawdzającym poprawność wyniku…

Innym podejściem jest mierzenie ilości mocy konsumowanej przez kartę podczas jej pracy. Na podstawie pomiarów można opracować wykresy. Ponieważ karty nie są zbyt szybkie można w zasadzie zilustrować pobór mocy w każdym cyklu pracy karty jak to pokazano dla algorytmu DES na slajdzie. Na wykresie widać, kiedy algorytm wykonywał początkową permutację, następnie 16 rund szyfrowania blokowego, oraz końcową permutację. Powiększenie uzyskanego wykresu pozwala na odczytanie klucza bit po bicie.

Nawet karty zabezpieczone przed takim atakiem (ograniczające ujawnienie konsumpcję mocy podczas obliczeń) nie są bezpieczne na inną odmianę ataku związanego z mierzeniem poboru mocy. Nazywa się on różnicowym atakiem z pomiarem mocy. W tym ataku dokonuje się pomiaru poboru mocy podczas bardzo wielu procesów kryptograficznych. Jeśli tylko istnieją bardzo niewielkie różnice w poborze prądu mogą one zostać wydobyte i posłużyć do odkrycia klucza. Te ataki zostały odkryte i opublikowane przez Paula Kochera i jego kolegów w instytucji Cryptography Research i obecnie dosyć pokaźna gałąź przemysłu związana z ochroną danych zajmująca się tylko zapobieganiu takim atakom.

Jeśli chodzi o ataki związane z czasem, to są one stosowane nie tylko na urządzeniach typu karty chipowe. Można sobie np. wyobrazić procesor wielordzeniowy, na którym na jednym rdzeniu wykonywany jest algorytm szyfrujący, a na drugim zainstalowano kod osoby atakującej. Rdzenie współdzielą ten sam obszar pamięci cache. Atakujący może mierzyć i podglądać przypadki nietrafienia pamięci podręcznej (cache misses) <<sytuacja, gdy program odwołuje się do pamięci cache, ale nie ma w niej potrzebnej danej lub instrukcji i musi ona być sprowadzona z innego poziomu pamięci (cache wyższego poziomu lub RAM)>> jakie wywołał algorytm szyfrujący. Okazuje się, że tylko śledząc nietrafienia pamięci cache można zgadnąć klucz stosowany przez algorytm. Jeden rdzeń może wydobywać informację od drugiego rdzenia obserwując tylko nietrafienia pamięci cache. W konsekwencji implementacje szyfrów blokowych są subtelnym zagadnieniem, które musi brać pod uwagę nie tylko siłowe ataki ale również ATAKI Z ZASTOSOWANIEM BOCZNEGO KANAŁU.

Powyższe przykładu ataku powinny przekonać słuchaczy tego kursu, że próby wymyślania własnych szyfrów blokowych i samodzielna implementacja algorytmów kryptograficznych to nie jest najlepszy pomysł. Po pierwsze trzeba wtedy zadbać, żeby w systemie kryptograficznym niemożliwe były ataki bocznymi kanałami. Po drugie implementacja musi być odporna na ataki na błędy. Dobrą praktyką jest stosowanie standardowych bibliotek, np. OpenSSL, czy innych.

Liniowe i różnicowe ataki [BS'89,M'93]

Podejście:
Mając *wiele* par wiadomość/jej szyfrogram, czy można odkryć klucz w czasie krótszym niż 2^{56} .

Liniowa analiza kryptograficzna(ogólnie) : niech c = DES(k, m)
Dla losowych k, m:

$$\Pr[\, m[i_1]\oplus\cdots\oplus m[i_r] \oplus c[j_j]\oplus\cdots\oplus c[j_v] = k[l_1]\oplus\cdots\oplus k[l_u] \,] = \tfrac{1}{2} + \varepsilon$$

Podzbiór wiadomości | Podzbiór szyfrogramów | Podzbiór kluczy

Dla DES $\varepsilon = 1/2^{21} \approx 0.0000000477$

29

Teraz zostaną przedstawione bardziej zaawansowane ataki na szyfry blokowe. Uwaga zostanie skupiona na zastosowanie tych ataków na DES. Ataki zostały odkryte przez Biham'a i Shamir'a w 1989 roku. Omówiona dalej wersja ataku została zaproponowana przez Matsui w 1993 roku.

Dysponujemy wieloma parami wiadomość/jej szyfrogram. Wiemy, że c jest szyfrogramem wiadomości m zaszyfrowanej przy pomocy klucza k. Wybieramy losowy klucz i losową wiadomość i w jakiś sposób dostrzegamy zależność pomiędzy wiadomością, szyfrogramem i bitami klucza.

Dokonuję operacji xor na podzbiorze wiadomości, xor na podzbiorze szyfrogramów, a następnie wykonuje xor na dwóch wyznaczonych wcześniej ciągach bitowych. Otrzymany ciąg bitowy jest porównywany z xor wykonanym na podzbiorze kluczy. Wyznaczane jest prawdopodobieństwo, że dane wyrażenie zachodzi. Gdyby nie istniała żadna zależność pomiędzy kluczami, wiadomościami i szyframi, prawdopodobieństwo prawdziwości takiego zdarzenia wynosiłoby ½. Ale przypuśćmy, że prawdopodobieństwo wychodzi ½ + ε (jakaś mała liczba). Jeśli tak, to w DES zachodzi jednak jakaś relacja pomiędzy kluczami, wiadomościami i szyfrogramami. Okazuje się, że taka relacja rzeczywiście zachodzi, ponieważ piąty S-box został źle zaprojektowany. Operacje w nim zbliżają się do funkcji liniowej. Jak się okazuje ta liniowa funkcja wpływa na cały obwód DES i powoduje pojawienie się wspomnianej relacji. Wartość ε wynosi 2^{21}.

Atak liniowy

- Istnieje twierdzenie, które mówi, że jeśli weźmiemy $1/\varepsilon^2$ losowych par (m, c=DES(k,m)), i wykonamy na nich $[m[i_1,\ldots,i_r] \oplus c[j_j,\ldots,j_v]]$, to z prawdopodobieństwem $\geq 97.7\%$ otrzymamy xor z kluczy zastosowanych do szyfrowania ($k[l_1,\ldots,l_u]$)
- Istnieje matematyczna metoda wydobycia części bitów klucza

Atak liniowy na DES

Dla DES, $\varepsilon = 1/2^{21} \Rightarrow$

mając 2^{42} par wiadomość/szyfrogram możemy znaleźć xor kluczy dla tych par w czasie 2^{42}

Pomijając szczegóły: możemy znaleźć 14 bitów klucza w czasie 2^{42}

Resztę bitów uzyskujemy z zastosowaniem ataku siłowego:
$56-14=42$ bitów w czasie 2^{42}

Całkowity czas ataku $\approx 2^{43}$ ($<< 2^{56}$) pod warunkiem posiadania 2^{42} losowych par wiadomość/szyfrogram

31

Nauka

- Niewielka skłonność piątego S-boxa do funkcji liniowej prowadzi do możliwości przeprowadzenia ataku na DES z czasem 2^{43}
- Wniosek:
  - Nie projektujcie szyfrów samodzielnie

Ataki kwantowe

Ogólny problem przeszukiwania

Niech $f: X \longrightarrow \{0,1\}$ będzie funkcją.

Cel: znalezienie $x \in X$ takich, że $f(x)=1$.

Klasyczne komputery:
złożoność najlepszego algorytmu wyszukiwania = **$O(|X|)$**
(jest liniowa w zależności od liczby elementów zbioru X)

Komputer kwantowy [Grover '96] : czas = **$O(|X|^{1/2})$**
(czas przeszukiwania będzie proporcjonalny do pierwiastka kwadratowego z rozmiaru zbioru)

Działają już pierwsze komputery kwantowe…

33

Ogólny problem przeszukiwania można sformułować następująco. Mamy funkcję, której dziedziną jest zbiór X. Funkcja ma dwie możliwe wartości wyjściowe 0 i 1. Tak się dzieje, że w większości wypadków funkcja zwraca 0. Istnieje wejście funkcji, które powoduje, że zwraca ona wartość 1. Może się zdarzyć, że istnieje tylko jedna taka dana wejściowa, dla której funkcja zwraca 1. Naszym celem jest znalezienie tej danej.

Co robi klasyczny komputer, jeśli nic nie wie o zbiorze do przeszukania? Przeszukuje wszystkie możliwe elementy zbioru X w celu znalezienia tego, który zwróci 1. Złożoność obliczeniowa takiego problemu jest funkcją liniową liczebności zbioru X.

Dysponując komputerem kwantowym problem przeszukiwania można rozwiązać szybciej. Okazuje się, że złożoność obliczeniowa kwantowego algorytmu przeszukiwania (opracowanego przez Grover'a) jest funkcją pierwiastka kwadratowego z liczebności zbioru X.

Siłowy kwantowy atak szyfry blokowe

Mając m, c=E(k,m) definiujemy

$$f(k) = \begin{cases} 1 & \text{jeśli } E(k,m) = c \\ 0 & \text{w przeciwnym wypadku} \end{cases}$$

Funkja z dziedziną K → wszystkie klucze

Algorytm Grover'a

⇒ komputer kwantowy może znaleźć k w czasie $O(|K|^{1/2})$

dla DES: czas $\approx 2^{28}$, dla AES-128: czas $\approx 2^{64}$

jeśli do użytku wejdą komputery kwantowe będziemy musieli stosować klucze 256-bitowe (np. AES-256)

34

Komputery kwantowe będą miały duże znaczenie dla kryptografii. Poszukiwanie klucza można sprowadzić do algorytmu przeszukiwania. Jeśli zaatakowany takim algorytmem będzie DES, to zajmie mu 2^{28} czasu (ok. 200 milionów, czyli ok. 1 ms). Zwróćmy również uwagę, że szyfrogram AES z 128-bitowym kluczem mógłby wtedy być złamany z czasem 2^{64}. A taki czas łamania wskazuje, że algorytm nie jest bezpieczny. Wprowadzenie komputerów kwantowych spowoduje, że będzie trzeba posługiwać się kluczami o długości co najmniej 256 bitów (np. AES-256).